An Engineering Biology Approach and Infrastructure for Automating Workflow in a Biofoundry

Alexis L. Casas

Imperial College London
Department of Bioengineering

Submitted for Degree of Doctor of Philosophy

Copyright Statement

‘The copyright of this thesis rests with the author. Unless otherwise indicated,its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC).
Under this licence, you may copy and redistribute the material in any medium or format. You may also create and distribute modified versions of the work.
This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose.
When reusing or sharing this work, enure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes.
Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.’

Statement of Originality

All work and data contained within this thesis is my own, except where referenced.

The work presented in thesis is part of broader research being undertaken in the Kitney Lab. All the experimental work has been carried out in the London Biofoundry – SynbiCITE.

The PASIV method and the cMatch algorithms and software presented in Chapter 3 have been published in peer-reviewed journals (ACS Synthetic Biology and Frontiers in Bioengineering, respectively). The experimental work on automating the generation of standard curves presented in Chapter 4 has led to an article which has been submitted to a peer-reviewed journal and is currently under review.

The framework and systems presented in Chapter 6 and 7 enable the automation of both experimental and computational workflows in a Biofoundry utilising graphs as a structure. The Biofoundry Information Manager and the Biofoundry Automation Manager have been implemented and validated with the data obtained from the lycopene case study. In the thesis I present a system that we think offers both a better way to orchestrate workflow in a Biofoundry - and a unique and novel approach to engineering biology. This provides the ability to trace the provenance of any computational data analysis obtained from experimental data, back to any DNA biopart used experimentally.

Acknowledgements

I would like to express my heartfelt gratitude to Professor Richard Kitney for his unwavering support and encouragement throughout my PhD journey. His guidance has been invaluable, and I truly appreciate the insightful discussions and good laughs we shared during our weekly group meetings.

I'm extremely grateful to my dear colleague Dr. Matthieu Bultelle for his support, guidance, and advice throughout the project and the preparation of this thesis.

I would also like to express my gratitude to my PhD thesis committee members, Professor Karen Polizzi and Professor Anil Wipat, for their thought-provoking questions and discussions during my viva.

I am also thankful to my colleague Dr. Charles Motraghi, who generously provided his knowledge, expertise, and help in the laboratory. Special thanks to Dr. Marko Storch for his help, advice, and tremendous support at the London Biofoundry and to Dr. Matthew Haines for our various discussions and projects.

I would like to thank senior staff, colleagues, and students, who have helped me in one way or another at the London Biofoundry: Dr. Christopher Waite, Dr. Matthew Pope, Keltoum Boukra, Livia Soro, Benedict Carling, James Marshall, Dr. Kirsten Jensen, Dr. Soo Mei Chee, Dr. David Bell, Dr. Michael Crone, Dr. Miles Priestman, and Dr. Marta Ciechonska.

I would also like to thank everyone that I interacted with from the iGEM Engineering Committee, the Build-a-Cell community, the Bio-Design-Automation Consortium, Friendzymes, the Bioprotocols Working Group, and all the iGEM teams I had the pleasure of advising and working with: Paris-Bettencourt 2020-2022, Imperial 2020-2022, Marburg 2020-2022, Costa-Rica 2022, and Dr. Maria Savino, for the valuable discussions we had along the way.

In addition, I would like to mention Dr. Vasily Shenshin, who initially encouraged me to explore the work of the Kitney lab and join the London Biofoundry.

Finally, I would like to thank my family and my friends, especially Dr. Marie Heintz and Dr. Lionel Friedmann, Carole Bui-Brandt and Valentin Brandt , Dr. Antoine Levrier, Santino Nanini, Maksim Baković, and Dr. Olivier Borkowski for their steadfast support during this endeavour.

Abstract

Engineering biology applies engineering principles to the design and construction of biological systems. Since its inception twenty-five years ago, approaches to engineering biology have transitioned from traditional wet lab manual methods, to using individual automated devices, to a complete integration of workflows in fully automated laboratories called Biofoundries.

Engineering biology applications require the navigation of large design spaces and the control of assays. To explore large design spaces in reasonable time and cost statistical methods need to be employed. The solution is automation, which provides a throughput high enough to enable the repetition of large numbers of assays. Automation provides higher control and reduces the influence of human operators. High throughput automation generates large measurement datasets which need to be dealt with computational automation.

The work presented in this thesis took place in the London Biofoundry - SynbiCITE. The contribution of the work presented here is a framework and a software system for supporting infrastructure in a Biofoundry. To study the problems that arise from automating an engineering biology application, a case study based on Lycopene was undertaken.

The lycopene application was chosen because it is a good metabolic engineering case study, based on a model pathway with a large design space. The goal of the case study was the optimisation of bioproduction of a metabolite of high bioeconomic interest, lycopene. Through the development of the lycopene case study, lessons were learned: the necessity to design a scoping study and the necessity to care for errors induced by automated methods. The work presented in the thesis utilises lessons learned from the case-study and presents an infrastructure/framework to support automation orchestration in a Biofoundry. Our framework was validated on data collected with the lycopene application.

Table of Contents

Chapter 1 Introduction 13

1. Overview 13

1.1 Engineering biology 13
1.2 Core aims of this thesis 14

2. The Need for Reproducibility and Replicability 16

2.1 Definitions 16
2.2 The problem 16
2.3 Reproducibility and replicability in engineering biology 16

3. Automation 17

3.1 Definition of automation 17
3.2 Levels of automation 18
3.3 Laboratory automation 19
3.4 Benefits of laboratory automation 20
3.5 Limitations of automation 22

4. Biofoundries 23

5. Strategy Overview 24

5.1 Goal and structure of the thesis 24
5.2 Application 25
5.3 Initiation of the cycle of a design of experiments (DoE) application 26
5.4 Automated standard curve 26
5.5 Automated workflows 27
5.6 The Biofoundry Information Manager 27
5.7 The Biofoundry Automation Manager 28

Chapter 2 Application 29

1. Metabolic Engineering 29

1.1 Introduction to metabolic engineering 29
1.2 Rational metabolic engineering 31
1.3 Evolutionary approaches 32

2. Combinatorial Pathway Engineering 33

2.1 Combinatorial metabolic engineering 33
2.2 Strategies 34

3. Combinatorial Pathway Engineering - Challenges and Solutions 36

3.1 Large design spaces 36
3.2 DNA assembly 37
3.3 High-throughput automation 39
3.4 Learning methods - DoE and machine learning 39

4. Application Workflow 41

4.1 The design-build-test-learn (DBTL) cycle 41
4.2 Design 41
4.3 Build 42
4.4 Test 43
4.5 Learn 43

5. Case Study: Optimisation of Lycopene Production 44

5.1 Lycopene 44
5.2 The lycopene synthetic pathway 45
5.3 Design of the lycopene operon 47
5.4 Build with BASIC DNA assembly 48
5.5 Test 50

6. Summary 51

Chapter 3 Initiation of the DoE Cycle 52

1. Introduction 52

2. Failure to Launch the DoE Cycle because of Burden and Toxicity Issues 53

2.1 Replication study - observed experimental failure 53
2.2 Failures and failure modes 53
2.3 Initiation of the DoE cycle 56

3. The PASIV Approach 58

3.1 The PASIV workflow 58
3.2 Application to the lycopene exemplar 60

4. Experimental Results (PASIV) 62

4.1 Preliminary investigation of the design space 62
4.2 Screening results 66
4.3 Location and multiplicity 68
4.4 Data analysis of the titration assay of lycopene concentration 72
4.5 Identification from the growth regions in the design space of a region for further investigation 75

5. cMatch (a Construct-Matching Software) 77

5.1 Need for cMatch 77
5.2 Construct-matching with cMatch 79
5.3 cMatch algorithms for construct-matching 83

6. Results (cMatch) 92

6.1 Synthetic constructs violacein 92
6.2 Testing algorithm CM_0 92
6.3 Testing algorithm CM_1 93
6.3 Testing algorithm CM_2 96
6.4 Final comparison of the algorithms 96

7. Summary 98

Chapter 4 Automated Liquid Handling 100

1. Introduction 100

2. Automatic Liquid Handling 103

2.1 Automating pipetting 103
2.3 Robotic workstations 104
2.3 Challenges with liquid handling 104

3. Automation Method for Creating a Standard Curve 106

3.1 The challenges of lycopene and DMSO 106
3.2 Choice of dilution schemes 107

4. Results 111

4.1 Phase 1 - preliminary investigation - geometric scheme 111
4.2 Phase 1 - preliminary investigation - linear dilution scheme 117
4.3 Error assessment methods 119
4.4 Phase 2 - lycopene in DMSO - linear dilution scheme 122
4.5 Phase 3 - lycopene in DMSO - final study 128

5. Summary 132

Chapter 5 Automated Research Workflows 134

1. Introduction - Where we are Now - What are the Needs ? 134

1.1 Automated research workflows 134
1.2 The rise of scientific workflows 135
1.3 Lessons from the case study 135

2. Tasks and Workflows 139

2.1 Tasks 139
2.2 Workflows 142
2.3 Tasks, workflows and composability 146
2.4 Protocols 153
2.5 Examples of real world workflows 155

3. Conceptual Framework and Digital Twins 158

3.1 Digital twins 158
3.2 Modelling the physical world 159

4. Experiments in Synthetic Biology 160

4.1 Studying a biological cohort 160
4.2 The operational pipeline 161

5 Summary 162

Chapter 6 Biofoundry Information Manager 163

1. Model graph - Representing Workflows 167

1.1 Structure: directed graphs 167
1.2 Model graph 168

2. Execution graph: Representing Execution 170

2.1 Workflow execution and runs 170
2.2 Execution graph 174

3. Data Dependency Structuring and Data Provenance Querying 178

3.1 The importance of data provenance in experiments 178
3.2 Example of the automated dilution standard workflow 181

4. Implementation 185

4.1 Using a graph database 185
4.2 Workflow libraries 186

5. Exemplars 189

5.1 Workflow examplar 1 - BASIC DNA assembly (computational) 189
5.2 Workflow examplar 2 - automated serial dilution 193

6. Summary 197

Chapter 7 Biofoundry Automation Manager 198

1. Workflow Orchestration 203

1.1 Definitions 204
1.2 The different roles of the orchestrator. 207
1.3 Examples of orchestrators 208

2. Biofoundry Automation Manager 208

2.1 Introducing the BAM 208
2.2 The BAM architecture and software system overview 210
2.3 Workflow libraries 213
2.4 The choice of Airflow as an orchestrator 214
2.5 The BAM implementation 216

3. Orchestration Repeats with the BAM 219

3.1 Repeats and dealing with repeats in workflows 219
3.2 Implementing the right strategy for repeats 223

4. Examples of Workflow Execution 226

4.1 Generic example 226
4.2 Computational example (BASIC DNA assembly) 236
4.3 Experimental example (automated standard curve) 239

5. Summary 244

Chapter 8 Discussion 246

1. Strategy Overview - Goal and Application 246

2. PASIV and cMatch - Discussion (Chapter 3) 247

2.1 Overview 247
2.2 PASIV - discussion 248
2.3 cMatch - discussion 252

3. Development of Automated Standard – Discussion (Chapter 4) 257

3.1 Overview 257
3.2 Discussion 259
3.3 Design and methodologies 259
3.4 Lycopene in DMSO 260
3.6 Conclusion 263

4. Automation of Workflows in the Biofoundry - Discussion (Chapter 6-7) 264

4.1 Overview 264
4.2 Discussion 265
4.3 Conclusion 268

Chapter 9 Future Work 269

1. Improvements to cMatch 269

2. The Biofoundry Information and Automation Manager 271

Bibliography 274

Appendices C3 306

C3 - Methods 306

BASIC DNA assembly 306
Preparation of BASIC linkers and BASIC bioparts 306
BASIC reaction 306
Magbead purification 307
Assembly reaction 307
Transformation 307
Pooled transformation 308
Cloning and cultivation 308
Extraction and measurement of lycopene 308
Sequence analysis and part-matching 309
Yield calculation 310
Lycopene Absorbance 310

C3 - Promoter Strength 311

C3 - Pool Number - Gene Order 311

C3 - Translation Rates 312

C3 - Visualisation By Gene Order 314

C3 - Visualisation By Promoter 322

C3 - JSON Template 324

C3 - cMatch Logs 326

Appendices C4 328

C4 - Methods **328**

Preparation of the lycopene standard 328
Automated dilution liquid-handler setup 328
Food colourant 329
Spectral measurements 329

C4 - Section 1 - Implemented Dilution Schemes **330**

1.1 - Geometric dilution scheme 330
1.2 - Linear dilution scheme 331

C4 - Section 2 - Geometric Dilution Scheme - Food Colorant and Water **332**

2.1 - Run 55 - Plate 01 - automated 332
2.2 - Run 55 - Plate 02 - manual 335
2.3 - Run 57 - Plate 01 338
2.4 - Run 57 - Plate 02 340
2.5 - Run 55 & 57 - comparison automated vs manual 342
2.6 - Assessing the effects of the iterative transfers 343

C4 - Section 3 - Geometric Dilution Scheme - Lycopene in DMSO **346**

3.1 - Run 62 - Plate 01 - automated 346
3.2 - Run 63 - Plate 01 - manual 348
3.3 - Assessing the effects of the iterative transfer 350

C4 - Section 4 - Linear Dilution Scheme - Preliminary Studies **355**

4.1 - Development runs - food colorant in DMSO 355
4.2 - Stratified analysis 358
4.3 - Main analytical approach 359
4.3 - Spectrum-based approach 363

C4 - Section 5 - Linear Dilution Scheme - Lycopene in DMSO **365**

5.1 - Development runs lycopene in DMSO 365
5.2 - Measurement study 371

C6 - Automated Protocols **373**

6.1 - Automated serial dilution protocol 373
373

Appendices C6 381

C6 - Model Graph **381**

C6 - Examplar 2 – Serial Dilution for Standard Curve **385**

Appendices C7 398

C7- Examples **398**

398

This page was intentionally left blank.

Chapter 1

Introduction

1. Overview

1.1 Engineering biology

The field of synthetic biology – a synonym to engineering biology – is fundamentally an engineering discipline (Knight, 2005). Synthetic biology strives to make the engineering of biology easier and more predictable (The Royal Academy of Engineering, 2009). Synthetic biology lies at the intersection of engineering, biological sciences, and computational modelling (Polizzi & Kontoravdi, 2013).

Engineering biology has been enabled by a number of technological advances: molecular techniques used to assemble recombinant DNA molecules, faster and cheaper DNA sequencing technology approximating Moore's Law, which significant landmark has been the whole human genome project (Shendure et al., 2017) and cheaper DNA synthesis which follows Carlson's curve (Carlson, 2003; The Economist, 2006).

Synthetic biology applies to biology techniques normally used in engineering design development where the specification of a part, device or system is defined and characterised. The system gets built and successive iterations aim to improve it. This approach is known as the engineering cycle (or Design-Build-Test-Learn (DBTL) cycle). The Royal Academy of Engineering defines synthetic biology in simple terms: 'Synthetic biology aims to design and engineer biologically based parts, novel devices and systems as well as redesigning existing, natural biological systems.' (The Royal Academy of Engineering, 2009). In the work presented in this thesis we will use both terms 'engineering biology' and 'synthetic biology' interchangeably.

Synthetic biology started twenty five years ago when Gardner, Cantor & Collins (2000) and Elowitz & Leibler (2000) demonstrated, respectively, a genetic toggle switch and a genetic oscillator. What set these two works apart from other biology papers was that engineering

principles were applied to design circuits encoded in DNA (Peccoud, 2016).

Synthetic biology has been enabled by a convergence of three fields: biology, physical science and engineering.

In his seminal paper on engineering biology, Endy (2005) lists three key engineering ideas that can be applied to engineering of biology: standardisation, decoupling and abstraction.

In the past twenty-five years, in order to build the aforementioned designed DNA circuits, the early approaches employed traditional wetlab manual methods, which evolved into using individual automated liquid handling devices and we now have integration of devices in fully automated Biofoundries, as illustrated in Figure 1-1.

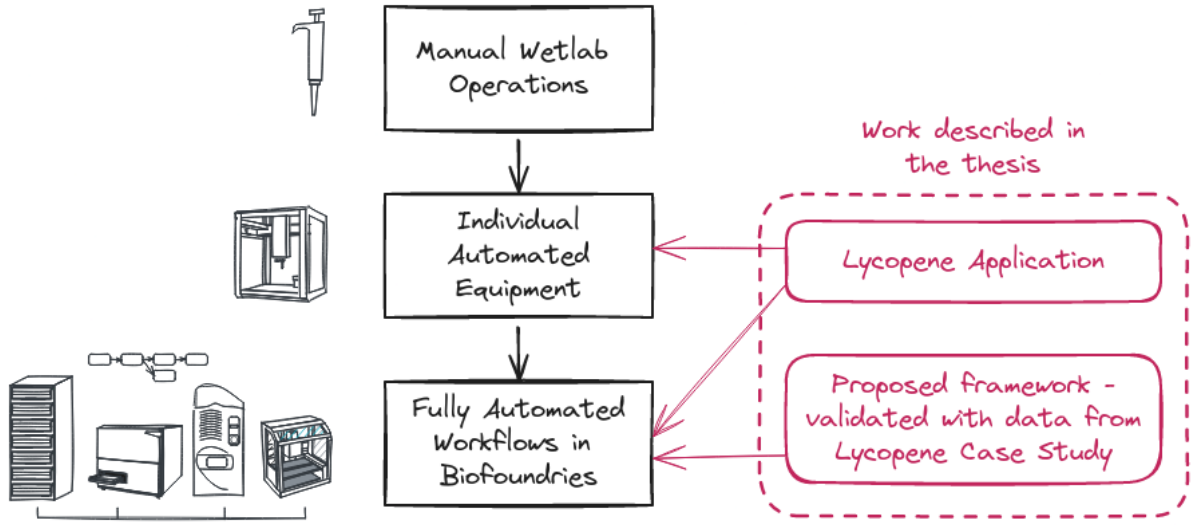


Figure 1-1 – From manual wetlab work to fully automated workflows in Biofoundries.

1.2 Core aims of this thesis

Applications in engineering biology require the navigation of large design spaces and the control of potentially large numbers of design factors and confounding variables. To explore large design spaces and assays for a fixed set of controlled parameters in a reasonable time and cost frame, statistical methods need to be employed. These methods require the repetition of the construction of a large number of biological systems and their corresponding assays. Additionally biology is stochastic. To overcome the variability of the measured features of engineered biological systems, a larger number of measurements are required, ideally, a statistically significant number of assays need to be done.

The solution to working with large design spaces is automation which provides a working throughput that is high enough to enable the repetition of a large number of assays.

Additionally, automation promises higher control, as it reduces the influence of human operators. High throughput automation generates large measurement datasets which in turn creates another problem of dealing with large quantities of data. Information organisation problems can be solved with computational automation (Peccoud et al., 2022).

The strategy that underlies the thesis is illustrated in figure 1.1. The central column of the diagram illustrates the evolution of workflows in engineering biology. This starts with fully manual operations in the wet lab. Over the last few years has evolved (in many laboratories around the world) into a combination of manual and automatic operations on individual piece of equipment – as illustrated in the centre block of the diagram. The final stage in the evolution is fully automated workflows in Biofoundries. To date, this has not been fully realised. The core of the thesis therefore comprises the development of a strategy and implementation of automated workflows in a biofoundry. In order to develop and test the strategy and its implementation it was necessary to develop a wet lab case study based on lycopene. Consequently, the goal of the work presented in the thesis was to develop a framework and software to support workflow automation in a Biofoundry.

The work presented in this thesis has taken place in the London Biofoundry – SynbiCITE. Biofoundries are biology laboratories equipped with automated equipment on which to run consecutive automated assays. This automated equipment allows individual repetition of tasks but assays are constituted from an ordered succession of multiple tasks. it is therefore necessary to enable the integration of multiple tasks into workflows, and automate these workflows.

The lycopene application was chosen because it is a good metabolic engineering case study based on a model pathway with a large design space. The first part of the thesis aims to identify problems that can arise in a metabolic engineering application (here the Lycopene study). The goal of the application is the optimisation of production of a metabolite, lycopene, in bacterial cells. Lycopene is a carotenoid present in many plants and organisms, with high bioeconomic interest. Additionally, lycopene is encoded with a simple three gene operon which can be built from a large combinatorial of genetic parts which yield a combinatorial space which is large enough for us to study but not too large (time and costs).

Through the development of the lycopene case-study, we learned a number of lessons: the necessity to design a scoping study, the necessity to care for errors induced by automated methods. From these lessons we reached a number of conclusions and developed some guidelines. On this basis, we developed a framework and a system to manage laboratory automation both mechanically and computationally in a Biofoundry. The framework was validated on data collected with the lycopene application.

2. The Need for Reproducibility and Replicability

2.1 Definitions

Reproducibility and replicability are cornerstones of the scientific method (McNutt, 2014)

The terms reproducibility and replicability have different meanings and uses across science and engineering (Plesser, 2018). The meaning and definition of these terms has led to confusion in collectively understanding problems in reproducibility and replicability (Committee on Reproducibility and Replicability in Science et al., 2019).

2.2 The problem

Marcia McNutt, *Science* editor-in-chief in 2014, starts her editorial of the January *Science* issue with the statement: 'Science advances on a foundation of trusted discoveries. Reproducing an experiment is one important approach that scientists use to gain confidence in their conclusions' (McNutt, 2014). McNutt raises the issue of recent reports which alarmed the scientific community, which concluded that peer-reviewed preclinical studies were not reproducible. More recently a committee of the U.S. National Academy of Sciences, Engineering and Medicine published a very complete report on the topic of Reproducibility and Replicability in Science (Committee on Reproducibility and Replicability in Science et al., 2019). The progress of science relies on the generation of replicable data. Despite the critical significance of reproducibility, there has been limited exploration into the reproducibility of life science publications (Jessop-Fabre & Sonnenschein, 2019).

2.3 Reproducibility and replicability in engineering biology

The terms reproducibility and replicability are defined by the U.S. National Academy of Sciences (Committee on Reproducibility and Replicability in Science et al., 2019). We will adopt these two definitions along the thesis:

'**Reproducibility** is obtaining consistent results using the same input data; computational steps, methods, and code; and conditions of analysis.' This definition is synonymous with 'computational reproducibility'.

'**Replicability** is obtaining consistent results across studies aimed at answering the same scientific question, each of which has obtained its own data. Two studies may be considered to have replicated if they obtain consistent results given the level of uncertainty inherent in the system under study.'

Jessop-Fabre & Sonnenschein (2019) state that synthetic biology aims to introduce

engineering principles into Life Sciences in hope that it can greatly improve the reliability of the DBTL cycle and that any engineering discipline must be predictable and reproducible. We are now in the third decade of the discipline and significant technical progress has been made and synthetic biology products impact industries such as healthcare and manufacturing (Voigt, 2020). However Lux, Strychalski & Vora (2023) argue that the efficiency and success rate of such transitions are far lower than anticipated given all the resources – financial, experimental and intellectual – that have been poured into the domain. Similarly to the other fields of science, synthetic biology faces the reproducibility challenge too.

Peccoud et al. (2022) state that designing experiments with replicability in mind calls for an analysis of factors that might contribute to the variation of the observed data and that controlled factors in an experiment are expected to cause the effect observed in data. The authors give the example of plasmid designs that may lead to different levels of gene expression but emphasise that extraneous factors influence the observed data in unexpected ways. Factors such as the difference in operators performing the experiment or the different version of the protocol or the different kits used may have unanticipated effects on the outcome of the experiment. Thus it is necessary to unwind how different factors contribute to the variation of data by planning and documenting data collection which help associate the controlled variable of interest with the observed variation in experimental outcomes (Peccoud et al., 2022).

It is worth noting that replicability studies are rare in synthetic biology. The most prominent have been conducted in the context of iGEM interlab studies (Beal et al., 2018; Cole et al., 2019; Beal et al., 2020), and DARPA (Goldman et al., 2022) – where identical methodologies were used by participants with markedly different outcomes.

3. Automation

The successful implementation of automation technology can help to improve both the throughput and reproducibility of experiments (Jessop-Fabre & Sonnenschein, 2019).

3.1 Definition of automation

Dictionaries define automation as ‘The use of machines and computers computers that can operate without needing human control’ (Cambridge) and ‘The use of machines and computers to do work that was previously done by people’ (Oxford).

Biermann et al. (2021) define automation in the context of manufacturing as the technology by which a procedure is executed without human assistance.

In this thesis we will discuss automation as the process to automate a task or a series of tasks in the context of wetlabs and Biofoundries.

3.2 Levels of automation

Frohm, Lindström & Winroth (2008) note that automation is not all or nothing but it should be seen as a continuum of automation levels defining the lowest level as fully manual to the highest being full automation without any human involvement.

Frohm et al. define two basic classes of automation: automation through mechanisation and automation through computerisation (as illustrated in Figure 1-2). Mechanisation is defined as the replacement of human muscle power (material and energy transformation). Computerisation is defined as the replacement of cognitive tasks (mental activity and sensory processes) which includes collecting, storing and analysing information to control the manufacturing process (Frohm, Lindström & Bellgran, 2005).

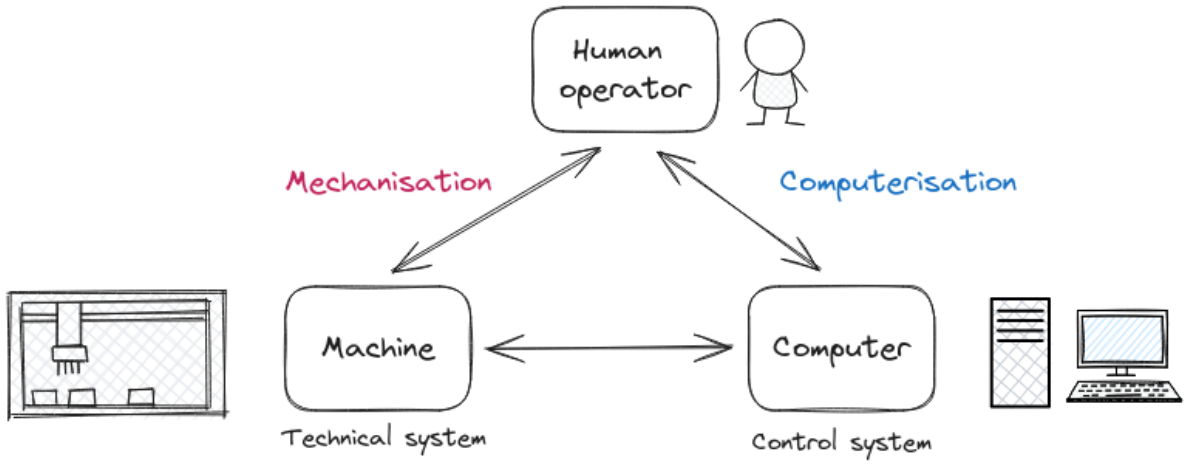


Figure 1.2 – Mechanisation and computerisation (adapted from Frohm et al., 2008)

Frohm, Lindström & Winroth (2008) provide an exhaustive review of the categorisation level of mechanisation and conclude that level of automation can be seen as three levels: manual, semi-automatic and automatic. All tasks ranging from manual tasks to fully-automated tasks fall into these three levels. Most manufacturing systems consist of both humans and automation in connection and thus most tasks fall between manual and full automation.

Computers are implemented in modern manufacturing operations for optimal performance control (Frohm, Lindström & Bellgran, 2005). Many taxonomies for classifying the different levels of computerisation have been reviewed by Frohm, Lindström & Winroth, (2008).

3.3 Laboratory automation

Some researchers state that automation is revolutionising biology (Glökler, Schütze & Konthur, 2010; Gallup, Ming & Ellis, 2021). However Jessop-Fabre & Sonnenschein (2019) argue that for many years laboratory automation has promised this revolution but academic laboratories have only minimally embraced conventional lab automation tools like liquid handling robots. Nevertheless, the methods frequently employed in synthetic biology are often routine tasks, making them prime candidates for automation. Automation technology can help improve the throughput and reproducibility of experiments. Although high costs and lack of flexibility might have slowed its widespread adoption in academia (Jessop-Fabre & Sonnenschein, 2019). The pharmaceuticals industry has been a major user of laboratory robotic equipment and Bogue (2012) argues that the history of lab automation parallels the development of modern drug discovery within this industry.

Many techniques used within synthetic biology on a routine basis are suitable for automation (Jessop-Fabre & Sonnenschein, 2019) – Holland and Davies (2020) indeed noted that 89% of articles studied by Groth and Cos (2017) featured a manual protocol that has an automated alternative. However, the degree of automation in biotechnology remains very low compared to classic manufacturing such as the automotive industry (Biermann 2021).

Laboratory experiments generally include multi step protocols containing many liquid handling and processing steps – which are highly labour intensive and inefficient when performed manually (Najmabadi, Goldenberg & Emili, 2006). Despite such inefficiencies, the adoption of traditional laboratory automation technologies such as liquid handling robots has been slow in academia (Jessop-Fabre & Sonnenschein, 2019).

Advances in automation technologies combined with new protocol sharing methods and protocol management systems allow researchers to gain the benefits of automation (Jessop-Fabre & Sonnenschein, 2019). A notable gap between academia and industry remains as significant barriers to automation adoption remain for those without the appropriate resources and expertise (Stephenson et al., 2023).

The flexibility of a laboratory automation system is defined by Najmabadi, Goldenberg & Emili (2006) as ‘the capability of an automation system to be adapted to a new laboratory protocol with the minimum cost and downtime of the existing system’. Automation system flexibility is thus very important for biotechnology research laboratories where working protocols are continuously modified as part of the research. In contrast to human scientists and technicians who are trained to execute a vast array of laboratory operations manually and can adapt, almost all existing laboratory automation equipment is designed to do one or a few related tasks and consequently lack flexibility. This lack of flexibility of commercial products is one of the most frequent complaints of laboratory automation system end users

(Linshiz et al., 2013), and ‘one of the great challenges for laboratory automation is the design of equipment that can be reconfigured to execute a wide range of experimental tasks’ (King, 2011).

One of the fundamental principles of automation is that there is a trade off between flexibility and throughput: If high flexibility is required, then throughput is generally low. Figure 1-3 illustrates this principle. An extreme example is the craftsman or artist producing a work to order. At the opposite extreme, a production line can be dedicated to a single unvarying model. Somewhere between these extremes lies the majority of modern industry – often relying on programmable machinery and robotics to allow rapid re-configuration between models. The leading automation vendors are constantly seeking to maximise both flexibility and throughput (Harding, Bradford & Brown, 2002).

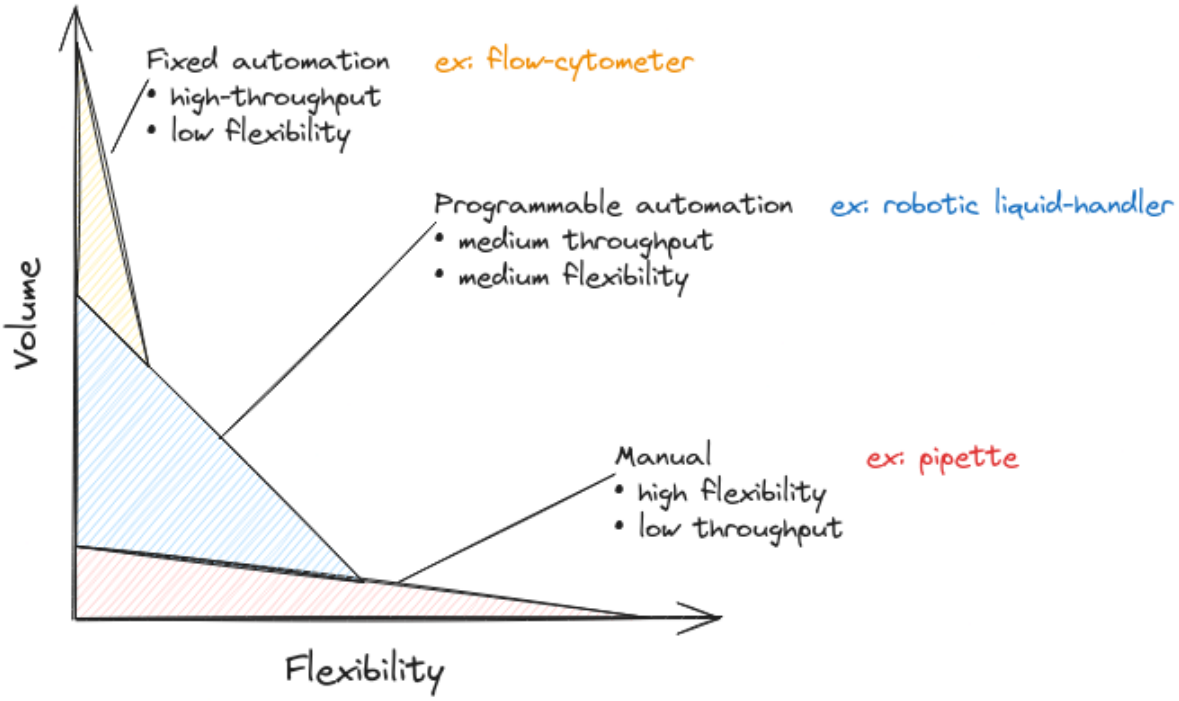


Figure 1-3 – Volume vs flexibility in the context of lab automation (adapted from Harding et al. (2002)).

3.4 Benefits of laboratory automation

We now list the benefits of laboratory automation.

- **Higher throughput**

  ‘Automation opens up the possibility of establishing high-throughput synthetic biology, similar to modern engineering technologies, where thousands of biosystems can be potentially built in short periods of time’ (Tellechea-Luzardo et al., 2022). The development of high-throughput screening (HTS) methods in the pharmaceutical

industry has been of particular importance for the development of laboratory automation since the 1980s. HTS aims to increase the number of samples processed per unit of time (Thurow, 2022). The only effective way to fully use the potential of current biotechnology research is to facilitate the use of HT (high-throughput) automated systems (Najmabadi, Goldenberg & Emili, 2006).

- **Replicability**

  Biological engineering is not an exception when it comes to the issue of replicability of experiments and a study from Goldman et al. (2022) designed an extensive replication experiment campaign, assessing the performance of novel logic gates implemented in *S. cerevisiae* yeast cells by Gander et al. (2017). The experimental campaign aimed to replicate the results of the original study using a very high degree of automation. They conclude that their results are consistent with the results from the original experiments, but they note that the performance of the current set of designs was unreliable for many gates (Goldman et al., 2022).

  This is problematic for the field of synthetic biology which applies engineering modularity concepts and designs standardised modular genetic parts (Knight, 2003) as parts need to be characterised and parts are supposed and expected to behave in the same way in varying contexts, which is not completely possible (Mutalik et al., 2013).

- **Reducing human error**

  The work of Yeow (2014) shows that human error is significantly affected by stress, repetition and fatigue in the manufacturing environment. Automation can be seen as a way to reduce the errors of tedious and repetitive tasks that are highly error-prone when conducted by humans (Jessop-Fabre & Sonnenschein, 2019). Automation can assist in improving replicability in three ways: a reduction in human-induced variability, an increase in the rate of data generation, and a decrease in contamination (Holland & Davies, 2020).

- **Laboratory efficiency**

  Holland & Davies (2020) define laboratory efficiency as the rate of production divided by the resources needed to accomplish this rate (ex: labour, input materials). Similarly to a manufacturing company investing in automation to increase its rate of production and reducing its resources, a life science research laboratory which invests in automation can improve the efficiency of its researchers and can achieve a greater rate of experimental output (Holland & Davies, 2020). In the context of academic research, laboratory automation benefit is the time saved by researchers.This differs from the industry, in academia efficiency is measured as time input to experimental output ratio and the real benefit from laboratory automation is

therefore the time saved by researchers (which can be spent on other experiments). Automation enables the transition from manual labour to cognitive labour (Kaber et al., 2009). Automation equipment allows an operator to set a protocol and walk away to think and focus on other tasks. Researchers often work on multiple projects in parallel and have many responsibilities outside of the laboratory.

- **Faster translation**
  Automation plays an important role in laboratories engaged in applied research who develop applications such as cell-based therapies, pharmaceutical developments or tissue engineering (Holland & Davies, 2020). The transition of these technologies from a research domain to final usage in a clinical setting is difficult (Ochs et al., 2017; Hua, De Matos & Metselaar, 2018). An early integration of automation enables process transitions at later stages of clinical development and commercialisation where automation becomes necessary (Rafiq & Thomas, 2016).

3.5 Limitations of automation

Automation in laboratories comes with its own set of limitations too. First automation comes with a high cost of setting up and running infrastructure; additionally to purchasing automated robotic and high-throughput equipment, costs include consumables, software and support for skilled personnel to set up and run the infrastructure. (Holowko et al., 2021)

A new automated infrastructure implies new ways of working, creating and programming new methods and thus implies some additional training and knowledge for the researchers.

Using BioFoundries effectively requires a paradigm shift in how we do biological engineering, and training the next generation to effectively exploit these technologies is also extremely important.

Although being a beneficial addition to the wetlab (as seen in Section 3.5), automation comes with a series of technical limitations:

- **Loss of flexibilty**
  Goldman et al. (2022) use of automation allowed them to ‘scale up the experiments by orders of magnitude’ but lost some flexibility in other aspects of the original experiments for example, they were not able to vary the temperature smoothly by degrees.
- **Dead volume in automated liquid-handling**
  Traditional liquid handlers are also limited in ways such as requiring ‘dead volumes’ of reagents inside wells, which can increase the cost of experiments (Jessop-Fabre & Sonnenschein, 2019).
- **Requirement for frequent quality assessment**

Automatic liquid handling devices need strict and frequent quality assessments to maintain high levels of accuracy (Chai et al., 2013). This is especially important for HTS in order to minimise false-positive and false-negative rates (Malo et al., 2006).

- **Additional time for set-up and optimisation of new protocols**
  There is an additional investment of time required for the set-up and optimisation of new protocols (Jessop-Fabre & Sonnenschein, 2019)
- **Less precision than a human**
  Jessop-Fabre & Sonnenschein (2019) write that counterintuitively, robots may be slower and less accurate than humans under certain conditions and mentions the work of Zielinski et al. (2014) who reported that their developed pipetting protocol had up to a 3x larger coefficient of variation and took twice as long to perform when carried out by a robotic liquid handler rather than by humans. This difference in liquid-handling precision is something that we also confirm and measured when we compared accuracy of a serial dilution between a human and our automated dilution protocol (see Chapter 4).

4. Biofoundries

In this section we introduce Biofoundries, what they are and what is their purpose.

The natural continuation of automating mechanised and computerised laboratory processes is to integrate both components together. This integration leads to the creation of biofoundries. Like any other molecular biology laboratories, Biofoundries require traditional equipment however additionally Biofoundries are equipped with high-end automated hardware suitable for high-throughput experiments.

Tellechea-Luzardo et al. (2022) define biofoundries as specialised laboratories that combine software-based design and automated or semi- automated pipelines to build and test genetic devices. A biofoundry offers an integrated infrastructure to enable the rapid design, construction and testing of genetically modified organisms for biotechnology applications and research (Hillson et al., 2019). In addition to the integration of all the robotic liquid-handling equipment and high-throughput analytical equipment biofoundries provide software personnel and data management systems (Holowko et al., 2021).

Biofoundries have two sets of purposes. On a technical level, Biofoundries bring accessibility to automation and high-throughput equipment, computer-aided design (CAD) software, new workflows and a variety of tools and thus allow the acceleration of academic and translational research in engineering/synthetic biology (Hillson et al., 2019). Biofoundries deliver capabilities and infrastructure that allow for an accelerated approach to synthetic

biology research and applications (Holowko et al., 2021). On a wider economic level, Biofoundries enable the development of bioengineered products and organisms relevant to the bioeconomy. They accelerate commercialisation of synthetic biology products and biomanufacturing process engineering (Hillson et al., 2019). Biofoundries play an important role for industrial translation. Public-funded biofoundries bridge academia and industry. Biofoundries help create academic research spin-outs and the creation of start-up companies by providing access to expensive infrastructure (Farzaneh & Freemont, 2021). The interest and activity of biofoundries have been growing worldwide (Holowko et al., 2021). Biofoundries support the strategy for countries to develop their bioeconomy. Over 50 countries (Kitney et al., 2019; Holowko et al., 2021), including all the G7 nations, have adopted national strategies related to the bioeconomy.

Biofoundries are facilities with a high level of automation. Automation can be from the equipment itself like the liquid-handling robots or programmable devices or devices like grippers and robotic arms that can move plates from one piece of equipment to another. The cost of DBTL cycles has decreased in time and labour by automating time-consuming operations (Chao et al., 2017). This changes the paradigm of research. Conventional biology engineering approaches favour heuristics because the cost of each is turnover is high but as the costs drop down, more exhaustive approaches can be adopted like designing many variants and generating and testing them all at higher throughput with the help of automation.

5. Strategy Overview

5.1 Goal and structure of the thesis

The goal of the work presented in the thesis is to develop a framework and software to support workflow automation in a Biofoundry. The thesis comprises two distinct parts: In the first part, an exemplar metabolic engineering (ME) application (Lycopene case study an application for the production of a metabolite: lycopene) is presented. The case study aims to identify the problems that can arise in a metabolic engineering application. The second part utilises lessons drawn from the case-study and presents an infrastructure framework to support automation in a Biofoundry.

For the exemplar application, we decided to use an application from metabolic engineering.

ME is relevant because it is an industrial application. ME applications typically require to navigate large design spaces therefore automation is needed. Combinatorial pathway engineering applications are engineering based (multifactorial design) which is the approach we focus on. Additionally, the bioproduction of compounds relies on host microorganisms

expressing metabolic pathways of the target compound from single-cell hosts to large-scale fermenters (Otero-Muras & Carbonell, 2021) but we do not factor in scale-up.

The goal of the application is the optimisation of production of a metabolite, lycopene, in bacterial cells. Lycopene is a naturally produced bright red pigment, a carotenoid present in many plants and organisms, with high bioeconomic interest.

The problem will be tackled with an engineering methodology called Design-Build-Test-Learn (DBTL) in engineering biology, which consists of iterative engineering cycles (we present the DBTL cycle in more detail in Chapter 2 - Section 4.1). The combinatorial metabolic engineering lycopene application comes with its own set of challenges – first of all the exploration of large combinatorial spaces. The exploration of large spaces was optimised with methods such as Design of Experiments (DoE) that enable efficient, systematic exploration and exploitation of complex design spaces. The screening of large combinatorial libraries requires high throughput methods which can only be achieved in a reasonable time-scale by automation.

Automation comes with its own set of practical mechanisation problems in the wetlab (Albert, 2007). Additionally, high throughput methods generate large measurement datasets which in turn creates another problem of dealing with large quantities of data. Information organisation problems can be solved with computational automation (Peccoud et al., 2022).

Through the development of the lycopene case-study we learned lessons: the necessity to design a scoping study, the necessity to care for errors induced by automated methods. From these lessons we learned a set of teachings and came up with some guidelines. From the needs to overcome the constraints of our class of application we designed a solution, a framework and a system to manage laboratory automation both mechanically and computationally in a Biofoundry.

5.2 Application

In Chapter 2 we introduce the notion of metabolic engineering and present an overview of the different strategies to combinatorial engineering. We discuss the problems introduced by large combinatorial pathway approaches: the search space is very large, a lot of data gets generated and it requires many rounds of experiments. The advent of high-throughput automation and modular DNA assembly techniques have made the combinatorial pathway approach feasible and engineering methods such as DoE help reduce the number of experiments to run to converge to an optimal solution in a reduced amount of time.
We present an engineering workflow to investigate combinatorial metabolic pathways and illustrate this workflow with the chosen case study of an application for the production of a metabolite: lycopene, which has a strong industrial interest and is a model pathway that has a size that large enough but not to large to explore the combinatorial space.

The lycopene application presented in Chapter 2 is a good case-study because lycopene has a strong industrial interest. The search space is very large. The lycopene operon is a model pathway that has a size that is large enough but not too large to explore the combinatorial space. A combinatorial metabolic engineering application generates a lot of data and the application requires many rounds of experiments.

5.3 Initiation of the cycle of a design of experiments (DoE) application

Chapter 3 focuses on the initiation of the application cycle. Along with the development and implementation of the different steps of the lycopene application workflow (the application comprises a succesion of tasks which constitute the workflow, each iteration of the workflow constitutes one cycle of the DoE application, the DoE application consists of multiple cycles) we have identified some problems and developed various laboratory protocols and computational methods to address them.
After developing the lycopene application workflow presented in Chapter 2, we identified a first problem that would prevent the initiation of a DoE cycle. As we undertook a replication study previously conducted in the lab (Exley et al., 2019), we couldn't observe growth for some of the constructs. Consequently, the absence of growth would hinder the measurement of lycopene produced by the cells. We believed that toxicity and burden from lycopene and the construct are responsible for this absence or slowdown of growth. For this reason and to enable us to initiate the DoE cycle we designed a method that we call PASIV (Pooled Approach, Screening, Identification and Visualisation). Within PASIV we developed an algorithm and a software called cMatch to allow a fast identification of constructs after a DNA sequencing step.

5.4 Automated standard curve

Chapter 4 focuses on the construction of a calibration curve because the lycopene application requires the measurement of the concentration of the lycopene produced by the cell expressing the genetic constructs. In the application pipeline, lycopene is extracted from the cells with DMSO. An absorbance measurement of lycopene in DMSO is carried in a plate-reader and the lycopene concentration value is derived from a lycopene standard curve. The construction of a standard curve requires the controlled generation of a set of samples of known concentrations on measurement microplates – and for the process to be repeated over a series of assays – which is achieved by using automated liquid handling. An automated protocol to create a standard curve was developed. Automation offers many advantages: automation allows to implement any arbitrary dilution pattern over a plate in a programmatic way, automation reproduces the same mechanised liquid transfer movements when repeating the protocol, thus ensuring reproducibility of the curve, automation removes

mechanised human induced errors. Automatic liquid handling and the challenges that come with the automation of liquid handling are introduced. Many issues have been identified (in the literature) as obstacles encountered during automated liquid handling operations and result in errors. Two dilution schemes were implemented and studied: the geometric scheme and the linear scheme and present the different sources of error that might be induced by one scheme or another.

5.5 Automated workflows

In Chapter 5 the need for automated research workflows is introduced. The lycopene case study helped us identify problems which are solved by introducing automated workflows. We define tasks and workflows and what are protocols and establish the relation and equivalence between the concepts. Two examples from our case study application are used: BASIC DNA assembly and the Automated Dilutions for Standard Curves. We illustrate how we create, respectively, an automated computational workflow and an automated wetlab workflow.

We introduce our conceptual framework and the concept of digital twins. We show how we can have a digital representation of the physical world and that it is desirable to link the digital representation of a physical object and its observable/observed interactions with other physical objects. In this chapter we define our world and our objects and how we represent experiments in synthetic biology. We define the operational pipeline as the successive actions on physical objects and their observations.

5.6 The Biofoundry Information Manager

Experiments run in a biofoundry make use of various pieces of equipment and measurement instruments. Each experiment generates data and the volume of data can be large. All the data produced needs to be organised for further analysis. Additionally to measurement data, there is metadata generated of which the collection helps organising the data for an easier recollection later down the experimental pipeline. It is thus necessary to have an information system to collect and organise all the data and metadata generated during the experiments. We call this system the Biofoundry Information Manager (BIM).

In the chapter we introduce two data structures – directed graphs (DGs) and directed acyclic graphs (DAGs) – that allow us to model experiments and their execution as well as to structure the collection of their resulting data and metadata. We also introduce the model graph, a DG that links the dependencies of all the data being produced along the workflow. Workflows can be executed as runs and the result of these runs gets stored in a DAG that we call an execution graph. We discuss their technical implementation with computational

libraries and graph databases. Structuring the data in a DAG and storing it in a graph database allows us to record a history of the runs and query the graph database to find data provenance.In addition, this way of structuring experiment in workflows built on top of DGs has another advantage, by following the graph edges of the DG, it is possible to read the order of the task and schedule of the execution of the tasks that it comprises. This allows the DG to be used as a blueprint for the orchestration of tasks and automation of the execution of workflows in the biofoundry.

5.7 The Biofoundry Automation Manager

In Chapter 7 we describe how we use the structure of the graph, and, more particularly, the model graph*,* as a template for automating the execution of the workflow introduced in Chapter 6. We describe how an automated laboratory can be built when workflows are stored as model graphs and coupling the workflow to software called an orchestrator. Our solution to automated workflows is introduced: the Biofoundry Automation Manager (BAM). The BAM automatically manages workflows and their execution on a biofoundry's infrastructure. Different strategies for testing the BAM's functionality and operation are presented. Examples of workflow executions, using the Biofoundry Automation Manager (BAM) and their results are discussed. The execution of a generic workflow that contains the difficult case of repetitions is demonstrated. In addition, an example of a computational workflow of 'BASIC DNA Assembly' for *in silico* DNA assembly is presented – together with a wetlab workflow for an 'Automated Standard Curve', to automatically dilute a reagent to produce a standard curve.

Chapter 2

Application

In this chapter we present our case study: the production of a metabolite with important industrial applications – lycopene.

First we review the field of metabolic engineering and focus on a recent approach to metabolic engineering, combinatorial pathway engineering, which has been greatly influenced by the core principles of synthetic biology. Then we discuss the challenges encountered when dealing with large scale spaces – as combinatorial pathway engineering is prone to yielding – and the existing approaches to solving them before presenting how these solutions are implemented and how their corresponding workflows are automated or are automatable. Finally, we discuss the metabolite of interest, lycopene, and we show how its pathway is a good model pathway, because the combinatorial possibilities from a small set of genetic regulators, are large enough for exploration but not too large.

1. Metabolic Engineering

1.1 Introduction to metabolic engineering

Over the past decades metabolic engineering (ME) has enabled the development of microbial strains capable of efficiently producing useful chemicals and materials and also helping degrade chemicals and contaminants (Kim et al., 2023). Metabolic engineering in microorganisms has produced hundreds of chemicals and materials, including biofuels, bioplastics, industrially important bulk chemicals used as solvents, polymer precursors, adhesives, food additives, cosmetic ingredients, and pharmaceuticals (Kim et al., 2023).

A report by Market Research Future in 2019 estimated the global market size of bio-based chemicals to reach USD 97.2 billion by 2023 (Ko et al., 2020).

Metabolic engineering has historically been applied to biomanufacturing and has led to the industrial production and commercialisation of products, as an example, in 1997 the companies such as DuPont and Genencor applied metabolic engineering to convert glucose

to 1,3-propanediol (PDO), a building block in the production of polymers, and its production reached commercial scale in 2006 (Clarke & Kitney, 2020).

Historically the first approach to metabolic engineering has been rational metabolic engineering as opposed to an evolutionary approach. Combinatorial metabolic engineering is an extension to rational metabolic engineering.

Metabolic engineering (ME) is the deliberate modification of intermediary metabolism using recombinant DNA technology (Cameron & Tong, 1993). As stated by Volk et al. (2023) ME's primary objective is to produce functional molecules such as chemicals, fuels, materials and proteins, using cell factories (Volk et al., 2023) – with economically viable metabolite yields and production rates that are consistent and reproducible. ME seeks to achieve this by altering native metabolic pathways and/or introducing heterologous pathways (derived from a different organism (Wikipedia, 2021)). Combinatorial pathway engineering is a recent and significant approach in the field of metabolic engineering. Here, one or more pathway components are simultaneously modified – unlike the classic approach that focuses on identifying bottlenecks and optimising them in a step-by-step manner (Yadav et al., 2012; Xu et al., 2017).

ME begins with the selection of a target product which has a promising application, then the selection of a host strain (the cell factory) which can be engineered to produce the desired product at a high titer, rate and yield (TRY) (Volk et al., 2023). Then an iterative process of design, build, test and learn (DBTL) paradigm is used to design and build a native or heterologous biochemical pathways, which will convert a substrate to the target product. This is followed by the optimisation of the metabolic fluxes using systems biology or synthetic biology tools (Volk et al., 2023).

Several strategies exist to rewire the metabolic network of an organism to achieve the desired objectives that rely on a high degree of understanding of cell metabolism (principally its kinetics and regulation) (Liu & Stephanopoulos, 2021). Collectively, these approaches are known as 'rational metabolic engineering'. Historically, they have been developed and applied since the early days of metabolic engineering (Bailey, 1991; Stephanopoulos & Sinskey, 1993).

In recent years the development of methods and techniques – such as automated DNA synthesis methods, robotics and high-throughput screening methods – have made a new approach to create and optimise for better production phenotypes possible. This extends the rational approach and aims to navigate large spaces of genetic variants of a base strain (Liu & Stephanopoulos, 2021). This approach, called combinatorial metabolic engineering, relies less on the understanding of biochemistry and physiology, but instead employs systematic search processes (Liu & Stephanopoulos, 2021). We will discuss this approach in detail in

section 2.

1.2 Rational metabolic engineering

An understanding of the reaction (kinetics, thermodynamics, regulations of all the biological network) would allow the determination of the optimal enzymatic profile that will maximise the desired product. Ideally there is a complete metabolic model that captures all the characteristics and interactions in the cell – and the optimal genetic modifications are determined mathematically. Unfortunately, such complete understanding of the enzymatic network is rarely available and in practice researchers rely on experiments to provide the missing information (Liu & Stephanopoulos, 2021). As defined by Liu & Stephanopoulos (2021) a generalised workflow of rational metabolic engineering experiments comprises three steps: 1. design the biochemical pathway to convert the substrate to the final product 2. source the genes that encode the enzymes of the pathway 3. introduce the pathway into the host organism using molecular biology tools and optimise gene expression and the enzyme kinetics.

Various strategies for metabolic engineering have emerged over the last 30 years. Volk et al., (2023) provide the most recent review of the various strategies implemented based on rational design – such as push-pull block, cofactor engineering, spatial engineering and transporter engineering definition.

The Push-pull block strategy, which as the name suggests consists of: 1. pushing, driving an increase of flux toward pathway precursors via up regulation of enzymes. 2. pulling, drawing higher flux along the biochemical pathway by overexpression of terminal enzymes; and 3. blocking, removing competing pathways consuming pathway intermediates

The cofactor engineering strategy consists of providing the adequate supply of cofactors. Cofactors are organic or inorganic molecules that assist the enzymes as they bind the proteins to help them carry out their catalytic functions. When biosynthetic pathways are constructed, the heterologous enzymes of the pathways might not function properly due to the absence of certain cofactors. Cofactor strategies include building a cofactor regeneration system, enhancing cofactor levels by regulating the host machinery or even modifying the enzyme's cofactor preference (Volk et al., 2023)

The spatial engineering strategy is an emerging strategy to spatially organise the location of the enzymes of the biosynthetic pathways inside the cell. The organisation of the pathway can boost the enzyme local concentration, remove diffusion limitations, limit the accumulation of toxic intermediates and bypass competing enzymes and inhibitory regulation networks (Volk et al., 2023). Volk et al. (2023) identify three categories of spatial engineering: fusion proteins, synthetic scaffolds and compartmentalisation.

Transporter engineering is a strategy that optimises the transportation of molecules inside, through and outside of the cell as the process of transportation is involved in all aspects of the metabolic pathway (Volk et al., 2023). The process of transportation starts with the intake of the substrate, the intermediate transportation to the export of the final product.

1.3 Evolutionary approaches

Evolution methods are close to combinatorial metabolic engineering. These techniques borrow from natural evolution. The goal of related strategies is to streamline the library construction and screening processes by applying adverse selection pressures to gradually enrich a desirable subpopulation of the library (Liu & Stephanopoulos, 2021).

One example of such an evolutionary method is directed evolution which goal is to generate new enzymes that have higher catalytic efficiency or that can catalyse new chemical reactions not found in nature (Liu & Stephanopoulos, 2021). A review on the topic can be found in (Arnold, 2018)] and (Wang et al., 2021). Such a technique can also be applied for strain optimisation; the selection pressure is designed in a way that makes the variants with the desired characteristics outgrow other strains that do not have such characteristics (Liu & Stephanopoulos, 2021) and a review of gene diversification strategies for directed evolution can be found in (Wang et al., 2021) and (Vidal et al., 2023).

The next step after the library has been created is the screening process to test the variants individually. (Liu & Stephanopoulos, 2021) emphasise the importance of choosing the correct characteristic to benchmark each variant and call it the most important factor as it not only reflects the desired phenotype but also enables a rapid and accurate screening (Liu & Stephanopoulos, 2021). To shorten run times per sample, screening processes make use of spectrophotometry or fluorescent techniques such as multiwell plate assays or flow cytometry (Dietrich, McKee & Keasling, 2010; Liu & Stephanopoulos, 2021).

In practice, combinatorial metabolic engineering can be applied to an organism as a whole or to a specific enzyme. The approach starts with the rapid generation of a library of a large number of variants of the targeted genetic parts (Liu & Stephanopoulos, 2021).

There are different methods to build libraries: by random or semi random mutagenesis techniques such as UV irradiation, chemical mutagens and error-prone PCR (Adrio & Demain, 2006). The mutants in the created libraries contain a wide spectrum of various characteristics which makes them better than those of the wild-type variant (Liu & Stephanopoulos, 2021). Liu & Stephanopoulos, (2021) identify two challenges to library construction: library coverage of the design space in order not to miss potential hits and the size problem of generated libraries that are too large – when only a small subpopulation shows meaningful changes. Thus one approach that Liu & Stephanopoulos, (2021) present is to only work within a predetermined genetic space, for example knowing that an enzyme is

rate limiting, (it is only necessary to design strains with variants of that enzyme only).

The advent of DNA synthesis and assembly techniques has made it easy to construct mutant libraries that might contain an improved mutant.

2. Combinatorial Pathway Engineering

2.1 Combinatorial metabolic engineering

The rational approach to metabolic engineering (presented in the previous section) requires a detailed knowledge of the pathway – including the enzymes catalysing its reactions (Liu & Stephanopoulos, 2021) – and its interaction with the host organism. Such a heavy constraint – Jeschek, Gerngross & Panke, (2017) called it impractical – has led researchers to develop combinatorial approaches where a large number of the permutations are introduced within the cell's genetic material in order to affect the metabolic flow in the pathway. A systematic search within the space of genetic changes is conducted to optimise the production of the metabolite(s) of interest. The main advantage of such an approach is to reduce the need for understanding the biochemical factors that affect the pathway (Liu & Stephanopoulos, 2021).

Jeschek, Gerngross & Panke, (2017) present our ‘engineering’ description of the approach. They define the term 'combinatorial' as a modification of more than one variable of a system at the same time. In the context of pathway engineering, this means diversifying two or more genetically encoded components of the pathway. The parts that can vary comprise regulatory elements (promoters, ribosome binding sites (RBS), terminators), coding sequences (CDS) and chassis (the host organism of the new heterologous pathway). Variation can happen at the concentration level of the components (mRNA levels) or on the parts themselves (homologue genes). As a consequence of these modifications, the pathway becomes unbalanced. This leads to many obstacles – such as accumulation of intermediates which can be toxic, side product formation and metabolic burden which can lead to growth inhibition and a very low end-product yield (Jeschek, Gerngross & Panke, 2017). The challenge that metabolic engineers face then becomes one of improving the metabolic flux of the target pathway and to create microbial cell factories for industrial applications (Jeschek, Gerngross & Panke, 2017).

Thanks to Panke’s abstraction, it is relatively straightforward to make an analogy between metabolic engineering and sound engineering. In the same way a sound engineer controls the various parameters on a mixing table, the metabolic engineer adjusts the parameters for the genetic regulation of a pathway, which results in a controlled metabolic flux (e.g. a enzymatic cascade) (as illustrated in Figure 2-1). The bottleneck-lifting approach is equivalent to using one part of the board – and keeping it fixed for further adjustments,

whereas the full combinatorial approach is equivalent ot using the whole board to achieve a global solution.

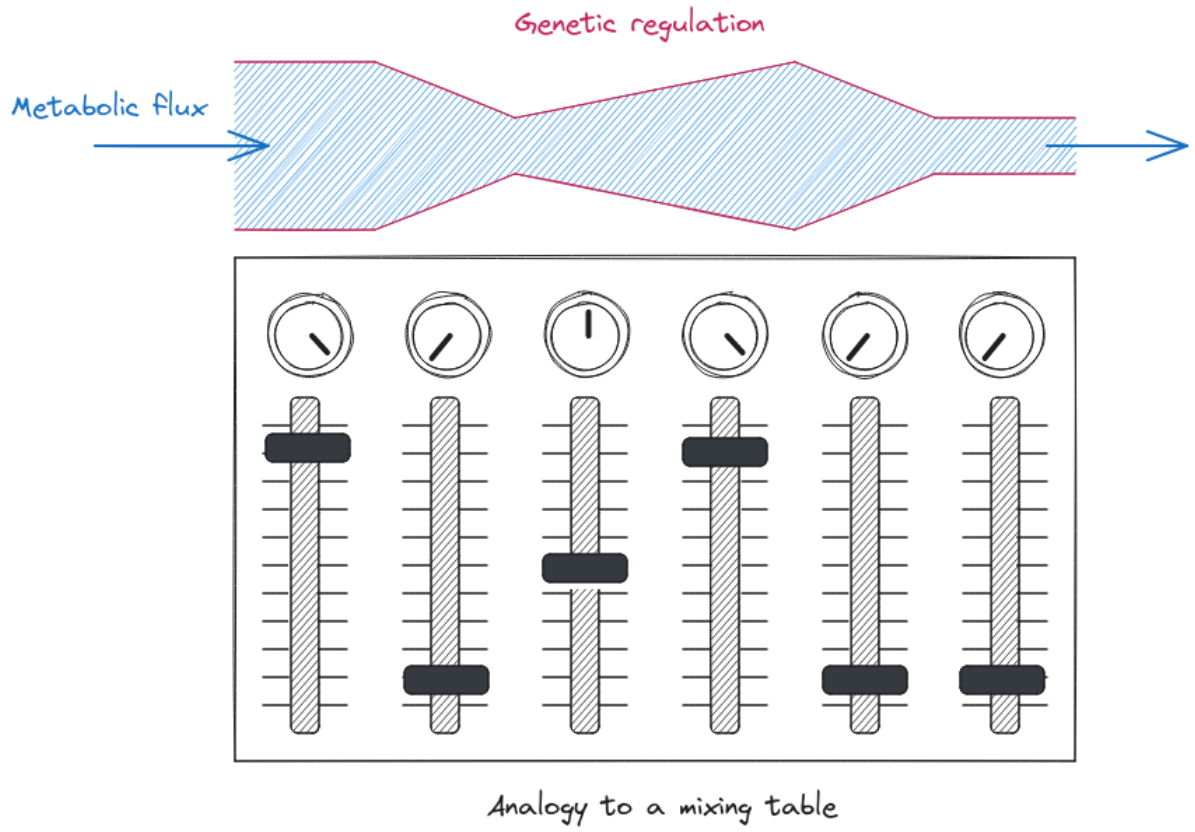


Figure 2-1 – Analogy to a sound mixing table. The faders of the mixing table control the genetic regulation of the metabolic flux of the pathway.

2.2 Strategies

Several strategies to introduce genetic diversity have been successfully developed. The most common strategies will now be reviewed. We focus on strategies that apply to metabolic engineering in bacteria as all the work presented in this thesis and done in the laboratory has been undertaken in a bacterial chassis (*E. coli*).
These strategies can be split into two families – methods that alter the function/activity of the enzymes and methods that alter their levels in the cells.

2.2.1 Engineering variation in the coding sequence (CDS)

There are two main strategies for varying the coding sequence (CDS) (Jeschek, Gerngross & Panke, 2017). In the first strategy different gene homologues can be used. Their corresponding enzymes catalyse the respective reaction steps. Such a strategy has been, for example, recently applied by Babaei et al. (2020) to produce Rosmarinic Acid in

Saccharomyces cerevisiae. This has been achieved by engineering multiple biosynthetic pathway variants, carrying only plant genes or a combination of plant and *Escherichia coli* genes. The second strategy uses random or targeted mutagenesis to diversify the CDS. As an example, such a strategy has been employed by Lv et al. (2016) to engineer the isoprene pathway by directed co-evolution of the genes dxs/dxr/idi to generate a mutant library. After using error-prone PCR to generate the mutant library, Lv et al. (2016) quantified the production of isoprene and lycopene through high-throughput screening methods. The problem with such an approach is when the mutagenesis not only targets the CDS, but also, affects the genetic regulatory elements (promoters, RBS). The observed improvements of the pathway may be attributed to some changes in expression levels rather than to the functional amino acid substitutions (Jeschek, Gerngross & Panke, 2017).

2.2.2 Engineering variation at the gene expression level

Balancing the pathway flux to get the desired product requires an optimisation of absolute and relative expression levels of the genes of the pathway (Jeschek, Gerngross & Panke, 2017). Fine tuning of the expression level can be achieved in several ways, including by controlling gene dosage, translation levels or transcription levels.

(i) Gene dosage

Gene dosage can be achieved by altering the number of copies of the enzymes per cell. This can be achieved by employing gene systems that range from single-copy (when integrated in the genome or using artificial chromosomes (Sjöberg et al., 2019) or using multiple copies of plasmids that have higher copy numbers. More recently Joshi, Yong & Gyorgy (2022) have introduced TULIP (TUnable Ligand Inducible Plasmid), a plasmid system in *E. coli* with inducible copy number control. Combining plasmids with different copy numbers can balance the expression of different pathway modules (each module on one plasmid) (Wong et al., 2021).

(ii) Transcription levels

Fine tuning gene transcription level is being done by engineering promoters to increase the transcription efficiency or by introducing hairpins and RNase sites into the mRNA enhancing the RNA stability (Alper et al., 2005; Blazeck & Alper, 2013; Xu, Wei & Liu, 2019; Cazier & Blazeck, 2021). Various promoter libraries have been created that span several orders of magnitude in expression levels (Hammer, Mijakovic & Jensen, 2006).

Computational tools exist to engineer such libraries (Fleur, Hossain & Salis, 2021) and have been assessed in comparative studies (Cassiano & Silva-Rocha, 2020; Zhang et al., 2022). More recently, machine learning tools have been employed to generate large promoter libraries in prokaryotic hosts (Wang et al., 2020; Zhao et al., 2022).

Promoter engineering expression of transcription can be expressed conditionally (by using an inducible promoter) or constitutively. Inducible systems require the expression of additional activator or repressor proteins which adds an extra burden on the host cell.

This burden adds up, in the case of combinatorial optimisation where multiple induction systems are required for the various pathway elements expression.

The use of constitutive promoters reduces any additional complexity, but can lead to growth inhibition. There is no separation between the phase of growth of the biomass and the production phase which is a feature required in industrial processes (Rugbjerg & Sommer, 2019).

(iii) Translation level

In prokaryotic hosts engineering ribosome binding sites (RBS) is another way of tuning the expression levels. The RBS in prokaryotes is a region upstream of the start codon. This region of the mRNA has the 5'-AGGAGG-3', also called the Shine-Dalgarno (SD) sequence. (Wikipedia, 2023). The change of a few bases in the sequence of an RBS (which are usually short sequences) spans a wide range of expression levels (Zhang et al., 2021). Predictive biophysical models can be employed to engineer RBS (Tian & Salis, 2015; Reis & Salis, 2020) and large RBS libraries that have been designed in silico are available (Jervis et al., 2018).

3. Combinatorial Pathway Engineering - Challenges and Solutions

Liu & Stephanopoulos (2021) identified the drawbacks of the strategies presented in the previous section, which are that the genetic search space can be quite large. This demands an effective high-throughput screening method to complete within a reasonable time to which we would add the problem of construction (Jeschek, Gerngross & Panke, 2017). These problems will be encountered in our lycopene case-study.

3.1 Large design spaces

The following definitions are used in the rest of the chapter:

- Design parameters: genetic elements that are varied
- Design space (or set of possibilities): all the possible combinations of values of design parameters on the general design constraints. As an example of a design constraint, the template of a transcription unit imposes an order of the various types of genetic parts: first the promoter followed eventually by an genetic insulator such as a RiboJ, followed by an RNA-coding sequence and ending with a terminator.

The various strategies mentioned in the previous section can be combined together e.g. in the work of Smanski et al. (2014). The authors have designed a heterologous combination of genes for nitrogen fixation in *E. coli* and assembled a 16-gene constraint-based design where they vary promoters, RBS, terminators and the order of genes (Smanski et al., 2014).

However, introducing diversity on multiple levels leads to very large design spaces and possibilities – a process known as 'combinatorial explosion'. For example, consider the violacein pathway (5 genes) (Hoshino, 2011) and an operon design – yielding 10^5 possible combinations for even a small library of 10 RBS. Varying more components in the design (e.g. the promoter driving the operon or adding degradation tags to the enzyme to control their expression) expands the design space by further orders of magnitude and easily reaches millions of potential combinations.

When optimising a metabolic pathway with a combinatorial approach, one of the main issues is scale and how to deal with large design spaces. For time and cost reasons, huge design spaces and possibilities cannot be exhaustively screened (brute-force) by any available experimental method. Consequently, smart strategies for search space reduction need to be applied to keep the screening effort manageable.

3.2 DNA assembly

The first challenge when navigating large spaces of possible designs is construction.

Approaches to pathway optimisation employing combinatorial pathway engineering methods have become feasible due significantly reduced cost of commercial DNA synthesis and reliable in vitro an in vivo DNA assembly techniques (Jeschek, Gerngross & Panke, 2017)

Recombinant DNA technology started in the 1970s (Zimmerman et al., 1967; Cohen et al., 1973; Lobban & Kaiser, 1973) and has enabled genetic engineering and the biotechnological revolution (Kelwick et al., 2014). Various reviews on the different DNA assembly methods have been published over the past decade (Ellis, Adie & Baldwin, 2011; Chao, Yuan & Zhao, 2014; Casini et al., 2015). Kelwick et al., (2014) reviewed development of tools for synthetic biology and highlight recombinant DNA technologies in which DNA sequences are cut and pasted together via restriction enzymes and DNA ligases. Kelwick et al., (2014) categorise the techniques according to the assembly strategies listed by Chao, Yuan & Zhao, (2014): restriction enzymes based, overlap-directed, recombination-based and direct DNA synthesis.

Methods such as Gibson assembly which is based on exonuclease digestion ligation (Gibson et al., 2009), assembly PCR (Polymerase Chain Assembly (PCA) (TerMaat et al., 2009)) or in vivo assembly based on the host DNA repair and homologous recombination abilities (Watson & García-Nafría, 2019) are popular but their drawback is that they only accept linearised DNA parts as substrates and therefore depend on time consuming DNA

purification steps (Bird, Marles-Wright & Giachino, 2022). By contrast, Golden Gate (GG) – a restriction enzyme-mediated assembly using Type IIS (shifted) endonucleases – has become quite popular in the synthetic biology community (Bird, Marles-Wright & Giachino, 2022).

Golden Gate is a method that can combine a large number of DNA parts in a one-pot assembly reaction. DNA assembled with the Golden Gate method can be transformed directly into the recipient strain. Golden Gate accepts both linear and circular DNA – thus enabling the creation of standardised libraries of ready-to-assemble parts in storage plasmids. This makes libraries of GG compatible parts on plasmids vectors easy to propagate, purify and distribute – as opposed to an assembly method, such as Gibson Assembly, which requires the need to linearise and purify individual DNA parts before the assembly (Bird, Marles-Wright & Giachino, 2022).

A popular GG-based DNA parts library/toolkit is CIDAR MoClo (Iverson et al., 2016) that comes with a collection of standardised vectors and parts. (Bird, Marles-Wright & Giachino, 2022) have written an extensive review focusing exclusively on Golden-Gate-based assembly methods.

The biobricks standard assembly RFC10 (Røkke et al., 2014) has been popularised by the iGEM competition for many years, but as surveyed by Kahl & Endy (2013), since the beginning of the competition in 2003, teams slowly moved from Biobricks assembly to Gibson assembly and using DNA de novo synthesis. More recently (2021) iGEM has started recommending the iGEM Type IIS as specified by RCF 1000 (Golden Gate) and the iGEM Distribution Kits ((iGEM Engineering Committee, 2023, 2023)) are now given out as Level 0 parts compatible with MoClo and Loop TypeIIS. Indeed the biobrick assembly is an inefficient way to create large multi-part constructs as it is limited to assemble two bioparts per reaction (Kelwick et al., 2014).

Methods that leave scars may alter the behaviour of flanking bioparts or can prevent the creation of fusion proteins and can be undesirable (Anderson et al., 2010; Ellis, Adie & Baldwin, 2011; Kelwick et al., 2014). However there exist scarless methods such as Scarless Stitching (Smanski et al., 2014) and Start Stop (Taylor, Mordaka & Heap, 2019).

Young et al. (2021) have written a review with the focus on DNA assembly methods that can be used for combinatorial approaches. Combinatorial DNA assembly allows the optimisation of pathways in absence of predictive modelling in silico (Young et al., 2021). DNA parts can be combined and tested by randomly generating assemblies from large pooled libraries. Our approach is to use a controlled approach such as Design of Experiments (DoE) techniques to sample efficiently in the design space to converge to a combination of genes and regulatory elements that maximises the yield of a bioproduct (Young et al., 2021).

Modular DNA assembly methods are convenient for combinatorial pathway engineering, as it allows working with a set of collection of DNA parts. There also exist DNA assembly methods that allow (or can be adapted for) one-pot assembly (Casini et al., 2014, Pryor et al., 2020), some coupled with microfluidic device (Perry 2021); (creating all the constructs in one reaction) and fast track the multiple constructs assembly process.

3.3 High-throughput automation

High-throughput is the feature most associated with automation. Bogue (2012) argues that the history of lab automation in biotechnology parallels the development of modern drug discovery and high throughput screening. The development of drugs routinely involves testing an ever larger number of compounds (hundred of thousands being common) against target diseases. This can only be achieved within a reasonable timeframe by first miniaturising the assays, and then automating them.

Solutions like DoE (discussed in the next section 3.4) methods require testing 3% of a large design space (as a rule of thumb, as mentioned by Lee et al. (2013) and Gilman (2021)). Even what can be considered a small design space can lead to quite a large number of assays. The constructs (or points) required to explore the design space need to be built. Thus employing automating techniques to assemble DNA increases the throughput. Once the constructs have been built we need to measure and screen their expression, which requires an additional huge number of operations in the measurement pipeline.

Additionally, if repeats are undertaken to control the variance of the constructs expression, this hugely increases the number of data points. Thus automation can be a solution to achieve higher throughputs to conduct these assays.

3.4 Learning methods - DoE and machine learning

The set of methodologies known as Design of Experiments (DoE) (Gilman, 2021) have become popular to deal with the design and highly-complex multifactorial optimisation of large-scale combinatorial spaces. DoE methods are very amenable to automation, as works by Carbonell et al. (2018) and Rajakumar et al. (2019) have demonstrated. These methods have been adapted to identify the media and culture conditions that maximise yield of the metabolic pathway (Singleton et al., 2019; Azubuike et al., 2020), or the optimisation of a cell-free system (Spice et al., 2020). In the context of metabolic pathways, these methodologies have been applied to choose regulatory elements such as promoters (Blazeck et al., 2012) and 5′UTR (Salis, Mirsky & Voigt, 2009), or to modify other popular dosage parameters such as plasmid copy number (Ajikumar et al., 2010) and codon usage (Hanson & Coller, 2018; Wu et al., 2019). Xu et al. (2017) showed how the five gene violacein pathway could be simplified to three operons as two of then encoded enzymes had

no impact on the output. DoE methods can be unsuited to the estimation of parameters. Optimal Experimental Design (OED) can be used when a model is available for the pathway. The goal then becomes to design a set of experiments so the model parameters may be estimated more reliably (Smucker, Krzywinski & Altman, 2018; Abt et al., 2018). OED typically aims to maximise the information content (computed from the Fisher Information Matrix or with some Bayesian approaches) of the new experimental iteration (Nishii, 1993). Despite the proven record in reducing the number of experiments and improving reliability, both sets of methods still require sampling at least a few percent of the design space (Lee et al., 2013) – potentially a very large number.

A variety of software (as mentioned in Gilman (2021) review on Statistical Design of Experiments for Synthetic Biology) can be used for this step. The authors mention commercially available software such as software include JMP software (SAS, 2019) or Minitab (Minitab, 2023), Design-Expert (Design-Expert | Stat-Ease, 2023). There is a plethora of open-source DoE python packages available (GitHub topics design-of-experiments, 2023) that could be used in our pipeline.

A more recent computational approach, machine learning (ML), has gained popularity in synthetic biology (Kim et al., 2020). Machine learning is a set of methods and techniques that can predict the effect of genotype changes on a phenotype, or extrapolate the large-scale behaviour from small-scale experiments, without a detailed mechanistic understanding (Carbonell, Radivojevic & García Martín, 2019). Machine learning has been used to improve genetic regulatory components such as promoters (Fleur, Hossain & Salis, 2021), or for the combinatorial assembly and design of enzymes (Lipsh-Sokolik et al., 2023). Machine learning techniques have also been used in synthetic biology tools such as the Automated Recommendation Tool (ART). This has been used to explore design spaces and provide recommendations on the next engineering cycle and have demonstrated its application on three real metabolic engineering cases: limonene in *E. coli* for biofuel production, linalool and geraniol production in yeast as flavouring molecules during the brewing process and, with less successful results, for dodecanol, a fatty-acid used in detergents, emulsifiers, lubricants, and cosmetics, production in *E. coli* (Radivojević et al., 2020).

Kitano et al. (2023) state that improving biological components is relatively easy to achieve providing that the dataset to train ML algorithms is large enough. The real challenge to advance synthetic biology is to facilitate the prediction of biological design with the desired characteristics, by predicting the associations between phenotypes and the combinations of genetic parts and genotypes.

4. Application Workflow

In this section we introduce a product development workflow based on an engineering paradigm called Design-Build-Test-Learn (DBTL). The DBTL paradigm is a product development workflow that can be applied to the development of any application. This section focuses on combinatorial pathway metabolic engineering applications, and presents principles for its automation. The resulting workflow was applied to our case-study, aiming to optimise lycopene production (and further discussed in Section 5).

4.1 The design-build-test-learn (DBTL) cycle

Academic and industrial biofoundries are increasingly adopting an engineering approach based on the iterative application of the Design-Build-Test-Learn (DBTL) cycle that has long been a central element of product development in traditional engineering disciplines (Carbonell et al., 2018). The DBTL cycle is illustrated in Figure 2-2.

An automated pipeline for engineering biology comprises metabolic pathway design tools (Design); automated DNA assembly and strain engineering (Build); high-throughput product quantification (Test); and data analysis and redesign through methods such as Design of Experiments (DoE) (Gilman, 2021) and machine learning (Learn) (Carbonell, Radivojevic & García Martín, 2019).

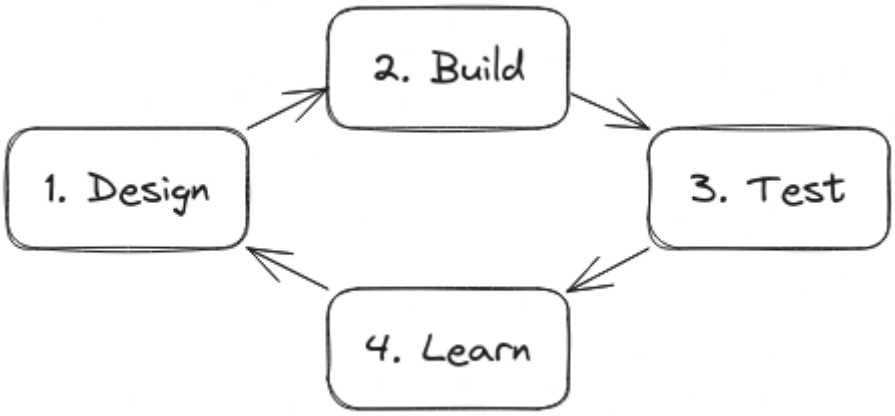


Figure 2-2 – The Design-Build-Test-Learn engineering cycle.

The following steps comprise the automated DBTL cycle as described in (Carbonell, 2019):

4.2 Design

The Design step starts with a target compound of interest. Various tools and algorithms

(Long, Ong & Reed, 2015; Fernández-Castané et al., 2014; Delépine et al., 2018) exist to explore known and new biochemical routes connecting the host organism to the target compound.

The encoding DNA sequences of enzymes that catalyse the reactions in the pathway are identified and combined with regulatory genetic elements. Combinatorial libraries are selected to explore the relationships between the design factors.

The first step of the workflow produces the design of multiple combinatorial constructs. First, the design starts with an initial generic construct design (i.e. the synthetic pathway). The constructs derive from a constraint-based construct design template and a list of positional genetic elements for every position in the construct. The various genetic elements (promoter, RBS, CDS, terminator) come from a repository of genetic parts, such as the iGEM parts repository (parts.igem.org, n.d.) or Synbiohub (McLaughlin et al., 2018). Then DNA assembly software takes the construct design template and the list of positional elements and outputs all the different combinations of the elements and returns the design of all the possible constructs. See Section 5.4 for our implementation of this generic step for our case-study.

4.3 Build

The Build step makes use of various automated platforms for the genetic cloning process: various selected DNA parts are assembled and inserted into a plasmid vector which is transformed into a host organism for replication, selection and screening. The different DNA parts are combined through one of the different DNA assembly methods (Young et al., 2021) like Golden Gate and Gibson (see Section 3.2). The resulting recombinant DNA is then inserted into the host organism (called the chassis). This process is known as transformation for bacterial hosts such as *E. coli*. The bacterial cultures are grown in selected media conditions and if the pathway contains inducible elements, the inducing molecules are added into the media at selected induction times.

Cell-free systems are an alternative to using a chassis (Laohakunakorn et al., 2020; Garenne et al., 2021) where the functional plasmid gets expressed in a mix obtained from a host cell lysate (Wagner, Jules & Borkowski, 2023) or from recombined from proteins such as the PURE system (Matsubayashi & Ueda, 2014; Lavickova & Maerkl, 2019).

The genetic construct can be assembled using various DNA assembly techniques. The BASIC assembly method (Storch et al., 2015) was used in the project; which is amenable to automation (Storch, Haines & Baldwin, 2020). The BASIC DNA assembly method requires the creation of specific genetic linker parts that can be designed with the help of software (Storch et al., 2015). Some in-house scripts have been created additionally which then evolved into the ‘Basicsynbio’ software created by Haines (2021).

Before moving on to the next step in the workflow, which consists of testing the constructs in a chassis, it is desirable to add an intermediary step of quality control (QC) to verify that the assembled constructs are correct. For this step a combination of laboratory work steps (cloning and growth) are required, followed by a sequencing step. The QC step is done with a software developed in the project, called cMatch, which is presented in further details in the next chapter (Chapter 3).

4.4 Test

The Test step is where samples are prepared and processed for high-throughput automated screening. Various technologies enable large-scale quantification of the cell response, such as proteomics to determine protein concentrations, transcriptomics to quantify expression levels and metabolomics to quantify metabolite concentrations (Amer & Baidoo, 2021).

The Test step comprises transforming and growing the various constructs. This is followed by an extraction process and a measurement assay, (to quantify the molecule of interest), which will be described in detail in Chapter 4.

4.5 Learn

The Learn step is where the quantified experimental data is analysed to identify the effects and relationships between the designs and the resulting responses. Predictive models are developed to infer design rules for the next iteration of the cycle (Carbonell, 2019). The measurements acquired through a first iteration of the pipeline drive the subsequent iterations.

Large combinatorial design spaces remain challenging. The DBTL cycle engineering approach can help overcome such a challenge. As an example, the number of candidate enzymes and genetic regulatory elements in a typical metabolic pathway lead to a combinatorial explosion of possible designs. Even with a streamlined pipeline and automated robotic platforms, the full design-space cannot be explored. Such a bottleneck can be lifted by using the data acquired through the first iteration of the pipeline to drive the next iterations.

Classical metabolic engineering relies on the iterative process of knowledge-guided DBTL and although this approach is widely used it is time-consuming and labour-intensive (Volk et al., 2023).

Once we reach the last steps, depending on the results of the testing step (Step 4) we can take a decision on whether to go back to optimise the initial combinatorial design or go for another round of construction. DoE software such as those mentioned in Gilman et al.’s

review on statistical methods for experiments in synthetic biology (Gilman, 2021) can be used for that step.

5. Case Study: Optimisation of Lycopene Production

For the development of the application, we have used a popular pathway in metabolic engineering: the lycopene pathway. We used an operon pattern to encode the lycopene synthetic pathway and varied the regulation of its genes. The DNA assembly technique that we used to build the operon is the BASIC DNA assembly technique which is amenable to automation.

5.1 Lycopene

Lycopene is a naturally produced bright red pigment. It is a carotenoid present in many plants and organisms (Hernández-Almanza et al., 2016) and is of high bioeconomic interest for the pharmaceutical (thanks to its antioxidant properties) and cosmetic and food industries (as a colouring agent) (Ciriminna et al., 2016; Wang et al., 2019).

The demand in the industry for carotenoids such as lycopene in medical and pharmaceutical applications keeps increasing (Kim et al., 2019; Jung et al., 2016; Yen, Palanisamy & Su, 2019) but its chemical synthesis is limited by high cost, low yield and quality [(Sevgili & Erkmen, 2019). Microbial lycopene production is an alternative promising strategy and has been developed in various hosts such as yeast (Ciriminna et al., 2016; Yamano et al., 1994) *Bacillus subtilis* (Wu et al., 2017), *Pichia pastoris* (Zhang et al., 2023) or *Yarrowia lipolytica* (Schwartz et al., 2017) and *E. coli* (Kim et al., 2019; Gallego-Jara et al., 2015; Kim et al., 2011; Yoon et al., 2007).

5.2 The lycopene synthetic pathway

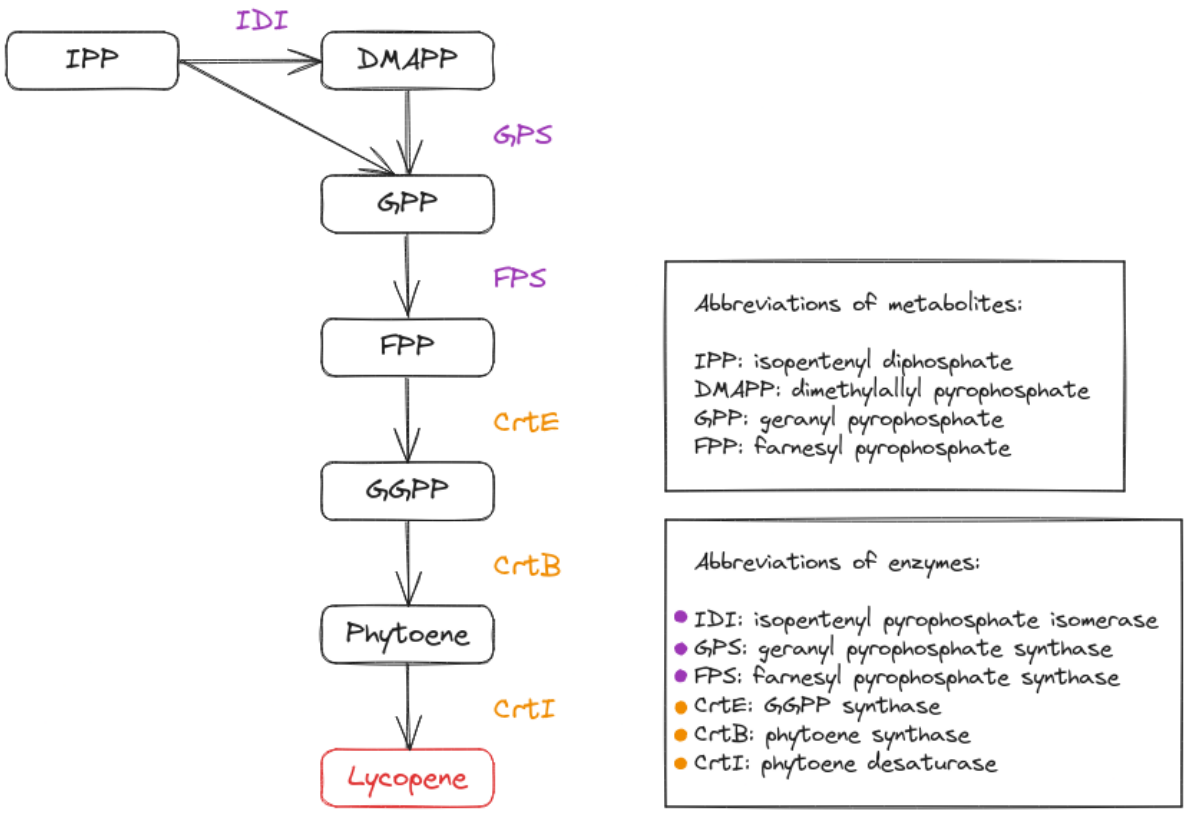


Figure 2-3 – The lycopene synthetic pathway. Condensation of IPP and DMAPP to generate geranyl diphosphate (GPP) is catalyzed by diphosphate synthase (GPS). GPP is condensed by one molecule of IPP by action of FPP synthase (FPS) to form farnesyl diphosphate (FPP). Then, FPP and IPP are condensed by the catalytic action of GGPP synthase (CrtE) to form geranylgeranyl diphosphate (GGPP). Subsequently, two GGPP molecules are condensed by phytoene synthase (CrtB) to form phytoene. Finally, lycopene synthase (CrtI) catalyzes phytoene desaturation in lycopene.

The lycopene synthetic pathway is, by its nature (synthetic), heterologous to the chassis and thus not metabolically balanced and it is necessary to re-balance its metabolic flux to optimise the production of lycopene.

The lycopene synthetic pathway (Figure 2-3) comprises three successive enzymatic reactions catalysed by the enzymes CrtE, CrtB and CrtI (Yoon et al., 2006; Misawa et al., 1990) – such simplicity, coupled with the fact all the associated regulatory mechanisms have been characterised in several strains, makes the lycopene pathway an appealing exemplar case study for optimisation problems in metabolic engineering. Lycopene production can be crippled by metabolic burden (strong production of lycopene and intermediaries take away resources otherwise used for cellular growth) (Borkowski et al., 2016) and creates toxicity to

the cell – lycopene having been proven to be toxic to the cell (Wang et al., 2019; Taylor, Hitchcock & Heap, 2021), due to its accumulation in the cellular membrane.

This pathway is heterologous to the chosen bacterial chassis (*E. coli*). *E. coli* is one of the main model organisms for terpenoid production (A. Rinaldi, A. Ferraz & S. Scrutton, 2022) The three genes GGPP synthase (crtE), phytoene synthase (crtB) and phytoene desaturase (crtI), come from *Pantoea ananatis*, a plant pathogen bacteria. Most terpenoids, like lycopene, are made in nature from the precursors isopentenyl diphosphate (IPP) and dimethylallyl diphosphate (DMAPP), produced from central metabolism via the 2-C-methyl-D-erythritol 4-phosphate (MEP) or the mevalonate (MVA) pathways (A. Rinaldi, A. Ferraz & S. Scrutton, 2022). Most organisms use the natural MEP or MVA pathways to make terpenoids and bacteria use the native MEP pathway (A. Rinaldi, A. Ferraz & S. Scrutton, 2022). The precursor pathways (MEP or MVA), not shown in Figure 2-3, produce both isopentenyl diphosphate (IPP) and dimethylallyl diphosphate (DMAPP) at a ratio of 5:1 in *E. coli*. Both IPP and DMAPP are produced by the MEP pathway (A. Rinaldi, A. Ferraz & S. Scrutton, 2022). As shown in the lycopene synthetic pathway in Figure 2-3, condensation of IPP and DMAPP to generate geranyl diphosphate (GPP) is catalyzed by diphosphate synthase (GPS). GPP is condensed by one molecule of IPP by action of FPP synthase (FPS) to form farnesyl diphosphate (FPP). Then, FPP and IPP are condensed by the catalytic action of GGPP synthase (CrtE) to form geranylgeranyl diphosphate (GGPP). Subsequently, two GGPP molecules are condensed by phytoene synthase (CrtB) to form phytoene. Finally, lycopene synthase (CrtI) catalyzes phytoene desaturation in lycopene (Méndez-Carmona, Hernández-Almanza & Aguilar, 2022).
High-yield production of carotenoids in engineered microbial hosts requires optimisation of the available precursor pool of IPP and DMAPP – and balancing the expression of carotenogenic genes for the efficient transformation of the precursors to the desired carotenoid compounds (Yoon et al., 2006). In the project we focused on balancing the three genes.

5.3 Design of the lycopene operon

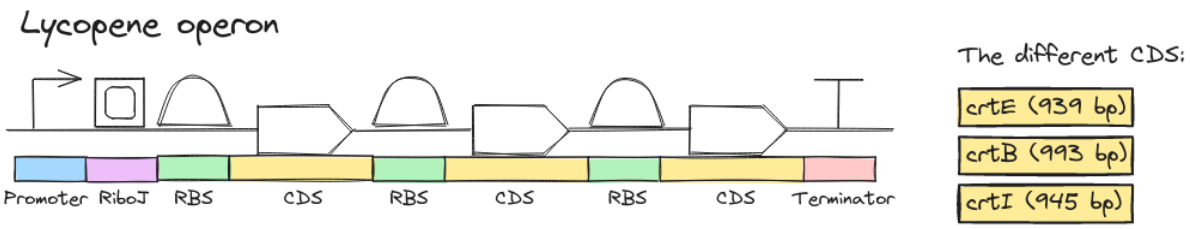


Figure 2-4 – The lycopene operon. The design includes these 3 genes (crtE, crtB and crtI) arranged in an operon pattern. All gene orders are permitted. A constitutive promoter drives the operon and an

insulating RiboJ is inserted post promoter.

The construct design used in this application is the design already used in (Exley et al., 2019) in the context of the use of well-characterised parts for DoE workflows. The three CDS of the operon produce, in a controllable manner, the three different enzymes (CrtE, CrtB, CrtI) that produce lycopene. The construct design (see Figure 2-4) is based on the following principles:

- An operon design is used as it is a common design feature in nature for prokaryotic systems and in synthetic biology (Gerngross, Beerenwinkel & Panke, 2022). It also reduces the design space compared to a design where each gene has its own transcription unit and reduces chances of homologous recombination if promoters are reused.
- A constitutive promoter drives the operon. Five constitutive promoters were chosen from the SynBIS library (Bultelle, de Murieta & Kitney, 2016), to span a range from weak (K137085) to strong (Kelly's reference promoter J23101) (Kelly et al., 2009), to very strong (BioFab promoter apFab32) (Mutalik et al., 2013) for their relative strengths.
- An insulating element RiboJ (Clifton et al., 2018) was inserted between promoter and first UTR-RBS as a direct consequence of using the BASIC assembly method and its library parts.
- All three RBS in the operon can be varied, as per Blazeck et al. (2012) and Salis, Mirsky & Voigt (2009). For each position, three RBS parts from a subset of the Biolegio library (referred to as RBS 1, 2 and 3 in the rest of this paper) were used.
- Finally, the design does not fix the gene order and allows for permutation instead – thus permitting a re-investigation of previous studies such as Nishizaki et al. (2007) with the carotenoid pathway in *E. coli* (and yeast), which demonstrated the influence of said gene order.

The DNA parts were sourced from a DNA repository created for the project where parts are stored as Genbank files. In our application the parts in the local repository have been sourced from online databases such as the Registry of Standard Biological parts (iGEM BioBricks) (parts.igem.org, n.d.) and SynbioHub (McLaughlin et al., 2018). In a second step, the DNA parts assembled from the lycopene operon construct design template; the design template data structure is detailed in the next chapter (Chapter 3). In a last step, a DNA assembly software produces the multiple combination of parts and outputs all the combinatorial DNA constructs to be assembled.

The original choices by Exley et al. (2019) led to a design space of 810 promoter/RBS/gene order combinations (6 permutations of the gene order, 5 promoters and 3 RBS in 3 positions). The size of the design space (810 constructs) made this combinatorial optimisation problem a suitable test bed for their integration of automation and software. It was small enough to build all combinations using standard modular plasmid construction methods and large enough to employ DoE.

5.4 Build with BASIC DNA assembly

The Biopart Assembly Standard for Idempotent Cloning (BASIC) (Storch et al., 2015) is a method and a standard to enable highly accurate multi-part DNA assembly at both manual bench and fully automated biofoundry scale (Storch, Haines & Baldwin, 2020). BASIC makes use of computationally designed linkers to join parts together (Casini et al., 2014; MacDonald & Siciliano, 2017). The DNA parts are available in a single storage format with a prefix and a suffix sequence flanking the part and stored in a high-copy vector (Storch, Haines & Baldwin, 2020). Linkers also act as composable parts which enriches the design space: UTR-RBS linkers enable tuning ribosome binding site (RBS) strength within a consistent 5'-unstranslated region (UTR). Linkers with a complete coding sequence can be used to create fusion proteins (Storch et al., 2015). The various steps of the BASIC assembly are illustrated in Figure 2-5.

To build our DNA constructs we used the BASIC DNA assembly method. For the following reasons:

- The initial study of the lycopene construct being done in the Kitney lab made use of BASIC assembly (Exley et al., 2019) and has been proven to be successful.
- The BASIC assembly method has been proved successful in some projects (Casini et al., 2015; Storch, Haines & Baldwin, 2020; Team Imperial College, 2020; Hérisson et al., 2022)
- BASIC DNA assembly has been partly developed and tested at the London Biofoundry (Exley et al., 2019) and a collection of the BASIC linker parts (basic-assembly.org, 2023) is available commercially (Biolegio, 2023) in the form of a kit.
- The method is amenable to automation (Storch, Haines & Baldwin, 2020)
- The method is favourable to build multi-part operons (Storch, Haines & Baldwin, 2020)
- BASIC DNA assembly has taken advantage of the availability of a large collection UTR-RBS linkers which enabled us to extend our design space
- The method allows the creation of a one-pot assembly strategy (See the PASIV

method in Chapter 3) to randomly build the constructs for our whole design space in one-pot reaction by mixing the various UTR-RBS linkers equimolarly (Casas et al., 2022)

To assist the planning of BASIC DNA assemblies we have worked on our own software scripts which then evolved into the 'Basicsynbio' software created by Haines et al. (2022). 'Basicsynbio' is an open-source Python package that enables the construction of BASIC construct design programmatically, users can access commonly used BASIC parts and linkers while designing new parts and assemblies with exception handling for common errors. Sequence data can be exported and instructions for manual or acoustic liquid-handling platforms can be created.

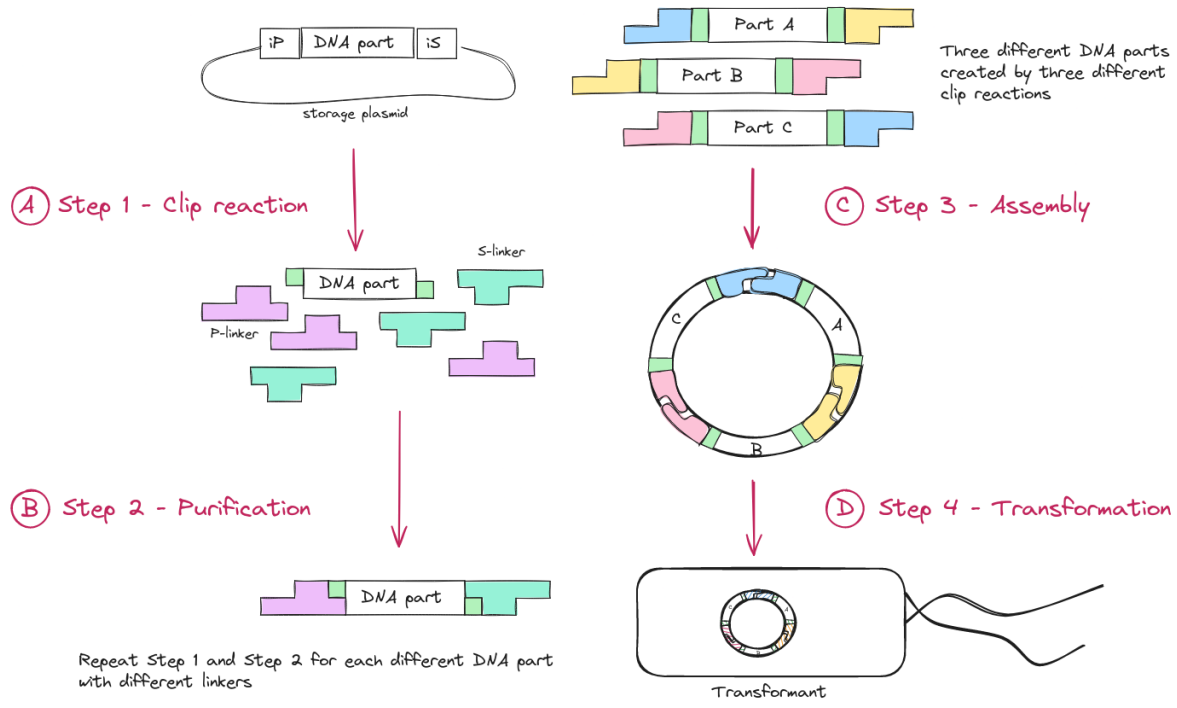


Figure 2-5 – BASIC DNA Assembly steps. (A) Step 1: Clip reaction: simultaneous digestion and ligation attaches Prefix (P-) and Suffix (S-) half-linkers to parts. **(B)** Step 2: Purification: clips are purified from the reactions removing excess linkers and enzymes. **(C)** Step 3: Assembly: purified clips are annealed and form the circular constructs. **(D)** Step 4: Transformation: assembled constructs are transformed into *E. coli*.

The design of the DNA clipping reactions required by the BASIC DNA assembly method was done in silico using the ‘Basicsynbio’ software (Haines et al., 2022).

After the step of BASIC DNA assembly, a quality control (QC) step was inserted to verify the various DNA constructs that have been assembled correctly. After transformation, DNA extraction and DNA purification was performed and DNA concentration was then verified. DNA was then sent for sequencing (Sanger sequencing) after a PCR step.

5.5 Test

Step 3 of the DBTL cycle consists of testing the operon in vivo and measuring its output. Transformation of the constructs were performed and after incubation, growth of the chassis was observed during which the molecule is being produced by the enzymatic cascade. The molecule of interest, lycopene, was extracted with an automated protocol that we implemented on a CyBio Felix (Analytik Jena AG, n.d.a) automated liquid-handling platform. Finally the final concentration of lycopene was measured with a UV-vis method in a plate-reader (BMG Clariostar Plus).

6. Summary

In this chapter we have introduced the notion of metabolic engineering. We have presented an overview of the different strategies to combinatorial engineering and discussed the problems introduced by large combinatorial pathway approaches: the search space is very large, a lot of data gets generated and it requires many rounds of experiments. The advent of high-throughput automation and modular DNA assembly techniques have made the combinatorial pathway approach feasible and engineering methods such as DoE help reduce the number of experiments to run to converge to an optimal solution in a reduced amount of time.

Finally, we have presented an engineering workflow to investigate combinatorial metabolic pathways and we illustrate this workflow with the chosen case study of an application for the production of a metabolite: lycopene, which has a strong industrial interest and is a model pathway that has a size that large enough but not to large to explore the combinatorial space.

By implementing this workflow in the biofoundry we have encountered obstacles that we had to overcome by introducing new methods that we will present in the next chapter.

Chapter 3
Initiation of the DoE Cycle

1. Introduction

Along with the development and implementation of the different steps of the lycopene application workflow presented in the previous chapter (Chapter 2), we have identified some problems that were addressed by developing various laboratory or computational methods.

The first problem relates to the problem of initiating a DoE cycle. We define the initiation of the DoE cycle as the initial random selection of constructs from the design space which we build by assembling DNA parts and transform in bacterial cells, which we grow and to test with a measurement assay. Due to the toxicity of lycopene to the cell (Wang et al., 2019; Taylor & Heap, 2020), the growth of some constructs could not be observed. The absence of growth makes it impossible to test the constructs by measuring the lycopene being produced by the cells. To overcome this issue and to be able to bootstrap the DoE cycle we designed a method that we call PASIV (Pooled Approach, Screening, Identification and Visualisation). Which will be presented in detail in Section 2 of this chapter.

The second identified problem is the need for identification of the constructs. In the traditional approach of building and growing constructs, each construct is being given a unique identification identifier (the constructs being identified in all the wells of the labware used along the workflow).The PASIV method builds constructs using a pooled approach. This means that all the combinations of parts are randomly assembled in a one-pot reaction, simultaneously. We thus cannot keep track of which construct is which, and thus require to add a construct identification step. A software called cMatch, presented in Section 3 of this chapter, has been developed to allow a fast identification of constructs after a DNA sequencing step.

2. Failure to Launch the DoE Cycle because of Burden and Toxicity Issues

2.1 Replication study - observed experimental failure

We conducted a preliminary study for this work and replicated the same initial DoE workflow as Exley et al. (2019). In the work of Exley et al. (2019) a design of 810 possible configurations of the lycopene operon was considered (comprising 14 distinct parts, with varying promoters (5 different promoters), RBS (3 different RBS), CDS (3 genes crtI, crtB, crtE), one terminator, and 18 permutations of RBS and the gene location.); 88 constructs were randomly selected by the DoE software (JMP). The 88 constructs were assembled and tested. We picked an initial set of 8 constructs out of the 88 random constructs from the previous study (Exley et al., 2019). The set of 8 constructs was assembled and transformed in *E. coli*. The transformants were plated and left to grow in an attempt to isolate some colonies for a further culture – in order to extract and measure the produced lycopene. We observed that growth of the transformed colonies was either non-existent or very slow. The first colonies observed took as long as 72h to grow.

To further investigate the reason for the observed growth discrepancy, the colonies that grew were picked and sent for Sanger sequencing. Two sets of issues were identified:

- Sequencing results showed that many genes constituting the lycopene operon had either been deleted or mutated and construct parts could not be identified.
- A significant number of non-expressing lycopene colonies were observed growing on top of lycopene-expressing ones and outcompeting them on the petri-dish.

Both issues indicate a tendency of the cells to mutate or eliminate the operon. These failures can be attributed to two biological factors: metabolic burden (strong production of lycopene and intermediaries take away resources otherwise used for cellular growth) (Borkowski et al., 2016) and toxicity – lycopene having been proven to be toxic to the cell (Wang et al., 2019; Taylor & Heap, 2020) due to its accumulation in the cellular membrane. In practice these failures are worsened by two sources of technical failures – experimental failures, and assembly failures.

2.2 Failures and failure modes

In the context of a DoE cycle, burden and toxicity are major obstacles to an efficient, reliable and reproducible investigation of the design space. In general, burden leads to large variations in the growth of the cells, which is itself a major obstruction to any effort to operate colony picking since colonies cannot be picked within the same timeframe. This reduces the

throughput of the entire operation, whether conducted manually or with automation. In titration assays, growth discrepancy is also a significant source of variation in the data. Toxicity (and mutations) will restrict the viable portion of the design space – more or less severely, depending on the capacity of the chassis to tolerate said toxicity (Sikkema, de Bont & Poolman, 1995).

Debugging individual failures – i.e. identifying the reason for a failure to transform and grow a colony is difficult. For instance, transformation failures cannot be observed immediately after the act itself, but only later in the process. Separating failure modes at population level is easier since they are distributed very differently over the design space (as illustrated in Figure 3-1).

- Assembly failures are sparsely occurring events (Figure 3-1A) – common modular assembly methods are very effective when short pathways are assembled as is the case for the lycopene exemplar and can be assumed independent from the construct (see discussion on our solution PASIV).
- Assay failures are uniformly distributed: assay/experimental failures are independent from the construct; their frequency depends on the skills of the experimentalist (or the level of automation) and the protocol adopted (Figure 3-1B)
- Toxicity and burden issues are NOT independent of the construct. The constructs encode a set of enzymes, and the recorded output (viability, OD, concentration of a metabolite) depends on the concentration of enzymes that are produced. When toxicity and burden are too low to affect the constructs, the entire design space is, *a priori*, viable (Figure 3-1C, left). Reciprocally, when toxicity and burden are issues – only contiguous portions the design space are viable (Figure 3-1C, right)

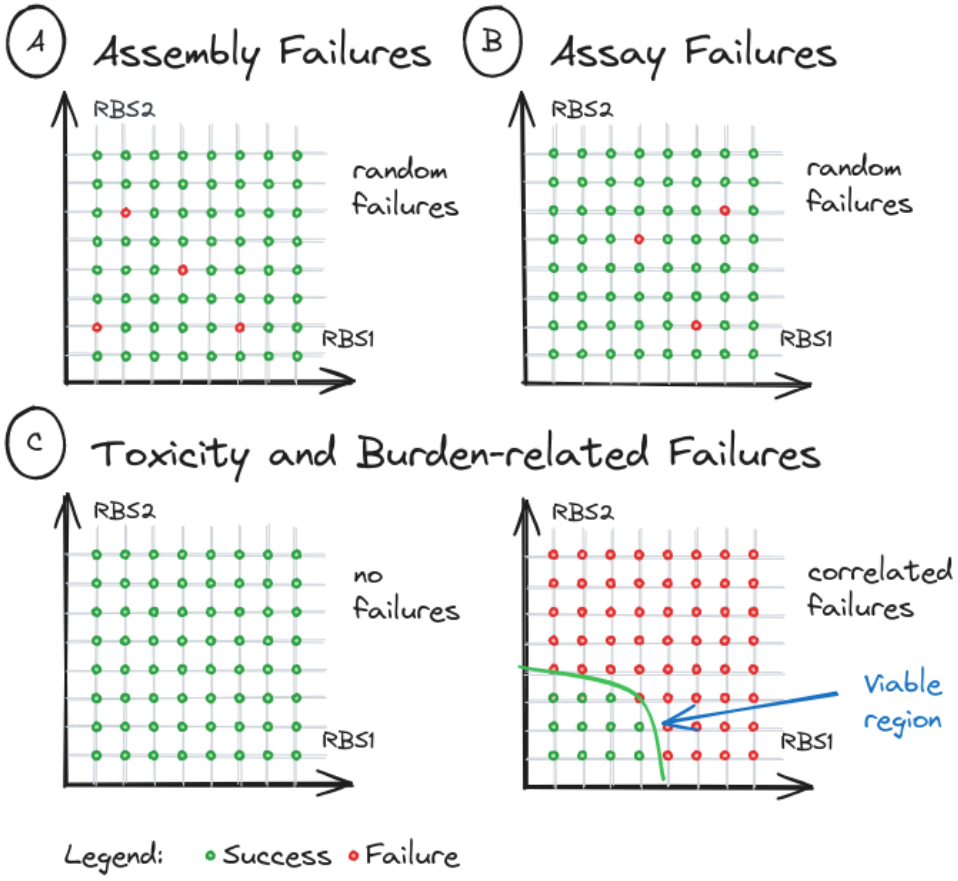


Figure 3-1 – Failure modes affecting the scoping round. Three types of failures are considered in this work. A) Randomly-distributed assembly failures. B) Assay failures (also randomly distributed). C) Toxicity-induced failures (highly correlated).

2.3 Initiation of the DoE cycle

A DoE workflow consists of a series of iterative experiments which are grouped in three categories: scoping, screening and optimisation (Gilman, 2021). The scoping experiment is used to confirm the range of the factors (the design parameters that we change and for which we want to find the effect in our application). The scoping phase of a typical DoE cycle consists of randomly drawing a small subset of the design space – to be followed by successive, targeted, draws based on data analysis of the collected data, and finally an optimisation phase once a narrow-enough region of interest has been identified. Figure 3-2A shows what can be expected at the end of the scoping phase for two extreme cases – when the whole design space is unaffected by toxicity and burden issues (non-toxic case – Figure 3-2A, left), and when toxicity and burden issues are severe but in a small region of the design space (toxic case – Figure 3-2A, right). In the non-toxic case, it is possible to transform, grow and assay for the vast majority of the initial constructs – failures can be solely attributed to technical issues – and enough data can be collected to proceed to the targeted rounds. The scoping round works as intended. Conversely, in the toxic case (for instance the exemplar lycopene study), few of the constructs that are initially drawn lie in this viable region. Failures may also be due to assembly or experimental issues, rather than burden or toxicity. Overall, only very few data can be collected at the end of this scoping phase – too few to analyse and proceed to the later phases of the DoE cycle. The scoping round fails.

Figure 3-2B shows the expected outcome of a scoping round if the whole design space is used as an input. In the non-toxic case, most colonies corresponding to the input constructs will grow and yield data – as before (the few) failures can be assigned to assay or assembly issues. In the toxic case, colonies will only grow for constructs located in the viable region. The viable region reveals itself by continuity, when successful assays are mapped in the construct space. Using the whole space as input is, of course, totally impractical – and implementing a brute force approach is antithetic to the idea of DoE. It is possible and tractable however, to randomly sample large sets by using multiplex construction in a pooled approach. The high density of the input dataset will then still allow us to exploit the different distributions of the failure modes and identify the continuous regions.

A novel workflow is presented in the next section – coined PASIV. This implements the methodology, and, through an interplay between biology (multiplex construction in a pooled approach) and software (a range of bioinformatics tools, including a novel construct-matching tool, to overcome issues inherent to pooled approaches) offers a novel way to perform the scoping phase of the DoE cycle, and identify the (viable) region of interest.

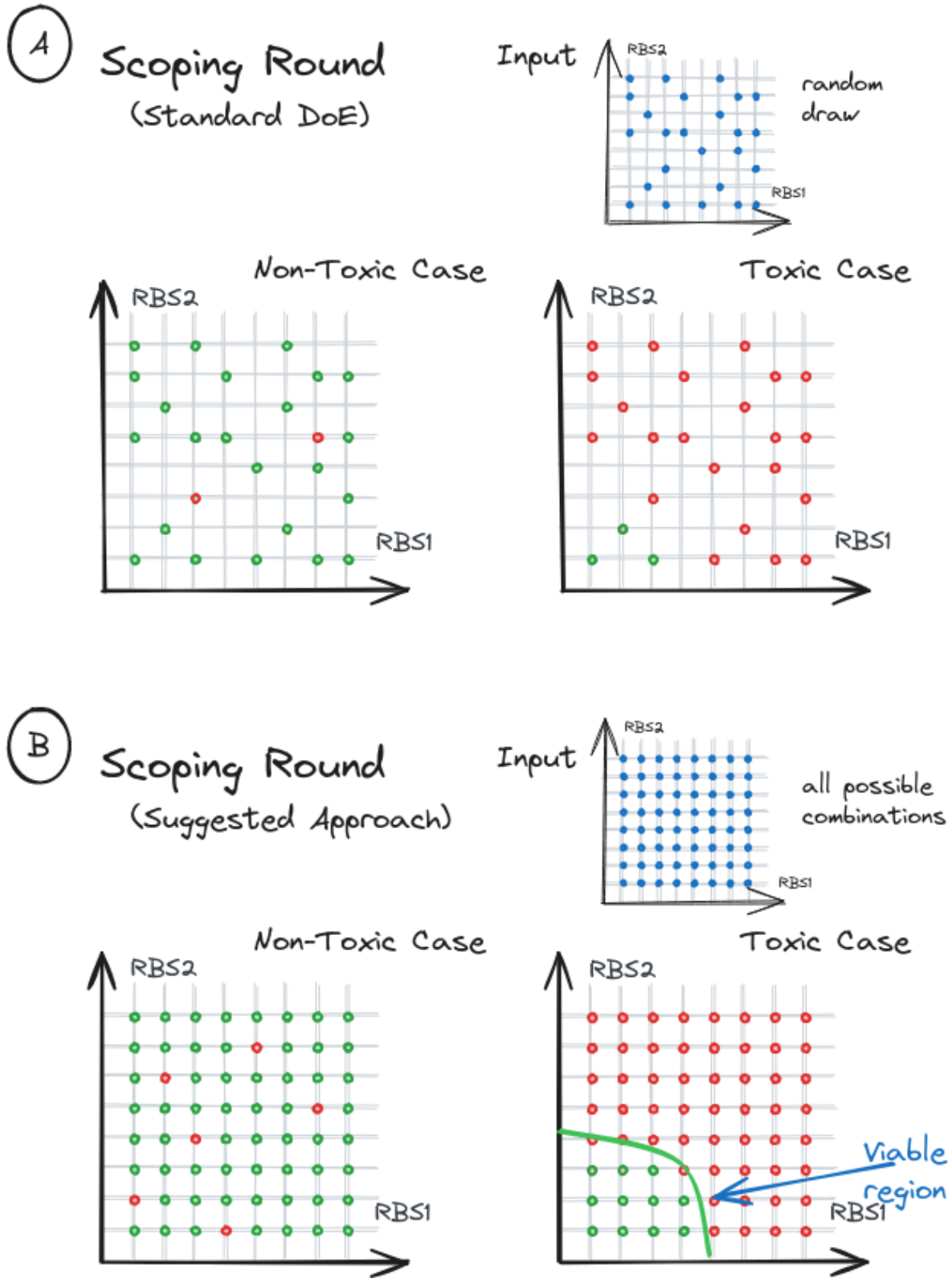


Figure 3-2 – Expected outcome to a scoping round in the non-toxic and toxic cases and for two different sets of input constructs. (A) A subset generated by a random draw from the design space is used as input. **(B)** The whole space is used. In the toxic case, colonies will only grow for constructs located in the viable region. Only by using a dense input dataset will the viable region reveal itself by continuity – this simple remark is the foundation of the PASIV method.

3. The PASIV Approach

The acronym PASIV stands for Pooled Approach, Screening, Identification and Visualisation – the four distinct phases of the workflow. PASIV is also, as its name implies, a passive method, as the viable region of the design space is allowed to appear through the viability assay (unlike the more active standard approach of drawing random samples and inferring the regions of interest from them). The PASIV approach comprises four successive phases.

3.1 The PASIV workflow

This section describes the four different phases of the PASIV workflow as illustrated in Figure 3-3.

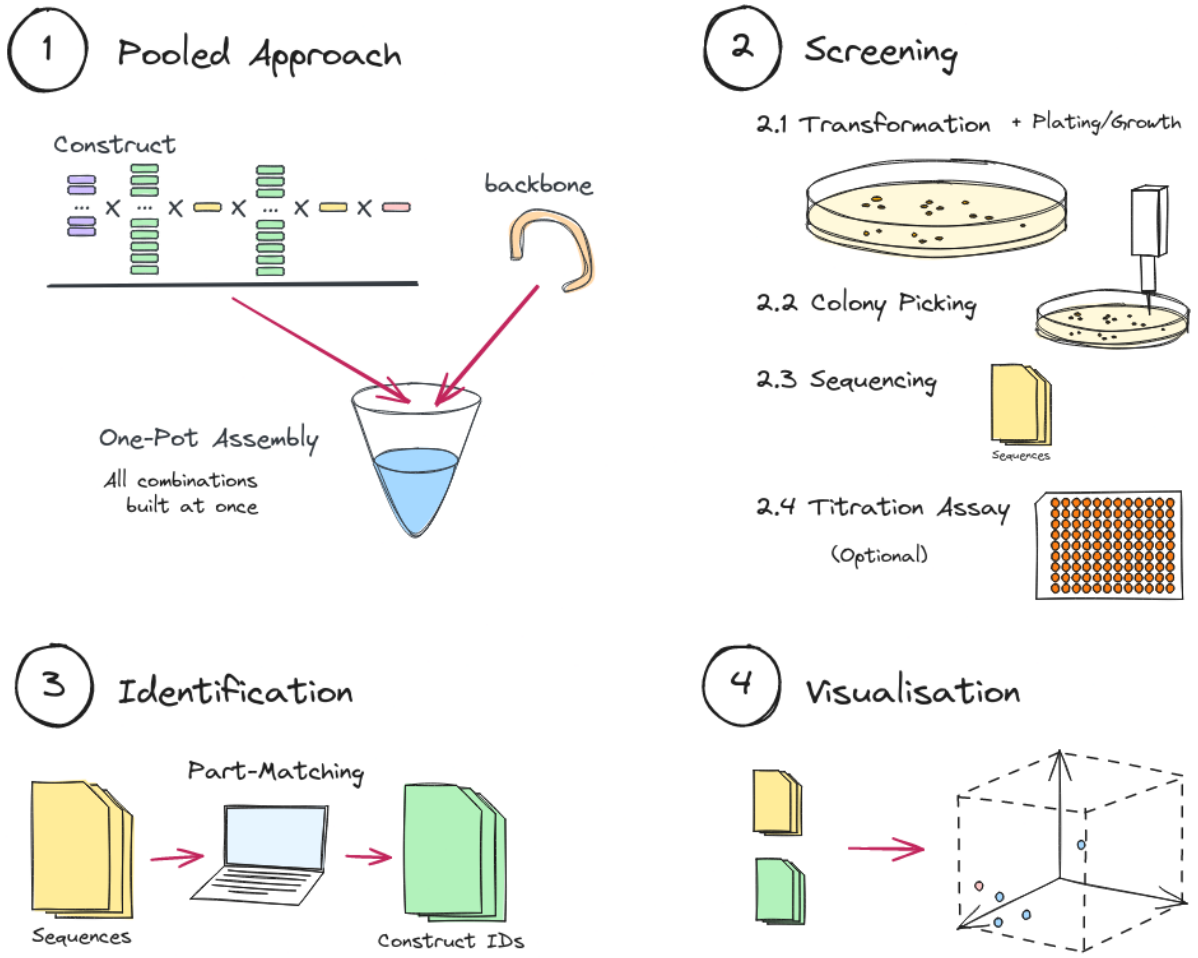


Figure 3-3 – The PASIV workflow. PASIV is made of four successive phases. **1 Pooled Approach** – a construction step using modular assembly in a pooled approach and aiming at building all constructs at once. **2 Screening –** a viability assay consisting of a transformation and culture step followed by a colony picking step (the assay may be followed by a titration assay where metabolite production by the yet unidentified colonies is measured. **3 Identification** – an identification step where the genetic content of the picked colonies is identified. **4 Identification of the Viable Region –** a visualisation/analysis and clustering phase step that combines all collected information.

Phase 1. Construction with a pooled approach

Instead of the targeted construction of a set of constructs (drawn at random), a multiplexed construction via a pooled approach is implemented. The purpose of this stage is to build all possible constructs simultaneously, in a fast and effective manner. Practically, it is expected to generate a large number of constructs from across the entire design space – with no obvious bias (large gaps, different relative frequencies). A pooled approach helps ensure the same experimental conditions apply to all the constructs that are built, thus minimising one of the potential sources of errors previously listed.

Phase 2. A viability screening assay

The cells are transformed with the genetic pool in another one-pot reaction. The transformed cells are then plated and grown for the selection of viable candidates. Viable candidates are picked up from the plate(s) and sent for sequencing. Selection should be at random and, in particular, should not be affected by apparent production (as assessed by the colour of the colonies). A titration assay can then be run with the viable candidates to collect data on metabolite production. This second assay collects standard titration features such as the optical density and the concentration of the metabolite of interest and intermediaries if available (also either as a time series or endpoint of assay).

Phase 3. Construct identification

Since the cells have been transformed with a mix containing all possible plasmids, there is little prior knowledge of the genetic content of the picked colonies – except that it matches a given template, was assembled from a known set of components and therefore lies in a, possibly large, space of possibilities. There is also no certainty that all the constructs will feature among the picked colonies; while some may be represented several times. This identification phase is crucial to the overall success of the method – without it, no consequent data visualisation and analysis is possible. Identification is achieved using a construct-matching software called *cMatch* (see Section 5) that was developed as part of PASIV.

Phase 4. Clustering and data visualisation – identification of the viable region

Finally, the collected data (colony content and assay measurements) are combined, analysed and visualised. The goal of this stage is the localisation of the viable region – that is the region least affected by toxicity and, therefore, eligible for titer optimisation.

Within PASIV, metabolite production is treated with a black-box approach. The constructs in the design space are seen as an enzyme producer. Enzyme production is itself viewed as a multi-dimensional stimulus to the chassis, which is controlled by tuning construct components (promoter, RBS ...). The bacterial chassis has its own limited resources, feedback systems, including resistance to toxicity, and capacity for producing a metabolite of

interest. No attempt is made to model these processes; instead, we rely on the data to estimate the dose-response relationship. After data combination, all constructs in the design space are assigned a multiplicity order (i.e. the number of times the construct has been identified among the picked colonies – this provides strong evidence as to whether the colony is viable and/or subject to burden issues) and the data collected during the titration assay (if any).

PASIV uses a minimal set of coordinates corresponding to estimates of the synthesis rates of all products encoded by the construct (i.e. the enzymes CrtE, CrtB, and CrtI in the case of the lycopene example). These coordinates – called 'modelling coordinates' in the rest of this chapter as they are based on the modelling of protein production – correspond to a representation of the stimulus amplitude. Modelling coordinates can be used as universal coordinates for constructs encoding the same functions (e.g. enzyme production) – and thus grant comparisons between constructs with different gene orders or based on different patterns (operon vs transcription units). Crucially, metabolic burden and toxicity effects are correlated to the amplitude of the stimulus – the higher the stimulus, the higher the burden and in the case of lycopene, the more lycopene is produced, the higher the toxicity for the cells. This makes such a set of coordinates ideal to differentiate toxicity effects from construction and experimental issues.

3.2 Application to the lycopene exemplar

We will now illustrate how the general PASIV workflow has been used in practice with the lycopene exemplar presented in Chapter 2. Three of the four phases of the workflow were customised as follows (the third phase of constructs identification which was done with cMatch (see Section 5) is completely independent from the application, and needs not be customised):

Phase 1. Pooled Approach

The BASIC assembly method (Storch et al., 2015) is used for the modular assembly of the constructs in a one pot reaction. When several gene orders are investigated, it is necessary to operate several pools – one for each gene order. This is, primarily, to ensure no combination lying outside the design space (constructs with multiple repeats of some CDS and missing CDS) is constructed. In the context of lycopene, where toxicity and burden are the most limiting factors. It is crucial to the success of the methods as constructs missing one of the pathway enzymes will exhibit a significant growth advantage (no lycopene made) and will therefore be overwhelmingly represented at the screening stage.

Six different pools – each corresponding to one of the six possible gene combinations for crtE, crtB and crtI – were created. For each of these pools:

- The other varying parts (promoters, RBS) are mixed in equal quantities.

- The cells are transformed with these genetic pools in a one-pot reaction.
- The transformed cells are plated and grown for the selection of viable candidates.

Phase 2. Screening of the Viable candidates

Viable and lycopene-expressing colonies were selected and picked. Practically, the transformed cells that have grown on the petri-dish and that express the characteristic orange colour hue phenotype of lycopene expression (Armstrong, 1997) were selected and colonies are picked and grown in liquid culture for 24h for further analysis: lycopene extraction and measurement, and sequencing. Lycopene was extracted with DMSO (see Appendices C3-Methods) from the liquid culture of each of the viable candidates. OD600 and absorption at 471nm, the characteristic wavelength of lycopene (Takehara et al., 2014) are measured and the lycopene yield is estimated.

Phase 4. Derivation of the modelling coordinates

In the lycopene study, production of each enzyme can be modelled with the standard constitutive expression model:

$$\frac{dm}{dt} = K_1 - d_m \times m$$

$$\frac{dE}{dt} = K_2 \times m - d_E \times E$$

Where K1 is the effective transcription rate, K_2 the effective translation rate, dm the effective mRNA degradation rate and d_E is the effective enzyme degradation rate. The synthesis rate is K_1K_2 / d_m and the corresponding steady-state concentration is $K_1K_2 / d_m d_E$.

Since visualisation, clustering and other operations conducted in Phase 4, can be conducted up to a multiplicative constant, not all quantities need be estimated and those who do can themselves be estimated up to a multiplicative constant.

The degradation rates of the 3 enzymes CrtE, CrtI and CrtB are constants of the problem. Thanks to the RiboJ's protection of mRNA at the 5' end, the mRNA degradation rate can also be assumed a constant of the problem – independent of the gene order, albeit hard to estimate. A decision was therefore made to use K_1K_2, that is the product of the effective transcription rate and the effective translation rate for effective synthesis rate, instead of the exact K_1K_2 / d_m to represent the synthesis rate. All synthesis rates used in the rest of this chapter are henceforth expressed in arbitrary units due to this simplification.

With the lycopene operon used in this study, the effective transcription rate K_1 for each gene is estimated from the characterisation data of the leading promoter (Appendices C3-Table-S1). All values are expressed in RPU (relative promoter units, a relative unit of strength common in promoter characterisation (Kelly et al., 2009), which has been estimated at

approximately 0.02 RNAP/s/promoter (Espah Borujeni et al., 2020). The use of the insulating element RiboJ (Clifton et al., 2018) ensures transcription and translation are decoupled from each other and these estimated promoter strengths could be used for the first gene of the operon. Effective transcription rates for the other positions were derived by multiplying its strength by a factor depending on its position down the operon, as suggested by Nishizaki (Nishizaki et al., 2007) whose experimental results proved that mRNA abundance decreased by roughly half from one gene to the next'.

The decoupling of transcription and translation is an example of favourable compositional context – unfortunately, no such simplification is possible for the estimation of the effective translation rate. The context is the set of interrelated factors that modulate the operation of biological processes. These factors are traditionally grouped as composition-specific, environment-specific and host-specific (Cardinale & Arkin, 2012). In general, the expression of functional components is affected by short adjacent upstream/downstream sequences (Espah Borujeni, Channarasappa & Salis, 2014; Tietze & Lale, 2021; Taylor, Mordaka & Heap, 2019). Calculator tools such as the RBS calculator (Salis, Mirsky & Voigt, 2009; RBS Calculator, 2021) now require 35 bp upstream and 60 bp downstream to estimate the translation rate of an RBS.

No attempt was made at experimentally characterising the RBS in context. The effective translation rate K_2 was estimated with the RBS calculator instead, for all the RBS in the Biolegio library and for all possible contexts with the operon design (upstream and downstream sequences from the flanking genes or RiboJ for the six (6=3x2x1) possible permutation orders of the CDS, six different pools were created corresponding to all the possible permutation order of the three CDS (crtI, crtB and crtE) (Pool 1: BEI, Pool 2: BIE, Pool 3: EBI, Pool 4: EIB, Pool 5: IBE and Pool 6: IEB)). Corresponding estimates can be found in the Appendices C3 - Translation-Rates and provide a vivid illustration of the influence of context on RBS expression. Not only do translation rates vary by as much as three orders of magnitude, but the ranking orders between RBS in the library are also not conserved when context changes. Such unpredictability contrasts sharply with transcription, which varies with gene order in a predictable manner by a factor of up to 4 (Nishizaki et al., 2007).

4. Experimental Results (PASIV)

4.1 Preliminary investigation of the design space

The first immediate application of the set of coordinates was to conduct a preliminary investigation (Figure 3-4 and Figure 3-5) of the design space generated by the operon

design and for the entire BASIC Biolegio library (Biolegio, 2023) and the chosen subset of {RBS1, RBS2, RBS3}. For all visualisations, the synthesis rates (synthesis(crtE), synthesis(crtB), synthesis(crtI)) are used as coordinates along three axes respectively named E, B and I.

Using the entire 15-strong Biolegio RBS library yields a design space made of 101250 constructs (5 promoters, 15 RBS in position 1, 15 RBS in position 2, 15 RBS in position 3 and 6 permutations for the gene order). The corresponding design space (Figure 3-4) is highly anisotropic. Constructs exhibit a larger density close to the origin, along the E and B axes and are also overwhelmingly located close to the E-B plane (The E-B plane being defined by the two axes E and B). (This is to be expected, as it corresponds to low crtI production and is a direct consequence of the effect of RBS context (Cambray, Guimaraes & Arkin, 2018). Having crtI downstream of the RBS drastically reduces the translation rate.) This was verified by conducting a codon optimisation of the crtI sequence (IDT Codon Optimization Tool, 2021) and comparing the estimated translation rates for both crtI and its codon-optimised sequence. Results (Appendices C3 -Translation-Rates) show an increase by an order of magnitude with the codon-optimised sequence. A wide region by the origin ([0,25000]x[0,25000]x[0,1000] in E-B-I coordinates) was densely sampled by the design space; while regions further and further from the origin less and less so.

Influence of the gene order can be observed thanks to the colour scheme used in the figure (see Figure 3-4). Separate visualisation of the constructs by gene order (Appendices C3 - Visualisation-By-Gene-Order) shows constructs tend to be located in different sections of space depending on their gene order – suggesting a possible reason why some gene orders are associated with better production and yield: over-representation in production sweet spots. Separate visualisation of the constructs by promoters (Appendices C3 - Visualisation-By-Promoter) show that the dense regions are not only the results of the expression of the RBS in different contexts and modulated transcription, but also of the weighting by promoters of different strength.

Figure 3-5 shows the 810-strong design space used in the constitutive study (5 promoters, 3 RBS in position 1, 3 RBS in position 2, 3 RBS in position 3 and 6 permutations for the gene order). The design space generated with this subset is much sparser as it is reduced by a factor of more than a 100 (810 instead of 101325). It also exhibits a truncated coverage as coordinates cannot go above 30000 along the E axis, and coordinates do not exceed 1000 along the I axis (Figure 3-5A). A zoom-in on the region close to the origin (Figure 3-5B) – corresponding to the weakest constructs, producing the enzymes in little quantity and thus expected to produce little lycopene, shows that the region remains densely packed (in the E-B plane), but only contains very low values along the I coordinates due to the effect of RBS context – i.e. the coding region of crtI upstream of the RBS affecting its translation rate.

Preliminary results obtained while attempting to reproduce Exley et al. (2019) have shown how difficult it is to obtain results even for weak constructs, therefore it was decided to stick to the original {RBS1, RBS2, RBS3} subset and not try to extend the design space with stronger (in context) RBS. Further post-hoc justification for this choice of the reduced RBS library was supplied by the experimental results presented in the next section, showing that all constructs yielded by the pooled approach were located well within this region of weak constructs.

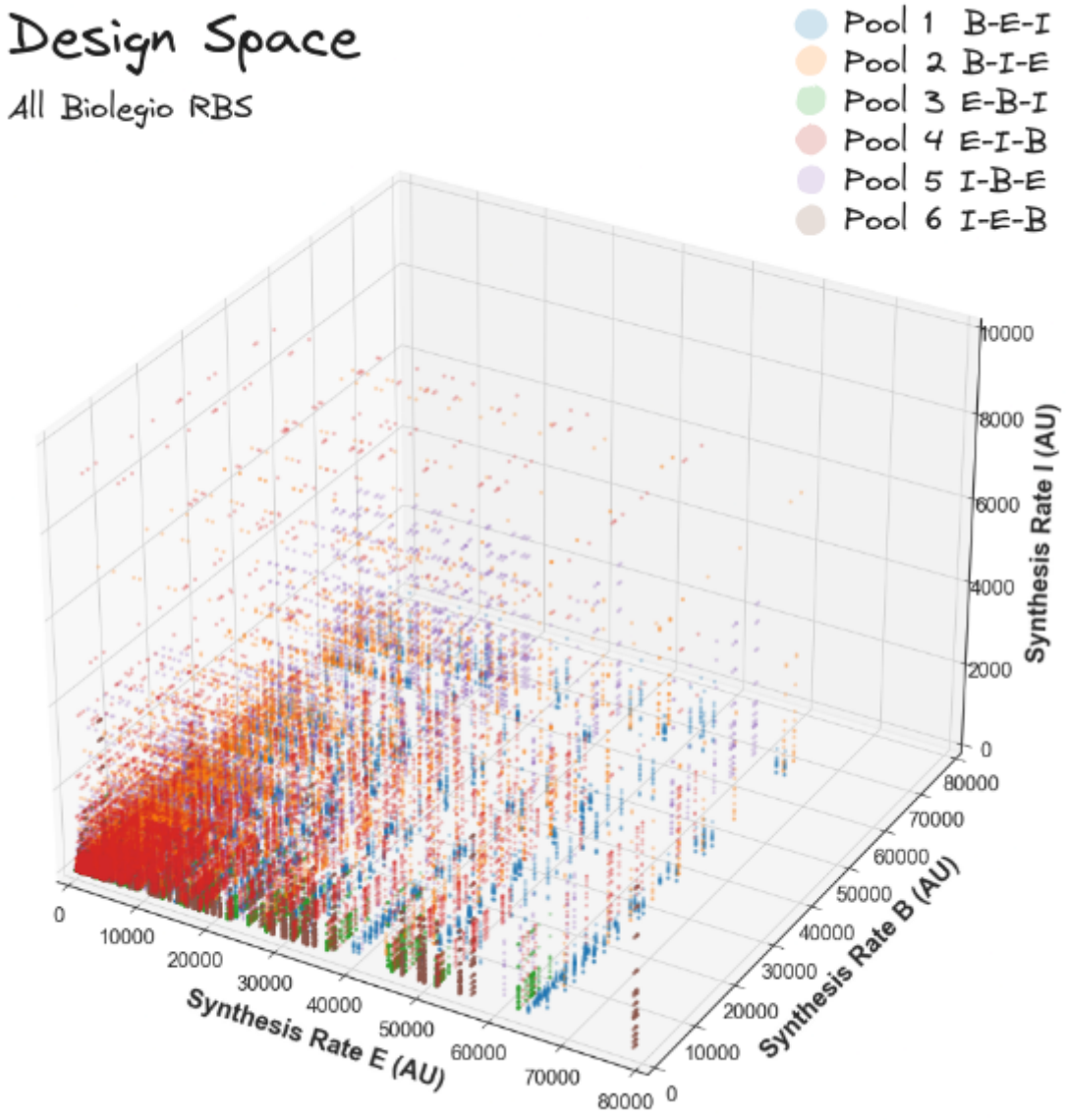


Figure 3-4. Design space generated from the entire RBS Biolegio library and the subset {RBS1, RBS2, RBS3}. Constructs are colour-coded according to their gene-order/pool. The design space generated from the entire Biolegio library has 101250 constructs and is heavily concentrated by the E-B plane.

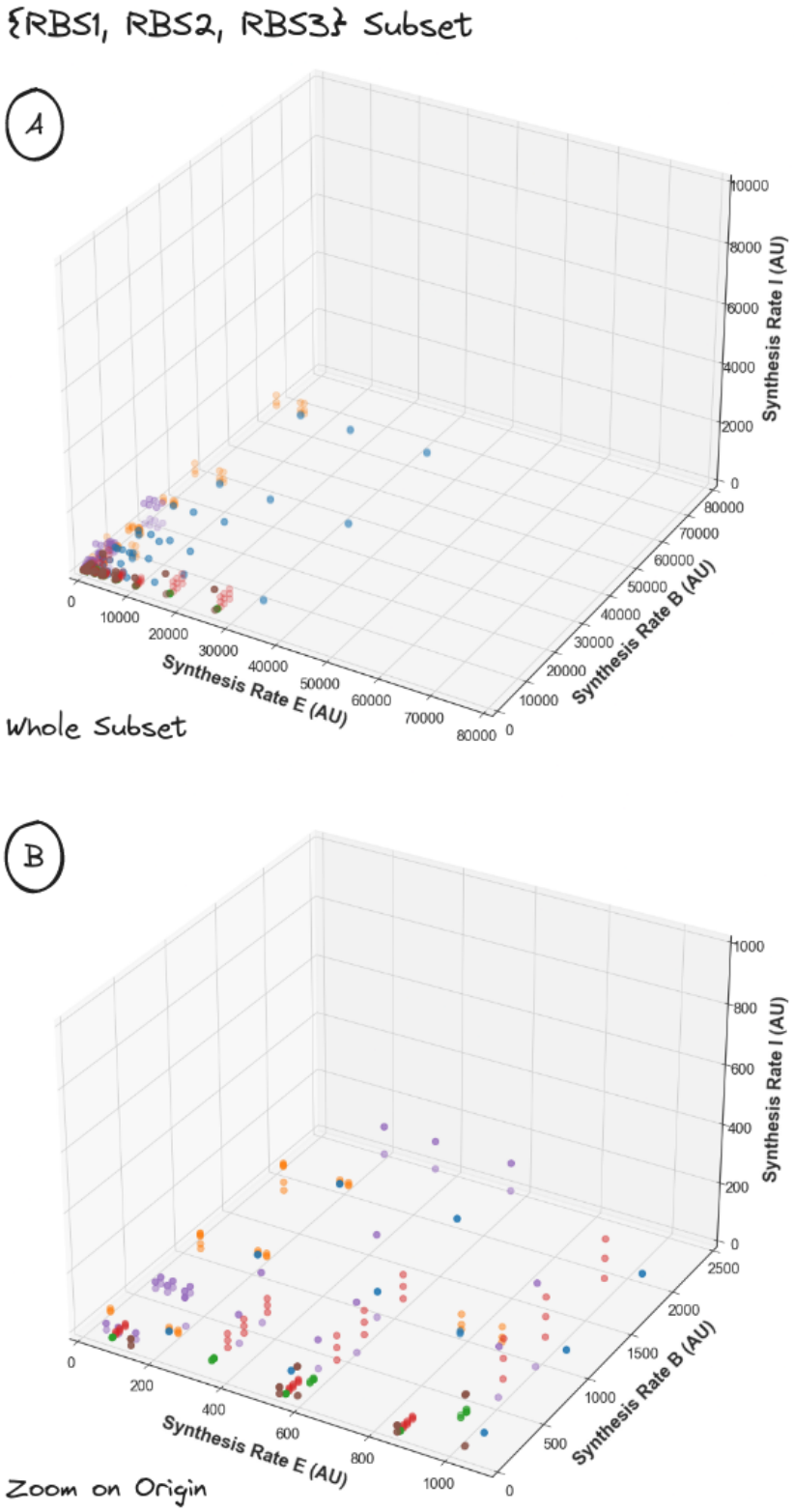


Figure 3-5. Design space generated from the entire RBS Biolegio library and the subset {RBS1, RBS2, RBS3}. Constructs are colour-coded according to their gene-order/pool. **(A)** The design space generated from the subset {RBS1, RBS, RBS3} only includes 810 constructs and covers a region

closer to the origin. **(B)** A zoom onto the origin shows dense sampling of this very weak region.

4.2 Screening results

In phase 2 of the PASIV workflow, viable colonies were picked to be sequenced and cultured for a titration assay. Colony picking was conducted irrespective of the colour of the colony. Only a few colonies were present – confirming previous attempts at cultivating the *E. coli* for lycopene production and how difficult it is. Overall, 99 colonies (from Pools 1-6) were isolated (Table 3-1). Pools 3 (EBI; 37/99) and 4 (EIB; 54/99) were overwhelmingly represented among these isolates. Pools 1,2 and 6 yielded very few viable colonies (all were picked), while Pool 5 yielded none.

Pool	Gene Order	Isolate Multiplicity	Construct Multiplicity
Pool 1	B-E-I	1	1
Pool 2	B-I-E	1	0
Pool 3	E-B-I	37	9
Pool 4	E-I-B	54	10
Pool 5	I-B-E	0	0
Pool 6	I-E-B	6	3

Table 3-1. Distribution of the Isolate Colonies (After Colony Picking) and after construct-matching. The predominant pools, 3 and 4, have been coloured in blue and purple, respectively. The cell in red indicates a mismatch identified by cMatch.

All picked colonies were sent for Sanger sequencing (see Methods for details on the primers). Returned sequences were then analysed with cMatch to identify each of the components of the constructs (promoters, RBS, CDS), and reconstruct them. Eighty-seven of these constructs were successfully sequenced. Seventy-three out of the sequenced 87 isolates could be identified with sufficient reliability – that is they met the minimum quality requirements for construct matching (each component could be matched with a sufficiently high score). Most of the identified isolates proved to be duplicates and, in the end, only 23 constructs were unique. Such a low number was anticipated, since preliminary experimental results hinted at a very limited viable region for the design space, and toxicity and burden effects having very noticeable effects on visible outcomes (cell viability and lycopene production).

Table 3-2 lists these 23 distinct constructs, their components, and their multiplicity among the 73 reconstructed sequences. Instead of listing construct components in template order (promoter, RBS in position 1, CDS in position 1 etc...), they are ordered in (promoter,

RBS_E, RBS_B_RBS_I, gene order) order where RBS_E (respectively RBS_B, RBS_I) is the RBS located in front of the CDS for crtE (respectively crtB, crtI). This was done to make the results easier to compare across gene orders and also to match the idea of modelling coordinates, which is independent of the gene order. Results indicate that:

- K137085 (the weakest promoter) is the most commonly represented promoter (12/23) and is used in all the constructs with high multiplicity (4 or more; all coloured in red). This is consistent with a scenario where toxicity is the limiting factor.
- Conversely, apFab32 (the strongest promoter) is not represented at all – again consistent with toxicity as a limiting factor.
- The RBS placed before crtE is almost always the weakest RBS in context RBS1 (22/23, all coloured in green), except in one case (RBS2, the second weakest one, in yellow) – again consistent with toxicity as a limiting factor.
- No construct uses the same RBS in all three positions at the same time. This observation contradicts projections by the EFM calculator (Jack et al., 2015; EFM Calculator, 2021) that repetitions of the RBS only has a minor effect on the overall RIP score (adding an extra 0.1), whereas the choice of promoter has a significantly larger effect (K137085 and its ATATATATATATATAT sequence being estimated to have add 50 points to the RIP score). A possible explanation is that the weight used in the EFM calculator corresponds to much more benign context than the toxic context our cells are under, and therefore under-estimates the possibility of recombination events when all RBS are identical.
- Pools 3 (EBI; 9/23; coloured in purple) and 4 (EIB; 10/23; in blue) remain the most dominant pools (Table 3-1 and Table 3-2). They also contain all the higher-multiplicity constructs (hence the relative drop in prevalence compared to the colony results) – hinting at a growth and/or production advantage for some constructs in these pools.

Name	Promoter	RBS E	RBS B	RBS I	Gene Order	Multiplicity
C_01	J23101	RBS1	RBS1	RBS2	I-E-B	1
C_02	J23101	RBS1	RBS1	RBS3	I-E-B	3
C_03	J23101	RBS1	RBS2	RBS2	E-I-B	1
C_04	J23106	RBS1	RBS1	RBS2	E-B-I	3
C_05	J23106	RBS1	RBS2	RBS1	E-B-I	1
C_06	J23106	RBS1	RBS2	RBS2	E-B-I	2
C_07	J23106	RBS1	RBS2	RBS3	E-B-I	1
C_08	J23106	RBS2	RBS3	RBS1	E-I-B	2
C_09	J23108	RBS1	RBS1	RBS2	E-B-I	1
C_10	J23108	RBS1	RBS2	RBS2	E-I-B	1
C_11	J23108	RBS1	RBS3	RBS2	E-B-I	2
C_12	K137085	RBS1	RBS1	RBS2	E-I-B	9
C_13	K137085	RBS1	RBS1	RBS2	I-E-B	1
C_14	K137085	RBS1	RBS1	RBS3	E-I-B	6
C_15	K137085	RBS1	RBS2	RBS2	B-E-I	1
C_16	K137085	RBS1	RBS2	RBS2	E-B-I	4
C_17	K137085	RBS1	RBS2	RBS2	E-I-B	9
C_18	K137085	RBS1	RBS2	RBS3	E-B-I	2
C_19	K137085	RBS1	RBS2	RBS3	E-I-B	12
C_20	K137085	RBS1	RBS3	RBS1	E-I-B	1
C_21	K137085	RBS1	RBS3	RBS2	E-I-B	5
C_22	K137085	RBS1	RBS3	RBS3	E-B-I	2
C_23	K137085	RBS1	RBS3	RBS3	E-I-B	3

Table 3-2 – Results of the sequence identification step. Constructs of higher multiplicity (4 and above) are coloured in red. The predominant pools, 3 and 4, are coloured in blue and purple, respectively. Finally, the RBS placed before crtE are also colour-coded (green for RBS1, yellow for RBS2) to illustrate the predominance of RBS1 in that position.

4.3 Location and multiplicity

When plotted in E-B-I (synthesis) coordinates, the identified isolates were all found to cluster close to the origin (Figure 3-6A). All constructs inherited the low-I distribution from the design space. Comparison with the whole design space (Figure 3-6B) illustrates how concentrated the constructs are close to the origin, compared to the available space sampled by the design space.

When construct multiplicity (as defined in Section 3.1) is added to the plots (Figure 3-7A), it can be seen that all higher-multiplicity constructs are not only located close to the origin – but, also, coincide with the lowest values for E-synthesis (Figure 3-7B) – illustrating results in Table 3-1, which show these constructs used the weakest promoter and the weakest RBS in front of E. These findings are consistent with the toxicity explanation, where cells die due to lycopene-induced toxicity (accumulation in the membrane), and therefore higher synthesis-levels of enzymes lead to higher lycopene production and accumulation in the membrane. In the E-B-I synthesis coordinates, all isolates were located in a box [0,2500] x [0,1500] x [0,200]. Highest-multiplicity isolates (exhibiting a growth advantage) were located in a smaller box [0,100] x [0,200] x [0,25].

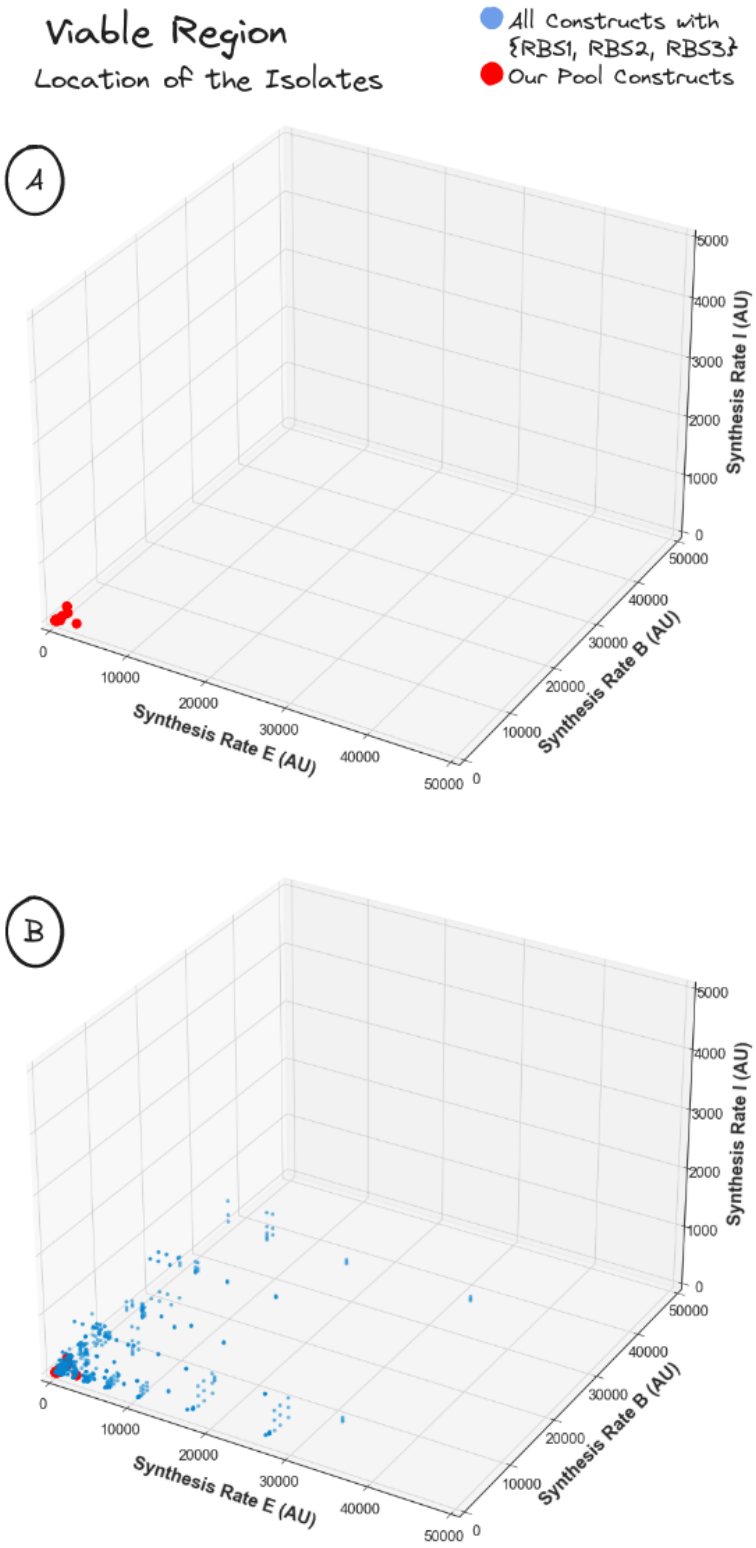


Figure 3-6 – Location of the identified isolates in the E-B-I coordinate system – Viable Region. **(A)** Identified isolates (red dots) cluster very close to the origin – **(B)** and only lie in a small portion of the available design space (blue dots).

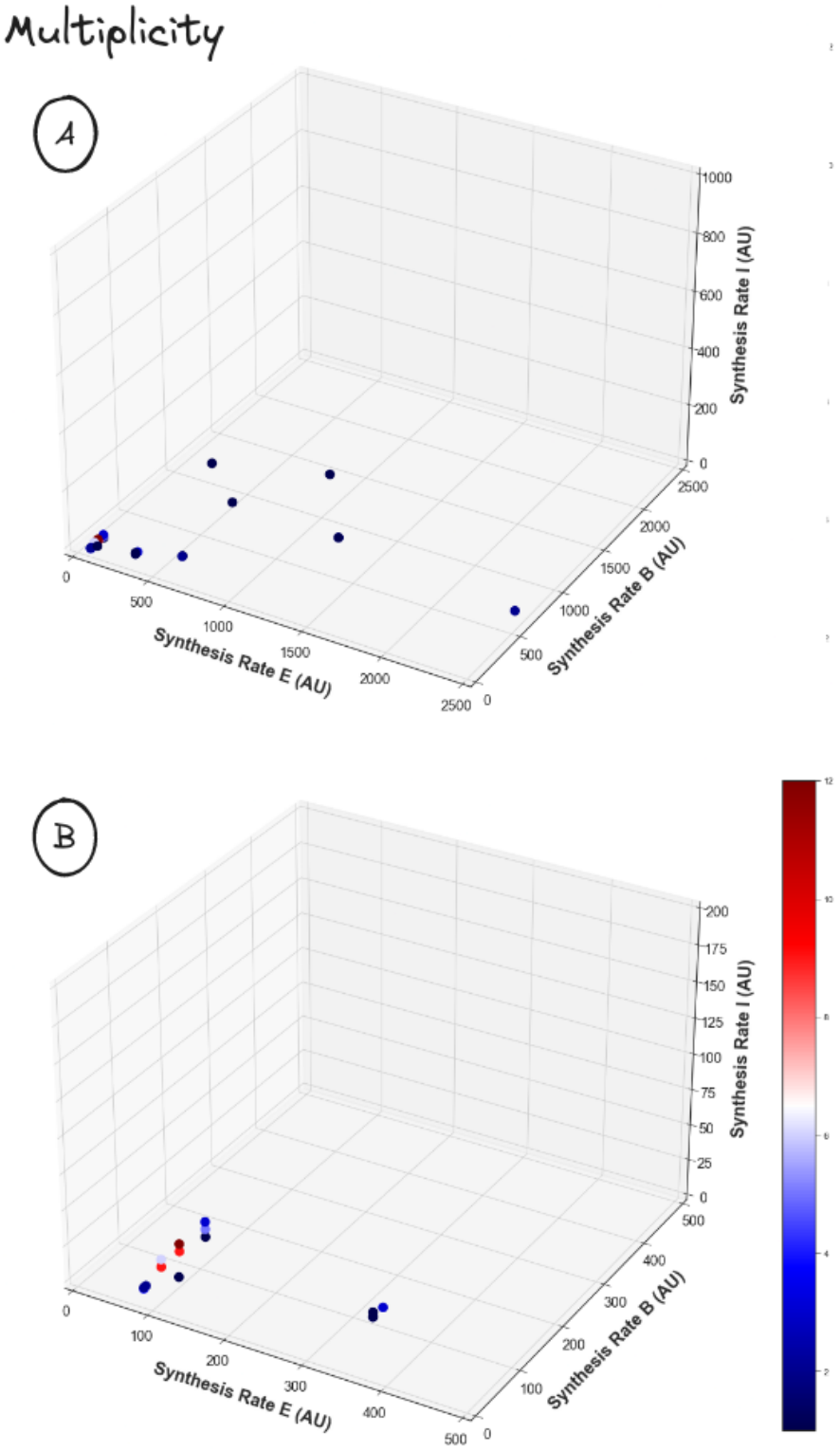


Figure 3-7 – Location of the identified isolates in the E-B-I coordinate system – Multiplicity. (A) The 3D plot shows that constructs with the highest multiplicity coincide with the lowest available value

of E. **(B)** The same region of space zoomed in.

4.4 Data analysis of the titration assay of lycopene concentration

Lycopene production and growth data were acquired as per the protocols described in the Methods (see Appendices C3 - Methods): Optical Density at 600nm (standard optical density) and 471nm. Optical Density at 600nm (OD600) was then converted into dry cell weight (DCW) (see Appendices C3 - Methods) and absorbance at 471nm was converted to lycopene concentration with the help of a standard curve. Corresponding yields (in mg/g of dry cell weight), were finally computed, as it is the most commonly used metric for metabolic performance.

When using PASIV, biological replicates are of two sorts. For each isolated colony that is picked up, several biological repeats originating from that colony are run as part of the titration assay (four in the exemplar study) – they are referred to as 'Type 1' in this section. Only after identification of their genetic material can the isolates, and associated repeats, be grouped according to their genetic material (the construct) – Type 2 repeats. For type 1 repeats, the biological samples are from the same colony, and it is expected that measurements can be safely aggregated and that the statistics derived from these operations will be indicative of the performance of the colony (and the accuracy of the measuring process). For type 2 repeats however, variations will also be indicative of the differences in terms of metabolic state for colonies with identical genetic material. This distinction between type 1 and 2 is especially relevant in the case of a constitutive design, placing significant duress on the cells (especially for the higher producing constructs), and where assays cannot be synchronised as they would be with an inducible design – and thus liable to result in significantly different metabolic states for colonies containing identical genetic material.

Table 3-3 lists the results for the yield for all the 23 distinct constructs. Three statistics (‘Mean Yield’, ‘Max Yield’, ‘New Yield’) are used to identify the best performers in terms of yield: mean of all the replicates associated with the construct, maximum over these replicates and a more robust statistic presented below to compensate for the possible variance in metabolic states. Both mean yield and maximum yield, identify a similar list of strong performers – the discrepancies between both lists being caused by the multiplicity of the construct. Average yields for the constructs with highest multiplicity (hence with the highest number of replicates) were often significantly lower than their max value (by as much as 80%) – indicating that genotypically identical constructs in the pool were liable to feature in disparate metabolic states. At the other end of the spectrum, construct C_01 with a multiplicity of only one returned similar values of the mean and max. The best performers

are, in all but two cases, among the constructs with higher multiplicity (located in the cluster closest to the origin). The best performer, C_22 (K137085, RBS1, RBS3, RBS3) is also located in the cluster. This was surprising, as constructs in the cluster (with some fitness advantage) were not expected to be strong producers (an activity imposing duress). The best yields corresponded to the strongest RBS (RBS2 and RBS3) placed in front of the final enzyme of the pathway (CrtI) – hinting that better yields could be found with stronger RBS in front of crtI and that our original choice of RBS was too restrictive (stronger RBS placed in front of crtI should be tolerated and lead to better yields).

Id	Mean Yield	Max Yield	New Yield	Multiplicity
C_01	26.10	31.85	26.10	1
C_02	58.03	80.69	68.92	3
C_03	1.29	2.42	1.29	1
C_04	4.83	9.66	9.00	3
C_05	0.94	1.31	0.94	1
C_06	9.86	20.08	17.80	2
C_07	7.63	8.26	7.63	1
C_08	7.19	34.78	11.52	2
C_09	2.84	4.49	2.84	1
C_10	8.25	11.29	8.25	1
C_11	9.17	25.31	17.62	2
C_12	22.73	86.07	57.09	9
C_13	3.44	4.39	3.44	1
C_14	17.54	40.57	30.64	6
C_15	20.85	22.82	20.85	1
C_16	25.05	65.71	38.78	4
C_17	16.54	67.63	58.40	9
C_18	9.49	11.52	10.54	2
C_19	24.48	73.78	65.71	12
C_20	5.80	6.58	5.80	1
C_21	23.23	55.33	44.69	5
C_22	63.18	108.62	80.86	2
C_23	15.38	38.28	30.97	3

Table 3-3 – Yield results (in mg/g of dry cell weight) **for all unique constructs drawn with the pooled approach.** Constructs of higher multiplicity (4 and above) are coloured in red. Best performers in terms of yield are coloured in purple (max yield) and blue (mean yield of the replicates). A new measure of yield is denoted as 'New Yield' – high performers according to this metric are coloured in magenta.

To decouple inter-isolate and inter-construct variation, the ‘New Yield’ statistic was added. The ‘New Yield’ statistic is computed in two stages. First, for all isolates, the average is computed for all features (DCW, lycopene concentration and yield). These average statistics are considered reliable indicators of the behaviour of the isolate. Then for constructs of multiplicity larger than one, the isolate the most representative of the potential of a construct is identified. The averaged features of that isolate are then assigned to the construct. In the present case, the isolate with the largest mean yield is selected as we are interested in lycopene production (mean lycopene concentration could also be used). Table 3-3 shows the value of this ‘New Yield’ statistic (alongside the mean and max) for all constructs. The new statistic does not penalise constructs with higher multiplicity as much as the mean of all repeats. Conversely, because it is based on some averaging, the ‘New Yield’ statistic does not reward outliers as much as the maximum.

The ‘New Yield’ statistic was used to separate the constructs into high and low producers. K-means clustering was used with all 23 constructs and two target clusters and using all three dimensions (yield, concentration and OD/DCW). (See Appendices C3-Figure-S18 and C3-Figure-S19).

Figure 3-8 shows the location of both clusters in EBI coordinates. Red dots are used for the strong performers; blue dots for weak ones. Red dots are all located in the cluster closest to the origin and associated with higher multiplicity – confirming that the cluster and nearby region should be investigated further if one wishes to find good, reliable performers. A black dot is used to label construct C_02, which has high yield but is also associated with low dry cell weight. As the construct is on its own, surrounded by blue dots (low producers), it either corresponds to an outlier (unlikely since it has a multiplicity of 3 in the pool) or is located in a region of the metabolic landscape (high yield, low OD/DCW) that is difficult to access via a pooled approach.

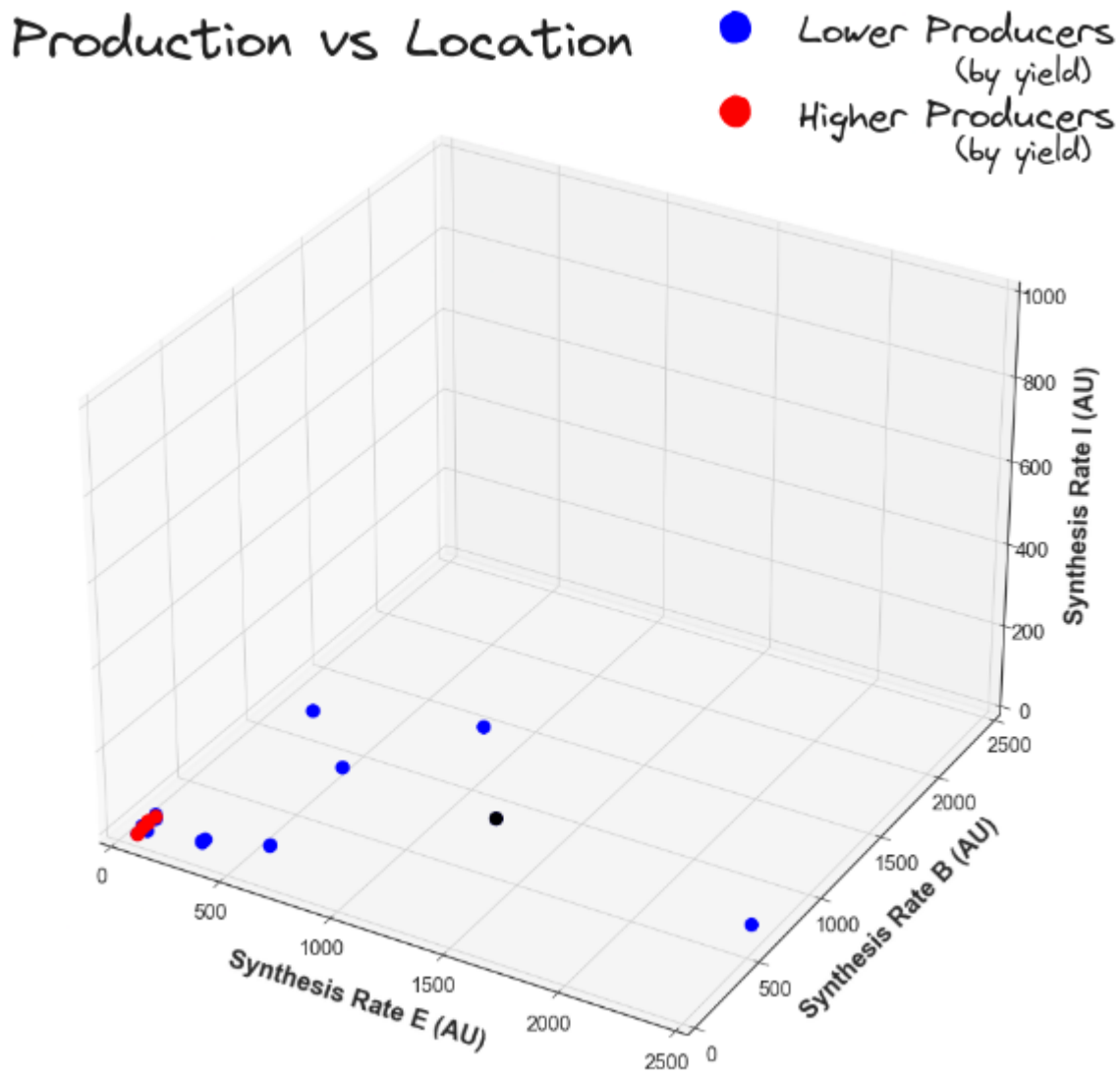


Figure 3-8 – Separating the high and low producers. Location of the high and low-producer clusters in EBI coordinates. Red dots are used for the strong performers; blue dots for weak ones. Strong performers are all located in the cluster closest to the origin and associated with higher multiplicity.

4.5 Identification from the growth regions in the design space of a region for further investigation

All viable constructs were located in the [0,2500]x[0,1500]x[0,200] box (in E-B-I coordinates), while the best performers were in a much smaller box [0,100]x[0,200]x[0,25] (Figure 3-9). The fact that higher producers were very heavily clustered inside the region, while lower producers were spread further from the origin, coupled with our expectations regarding the effect of toxicity and burden on production, and the results of the viability assay (very few

viable colonies) convinced us not to look for further regions of interest in the lycopene case. Instead, it was decided to concentrate on the region containing the most promising isolates, as it clearly was the region to favour for the screening phase. In parallel, the larger region containing all isolates was to be retained for a possible separate screening, as it had been much more sparsely sampled.

Another important feature of the construct design is gene order. Although a safe approach would be to consider all possible orders, using fewer gene orders would simplify construction workflows. Practically, it was decided it would be advantageous to only use pools 3, 4 and 6 (EIB, EBI, and IEB) since Pools 3 and 4 yield the most constructs and some of the strongest performers, Pool 6 yields fewer constructs but the best performers, other pools yield very few constructs and weaker ones than can be found in other pools.

Finally, as a way to account for possible errors in the process, and to make sure that as little as possible of the promising higher-producing region is left out, the region was extended along the B and E dimensions, the maximum value was doubled. Yield data for the stronger producers hint that it is advantageous to have stronger RBS than RBS3 in front of crtI, the maximum along the I dimension was therefore substantially increased, from 25 to 100.

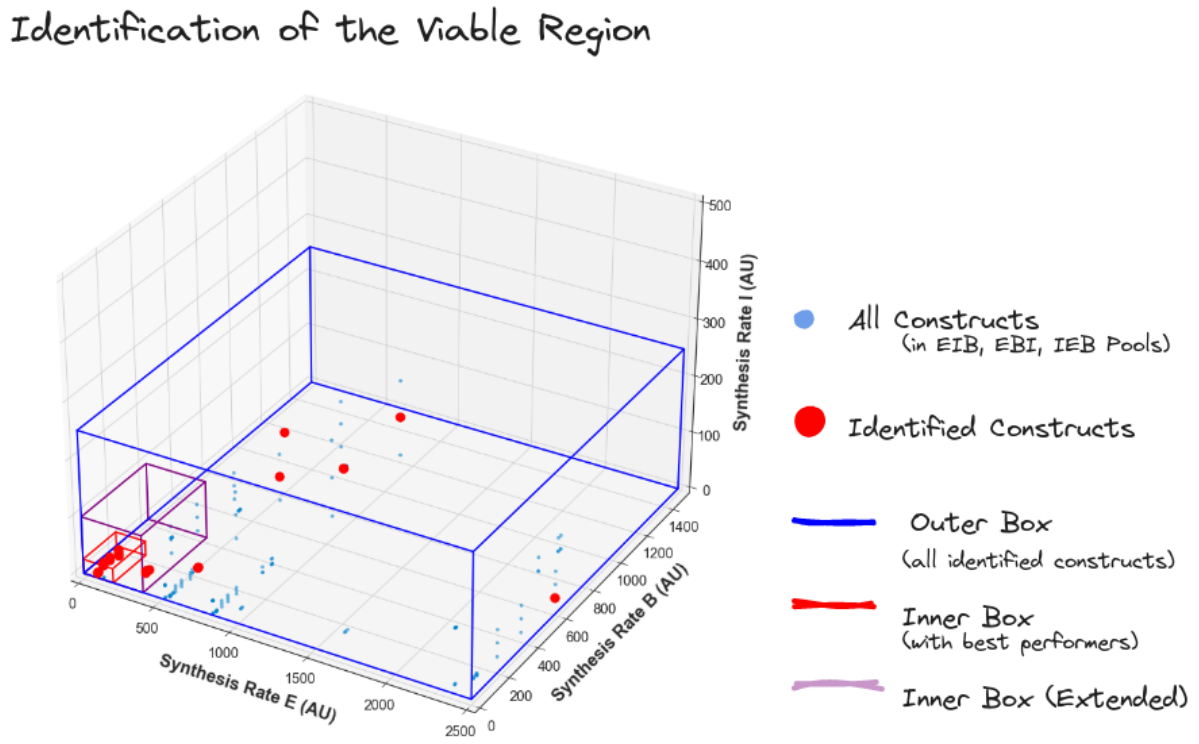


Figure 3-9 – Concluding the scoping study. The higher-producing constructs feature in the red box – close to the origin. To compensate for possible errors in the process, the region is expanded – most noticeably in the I direction – yielding the purple box, which is considered the most promising and should be the subject of further investigation.

Analysis of the data has identified two overlapping regions of the design space: the blue box (see Figure 3-9) containing all the identified constructs, and a red box containing the higher-producing region (in red) (see Figure 3-9). To compensate for possible errors in the process, it was decided to expand the higher-producing region (in red) – most noticeably in the I direction; the resulting region (in purple) (see Figure 3-9) is considered the most promising and should be the subject of further investigation.

The 3D visualisation is appropriate in the case of displaying points in a synthesis rate coordinate system for the three enzymes of the lycopene operon but other designs could lead to having more than three factors to vary. In this case a three dimensional representation would necessitate a dimensionality reduction technique such as principal component analysis (PCA) or linear discriminant analysis (LDA).
The importance of having an accurate measurement method is emphasised as variation due to measurement errors would compromise the aggregation of data.

5. cMatch (a Construct-Matching Software)

5.1 Need for cMatch

The development of cMatch was born out of the practical experience gathered during the development of the lycopene application (Chapter 2 Section 5) and the need to identify constructs made by the pooled-approach used in Phase 1 of PASIV (see Section 3.1). cMatch was specifically developed to exploit and tackle a set of five features and constraints that are typical of large-scale combinatorial spaces and metabolic engineering. These are as follows.

1. **Tackling the problem requires automation and programmatic access.**
   Construct identification quickly becomes a bottleneck when too many sequences need to be verified – as is the case with large design spaces. The standard way of identifying a genetic construct and its different components involves loading the results of a Sanger sequencing onto a DNA editing software, eg. Benchling and painstakingly searching for the possible components one by one. This quasi-manual process is extremely time consuming (up to several hours for a few sequences) and error prone. It also scales very poorly with the number of constructs and with the complexity of the constructs. Differentially identifying close constructs (e.g using closest RBS) is extremely challenging. The process also becomes vastly more cumbersome when a comparison is made against not one, but several members of the same design space. Finally, there is also no way to assess the reliability of the process – except by repeating it.

2. **The synthetic constructs share a well-defined structure,**
   thanks to their modular design. Synthetic constructs generated from a modular design share very specific features:the type and relative order of their components are known. Their components are drawn from known lists of elements. The interfacing of their components is also well-defined. Most designs will use non-overlapping components; when this is not the case the overlaps can be strictly defined as they are parts of the design.
3. **High level of precision is required in the identification of some components**
   because, the biological functions encoded in the constructs have sequences spanning several orders of magnitude (CDS are one kb or more, promoters around a 100bp, UTRs around 50 bp or less, while degradation tags are often less than 10 bp long). Some very short regulatory components such as RBS can be rationally designed to span specified ranges of strength (Jeschek et al., 2016), but minor changes in their sequence will have a significant effect on their strengths (Salis et al., 2009). These components must therefore be identified (sequence and location) with the utmost precision.
4. **High level of precision is also required for components interfaces.**
   Context is the set of interrelated factors that modulate the operation of biological processes – these factors are either composition-specific, environment-specific or host-specific (Cardinale and Arkin, 2012). Environment and host-specificity are irrelevant to the problem addressed here. Composition specificity is crucial on the other hand, and takes many forms. For instance, in an operon, gene order influences effective transcription rates (Nishizaki et al., 2007). Constructs are also at risk of several forms of genetic instability, most commonly deletions caused by homologous recombination and indels – risks that can be estimated with the EFM calculator (Jack et al., 2015). Of all composition factors, the interfacing of adjacent regions is the most important. Expression of functional components is indeed affected by short adjacent sequences (Espah Borujeni et al., 2014; Taylor et al., 2019; Tietze and Lale, 2021). A calculator such as the RBS calculator (Salis et al., 2009) now requires 35 bp upstream and 60 downstream to estimate the translation rate of an RBS.
5. **Sequencing data may be short because of the limits of the sequencing methods** (i.e. Sanger sequencing), but possibly spread over several sequences.

Typical constructs will only include a small number of genes (less than 20, often far fewer as pathways will be refactored over several plasmids if they are too long). Sequences to analyse will therefore be several kbp-long or less (considerably less than the length of

genomes). Practically, this implies that cMatch can rely on one of the most precise matching algorithms, instead of a popular algorithm such as BLAST (Altschul et al., 1990) which is better suited to genome length sequences. Finally, sequencing data for a construct will consist of one sequence spanning the whole construct, or several shorter sequences covering complementary fragments of the construct.

5.2 Construct-matching with cMatch

To exploit the structures of the constructs, cMatch operates at the functional (component) level. Since synthetic designs encode a set of biological functions, and add a set of positional constraints, cMatch searches for components encoding given functions, and checks if their combination is admissible (Figure 3-10A). We call this operation 'construct-matching'. It is a more thorough operation than simple sequence-matching or alignment, as it aims to answer a set of questions regarding the modular structure of the construct, as well as how well each component matches the data, including

- Structure matching: Does the construct match the design?
    - Are all intended components present ?
    - Does their order match the construct design ?
- Quality of the matching: How close is the matching?
- Are there any components affected by mutations? If so, to what extent ?
- Are there any insertions and/or deletions ?

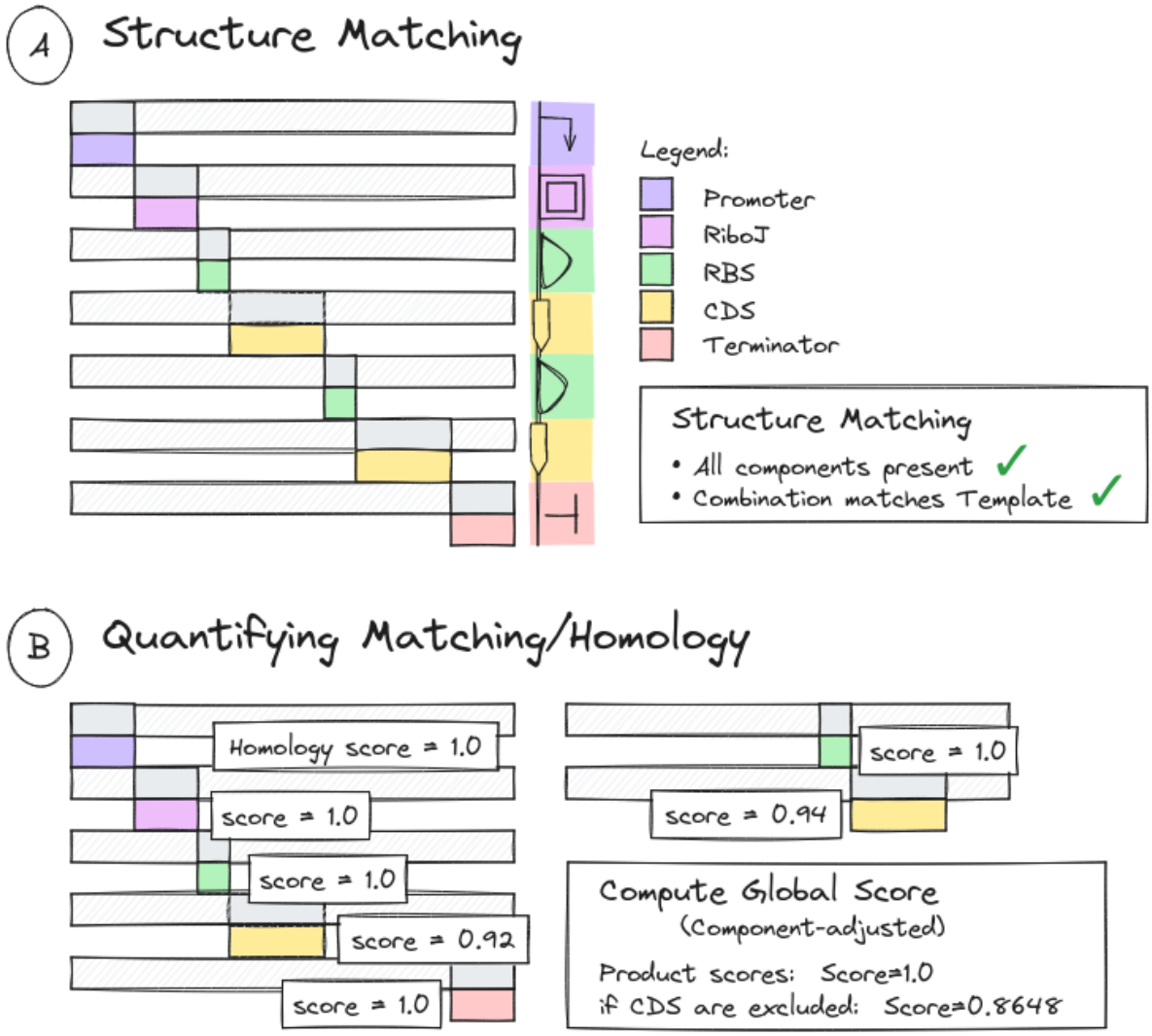


Figure 3-10 – Construct-matching in action (for a two-gene operon design). Construct-matching performs two complementary tasks. **(A) Structure-matching:** this step and checks whether the data correspond to an admissible construct, that is contains all the specified components in a combination that matches the design template. **(B) Quantification of the matching:** each component has its own matching/homology score, from which a global score is derived.

cMatch gives a standardised matching score for each component. These scores and their weighted average are then used to automate the decision-making and quantify the reliability of the prediction through a set of global matching scores (Figure 3-10B).

Separating the components in the construct helps address several of the previously listed challenges.

- All functional modules are dealt with in parallel. Homology (matching) scores can be standardised against their lengths, yielding an immediate insight into where the discrepancies between data and ideal sequences are located. This is important when dealing with short, regulatory components, and their adjacent regions.

- It is easy to generate global matching metrics from the individual component scores. Equal weight can be given to the components and the geometric mean can be calculated. Conversely, some components, such as CDS, may be omitted from the global scores if the user decides mutations in these regions are of little concern.

cMatch is built on top of the thoroughly validated and widely used (inudstry standard) Smith-Waterman algorithm (SW) (Needleman and Wunsch, 1970; Smith and Waterman, 1981). Majority of genome alignment tools use SW to determine similarity between the read and its global positions in the reference genome and SW remains the most popular algorithm (Alser et al., 2021). The Smith-Waterman algorithm is a search algorithm based on an earlier model named Needleman-Wunsch after its creators. The SW algorithm implements a technique called dynamic programming which takes alignments of arbitrary length and location in any sequence and determines if an optimal alignment can be found. The choice was born out of the need for a high level of precision in the identification of some components and their interfaces.

Our algorithms rely on the ability to perform a sequence alignment between the part to match and the sequenced construct. Sequence alignment can be divided into global alignments and local alignments – typically using the Needleman-Wunsch (NW) algorithm is used for global alignments while local alignments use the Smith-Waterman (SW) algorithm. Global alignments align complete sequences and local alignments identify only the most similar segments. Global alignments are usually done for comparing homologous genes like comparing two genes with same function or comparing two proteins with similar function. In general, global alignment algorithms (NW) return more accurate alignments for sequences of similar lengths and local alignment algorithms (SW) are better at identifying similar regions within sequences – when the sequences are not related over their entire length.

The Needleman-Wunsch algorithm (global alignment) could have been considered for a sequence to sequence matching in our ‘Sequence matching algorithm CM_0’ (see Section 5.3.2) since it performs a function that is suitable for aligning two closely related sequences (the expected sequence to match should be equal to one of the possible constructs generated from the template (with a perfect homology score of 1.0 if no mutations or deletions occurred)). NW attempts to align the entire sequence (end to end alignment). A global alignment contains all letters from both the query and target sequences (this is the case for the brute-force approach in CM_0 (see Section 5.3.2), whereas SW (local alignment) aligns a substring of the query sequence to a substring of the target sequence and finds local regions with the highest level of similarity between the two sequences.

We considered the SW algorithm to be most appropriate for our component-matching algorithms CM_1 and its extension CM_2 (see Section 5.3.3 and Section 5.3.4) since

components are, by definition, a sub-part of the template. Components can be as large as the largest component of the template (usually the CDS) or as small as the smallest components (a dozen base-pairs for RBS – changing only one base-pair can change the translation rate of many orders of magnitude) and thus a local alignment was considered.

The implementation of cMatch has been written in the Python programming language and makes use of the Biopython (Chapman & Chang, 2000) python package and more particularly the pairwise2.py module. The align() function of pairwise2 calls optimised functions in cpairwise2module.c a python C extension module written in the C language and compiled (thus being a very fast implementation). Switching from the current Smith-Waterman algorithm (local alignment) used in cMatch to the Needleman-Wunsch algorithm (global alignment) can easily be done in cMatch implementation code by only changing a function call. No attempt of benchmarking and directly comparing the two algorithms was made.

These stringent requirements place cMatch in sharp contrast to applications such as barcoding (Woodruff et al., 2017), where short barcode sequences (with a space of possibilities in the billions) are used to identify constructs in a design space of several thousands/millions. Exact string comparison, as implemented in annotation tools such as Benchling's Auto Annotate, is the fastest form of matching but it is impractical for construct matching, as it does not offer any way to deal with mutations or indels (insertions/deletions), and more generally fails as soon as data differ from the expected sequences (without being able to assess to what degree they differ). The two standard algorithms in bioinformatics Smith-Waterman and BLAST (Altschul et al., 1990) were considered for cMatch. BLAST is very popular thanks to its speed and is the algorithm of choice to deal with long and genome-length sequences (Goujon et al., 2010; Donkor et al., 2014) stored in the many biological databases (Lakshmi and Ramyachitra, 2020). But it is not as precise as Smith-Waterman, which is designed to return the optimal local alignment (Wieds, 2007). Smith-Waterman is much slower and is costly in terms of computer power and resources. This concern was overlooked, as the sequences to be analysed and the corresponding constructs are several kbp-long or less (considerably less than the length of genomes), which remains tractable. The fact the algorithm is deterministic (unlike BLAST), which is important for reproducibility, and the availability of a highly validated biopython implementation (Chapman and Chang, 2000) were compounding factors for our choice.

cMatch relies on knowledge of the structure of the ideal construct from which annotation can be derived. Consequently, cMatch can also be used as an annotation tool – although this is not its main objective. Comparisons with the recent range of auto-annotation tools for plasmids and constructs of similar length remain instructive. Beside its choice of sequence-matching method, the component-matching phase of cMatch differs from tools such as

Benchling Auto-annotate and pLannotate (McGuffie and Barrick, 2021) in several ways. Unlike pLannotate, it does not use external databases as sources for the components. cMatch also offers full programmatic access and lets the user change the matching parameters. Only the sequence-annotate features offered by the recent SYNBICT (Synthetic Biology Curation Tools) (Roehner et al., 2021) project offer a similar level of flexibility (and also use pairwise matching).

But where cMatch differs from this set of tools in an important way is at the second stage. The method goes much further than sequence annotations, as it performs a set of construct quality control (QC) operations on top of component matching. It also offers ways to quantify the entire process (only SYNBICT defines a threshold to accept/reject annotations, but it does not utilise a template in any way). cMatch exploits modularity in order to offer comparisons not against one construct, but an entire design space.

5.3 cMatch algorithms for construct-matching

We will now present our solution to the construct-matching problem. Two construct-matching algorithms are presented. The first algorithm, named CM_0 ('Construct Matching 0'), is a sequence-based algorithm based on a brute force investigation of the design space (Section 5.3.2) – it is included for comparison purposes and to highlight the benefits of the component-matching approach adopted by the other algorithms. Finally, the construct-matching algorithm CM_1 (Section 5.3.3), based on a component matching strategy and step-by-step reconstruction is presented, as well as an extension (CM_2) to deal with multiple inputs.

5.3.1 Construct-matching

Construct-matching has two distinct input components:

- **A construct to be identified** (the 'target' of the analysis) by way of sequencing results. All sequences take value in the IUPAC dictionary (IUPAC Codes, n.d.).
- **A search space.** The constructs contained in the search space will be referred to as ‘candidates' – they are possible output values for the identity of the target construct. A template describing the modular structure of the construct (type of the components and relative order) and thus providing a set of structural constraints for the space of possibilities. The search space is specified by two search constraints (they are the parameters of the algorithm):
    - A template describing the modular structure of the construct (type of the components and relative order)
    - For each component of the template, a library of admissible elements –

providing a description of the combinatorial element of the problem

Figure 3-11 shows the search constraints for a two-gene construct example:

- **Template** (Figure 3-11A): The construct has 6 components (1 promoter, 2 RBS, 2 CDS and a terminator) arranged in an operon pattern.
- **Combinatorics** (Figure 3-11B): A promoter library of 5, 2 RBS libraries of 10, 2 CDS libraries of 1 and a Terminator library of 1 are used.

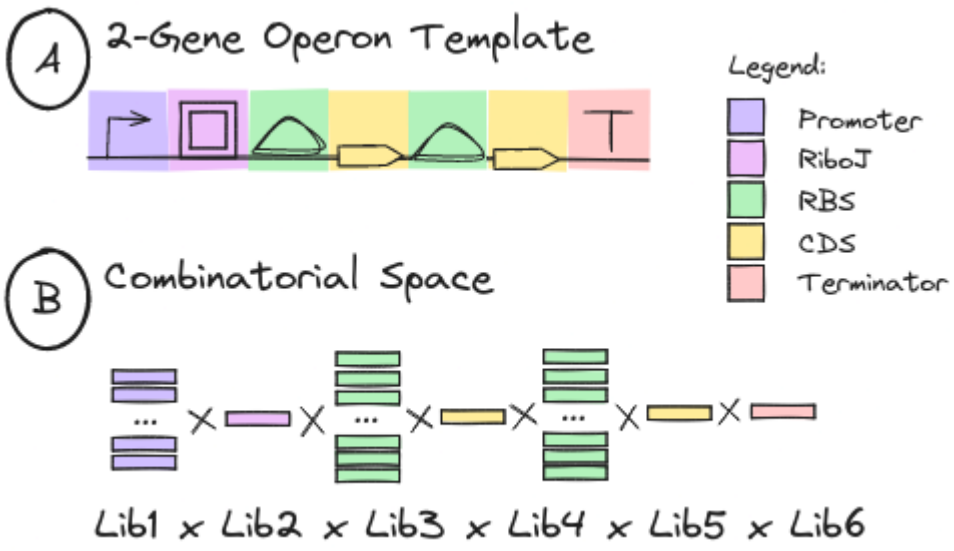


Figure 3-11 – The two sets of search constraints for a 2-gene operon. (A) The template specifies the types of the components and their positional arrangements. **(B)** The list of possible elements for each component yields the combinatorial element of the problem.

Even with such a simple example, the power of the template is apparent. There are 5 x 10 x 1 x 10 x 1 x 1 = 500 admissible combinations. If the relative order of the components is not imposed, there are 6! = 720 as many ways to arrange the components, i.e. 360000 (mostly non-functional). These numbers further increase exponentially if components can go missing or feature more often than once.

5.3.2 Sequence matching algorithm CM_0 (brute-force)

Given prior knowledge of a construct template, and its libraries of components, a simple solution to construct-matching consists of a purely combinatorial generation of all the candidates, followed by brute force matching of their sequences (as illustrated in Figure 3-12) :

1 **Combinatorial Step** (Figure 3-12A): generate all possible combinations in the search space from the template and libraries of components.

2 **Matching Step** (Figure 3-12B): for each generated sequence, compute the matching score to the target sequence with the Smith and Waterman algorithm.

3 **Output Step** (Figure 3-12C): return the combination(s) with the highest scores.

Such an approach does not seek to exploit the modularity of the constructs. Issues with such an approach are four-fold.

1 It is very rigid in the way it builds the candidate sequences, and makes it difficult to apply overlapping options.

2 The computational cost of a pure combinatorial approach fast becomes prohibitive, as the number of candidate constructs increases exponentially.

3 The performance of pairwise matching algorithms degrades fast as the sequences increase in length – due in no small part to their attempt at finding the best matches by inserting and deleting base pairs

4 The approach naturally gives more weight to the longer components like the CDS to the detriment of very short, yet important ones like RBSs, because it only computes a global matching/homology metric.

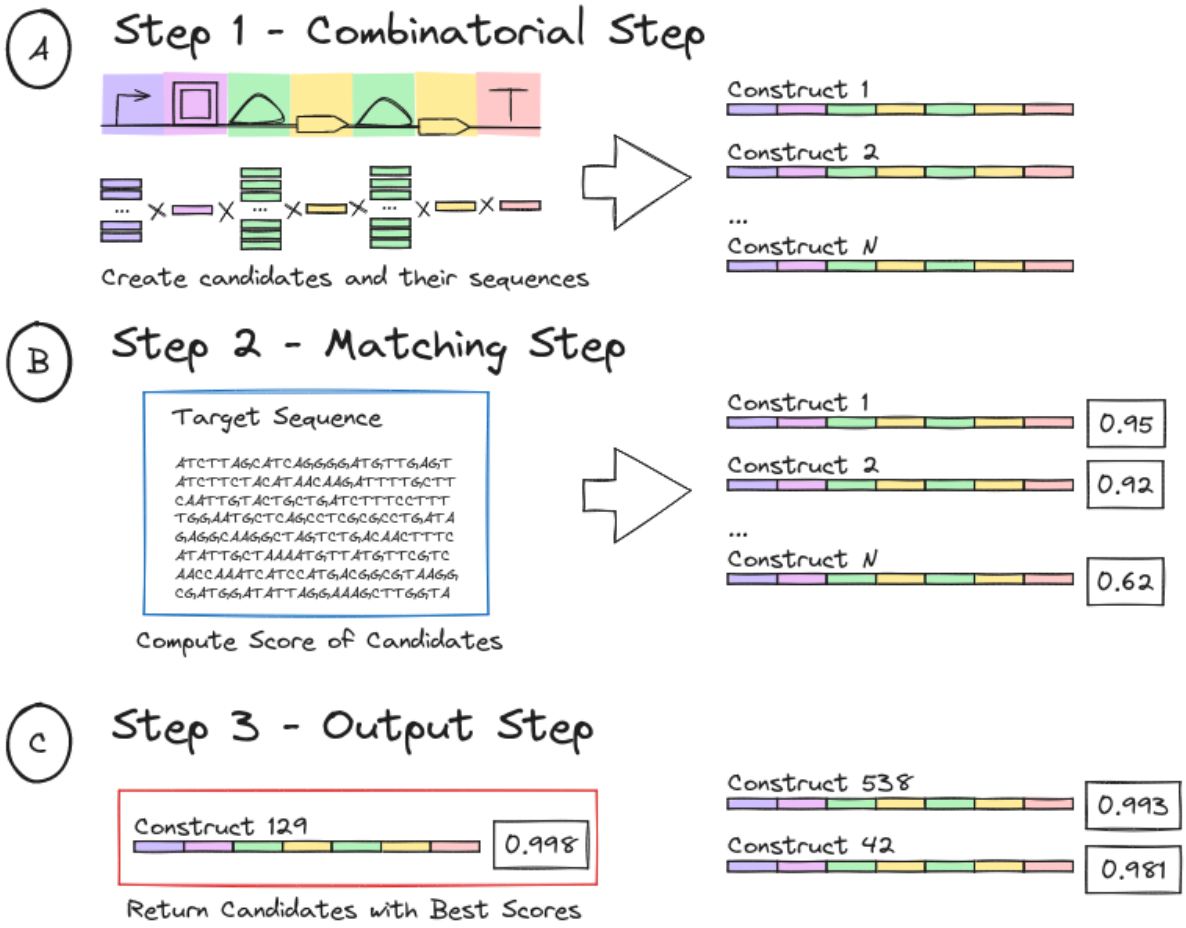


Figure 3-12 – Visual Representation of the Sequence-Matching Algorithm CM_0. The algorithm is made of three steps. **(A)** Step 1 – CM_0 generates the sequences for all possible constructs in the search space. **(B)** Step 2 – These candidate sequences are matched to the input sequence – returning for each a global matching score. **(C)** Step 3 – Constructs with the highest scores are returned.

5.3.3 Component-matching algorithm CM_1

We have developed a component-based algorithm (referred to as CM_1), which performs the operation we have called construct-matching. Unlike CM_0, CM_1 utilises the information contained in the template, and focuses on function-encoding components and their admissible combinations. CM_1 proceeds as follows (as illustrated in Figure 3-13) :

1. **Component Matching Step** (Figure 3-13A): This step looks for the components in the target – iterating over all libraries of components, to match each library element to a subsequence of the input sequence, and assigning a matching score (normalised by the length of the component) and a position to each of them in the process. Pruning follows: only matches with a score above a user-specified threshold are kept.
2. **Reconstruction and Pruning Step** (Figure 3-13B): Identified components are recombined into a list of admissible constructs. Rather than using a purely combinatorial approach, where all possible combinations are generated and then pruned against the template, an iterative reconstruction – based on dynamic programming has been implemented. A combinatorial approach proves extremely costly when the input contains repetitions and components are detected in several locations (See Section 3.1 for an example). Instead, the reconstruction applies positional constraints (e.g. 'component 3 must be located before component 4') as early as possible to prune out entire branches of the reconstruction tree.
3. **Output Step** (Figure 3-13C): At the end of the reconstruction, all remaining paths are assigned a global score (we use the geometric mean of the components scores).The combinations with the highest scores are returned

By performing component-matching, rather than mere sequence-matching, CM_1 returns a great deal more information than CM_0:

- identifies and locates all the components in the construct
- confirms their combination matches the template
- returns individual homology/matching scores for each of these components (a crucial source of information to identify where the mutations are) which allows us to deal with the components (however long or small) in parallel.

Working at component level opens up a set of strategies to limit computation times and resource utilisation. First it avoids matching long sequences to long sequences and can be parallelised. Second, it enables us to avoid combinatorial explosion, by only using the best matches from each library, and then pruning reconstruction paths as early as possible.

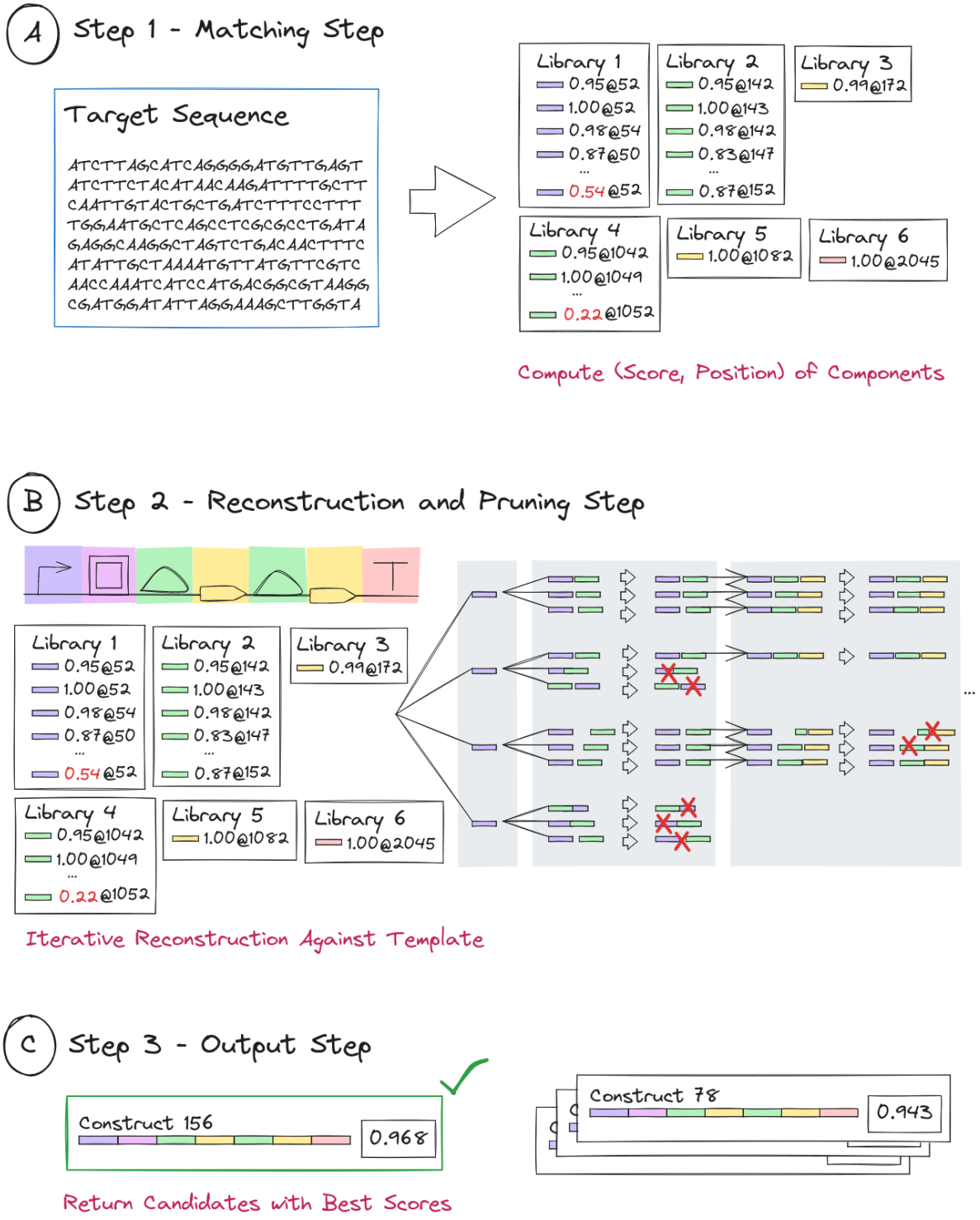


Figure 3-13 – Successive Steps of the Construct-Matching algorithm CM_1. (A) Step 1 – CM_1 first looks for all components in the target sequence. Poor matches (low score, in red) are pruned out at this early stage. **(B)** Step 2 – Components identified in Step 1 are recombined into a list of all admissible constructs. Dynamic reconstruction was implemented. A matching score is generated from the individual component matching scores for all possible constructs. **(C)** Step 3 – Constructs with the highest scores are returned.

5.3.4 Extended Component-matching Algorithm CM_2

CM_1 relies on a sequence being available that covers the whole construct, and that can therefore be used as an input. Such an assumption may not be met in general use. Using multiple sequences as inputs instead of a single one are very common with Sanger sequencing, as it only returns reads of a limited length (~500–700 bp). Using several sequences also gives flexibility regarding the portion of the construct to be identified. Such flexibility is desirable, as some components are more important than others for given applications, and their presence can be deduced thanks to prior QC checks.

In the lycopene application described in Section 3.2, reverse primers, originating in the CDS, were used that only covered a few hundred base pairs of the CDS – but gels were run to check the constructs were of the expected sizes.

Because our sequencing data from the lycopene exemplar application is done with Sanger sequencing (which returns reads of a limited length, ~500–700 bp, and our template sequence lengthens to 3.7kbp), we have extended CM_1 so it may support multiple sequences as inputs. The new algorithm (henceforth referred to as CM_2) utilises a split-apply-combine strategy (Wickham, 2011), where the first step of the process (the split phase) mirrors the sequencing of fragments of the construct. CM_2 proceeds as follows:

1 Partition Step: From the input data ($Sub_sequence_i$) and the global template, a set of sub-problems $\{(Sub_sequence_i, Sub_Template_i)\}_{i=1...N}$. are generated.

2 Sub-sequence Analysis: CM_1 is applied in parallel to all the sub-problems – which are also all cheaper to solve than the original problem due to their smaller size.

3 Recombination Step: The results of the individual analyses are combined into a final construct matching the original template

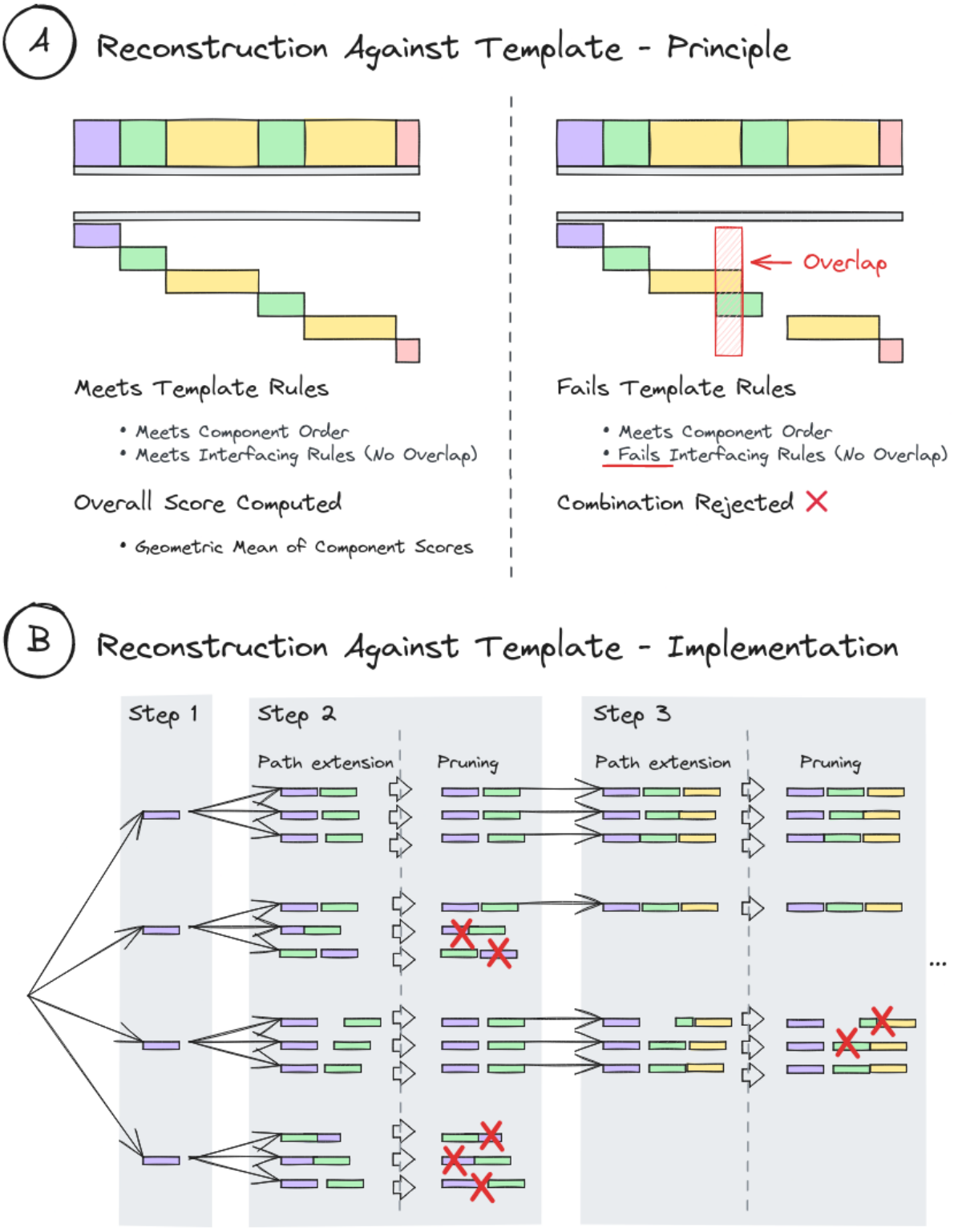


Figure 3-14 – Reconstruction against the template – Principle (A) and implementation (B). **(A)** On the left hand side, components are assembled in a way that matches the template (same order, admissible overlapping). On the right-hand side, the reconstruction is not rejected since two components overlap. **(B)** Only the first three steps of the reconstruction iteration are shown. Step 1: 4 possible promoters are added to an empty path. Step 2: 3 UTRs are added; 5 out of 12 paths are pruned. Step 3: a CDS is added and 2 paths are pruned – leaving only 5 possible paths after the first 3 steps.

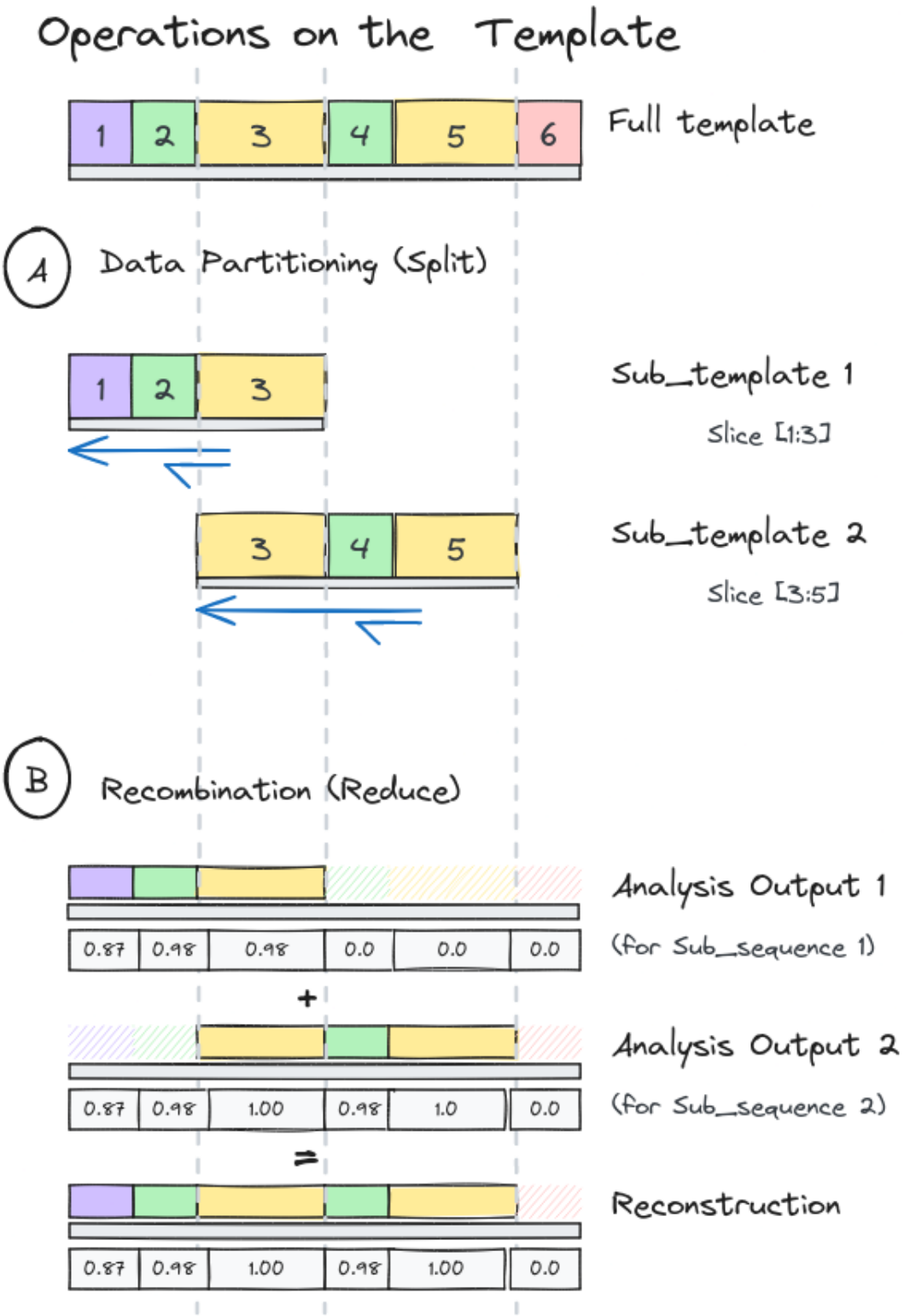


Figure 3-15 – Operations on the template: split and recombination. (A) Splitting: Two reverse primers (in blue) are used to generate two overlapping sub-sequences; the sub_templates are obtained by slicing the list containing the components of the template. **(B)** Recombination: The two sub-templates overlap in the first CDS region, yielding two estimates for that CDS. In the first step of the recombination the results are cast back into the complete template. Missing components are indicated by hatches (and a score of 0). Then the two objects are added – for 4 out of 6 components, addition is straightforward. In position 3 (the first CDS), 2 elements are possible and the element with the better score is retained, as it is deemed the best available estimation. For the last component (position 6, Terminator) the recombination returns an empty component – a direct consequence of the design of the primers.

6. Results (cMatch)

6.1 Synthetic constructs violacein

Construct-matching algorithms have been tested on synthetic data first and used the violacein pathway construct (Figure 3-16) comprising five enzymes. Violacein is a classic pathway which has been well characterised (Myeong et al., 2016). Violacein is a metabolite with industrial applications, it is used as a dye (vivid purple colour) or for medical applications (Durán et al., 2007).

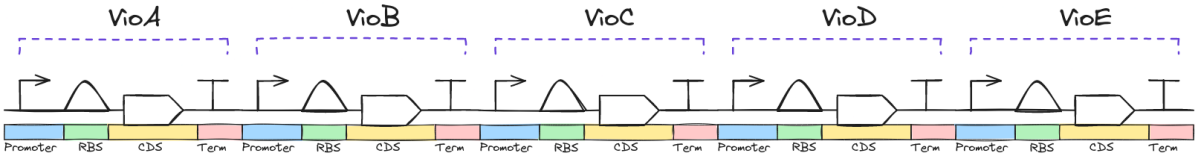


Figure 3-16 – The 5-Gene Violacein Pathway and its Implementation with a TU-Design. The design is made up of 5 transcription units (hence 20 components) in a set gene order. All units share the same promoter (BBa_J23101) and terminator (BBa_B0015) – providing a stiff test for the algorithm, due to the large number of potential repetitions.

The construct design is based on a simple TU (transcription unit) pattern (Figure 6):

- Each enzyme is encoded with its own TU. TUs share the same promoter (BBa_J23101) and terminator (BBa_B0015), both mainstays of the iGEM Registry.
- The design fixes the gene order to VioA-VioB-VioC-VioD-VioE.
- All UTRs can be varied, as per (Blazeck et al., 2012; Salis, Mirsky & Voigt, 2009). All positions use the same library of 5 RBS from iGEM registry (BBa_B0030 to BBa_B0033 and BBa_B0064, which only differ by a few base pairs).

This design comprises 20 non-overlapping components resulting in a design space of size 5^5=3125. The TU-based pattern is a challenging case for a construct-matching algorithm, due to the large number of repetitions in different locations (all constructs have 5 identical promoters, and 5 identical terminators). RBS libraries have been chosen to make reconstruction as difficult as possible (they are identical).

6.2 Testing algorithm CM_0

The predicted limitations of CM_0 can be illustrated with a few tests on the synthetic data. The first test with CM_0 was a 1-to-1 scenario; the construct Vio-0000 was used (a construct with 5 identical RBS B0030) then with 10 different constructs (Vio-0000 and 9 other constructs drawn at random). The second test is a 1-to-many scenario against different

targets (N=10 repeats for each target).

Performance of the CM_0 Algorithm

1-to-1 scenario with CM_0		
	Average Run time	**Global Matching score**
1 target (N=10)	36.60s ± 0.73s	1.0
10 targets (N=10)	37.89s ± 2.26s	1.0
1-to-many with CM_0		
	Total Run time	**Global matching score**
10 target (N=10)	370.72s ± 3.99s	1.0
100 targets (N=3)	56–58min	1.0

Table 3-1 – Performance of the CM_0 Algorithm. Results in the top part of the table for the 1-to-1 scenario show a perfect score of 1.0 – confirming that the Smith-Waterman algorithm correctly matches identical sequences. Results in the bottom part of the table for the 1-to-many scenario show a linear performance of CM_0.

CM_0 is expected to perform linearly with the number of candidates because each possible candidate (each possible combination in the search space) is being matched against the target sequence. Extrapolating from the results of the 1-to-many scenario, if the whole design space was used as search space (a 1-to-all scenario, against more than 3000 constructs), it would take more than 30 hours to identify a target construct with CM_0. In all instances, the correct construct was identified with a perfect score of 1.0, but several candidates returned scores above 0.99 since they varied only by a few base pairs from the target. Furthermore, a global match does not help locate the discrepancies between data and construct sequences and their significance.

6.3 Testing algorithm CM_1

CM_1 has two limiting steps: its matching step (affected by the length of the target sequences, the number and length of the elements in the component libraries), and its reconstruction step (affected by the repetitions in the construct and the matching threshold).

To test the reconstruction phase of the algorithm, tests were run with two threshold levels (0.99 and 0.75), and for two targets of opposite complexity:

- Vio-0007 – a construct with 5 different RBS (B0030-B0031-B0032-B0033-B0064)

- Vio-0000 – a construct with 5 identical RBS (all B0030)

Vio-0007 is the easiest case for reconstruction: after the matching step promoters and terminators are identified in 5 possible locations – but UTRs are identified in a single location. The reconstruction step needs to prune a space of 5^{10} possibilities down to one construct.

Vio-0000 is the most challenging case: after the matching step, its UTRs are identified at 5 possible locations. The reconstruction step needs to prune a space of 5^{15} possibilities. Brute force becomes intractable. We developed the step-by-step reconstruction for such cases and to prune combinations as soon as they break the template constraints.

To assess the influence of the number of elements in libraries – tests were run for the opposite cases of 1-to-1 (against the target itself) and 1-to-all (against all the 3125 members of the design space). Results for all tests are displayed in Table 3-1.

Performance of the CM_1 Algorithm

1-to-1 scenario with CM_1		
	Threshold = 0.99	**Threshold = 0.75**
Vio-0000	78.70s ± 6.61s	83.07s ± 7.26s
Vio-0007	78.80s ± 0.35s	79.12s ± 1.05s
1-to-all scenario with CM_1		
	Threshold = 0.99	**Threshold = 0.75**
Vio-0000	109.86s ± 15.10s	99.41 s ± 14.31s
Vio-0007	104.53s ± 4.47s	106.43 s ± 10.60s

Table 3-2 – Performance of the CM_1 Algorithm. Performance was assessed for constructs of opposite complexity and different reconstruction thresholds. All tests were repeated 10 times. Results in the top part of the table for the 1-to-1 scenario show computation times do not depend on the constructs and threshold. Results in the bottom part of the table for the 1-to-all scenario show similar results.

In the 1-to-1 case, with both thresholds and both targets, the reconstruction yielded only one combination (the target itself) with a perfect score of 1.0. Overall running time was larger with CM_1 than CM_0, and was independent of the target construct. The run time did not change against the threshold (as there was only one candidate for identification).

In the 1-to-all case, the reconstruction yielded (for both targets) only one possible reconstruction, the target itself, and did so with a perfect score of 1.0 for the higher

threshold. With the lower threshold, 2 UTRs passed the matching step for each template position, and the reconstruction yielded 32 possible reconstructions. Final selection then identified the correct construct (the target itself) with a perfect score of 1.0 among all the candidates. As expected, matching against the whole design space took longer than against one construct – between 100s and 110s. The difference corresponded to the time required to match all the components needed for a full comparison with the design space.

Split times (in the logs in Appendices C3 - cMatch-logs) confirm that, for both minimal and 1-to-all tests, reconstruction times were in the hundredths of a second and the component-matching step was the limiting step of the whole process. Variations between runs showed similar results as for CM_0 – times were either clustered around a main value close or spread in a tail. As with CM_0, these results were attributed to resource limitations in the system.

6.3 Testing algorithm CM_2

CM_2 has the same limiting factors as CM_1. All tests in this section were therefore identical to the tests run on CM_1. Results in both tests (Table 3-3) show CM_2 performed faster than for CM_1 by an order of magnitude. This was expected since it uses much shorter sequences as inputs, and the pairwise-matching algorithm performs faster with shorter sequences.

Peformance of the CM_2 Algorithm

1-to-1 scenario with CM_2		
	Threshold = 0.99	**Threshold = 0.75**
Vio-0000	10.32s ± 0.15s	10.22s ± 0.14s
Vio-0007	9.70s ± 0.08s	9.52s ± 0.33s
1-to-all scenario with CM_2		
	Threshold = 0.99	**Threshold = 0.75**
Vio-0000	10.71s ± 0.17 s	11.43 s ± 0.88s
Vio-0007	10.20s ± 0.03s	10.06 s ± 0.07s

Table 3.3 – Performance of the CM_2 Algorithm. Performance was assessed for constructs of opposite complexity and different reconstruction thresholds. Results in the top part of the table are for the 1-to-1 scenario, and show computation times are unaffected by the constructs and threshold. Results in the bottom part of the table are for the 1-to-all scenario, and also show computation times are unaffected by the constructs and threshold.

6.4 Final comparison of the algorithms

Finally, we wrote a test to compare the three algorithms. Since the reconstruction step of the algorithms is very efficient we built a test set of three constructs of increasing lengths and complexity: Vio-0000 ×1 (8.5kb, 5 TUs and 20 components); Vio-0000 ×2 (17kb, 10 TUs and 40 components), and finally Vio-0000 ×3 (25kb, 15TUs and 60 components). All three algorithms were then applied to all three synthetic constructs – for both scenarios. Input data for CM_2 were generated by slicing the constructs TU by TU as in previous tests. Results for these tests are displayed in Table 3.

Comparative performance of the algorithms for constructs of increasing complexity and lengths

	Vio-0000 ×1 8.5kb	Vio-0000 ×2 17kb	Vio-0000 ×3 25kb
CM_0	**1-to-1**	Failure	Failure
	36.60 s ± 0.73 s		
	1-to-all	Failure	Failure
	> 30 hours (est)		
CM_1	**1-to-1**	**1-to-1**	**1-to-1**
	78.70s ± 6.61s	524.04s ± 3.16s	1776.53s ± 4.74s
	1-to-all	**1-to-all**	**1-to-all**
	109.86s ± 15.10s	663.01s ± 2.88s	2197.00s ± 3.51s
CM_2	**1-to-1**	**1-to-1**	**1-to-1**
	10.32s ± 0.15s	18.86s ± 0.16s	27.72s ± 0.76s
	1-to-all	**1-to-all**	**1-to-all**
	10.71s ± 0.17 s	21.00s ± 0.16s	30.04s ± 0.26s

Table 3-4 – Comparative performance of the algorithms for constructs of increasing complexity and lengths. Performance was estimated for all three algorithms and in the 1-to-1 and 1-to-all cases. The test constructs were generated by concatenating the test construct Vio-0000. Failures are indicated by red cells.

The results reveal a crucial limitation of CM_0 (the tests failed due to lack of RAM): its range of application is limited to shorter sequences. CM_1's performance also suffered for the largest sequences. With Vio-0000 ×1, full analysis took less than 2 minutes per construct (a reasonable time for everyday use), but shot up to 10 minutes for Vio-0000 ×2, and 35 minutes for Vio-0000 ×3 (which would present challenges at scale). This is indicative of our experience – CM_1 performs well with sequences of less than 10kb, but becomes expensive for sequences above 15 kb. It is worth noting that when genetic diversity is generated with short regulatory elements, running 1-to-all only adds 20% to the run time (matching short sequences remains a cheap operation even for long targets). In everyday use, the run times can be improved by not matching the longer CDS components (or only fragments of them) and focusing on the shorter, regulatory components.

Finally, CM_2 which uses short sequences as inputs remains fast in all cases, and performs

linearly with the construct length. This was expected, since reconstruction and slice recombination have negligible cost, and slices take a similar time to be processed.

7. Summary

When we developed the lycopene application workflow presented in the previous chapter (Chapter 2) we identified a first problem that would prevent from initiating a DoE cycle. As we undertook a replication study previously conducted in the lab (Exley et al. 2019), we could not observe growth for some of the constructs. Consequently, the absence of growth would hinder the measurement of lycopene produced by the cells. We believed that toxicity ((Wang et al., 2019; Taylor & Heap, 2020) and burden from lycopene and the construct (Borkowski et al., 2016) are responsible for this absence or slowdown of growth. For this reason and to enable us to bootstrap the DoE cycle we designed a method that we call **PASIV** (Pooled Approach, Screening, Identification and Visualisation). Instead of drawing random samples similarly to a classic DoE approach and inferring regions of interest from them, PASIV identifies a viable region of the design space through a viability assay. PASIV is made of four successive phases:

- **Phase 1.** A construction step using modular assembly in a pooled approach and aiming at building all constructs at once.
- **Phase 2.** A viability assay consisting of a transformation and culture step followed by a colony picking step.
- **Phase 3.** An identification step where the genetic content of the picked colonies is identified.
- **Phase 4.** A visualisation and analysis step that combines all collected information.

In this chapter, we presented experimental results of the PASIV approach applied to the lycopene operon construct. The pooled approach building phase of PASIV resulted in 99 colonies – 87 of these constructs were successfully sequenced. 73 out of 87 isolates were identified with sufficient reliability (See Section 4.2). Most of the identified isolates proved to be duplicates and, in the end, only 23 constructs were unique. As described, the visualisation phase identified a viable region of the design space that enables the initiation of a DoE cycle.

The second problem identified, which arose in Phase 3 of PASIV, is the need to identify the constructs made by the pooled-approach used in Phase 1. This means that all the combinations of parts are randomly being assembled in a one-pot reaction, all at the same time. Consequently, it is impossible to keep track of which construct is which – such as in a

traditional approach of building constructs where we know what we build. Therefore, the PASIV method requires a robust construct identification step. To this end, we developed a novel, automatic software tool called cMatch**.**

cMatch exploits the structures of the constructs and operates at the functional (component) level. cMatch utilises the information contained in a modular template of the constructs. cMatch has been designed to identify long constructs from fragmented sequencing results (i.e. Sanger sequencing) by utilising split-apply-combine strategy.

We also demonstrated that cMatch is very efficient by running benchmarks on very large synthetic constructs. cMatch can automatically identify constructs in a matter of seconds. On the other hand, a construct identification operation performed quasi-manually by a human is tedious, extremely time consuming and error prone.

Chapter 4
Automated Liquid Handling

1. Introduction

We have presented the lycopene application in Chapter 2 and we have introduced a method to bootstrap a DoE cycle in Chapter 3. The measurement step of the lycopene application is critical, as the values measured from the samples (during the 'Test Phase' of the DBTL cycle) tells us how confident we should be of the results to determine the choice of the next points to choose in the design space for the next iteration of the cycle, and help us decide if we carry on with another iteration of the cycle or stop. An accurate and reliable method of measurement is therefore necessary. Consequently, we need to establish a relationship between the values that are measured and a measurement standard. We need to be able to convert measurements into metabolites concentrations.To this end, we create a calibration curve (also called a standard curve) – (two terms being used interchangeably in the rest of the thesis) from a known standard (a known concentration of lycopene, made from a pharma-grade commercial standard of pure lycopene crystals).

In the Section 1 of this chapter we present the need for a calibration curve. The lycopene application necessitates measuring the concentration of the lycopene produced by the cell. Lycopene is extracted from the cells with DMSO and lycopene concentration is deduced from the measurement of absorbance of lycopene in DMSO – by reading the values on a standard curve, derived from measuring and diluting a standard at known concentration. The construction of a standard curve requires the controlled generation of a set of samples of known concentrations on measurement microplates – and for the process to be repeated over a series of assays. This is achieved by using automated liquid handling. Because laboratory experiments combine multi-step protocols with many liquid-handling steps, robotic liquid-handling platforms are central to the transformation of research laboratories and their adoption of automation (Kong et al., 2012). It is therefore important that this fundamental step of experimental workflows should be reliably automated (i.e. it must yield consistent

outputs). From a replicability viewpoint, consistency of outputs is not enough. Liquid-handling errors should be well-understood (ideally predictable); controlled in magnitude (and unbiased); and should be prevented from propagating to successive parts of the experimental pipelines.

In Section 2 we present the evolution of automated liquid handling from pipetting automation to robotic workstations, and the different challenges that come with its automation. We approach the testing of liquid handling through the construction of a standard curve for a testing mix.

In Section 3 we present the challenges of handling a liquid comprising a combination of lycopene and DMSO which, due to its physical properties, is rather difficult to dispense and measure in an automated manner. Additionally, two dilution schemes (a geometric dilution scheme and a linear dilution scheme) for liquid dilution are discussed, which differ in complexity for their implementation. The choice of the implementation of an automated dilution protocol is important because it introduces different types of errors.

In Section 4 we present an automation method for the creation of a standard curve. The goal of creating a standard curve is dual: first, as introduced, accurate measurement necessitates calibration. Second, the testing of various dilution schemes in the development of automated liquid-handling protocols enables us to find the types of errors introduced by liquid-handling automation – and thus enables us to perfect our liquid-handling operations in our application to minimise the error introduced by the use of automation.

To develop the automated method for creating a standard curve, we conducted a study in two parallel tracks – one for each dilution scheme (geometric and linear) – each track includes three sections: a preliminary study, the study of the lycopene and DMSO mix and the final study that produces the lycopene standard curve. We present the results of these studies. In addition to producing a final calibration curve for our own use in the lycopene application, the studies provided learnings for practical recommendations regarding automation of liquid-handling.

The measurement in the testing step of the lycopene application (See Chapter 2 Section 5.5) is measured with a UV-vis method (ultraviolet–visible spectrophotometry) in a plate-reader. The ultimate value of lycopene concentration of a tested sample is deduced from the measured absorption value by looking up on a lycopene standard curve that we build. A standard curve (also known as a calibration curve) is ‘a regression of the assay response on the known concentrations in ‘standard’ samples. It is a model that fits data from standards and is used for calculating (calibrating) values of unknown test samples.’ (Markossian et al., 2004). The construction of a standard curve requires the controlled generation of a set of samples of known concentrations on measurement microplates – and for the process to be repeated over a series of assays. We only consider liquid handling schemes capable of

producing mixes of solutes and solvents by transferring variable volumes of these solutes and solvents into the wells of the microplates and mixing them *in situ*. This is because automatd liquid handlers manage resources more efficiently, are less labour intensive, scale better and can be used for any liquid handling steps in biological protocols.

Additionally, the construction of a standard curve offers several ways to test liquid handling:

- A (very large) space of plate patterns. Good practices from the design of experiments require that samples should be collected under identical experimental conditions. In our study we chose that all samples were located on the same plate – to control for other experimental effects. Some families of liquid-handling protocols are very flexible and can generate any patterns (including randomised ones) – others are more constrained.
- A large set of modifiers/testers. The liquid-handling scheme can then be stress tested by modifying the liquid that is dispensed. We chose a mix of lycopene in DMSO for reasons explained in the following section.
- Natural reliability metrics. The link with regression can be used to develop metrics quantifying the reliability of the liquid-handling process (consistency within plates and between plates). Regression comes with its own set of statistics (e.g. the coefficient of determination R^2 used in linear regression) and validation methods.

The creation of the standard curve has two goals. First, to test for liquid handling errors and to help us in the refinement a robust automated liquid handling protocol. Second, to create a calibration curve for the lycopene application to deduct lycopene concentration values from the absorption measurement of the samples.

2. Automatic Liquid Handling

This section introduces automated liquid handling with robots and presents the challenges of automating liquid handling.

2.1 Automating pipetting

We will now provide a brief description of the early precursors of pipettes and mouth pipettes techniques dating back to the 19th century with the work of French chemists Francois Descroizilles (Duval, 1951) and Joseph Louis Gay Lussac who coined the terms ‘burette’ and ‘pipettes’ (Szabadváry, 1978), or Marshal Barber's pioneering work in glass capillary pipetting methods (Terreros & Grantham, 1982), modern micropipettes can be attributed to the design of the German physician Heinrich Schnitger in the late 1950s (Klingenberg, 2005) which started to get commercialised in 1961 with the model of the ‘Marburg pipette’ through the German company Eppendorf (Eppendorf, 2021). In the 1940’s low density polyethylene (LDPE) was introduced and the blow moulding industry grew rapidly. Products could now be mass produced cheaply and quickly. In the 1950s polypropylene (PP) got commercialised (Baker, 2018). The first adjustable micropipette is attributed to Henry Lardy and Warren Gilson (Zinnen, 2004). The mass production and commercialisation of micropipettes made them widespread in the 1970s through companies such as Eppenderof and Gilson, some brand names still very present in today's laboratories. Ignoring the horror and dangers of mouth pipetting, daily extensive manual micropipette comes nonetheless with its own set of ergonomic problems (Björksten, Almby & Jansson, 1994) and thumb injuries (Norton & Fischer, 2023).

Turning now to micropipettes, there are two types of micropipette: single channel and multichannel. The single-channel piston pipette is both easy to handle and the most common type used in laboratories. Multichannel pipettes increase throughput by eliminating time-consuming and repetitive pipetting (Leung et al., 2018). The principle of dispensing is simple. A disposable tip is attached to the piston pipette, with the piston at the aspiration lower limit the tip is dipped in the liquid. When the piston is moved to the aspiration upper limit it aspirates the liquid. Micropipettes can be classified in two groups: fixed volume dispensing and adjustable volume dispensing (Batista, Filipe & Mickan, 2008). An extensive classification of single and multi-channel adjustable micropipettes can be found in (Khastsaev, 1993).

Modern Life Science laboratories are frequently equipped with high-throughput liquid handling for the purpose of efficiency. ‘However, the activities are tedious, time-consuming, and even impossible if performed by hand. Consequently, automation for life science has

become a very important field of study today.' (Kong et al., 2012)

Kong et al. classify the various liquid-handling technologies – with the two main categories being contact and non-contact dispensing techniques. In contact dispensing systems a touch-off is necessary to complete the liquid dispensing. This technique is popular for nano/microliter volume dispensing. The technique is simple, reliable and low cost. In noncontact dispensing systems, the liquid is ejected from an orifice. (Kong et al., 2012)

Techniques for non-contact systems are based on the inkjet printing technology.

The three main technologies of automated dispensing are solenoid, piezoelectric and acoustic:

- In solenoid dispensing the liquid presses against a solenoid valve and when the valve opens, the liquid gets ejected.
- In piezoelectric dispensing a piezoelectric crystal is coupled to the glass tube as a collar. A voltage signal induces the contraction of the collar and pressured is applied to the glass tube which ejects nanoliter drops
- Acoustic dispensing focuses acoustic energy on the liquid to eject a droplet without any physical contact. The droplet size is controlled by the intensity of the acoustic energy.

2.3 Robotic workstations

Robotic workstations present many advantages over manual liquid handling – since robots can work without fatigue, increase the throughput, perform consistently, and ensure accuracy and precision (Kong et al., 2012). Kong et al. (2012) divide robotic dispensers in two categories, which differ in scale and scope: single-tasking and multi-tasking:

- Single-tasking robots dispense samples into plates, (the rest of the work is left to human labour), this type of robotic dispenser is usually at a bench scale and dispenses liquid in a single- or multi-channel mode.
- The second category, multitasking robotic workstations have a greater degree of automation. Multi-tasking workstations offer a more complete process which may include operations such as sample transfer, liquid dispensing or substrate preparation. They often comprise the integration of a single-tasking workstation – expanded with accessories such as plate washers, plate readers and plate sealers. In such automated platforms, with integration and multi-function, the robotic liquid handler becomes only one part of the overall multi-functions.

2.3 Challenges with liquid handling

The primary challenge of automating liquid handling is to achieve reliable, robust, repetitive

and high-speed operation (Kong et al., 2012). Many issues have been identified by Kong et al.'s investigation on the obstacles encountered during automated liquid handling operations. These include evaporation control, viscous material handling, calibration, clogging detection, air bubbles dissolution and system integration. Evaporation control is not an issue for large volumes, but becomes a problem when handling nanolitre scale volumes. One solution to overcome this problem is to add an additional pool of water or to add/use extra dummy wells to surround the working wells to prevent edge effects on microplates. Other more sophisticated systems include sensors to precisely humidity and temperature control systems. Other solutions involve plate sealing, which introduce the other problem of peeling-off if the samples need further handling. Viscous materials are very often used in biology research and are challenging to dispense with automated liquid handlers, as they adhere to any tools they touch. Viscous materials are also tricky to handle with acoustic nanoliter dispensers (Labcyte, 2013). Many factors such as the environment conditions and liquid properties influence the dispensing process thus calibration is necessary to ensure accurate dispensing of volumes. Standard solutions can be dispensed successively at different volumes and the accuracy and precision can be calculated (Batista, Filipe & Mickan, 2008). Spectrophotometric methods can be affected by sample dilution (Dong et al., 2006). Clogging of tubes, valves and pipetting heads can become a problem and is caused by viscous material, dust and particles or accumulation of liquid debris. Air bubbles may form during aspiration or dispensing and cause inaccuracies in volume handling. Sensor methods can help detect bubbles in real-time (Kong et al., 2012) however even if the problem is identified, no solution exists for dissolution in the tube or the tips. One traditional method to remove air bubbles in plates is the use of a sonicator, another one often used in microplate-readers is orbital shaking of the plate.

Achieving robust, repetitive and high-speed automated liquid-handling operations is not without its challenges (Kong et al., 2012). A wide range of physical factors have been identified that play a role in the dispensation of liquids, including density and viscosity of the pipetted liquid, pre-wetting and environmental factors such relative humidity, vapour pressure, temperature and air pressure (Ewald, 2015). For instance, combinations of viscous material, dust, particles or liquid debris can result in clogging of tubes, valves and pipetting heads. Additionally, air bubbles may form during aspiration or dispensing and cause inaccuracies in volume handling. Such factors are an important source of error, arising from physical phenomena and it is not immediately clear how they relate to the protocol actions and parameters.

A wider, more practical view of liquid handling errors was presented by Albert (Albert, 2007), who proposed a classification tracing them back to ill-adapted equipment, ill-adapted pipetting-methods and ill-adapted dispensing-schemes. Noticeably, some of these errors

affect wells independently, while some introduce correlation within and across plates – e.g. errors linked to the travel of the tips and the evolution of common resources (e.g. the volume of reagent in a reservoir). Such errors will propagate down the liquid-handling workflows differently.

3. Automation Method for Creating a Standard Curve

This section describes the challenges of working with a combination of lycopene with DMSO. We then explain two dilution schemes (geometric and linear) that we have considered – and discuss how amenable they are to automation.

3.1 The challenges of lycopene and DMSO

The final component to be measured in the lycopene application is the mix of the solvent dimethyl sulfoxide (DMSO) and lycopene. We observed that the mix of lycopene and DMSO was difficult to pipette manually – going as far as likening it to a mix of tomato puree at higher concentrations. Albert et al. (2006) calls such mixes atypical liquids. They offer substantial challenges to automate their dilution (and opportunities to determine practical dos and don'ts regarding automation of dilution protocols and the control of liquid-handling errors).

The absorption spectra of carotenoids share a similar distinctive shape, with three separate peaks, in the visible part of the spectrum (Scott, 2001). Carotenoids interact to various degrees with solvents, resulting in shift and change of intensity of the absorption peaks. The spectrum of lutein is almost independent of solvents at various concentrations, unlike zeaxanthin, lycopene and β-carotene (Zang, Sommerburg & van Kuijk, 1997). The three peaks of the spectrum of lycopene have been located at 457, 483, 516 nm, in hexane (Chemat-Djenni et al., 2010), 448, 474, 505 nm in acetone, and 451, 479, 511 nm in DMSO (Zhao et al., 2020). Shape of the absorption spectrum and location of its peaks were used for validation at several points of our study. Lycopene undergoes two types of change during processing and storage – isomerization from all-trans to mono or poly-cis forms and oxidation. Due to its highly unsaturated nature, it is very susceptible to oxidation. Degradation is largely affected by time, reaction medium and environmental conditions such as heat, light and oxygen (Zhao et al., 2020; Saeid et al., 2016; Shi et al., 2003; Srivastava & Srivastava, 2015).

Lycopene is a hydrophobic molecule, but is soluble in lipophilic solvents such as hexane, acetone, and dimethyl sulfoxide (DMSO). Conventional methods for extraction of lycopene use non-polar organic solvents (Guerra et al., 2021), such as hexane, dichloromethane and ethyl acetate (Barba et al., 2006; Anarjan, 2020),. However, USFDA regulations state that

use of organic solvents, such as hexane and methanol, should be limited because of their toxicity. They are also highly volatile, which makes them cumbersome for high-throughput applications. Previous work at the London Biofoundry (Exley et al., 2019) used high-performance liquid chromatography (HPLC) to measure lycopene produced and used acetone for extraction. These researchers reported difficulties to maintain constant volume during the processing of the samples on 96-well plates. This led some researchers to use DMSO instead, as it offers a better trade-off between high throughput and process consistency (HamediRad et al., 2019). We chose DMSO for the following reasons: it is a polar aprotic solvent which can dissolve both polar and nonpolar compounds, and has a proven ability to enhance the permeability of lipid membranes (Notman et al., 2006). It is extensively used in biochemistry and cell biology (Wang et al., 2018) – including to extract lycopene from tomatoes, algae, and bacteria (Hamed et al., 2022; Wang et al., 2023). DMSO is a clear, colourless to yellowish viscous liquid with a characteristic bitter odour and taste (Pope & Oliver, 1966; Brayton, 1986). Few direct mentions of serious solubility issues of lycopene in solvents like DMSO could be found in the literature to back our observations. However, a study (Zang, Sommerburg & van Kuijk, 1997) of how solvents affect absorbance spectra of carotenoids pointed out the relation between carotenoid concentration and absorbance could become highly non-linear, and presented ‘differences in solubility leading to the formation of microcrystals’ as a possible explanation.

3.2 Choice of dilution schemes

Two different dilution schemes (see Figure 4-0 and Figure 4-1), representative of two different approaches to dilution, were used for this study. The first dilution scheme is a direct adaptation of a well-known manual protocol (the serial dilution protocol (Blaize, Suter & Corb, 2023; Ben-David & Davidson, 2014)). It is based on an iterative transfer workflow (transferring liquid from column to column) – we henceforth refer to it as the ‘geometric dilution scheme’. In our implementation (Figure 4-1A and Figure 4-XB), the first column is created by direct dilution so concentrations vary linearly, before iterative dilution is performed to generate concentrations of lycopene that halve with every column. The second scheme (the ‘linear dilution scheme’) is based on a direct transfer workflow (that only transfers liquid from reservoirs to wells). Our implementation (Figure 4-1-B) generates a linear range of dilutions between 0 and 1, with an increment of 5%. All dilutions were performed on a CyBio Felix (Analytik Jena AG, n.d.a) liquid-handling robotic platform with an 8-channel SELECT Head (Analytik Jena AG, n.d.b) – see Methods (Appendices C4 - Methods) for an overview of the protocols.

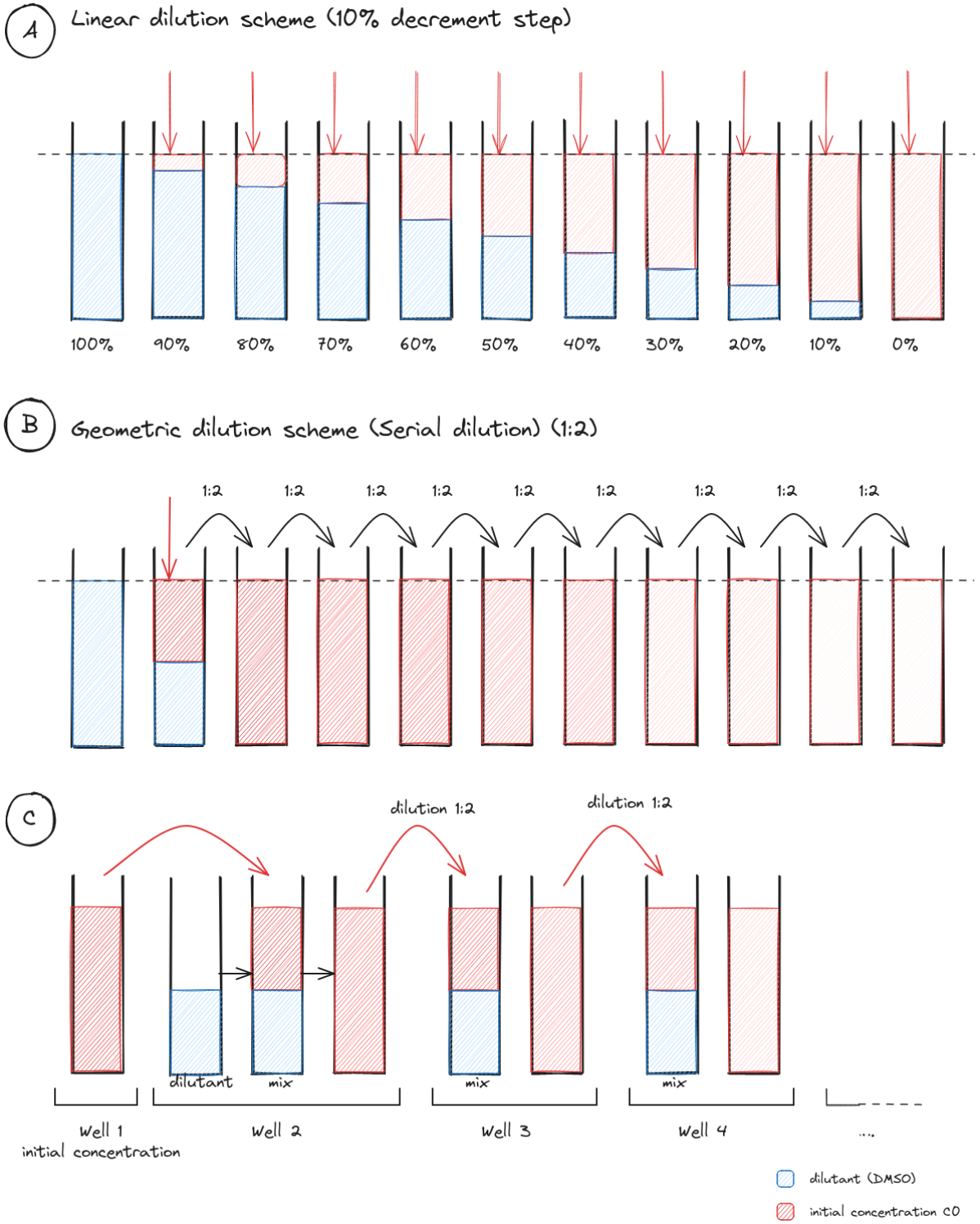


Figure 4-0 – Dilution schemes represented as liquid transfers (A) Linear dilution scheme with a decrement step of 10%. **(B)** Geometric dilution scheme with a 1:2 ratio. **(C)** Detail of the liquid transfer from well to well for a geometric dilution scheme with a 1:2 ratio.

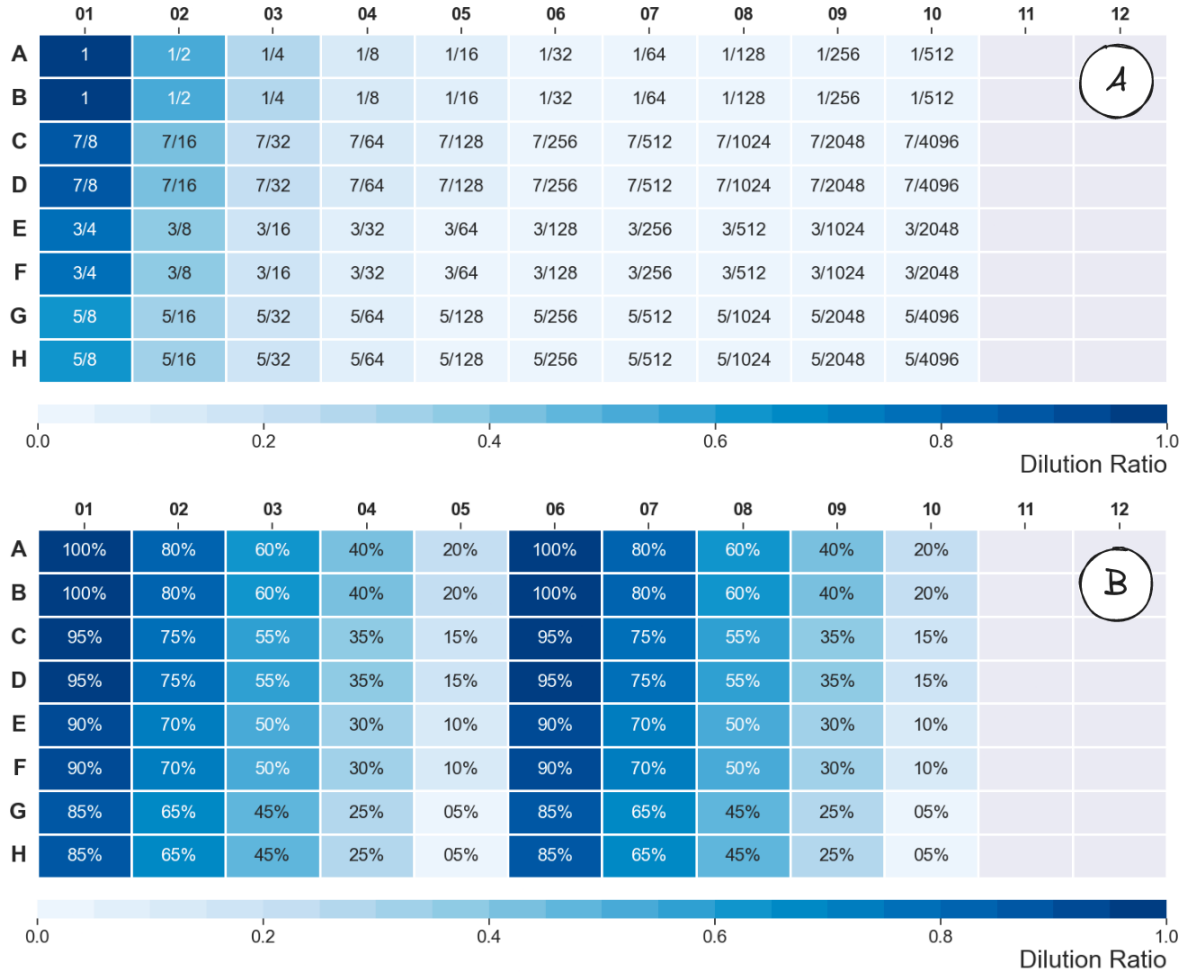


Figure 4-1 – The geometric and linear dilution schemes. (A) Geometric dilution scheme plate map of the 96-well measurement plate. Initial concentration of the standard is in well A1 and fractions of the initial concentration in C01, E01 and G01, respectively (7/8), (3/4) and (5/8) of the initial concentration (with duplicated technical repeats every other rows), the subsequent columns contain a dilution scheme of 1:2. Blanks (containing only the diluent) are located in columns 11 and 12. All wells contain an overall volume of 200µl. **(B)** Linear dilution scheme plate map of the 96-well measurement plate. Initial concentration of the standard is in well A1 and each successive dilution has a step of 5% going from top to bottom from rows A to H (with technical repeats every other row). The pattern contains two identical blocks (columns 01-05 and 06-10). The same blanks were used as for the first scheme. Likewise, all wells contained an overall volume of 200µl

Both the geometric and linear dilution schemes vary in terms of complexity for their implementation, which, in turn determines how amenable they are to manual implementation at even modest scales, or, equivalently how needed automation is for their use. They also vary in terms of flexibility and what plate pattern they can generate. Finally, they differ in another important way: the errors associated with their implementation and platform limitations (summarised in Table 1 using Albert's approach). A more thorough comparison between the two schemes are referred to Section 1 of the Appendices C4 (Appendices C4 – Section 1).

Possible sources of errors

Source of error	**Geometric scheme**	**Linear scheme**
Volumes transferred (depends on platform)	Only large (safe) volumes	**Any volume in theory**. Increments chosen so only safe volumes transferred
Contamination by falling proplets (depends on choice of tips and dispensed liquid)	**Minimal**. Only the first column is constructed by direct transfer. Column by column transfers are short. DMSO was dispatched first to minimise risks	**Possible**. The linear dilution scheme involves a lot of travels between wells and reservoirs – each with its own probability of causing contamination with falling droplets. Same safety measure for DMSO
Reservoir lycopene concentration	**Potentially severe**. Same as for the linear scheme will only affect column 1	**Potentially severe**. At high concentration, the mix is expected to be difficult to dispense – affecting all wells in columns 1-10
Liquid homogeneity (between wells)	**Potentially severe**. With the iterative transfers, most wells are used as sources. Pipetting precision may be affected by a one-size-fits-all approach for the aspiration and dispensation parameters.	**Irrelevant**. Only two homogenous reservoirs are dispensed from. Aspiration and dispensation parameters can be set up globally.
Type of error	**Geometric scheme**	**Linear scheme**
Independent errors	Yes. ● Contamination ● Direct transfers in first column	Yes. ● Contamination ● All direct transfers
Correlated errors	Yes. Transfer column by column propagates errors	None expected

Table 4-1 – Possible Source of Errors (after Albert's Classification).

4. Results

To test the validity of the method to automatically generate a standard curve we designed a study to be conducted in two parallel tracks – one for each dilution scheme (geometric scheme and linear scheme). Each track was to include three sections, which we detail below:

Phase 1. Preliminary investigation.
Testing runs were to be conducted using food colourant in solution in water (or DMSO). These are very common mixes for the development of liquid-handling applications, as they present no solubility or viscosity issues. The process was to be repeated using mixes of lycopene in DMSO. Two sets of lycopene mixes were used – both derived from an extremely concentrated mix of lycopene in DMSO (concentration 0.5 mg/ml) by one or two 1/10 dilutions – see Methods (in Appendices C4 - Methods). The high concentration mix, at concentration 0.05 mg/ml, was expected to exhibit strong non-linearity in the relationship between dilution and measurements – unlike the low concentration mix, at concentration 0.005 mg/ml. Experimental results were to be used for three different purposes: (a) to analyse the statistical properties of the collected data, and validate their suitability for the construction of a standard; (b) to validate the automated implementations (the manual implementations, when available, are to be used as gold standards); and (c) to probe the spatial patterns introduced to the dilution schemes (in particular the pattern between columns in the geometric scheme , since iterative transfers are liable to introduce specific classes of errors)

Phase 2. Lycopene in DMSO.
In the second phase, the properties of the mix of lycopene in DMSO were further investigated and quantified – including the stability of the mixes (stability of the absorbance spectrum and degradation studies). All assays used mixes of lycopene in DMSO – as the dilution protocol would have been validated (possibly to be tuned up still). Methods for the quantification of dilution errors were to be developed on data from Phase 2.

Phase 3. Final study.
The standard curve for lycopene in DMSO was to be constructed. A final data acquisition protocol – incorporating all relevant information from the previous phases – was to be designed. so measurements from several runs could be safely aggregated.

4.1 Phase 1 - preliminary investigation - geometric scheme

Figures 4-2 and 4-3 (next page) show the outcome to the exploratory analysis of the data collected with the geometric dilution scheme for a development run with food colourant in water (Run 55 - Plate 01). The plate was generated with the automated version of the

geometric protocol. Figure 4-2-A shows the absorbance spectra for all 80 samples. Repeats yield very close measurements for the first columns, and get more separate as the columns move to the left. Due to the distribution of dilutions generated by the dilution scheme, samples cluster close to a baseline. Figure 4-2-B shows the correlations between targeted dilution ratios and measurements at wavelengths between 350 nm and 650 nm.

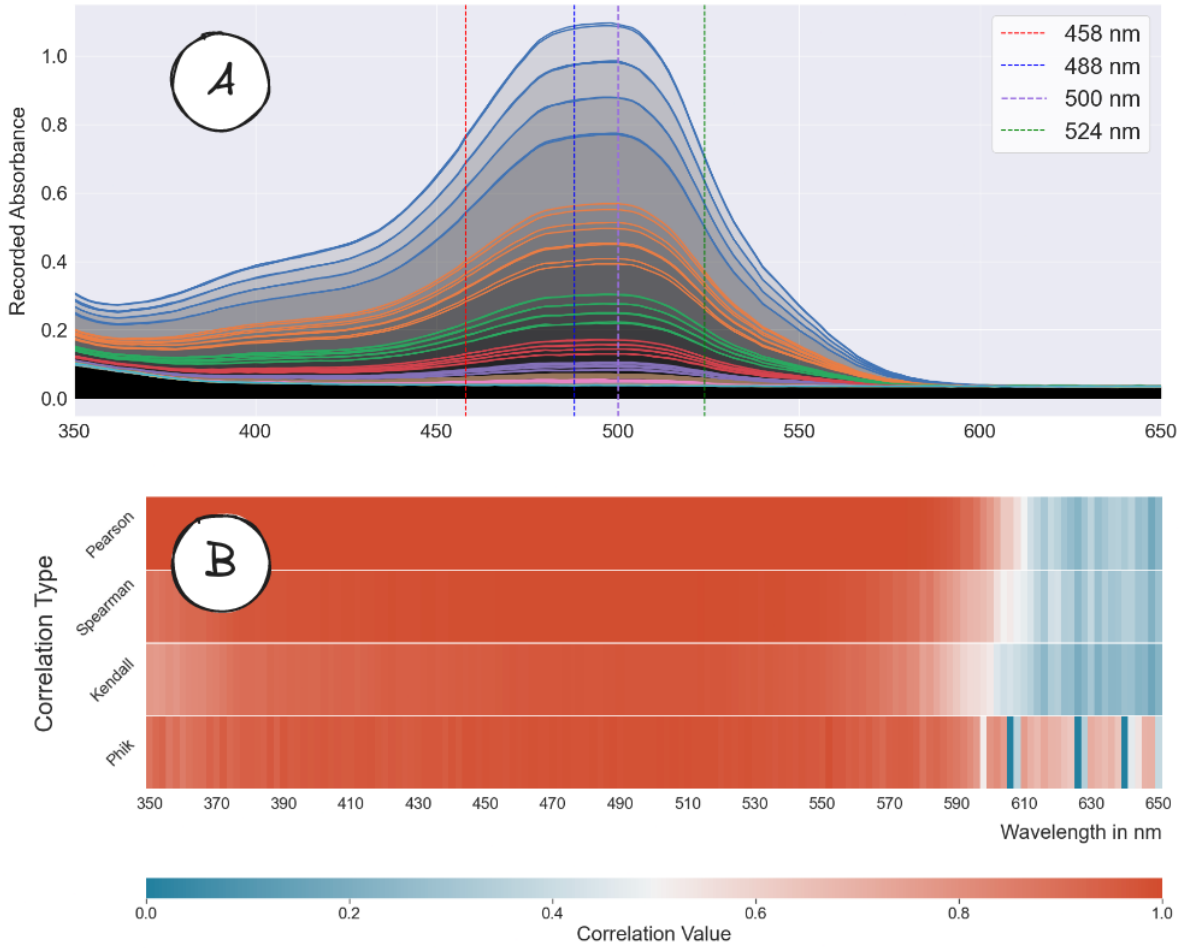


Figure 4-2 – Run 55 - Plate 01 (Food colourant in water) – Automated implementation of the geometric dilution scheme. (A) Absorbance spectra for all samples (for wavelengths ranging from 350 nm to 650 nm). Samples are coloured according to their rows (blue = columns 1, orange = column 2 etc…). **(B)** Correlations between the target dilution ratios and measurements at all wavelengths. All four standard measures of correlation (Person, Spearman, Kendall and Phi-k) are computed.

Four common correlation metrics (Pearson, Spearman, Kendall and Phi-k) were computed. All indicate a high degree of correlations for measurements between 400 nm and 550 nm, and gradual loss of correlation for higher wavelengths. The pair plot (Figure 4-3) for selected wavelengths gives further insight into these results. Measurements between 400 nm and 550 nm were very highly correlated. Measurements from 600 nm and above had not only low intensities but also were noisy – indicating that the limits of the instruments were

reached. Altogether, this set of results indicate the data as highly suitable for the construction of a standard.

To further validate the automated implementation of the geometric dilution scheme, a comparison with the manual implementation was conducted. Two of the development runs (Runs 55 and 57) contained a plate (Plate 01) generated with the automated implementation, and another (Plate 02) generated manually – both from the same mix freshly made on the day. Both assays were run by the same experimentalist, with the same pipetting equipment. Results can be found in the Appendices C4 - Section 2. They show measurements for manually and automatically-generated plates matching each other to a very high degree.

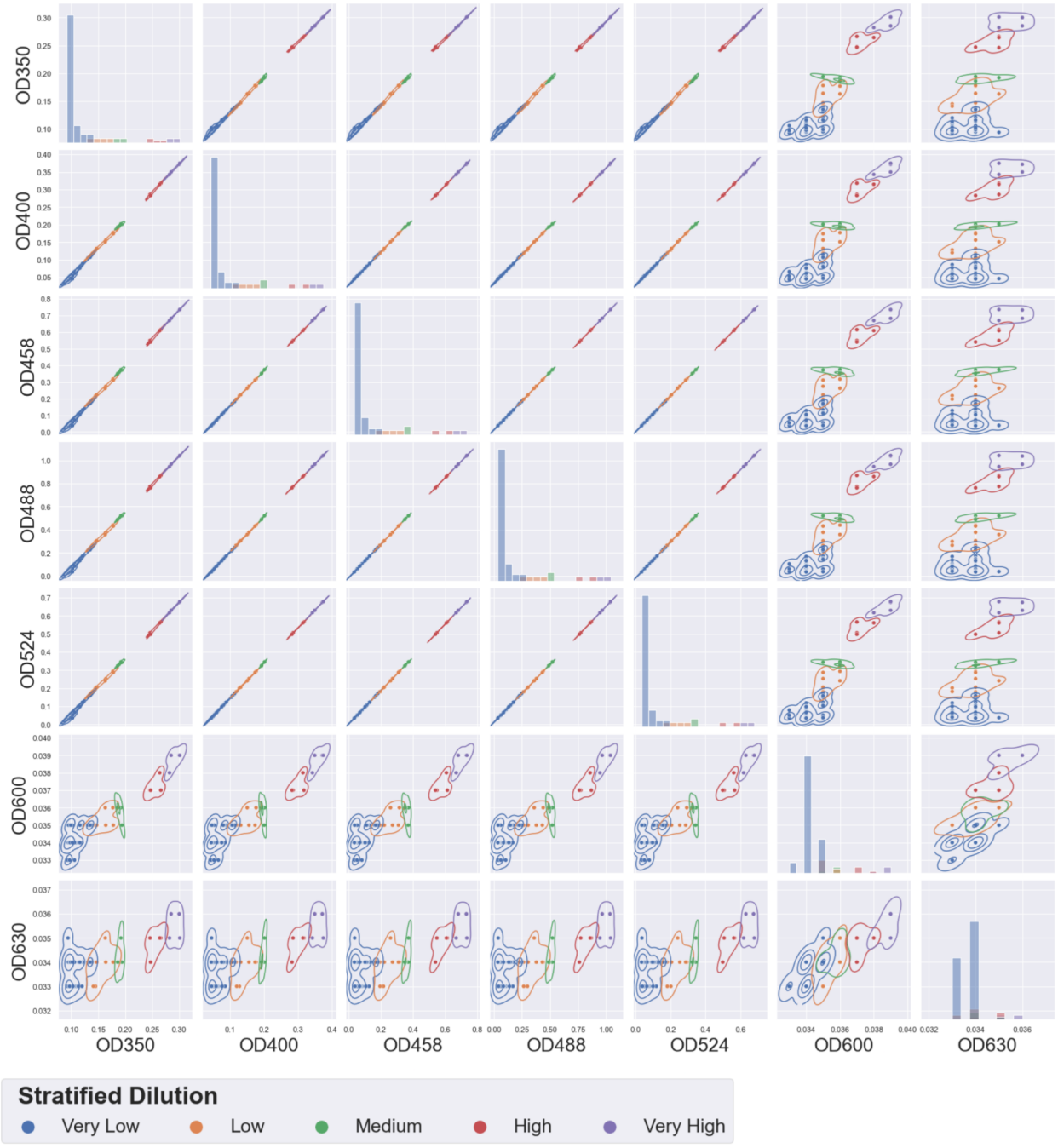


Figure 4-3 – Run 55 - Plate 01 (Food colourant in water) – Automated implementation of the geometric dilution scheme – Pairplot for selected wavelengths. Samples were stratified and assigned colours on the basis of their dilution (very low = dilutions <0.2, ... very high = 0.8 and above). Measurements at different wavelengths were highly correlated for wavelengths below 550 nm – Measurements from 600 nm and above had low intensity, were noisy and consequently their correlations with other measurements were poor.

The process was repeated for mixes of lycopene in DMSO (Runs 58 to 64) – using two sets of concentrations. (Results of the exploratory data analysis for some of these runs can be found in the Appendices C4 - Section 3.) These results were similar to the results for the testing mixes, but correlations between measurements (and with the dilution ratios) were less strong. Absorbance spectra between the runs seem to vary significantly too – this was attributed to the mix age in a later part of our study. These preliminary results confirm initial assumptions about the difficult nature of the combination of lycopene and DMSO (such mixes are qualified as atypical liquids by Albert et al. (2006)) – additionally, personal communication between members of SynbiCITE's London Biofoundry described the mix of lycopene and DMSO as difficult to pipette manually because of its viscosity.

Finally, we focused on the effects of the iterative transfers from column 1 to column 2 (2 to 3 etc..), as it is the operation that was identified as liable to introduce correlated errors. Accurate transfer from one column to the next would result in measurements that are distributed tightly around the regression curve, and columns showing trends that deviate little from the general trend of the dataset. Figure 4-4 shows results for two runs – one run with food colourant in water (Run 55 - Figures 4-4A and 4-4B – representative of this mix) and one for lycopene in DMSO (Run 63 - Figures 4-4C and 4-4D). For both runs, a plate was generated with automation (Figures 4-4A and 4-4C), and manually (4-4B and 4-4D). All measurements were conducted on the same measurement platform. For all datasets, the measure of trend is indicated by the confidence intervals (in red) obtained with a linear regression.

For the food colourant in water (Figures 4-4A and 4-4B), stratification by column reveals different trends (slopes and baselines) for columns 1, 2 and 3 . A few samples fell outside the confidence interval of the predictions. This was true for both manual and automated implementations, and was stronger with automation. Results are consistent with a transfer that is too large from one column to the next. The difference between automation and manual implementations was very noticeable for lycopene in DMSO (Figures 4-4C and 4-4D). Results for the manual implementation (Figure 4-4-C) were much smoother and closer to the runs with water and food colourant than the automated results (Figure 4-4D). They still exhibited the same jumps from one column to the next and the slope pattern as in Figures 4-4A and 4-4B. But overall the deviations were smaller. Results for several other automated runs (see Appendices C4 - Section 3) with lycopene and DMSO (at high and low concentrations) show similar results – large deviations from the dataset trend, very different slopes for each column but also large deviations in the trends for the same columns – all indicative of large errors in the transfers from adjacent columns, and a general lack of control of the operation.

Taken in conjunction with the results for the runs with food colourant, we concluded that two factors were at play. First, iterative dilution introduces a distinct class of (correlated) errors. Even in the case of a simple liquid mix, the errors were visible, so much so data should be analysed column by column in order to compensate for them. Second, the errors are significantly worse for the lycopene in DMSO. For both reasons the geometric (iterative) dilution scheme was concluded unsuitable for the construction of the standard of lycopene in DMSO. Further investigations with the geometric scheme were duly cancelled.

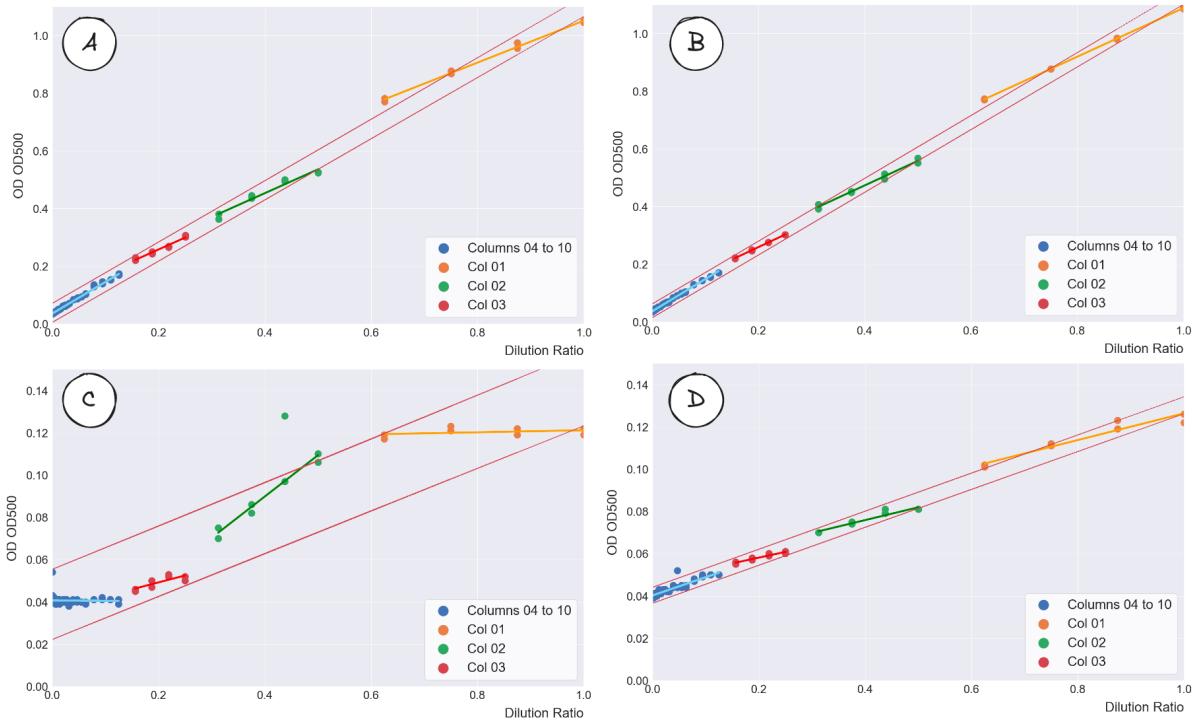


Figure 4-4 – Effects of the iterative dilutions at 500 nm – stratified by columns. (A) Run 55 - Plate - 01 (Food colourant and water) Automated Implementation. **(B)** Run 55 - Plate - 02 - Food colourant and water - Manual protocol. **(C)** Run 62 - Plate - 01 – Lycopene and DMSO – Automated implementation. **(D)** Run 63 - Plate - 01 – Lycopene and DMSO – Manual protocol. In all plots the samples were coloured according to their rows, and the dataset trend was indicated by the confidence intervals (in red) obtained with a linear regression for the whole dataset. All measurements, at the wavelength 500 nm, were conducted on the same platform (Clariostar A in our Biofoundry).

4.2 Phase 1 - preliminary investigation - linear dilution scheme

The preliminary study for the linear dilution scheme used once again food colourant, but this time in DMSO. Figure 4-5 shows the main results for Run 74 - Plate 01. The results are representative of all development runs (See Appendices C4 - Section 3 for more runs). All measurements were repeated in two different Clariostar platforms (A and B in our naming nomenclature – A being shown in Figure 4-5). The results were almost identical – hence they were not affected by the measuring instrument used (See Appendices C4 - Section 4). Absorbance spectra for all 80 samples are shown in Figure 4-5A. For all concentrations repeats, the results were close to each other and their magnitude was evenly spread – with a slight compression indicating some non-linearity at the highest concentrations. This was confirmed by the regression models built from the data (Appendices C4 - Section 4). As a consequence of the choice of sample concentration (spreading concentrations evenly between 0 and the maximum concentration in the solute reservoir), the pattern of correlations between targeted dilution and measurements was very pronounced (almost perfect correlation at wavelengths below 580 nm – Figure 4-5B). The same applies to correlations between measurements at different wavelengths (Appendices C4) – which were highly correlated for wavelengths up to 580 nm.

Unlike the previous study with the geometric dilution scheme, stratified analysis by column and by row showed no discernable effect differences (see Appendices C4 - Section 4).

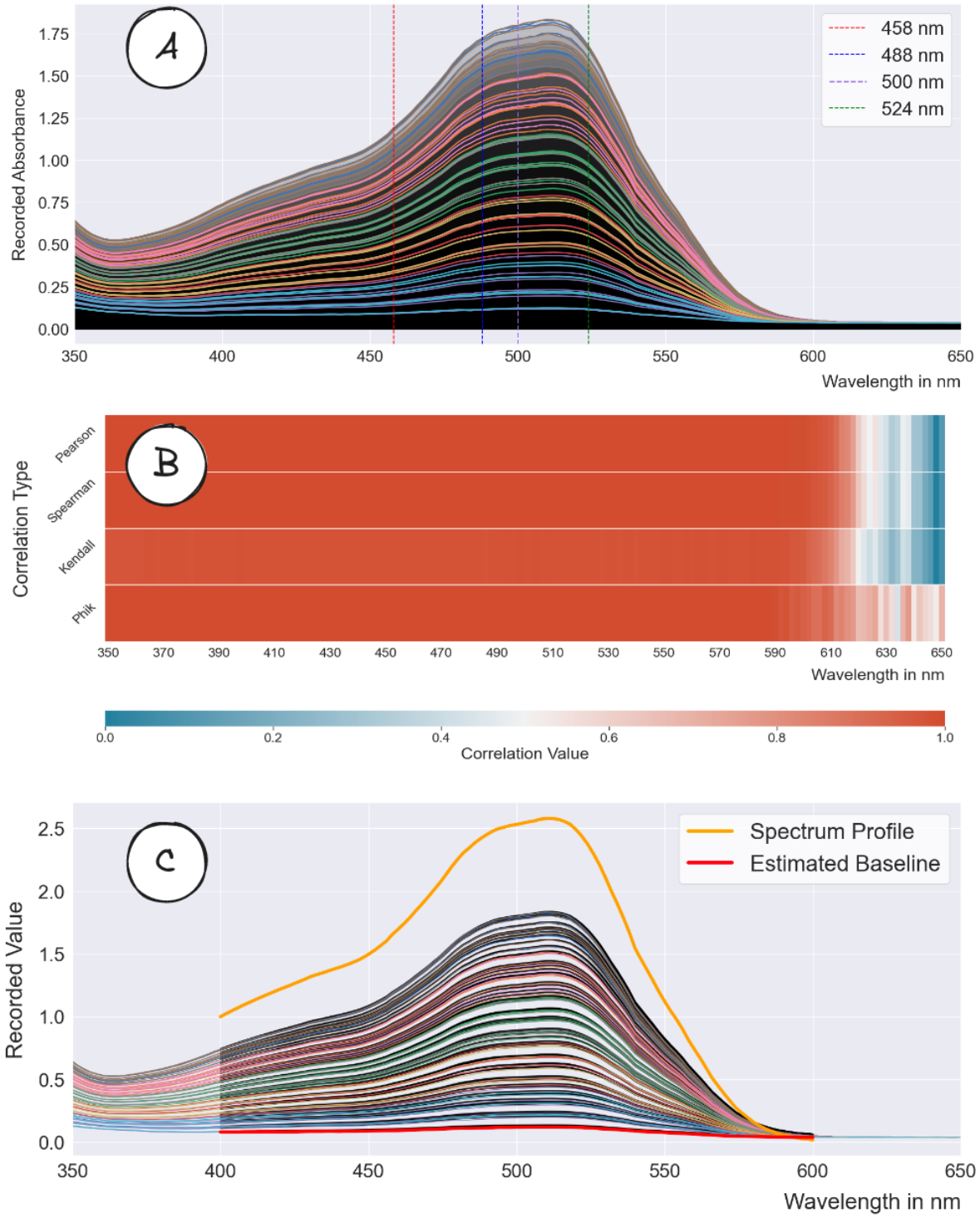


Figure 4-5 – Run 074 - Plate 01 (food colorant in DMSO) – linear dilution scheme. (A) Absorbance spectra for all 80 samples in the plate. **(B)** Correlations between the target dilution ratios and measurements at all wavelengths. (**C)** Estimation of the Spectrum of the mix and the baseline reading – the spectrum is normalised for legibility purposes.

The high degree of linear correlation between measurements at different wavelengths led to the construction of a model (called SM, for spectrum model) which binds measurements for any well over a range of wavelengths to a vector S – the spectrum of the solute – and a baseline B – the background measurement of the plate over the same range. The coefficient α_i is well dependent and indicates the quantity of signal producing-solute in the well. Our model imposes minimal constraints: all components of S and B and coefficients α_i are to be positive. A penalty function is used so the baseline B stays close to blank values.

$$(SM) \quad Y_i = \alpha_i \times S + B$$

Figure 4-5-C shows the results of the estimation of the parameters B,S (α_i not shown) from the plate data. The baseline B was plotted in red – and matches the absorbance spectra of the blanks on the plate. The spectrum S, was plotted in amber; and for legibility purposes its amplitude was normalised so its value is 1 at wl= 400 nm (See Appendices C4).

4.3 Error assessment methods

Ideally, the content of the wells would be known to a sufficient precision – either because liquid-handling makes small errors (a proposition which needs testing), or because the dispensed volumes can be measured accurately (which was not possible). Estimating dilution errors therefore needs an alternative way to estimate the concentration of the mix in the wells. Consequently we developed the following workflow (see Figure 4-6), built on regression modelling. The spectrum model estimates the absorbance spectrum of the mix, by concentrating on the part of the signals that is common to all the readings. This is possible because once a baseline (caused by the plate itself) is removed from the individual sample readings, they are proportional. Our workflow is concerned with these coefficients of proportionality, and proceeds as follows:

1. **Step 1. Regression**: First, a wavelength containing information representative of the concentration of solute is selected. We suggest to select a location near a peak of the spectrum, where the signal is largest. Then a regression model linking the independent variable (the targeted dilution ratio or the concentration) and the dependent variable (the absorbance measurement) is built.
2. **Step 2. Inversion:** The regression model is inverted and used to estimate the concentration for each sample (or equivalently the dilution ratio). This is where the framework deviates from simple regression and construction of a standard, to focus on estimation of liquid-handling errors. Without liquid handling errors, the relationship between estimated concentrations and targeted concentrations would perfectly fall on the Y=X line. In practice it does not, and (global) discrepancy metrics can be computed including the standard Pearson correlation

and mean square error (MSE, concerned with the magnitude of errors).

3. **Step 3. Maps (Local Error Analysis):** This final step is concerned with the detection of local patterns. For all samples, the residual error between predicted and targeted concentration is calculated, and converted into relative errors (by dividing the residual by the targeted value). Finer-grain analysis (e.g. stratification against the targeted concentration) and analysis of local effects may follow.

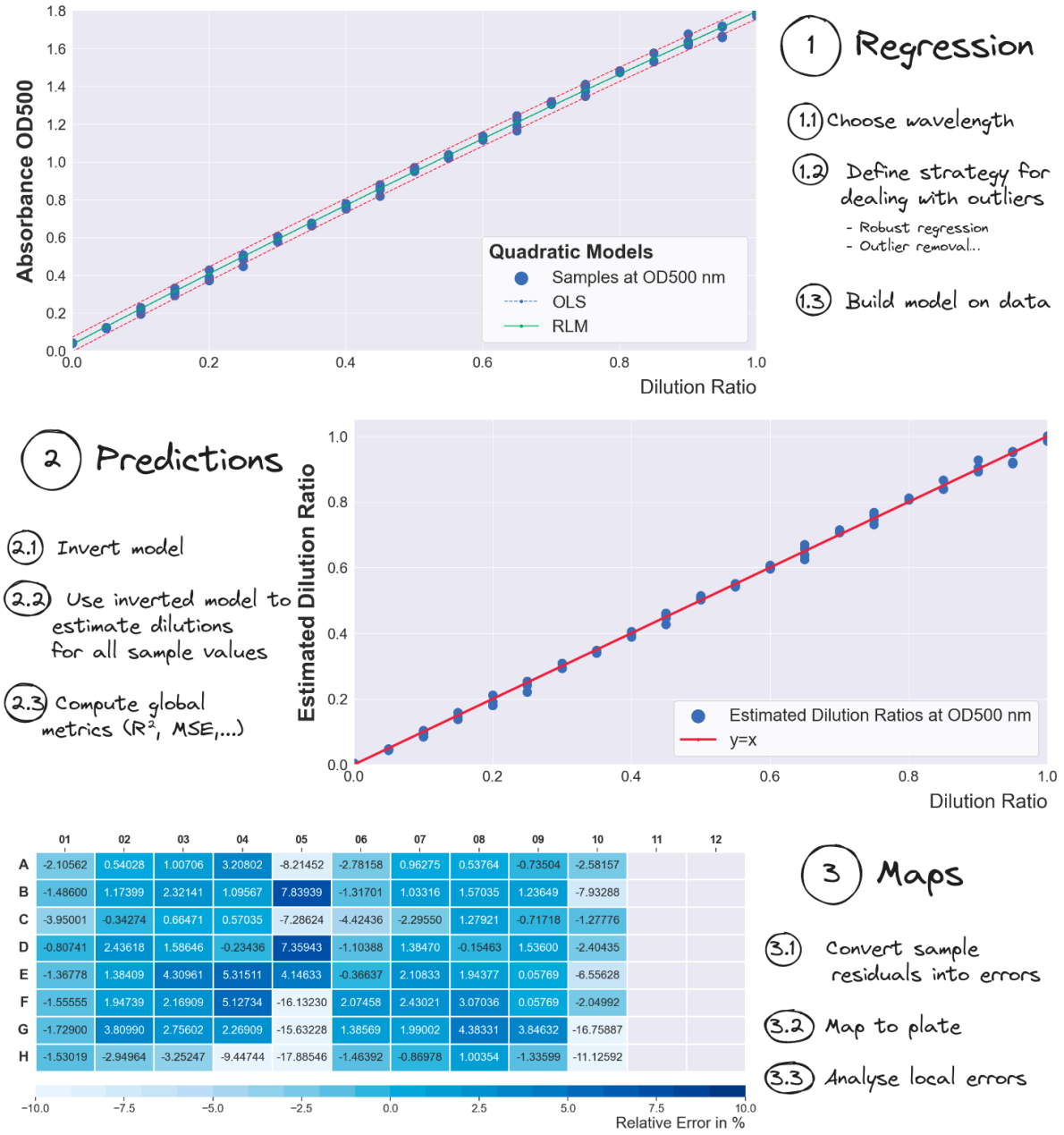


Figure 4-6 – Error Quantification workflow. The workflow is illustrated on Run 074 - Plate 01 - Food Colorant in DMSO (linear dilution scheme).

The framework is open with regard to what regression model is used in the first step. In our study all trends could be captured by linear models of degree 1 (affine model) or 2 (quadratic models, to capture a possible curvature indicative of saturation). The framework is also open with regard to how to deal with outliers. For every dataset, linear regression was performed with the whole dataset to obtain an insight into its trend and benefit from the set of statistics yielded by linear regression techniques – in particular the confidence intervals on the predictions and on the parameters. Then robust regression models were used (which adapt the weight of the sample data to mitigate the influence of outliers) as a refinement. The error-estimation workflow can also be adapted to use the estimated coefficients $\alpha = (\alpha_i)$ in the spectrum model instead of the measurements at the selected wavelength. The precision of the overall approach will then be very dependent on the validity of the spectral model. For high concentrations of solutes, various solubility effects may occur – and only a portion of the spectra may be sufficiently correlated.

Figure 4-6 illustrates the different steps of the framework on Run 074 - Plate 01. The results after inversion show an extremely good match between estimated and targeted concentrations – Table 4-2 shows nearly perfect global quality metrics (Pearson, Spearman correlations and square root MSE). Mapping relative errors onto the plate shows the largest relative errors are for the smallest concentrations of solute (5 to 15%); they are consistent with manufacturers statements of pipetting error ranges of a few microlitres (µl). Results also show nearly identical results between using the spectrum method and measurement data at 500 nm (close to the peak of the spectrum for the testing mixes).

Global quality metric values

Run 074 - Plate 01		
At 500 nm	**Model of order 1**	**Model of order 2**
Pearson correlation	0.9992221245	0.999427306
Spearman correlation	0.997359134	0.9973591339
√MSE	0.01295218	0.01112147
Spectrum method	**Model of order 1**	**Model of order 2**
Pearson correlation	0.9991950065	0.9994483427
Spearman correlation	0.9969024249	0.9969024249
√MSE	0.01317668	0.01091633

Table 4-2 – Global quality metric values. All results are for the development run Run 074 - Plate 01 (Food colorant in DMSO - linear dilution scheme). Instead of the MSE which yields small values for the dilution ratios, the square root of the MSE was used.

4.4 Phase 2 - lycopene in DMSO - linear dilution scheme

Several runs for the linear dilution scheme were conducted – to tune up the protocol and test the error assessment framework (See Appendices C4 - Section 4) – using both low concentration and high concentration mixes. Results for the Run 81 (low concentration mix) are shown in Figure 4-7 ; and Run 79 (high concentration mix) in Figure 4-8. Both used freshly-made mixes (starting from an original mix at 0.5mg/ml) and were representative of runs under such conditions. Table 3 contains the global quality metrics for both runs. The absorbance spectrum of Run 81 (7-A) displayed the characteristic shape of the carotenoids – with three distinct peaks at 458 nm, 488 nm and 524 nm, which were slightly higher values than reported in literature. Measurements at the peak wavelength 524 nm (7-B) were dotted around the regression curve (all remaining in the confidence interval of the regression) – but less tightly than for the food colourant (Table 4-2). Likewise the inverted model did not yield such good estimates of the concentrations as for the food colourant. Finally, when mapped onto the plate, some wells displayed large relative errors of 10-15%. Quantitative results (in Table 3) showed larger errors than with the food colourant mix (the root MSE was twice as large as that of Run 74 in Table 2), but still very good fits. Overall, the mix yielded more variation than the food colourant – some wells deviated and were borderline outliers.

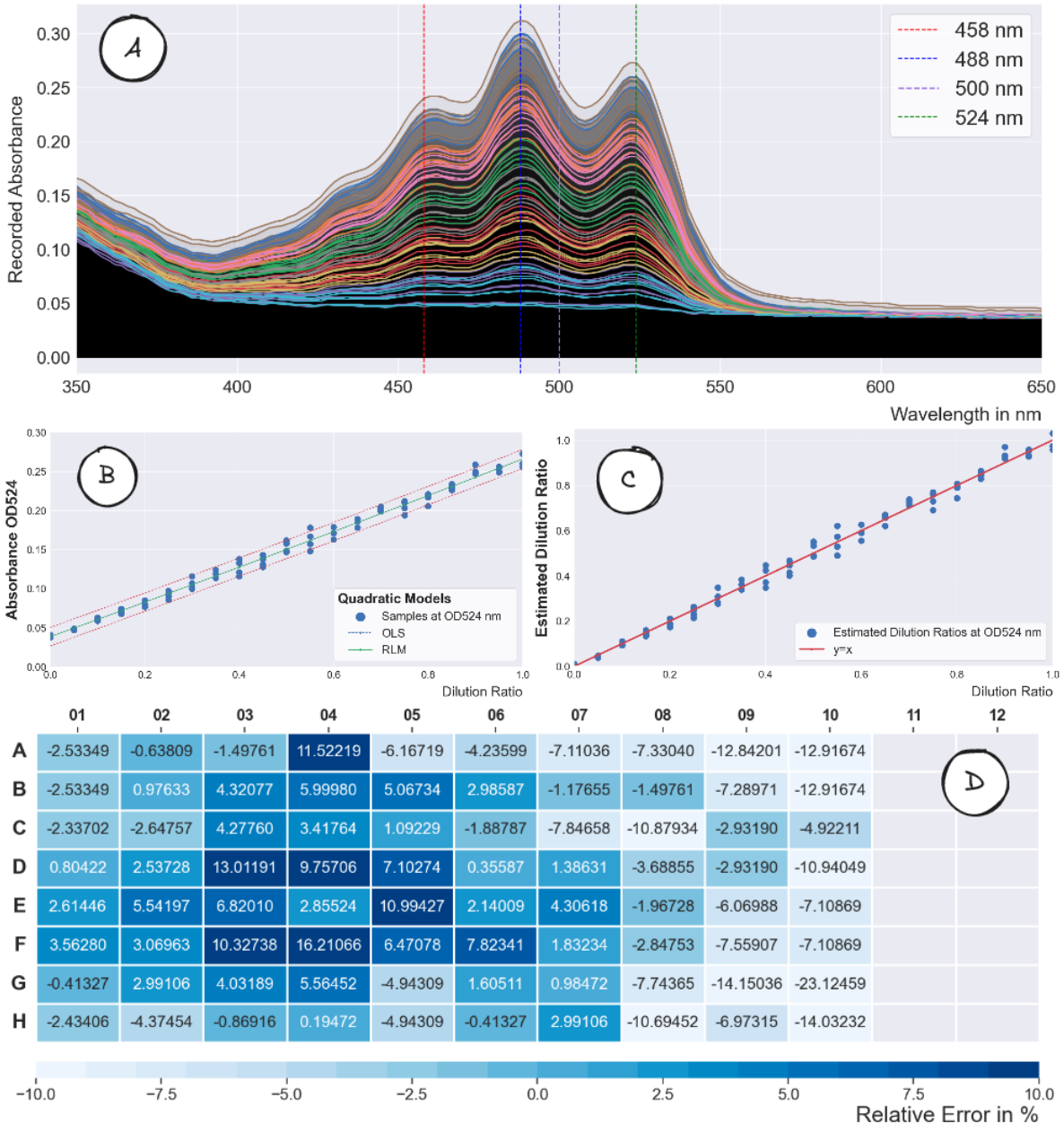


Figure 4-7 – Results for the typical Run 081 - Plate 01 (Low concentration mix). (A) Absorbance spectra for all 80 samples in the plate. **(B)** Correlations between the target dilution ratios and measurements at 524 nm (peak of spectrum). **(C)** Estimation of the real dilution ratios (the Y=X line is plotted in red for comparison). (**D)** Relative errors for all sample plates mapped onto the plate.

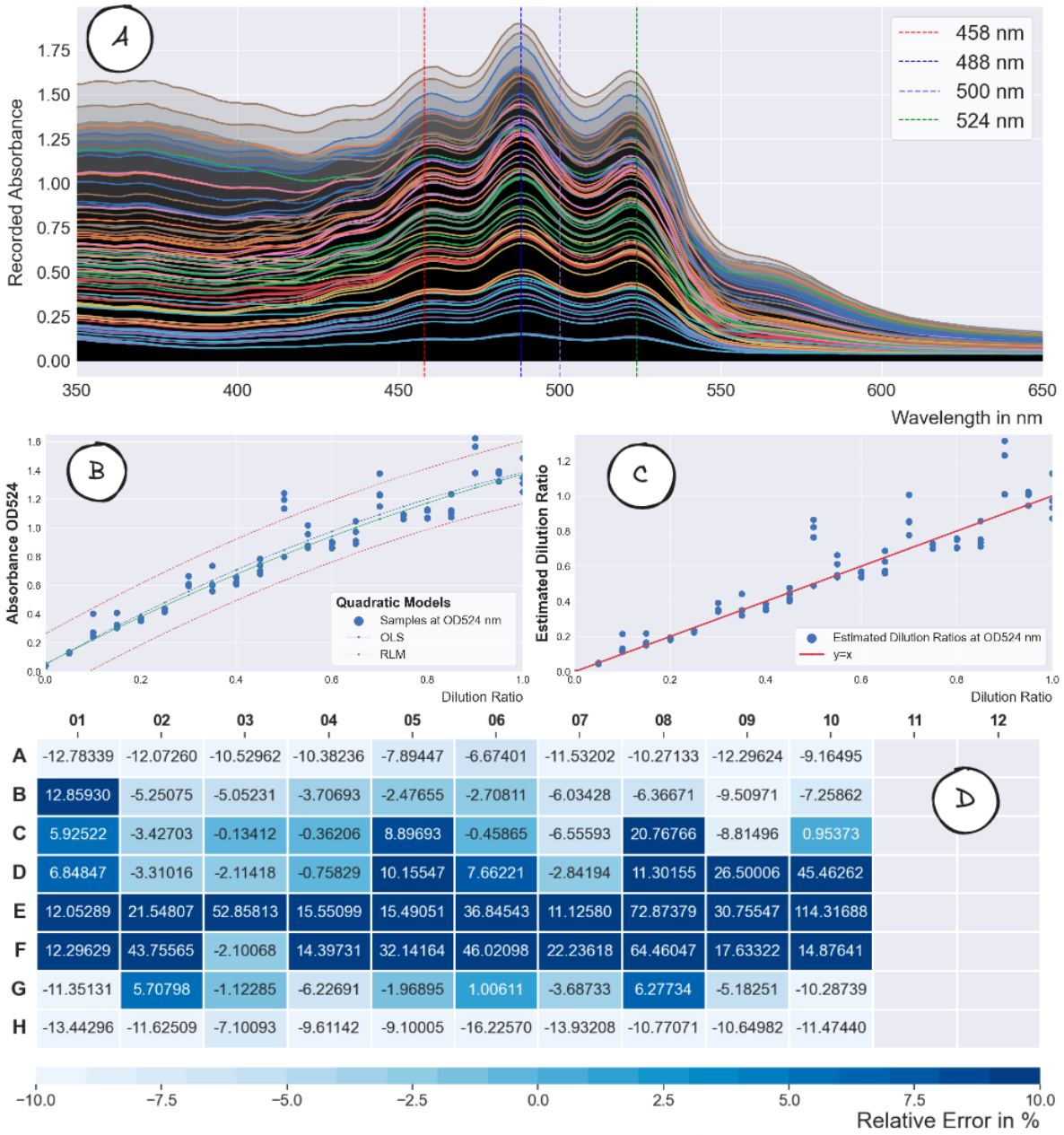
The absorbance spectrum of Run 79 (Figure 4-8-A) still displayed the characteristic shape of the carotenoids – its peaks still at 458 nm, 488 nm and 524 nm, but flatter than for the previous mix). Measurements at 524 nm (8-B) were much more dispersed around the regression curve (some samples being outside the already wide confidence interval of the regression). As a robust regression model was used, the regression captured the trend of most of the points in the dataset, and overlooked the rest – this dual behaviour is reflected in the estimations (Figure 4-8-C) and on the plate map where some wells displayed very large relative errors. The quantitative results (see Table 4-3) showed larger errors than with the food colourant mix (the root MSE was four times as large). Measures of correlation were still high – which is more a consequence of the spread of concentrations afforded by the linear dilution scheme. Overall, the high concentration mix yielded noisy data, indicative of imprecise liquid transfer, whose trend could nonetheless be captured due to good plate design.

Global quality metrics values – Lycopene in DMSO

Run 081 - Plate 01		
At 528 nm	**Model of order 1**	**Model of order 2**
Pearson correlation	0.99695184	0.99705951
Spearman correlation	0.99547733	0.99547734
√MSE	0.025680519	0.02519867
Spectrum method	**Model of order 1**	**Model of order 2**
Pearson correlation	0.99663404	0.99673157
Spearman correlation	0.99456228	0.99456228
√MSE	0.02702327	0.02660369

Run 079 - Plate 01		
At 528 nm	**Model of order 1**	**Model of order 2**
Pearson correlation	0.96857428	0.96053439
Spearman correlation	0.96810563	0.96810563
√MSE	0.08588909	0.10094636
Spectrum method	**Model of order 1**	**Model of order 2**
Pearson correlation	0.95149644	0.93771964
√MSE	0.11109348	0.13448692

Table 4-3 – Global quality metrics values – Lycopene in DMSO. Top: Run 081- Plate 01 (Lycopene in DMSO - Low concentration mix). **Bottom:** Run 079 - Plate 01 (Lycopene in DMSO - High concentration mix). Instead of the MSE which yields small values for the dilution ratios, the square root of the MSE was used.



	01	02	03	04	05	06	07	08	09	10	11	12
A	-12.78339	-12.07260	-10.52962	-10.38236	-7.89447	-6.67401	-11.53202	-10.27133	-12.29624	-9.16495		
B	12.85930	-5.25075	-5.05231	-3.70693	-2.47655	-2.70811	-6.03428	-6.36671	-9.50971	-7.25862		
C	5.92522	-3.42703	-0.13412	-0.36206	8.89693	-0.45865	-6.55593	20.76766	-8.81496	0.95373		
D	6.84847	-3.31016	-2.11418	-0.75829	10.15547	7.66221	-2.84194	11.30155	26.50006	45.46262		
E	12.05289	21.54807	52.85813	15.55099	15.49051	36.84543	11.12580	72.87379	30.75547	114.31688		
F	12.29629	43.75565	-2.10068	14.39731	32.14164	46.02098	22.23618	64.46047	17.63322	14.87641		
G	-11.35131	5.70798	-1.12285	-6.22691	-1.96895	1.00611	-3.68733	6.27734	-5.18251	-10.28739		
H	-13.44296	-11.62509	-7.10093	-9.61142	-9.10005	-16.22570	-13.93208	-10.77071	-10.64982	-11.47440		

Figure 4-8 – Results for the typical Run 079 - Plate 01 (high concentration mix). (A) Absorbance spectra for all 80 samples in the plate. **(B)** Correlations between the target dilution ratios and measurements at 524 nm (peak of spectrum). **(C)** Estimation of the real dilution ratios (the Y=X line is plotted in red for comparison). **(D)** Relative errors for all sample plates mapped onto the plate.

In addition to the analysis of liquid-handling errors, these runs (and the runs for lycopene-DMSO mixes using the geometric dilution scheme) were checked for spectrum stability. The absorbance spectrum of the mix is a strong indication of the state of the mix – so replicability requires such stability (else comparisons are impossible).

The variation present in the absorbance spectrum was unexpected (see Figure 4-9). It was expected that mix concentration would affect the spectrum (as in Figures 4-7 and 4-8). But

the influence of the age of the mix was not expected to be so pronounced after taking the precaution of storing all mixes in the dark in a fridge. Older mixes (2 weeks and older) essentially lost their distinctive peaks. At the same time, fresh mixes yielded similar absorbance spectra (albeit subject to the effects of mix concentration). It was therefore concluded that a fresh mix preparation was essential for the collection of replicable results.

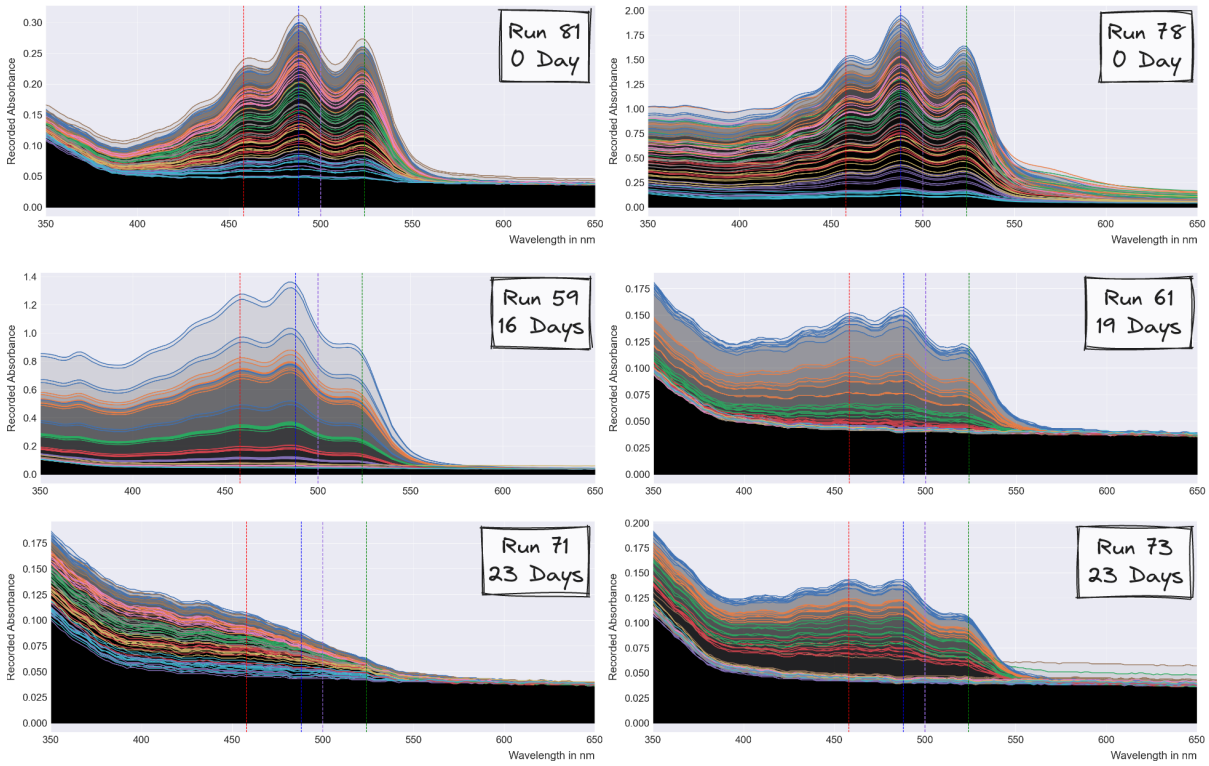


Figure 4-9 - The Spectral Jungle of Lycopene in DMSO. For each assay, the absorbance spectra for all 80 samples in the plate is plotted. The age of the mix used was estimated as the number of days between the creation of the original mix and the date of the assay.

Before conducting the final study, a degradation study was run. Its purpose was twofold. First, to determine a safe time scale for the processing of the plates and their measurements. This, in turn, helps decide the number of technical repeats (the more repeats the better in general, provided the mix retains its properties while the measurements are conducted). Second, to give further insight into the ageing effect of the mix. The degradation study was conducted on several plates with both low concentration mixes and high concentration mixes – since concentration was a factor in the appearance of the absorbance spectrum. The study consisted of monitoring a plate over 12 hours and measuring every hour the entire absorbance spectrum (350 nm to 650 nm). All assays used fresh preparations – less than an hour old to remove all prior ageing effects – and were run at room temperature – see Appendices C4 - Section 5 for the detailed results.

We postulate that the main driver for the transformation is a cristallisation process . Such a process would be concentration-dependent and would be compatible with the mentions of micro-crystals. A study of how solvents affect absorbance spectra of carotenoid (Zang, Sommerburg & van Kuijk, 1997) pointed out the relation between carotenoid concentration and absorbance could become highly non-linear, and presented ‘differences in solubility leading to the formation of micro-crystals’ as a possible explanation. Furthermore, the older the mix, the bigger the clumps of lycopene are – and the more denatured the spectrum. Some degradation by oxidation – mentioned in several publications – could not be ruled out as a secondary driver, but failed to explain the concentration-dependence.

4.5 Phase 3 - lycopene in DMSO - final study

The findings from the previous phases were incorporated into the following protocol. The protocol comprises three levels – each level is designed to account for a type of variability. When data were collected according to the protocol, all measurement repeats could be considered either technical repeats or measurements carried out on similar objects. Minimising the variation between batches. Only lycopene mixes (high and low concentration) that were prepared according to the protocol specified in the Methods sections were used. Data collected in the previous phases confirmed that they behaved as expected in terms of linear response to concentration. The absorbance spectrum was found to be a crucial piece of information on the state of the mix. The protocol only used fresh mixes (assumed free of all ageing effects). Their absorbance spectra were nonetheless checked a posteriori for sameness.

We applied the identical treatment to the plates from the same batches. As in the previous phases of the study, plates were filled from the same mix and processed in parallel on the same platform, at the same time, and using the same version of the protocol. The process was conducted in a time short enough that all degradation effects were negligible.

Minimising the variability between measurements. From the degradation study it was concluded that degradation was minimal over a 30-40 minute time scale. Measurements were therefore repeated every 2 minutes 6 or 10 times in a row in a Clariostar platform (Clariostar A) and then as many times in another (Clariostar B) in order to account for the choice of instrument.

Before generating the final data for the construction of the standard curve, a final series of runs was conducted to investigate the outliers (met with high concentration mixes in the previous phase). The goal was to assess whether the outliers could be explained as simple instrument issues (tackled with standard outlier detection), or by temporary phenomena which would disappear over a practical time scale (e.g. temporary solubility issues that will resolve themselves), or by more permanent issues (e.g. transfer of the wrong volume of

mix).

Results (see Appendices C4 - Section 5) show a more complex phenomenon. Successive reading for the same samples returned varying values – many regressing towards the trend of the dataset from the outset, some deviating (over a few measurements) before returning to the trend. A few did not return to the trend – at least over the timescale of the measurements. The phenomenon was more marked for high concentration mixes. Since the samples which values deviate over the timescale of the measurement were not located in the same positions on the plate, instrument malfunctions could be ruled out. We believe this is indicative of solubility issues in the wells – inhomogeneities (agglomerates) moving and interfering with the absorbance measurements. This also suggests that more thorough mixing in the wells should have been conducted.

These results led to the development of an additional methodology, so measurements for the same sample could be aggregated into a composite value that is not influenced by such fluctuations.

Instead of using statistics such as the median, our method aggregated repeat values on the basis of how much they deviated from the trendline of the whole dataset and, for each well, selected the value that deviated the least. Using this approach, for the majority of samples seen regressing towards the trend, the last data point is selected. Conversely for the samples with different trajectories, whose repeat values may cluster away from the trendline, a safer, more consistent value is returned. In our view such a strategy to aggregate repeat measurements, (based on dataset consistency and trend), is applicable to other complex mixes subject to extreme solubility issues.

Results of Figure 4-10 (two runs with high-concentration mixes – the most prone to errors) show the strategy to be highly effective. It selects among all the repeats and returns values which are close to the trendline – leading to much better quality metrics (the root MSE being reduced by a factor of 2 or 3). It is worth noting that not all samples are brought back to the trendline – in the significantly noisier Run 85 (Figure 4-10 - C and D), some of the samples for the highest dilution ratios remain very far off trend (either due to very high solubility issues, or failed liquid transfer).

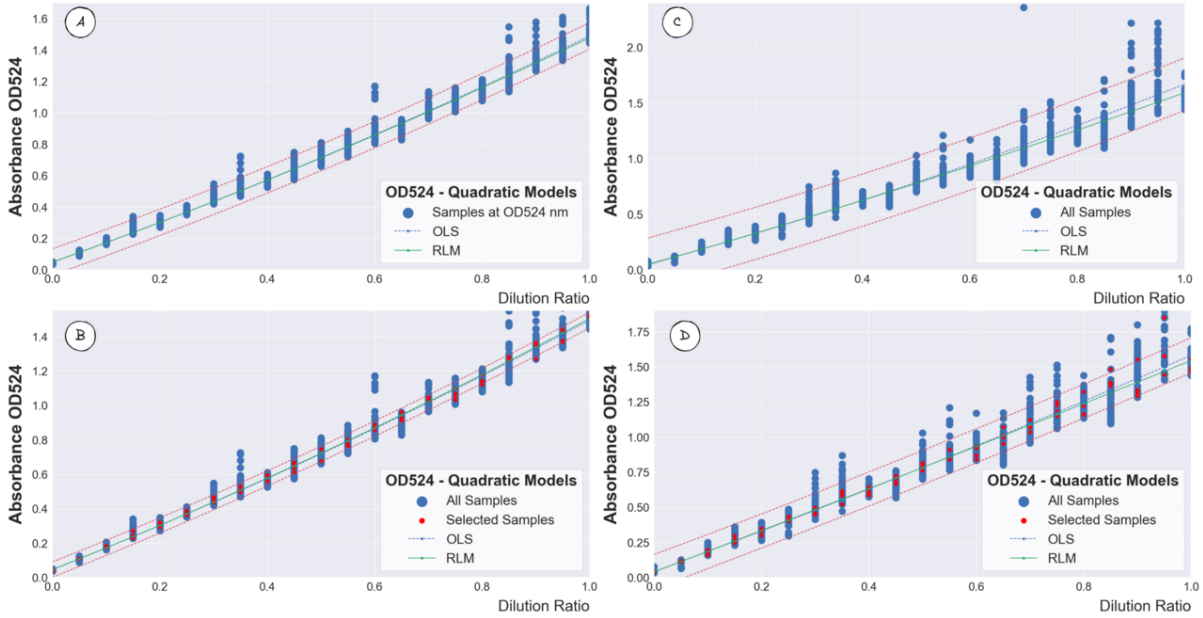


Figure 4-10 – Combining multiple repeat measurements. The standard curves were computed for two high-concentration mixes (Run 90 - left and Run 85 - right) without (top row) and with (bottom row) the aggregation method. **(A)** Run 90 - 20 repeats - without aggregation. **(B)** Run 90 - 20 repeats - with aggregation. **(C)** Run 85 - 12 repeats - without aggregation. **(D)** Run 85 - 12 - with aggregation.

Finally all the results for the assays with a high number of repeats were aggregated to estimate the absorbance spectrum of lycopene and construct the final standard curve (Figure 4-11). As before, we used 524 nm for the standard curve. The absorbance spectrum exhibits the well-known three peaks (located at 458 nm, 488 nm and 524 nm).

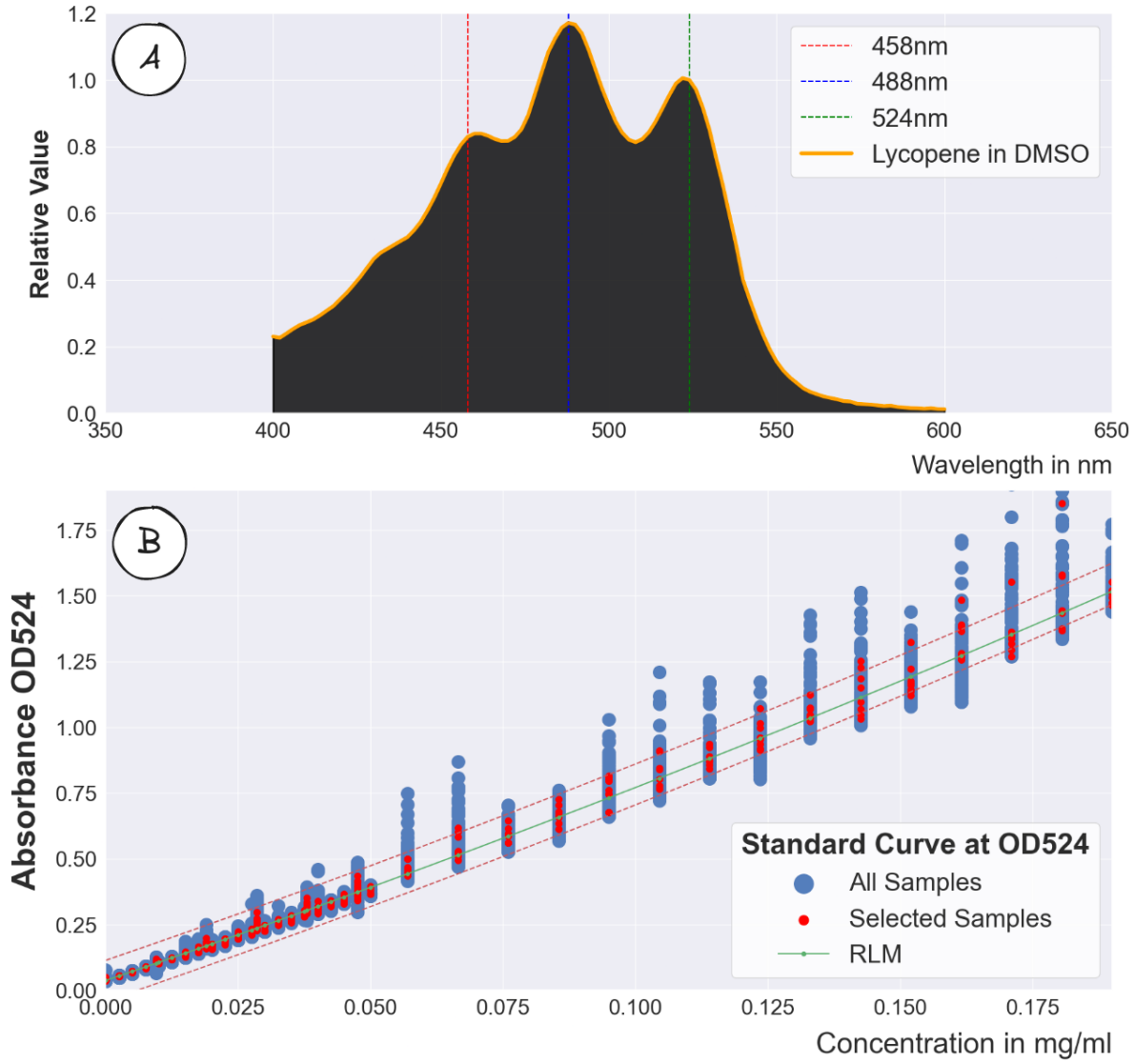


Figure 4-11 – Spectrum of lycopene in DMSO and standard curve at 524 nm. (A) The spectrum of lycopene in DMSO - its magnitude was normalised to be 1 at the 524 nm peak. **(B)** The final standard curve generated from several runs at low and high concentrations. For each run, the repeat measurements (plotted in blue) were first combined (the combinations are plotted in red), and then the standard curve was generated by running a final regression with all the combined data.

5. Summary

In this chapter we presented the need for a calibration curve. The lycopene application (see Chapter 2) requires the measurement of the concentration of the lycopene produced by the cell expressing the genetic constructs. In the application pipeline, lycopene is extracted from the cells with DMSO. An absorbance measurement of lycopene in DMSO is carried in a plate-reader and the lycopene concentration value is derived from a lycopene standard curve. The construction of a standard curve requires the controlled generation of a set of samples of known concentrations on measurement microplates – and for the process to be repeated over a series of assays – which is achieved by using automated liquid handling. We developed an automated protocol to create a standard curve because automation offers many advantages: automation allows to implement any arbitrary dilution pattern over a plate in a programmatic way, automation reproduces the same mechanised liquid transfer movements when repeating the protocol, thus ensuring reproducibility of the curve, automation removes mechanised human induced errors.

Section 2 described automatic liquid handling and the challenges that come with the automation of liquid handling. Many issues were presented that have been identified (in the literature) as obstacles encountered during automated liquid handling operations and result in errors. These liquid-handling errors have been categorised and their consequences need to be taken into consideration by protocol designers.

Section 3 discussed the challenges of automating liquid handling of a lycopene and DMSO mix, which qualifies as an atypical liquid known to be difficult to work with from a mechanised point of view. Additionally, the combination of lycopene and DMSO has physio-chemical properties that affects concentration and absorption measurement. Two dilution schemes were implemented and studied. were then introduced. This comprised the geometric scheme and the linear scheme – this highlighted the different sources of errors that might be induced by one scheme or the another.

Section 4 described the study conducted to develop an automation method to create a standard curve. The study was conducted in two parallel tracks, one for each dilution scheme (the geometric dilution scheme and the linear dilution scheme) and each track included a preliminary study (Phase 1) followed by a study of lycopene and DMSO mix (Phase 2) and a final study that produces a lycopene standard curve (Phase 3).

After conducting the preliminary study (Phase 1) of the geometric dilution scheme we concluded that iterative dilutions introduce a class of correlated errors. Additionally, a geometric dilution of lycopene and DMSO introduced even larger errors. Consequently, we

dropped the geometric dilution scheme and pursued the second track with the linear dilution scheme. A workflow to assess dilution errors based on a regression modelling was presented. We then ran many experiments to tune the automated linear dilution protocol and to test the error assessment workflow. Additionally, a study of the spectrum of the lycopene and DMSO mix was conducted to ensure that the mix is stable to make comparison between different runs possible. We ran a degradation study over many hours (12h). Importantly, we discovered that the age of the mix had a very strong effect on the spectrum. We concluded that all assays would then use fresh lycopene and DMSO preparations of less than an hour old to remove all ageing effects.

In a third phase a series of runs were conducted to investigate outliers to understand if outliers could be explained by instrument issues or by a temporary phenomena that would disappear (or not) over some time. The conclusion was that inhomogeneities appeared in the mix (especially at higher concentrations) and that some particles moving were interfering with the absorbance measurements. Sample fluctuations were compensated for by aggregating repeating measurements. Finally (repeat) measurement data from low and high concentration dilutions was aggregated and a final regression with all the combined data to produce a lycopene and DMSO standard curve. Additionally, the conducted studies provided new practical recommendations regarding the automation of liquid-handling.

Chapter 5 Automated Research Workflows

In this chapter we introduce the current need for automated research workflows and then show the needs that we identified in the case-study. These problems can be solved by introducing automated workflows.

In Section 2 we define tasks and workflows and what are protocols and establish the relation and equivalence between the concepts and give two examples of workflows, the computational workflow preceding BASIC DNA assembly and the workflow for automated dilutions for standard curve. Both examples have been presented in the previous chapters (Chapter 3 and Chapter 4).

In Section 3 we introduce our conceptual framework and the concept of digital twins. We show how we can have a digital representation of the physical world and that it is desirable to link the digital representation of a physical object and its observable/observed interactions with other physical objects.

In the last section, Section 4, we define our world and our objects and how we represent experiments in synthetic biology. We define the operational pipeline as the successive actions on physical objects and their observations.

1. Introduction - Where we are Now - What are the Needs ?

1.1 Automated research workflows

For centuries, the iteration of a workflow built upon experimentation and observation and analysis of the resulting data has been the main driver of scientific research. Computational and automation technologies have been central to research workflows and have supported the acquisition, processing and analysis of data. The same computational and automation

technologies can now also control the acquisition of data by designing new experiments or making decisions from observations. (NAS 2022).

The National Academy of Sciences has defined the term ‘Automated Research Workflows’ (ARW) as the integration of tools and technologies that perform tasks that comprise the research process. National Academy of Sciences definition: ‘Automated Research Workflows ARWs integrate computation, laboratory automation, and tools from artificial intelligence in the performance of tasks that make up the research process, such as designing experiments, observations, and simulations; collecting and analyzing data; and learning from the results to inform further experiments, observations, and simulations.’ (Committee on Realizing Opportunities for Advanced and Automated Workflows in Scientific Research et al., 2022).

Automated Research Workflows have the potential to effectively manage and exploit the exponentially increasing volume and availability of data (NAS 2022).

1.2 The rise of scientific workflows

‘The emergence of computational and data-driven science as the third and fourth paradigms increases the demand for modern technologies.’ (Hey, Tansley & Tolle, 2009).

Hey, Tansley & Tolle, (2009) name data-driven science as the fourth scientific paradigm,

computational and data-driven science increase the demand for modern technologies in science with the need for data analysis and statistical methods or data-mining techniques, to discover new knowledge from simulation, observation and experimental data. A data-driven approach to science involves a series of steps: moving the data from the sources to the computational resources, data cleaning, constructing a model, validating the model, visualisation and finally moving the results to a storage system (Liew et al., 2016). Such processes can be modelled as workflows which, as defined by Liew et al. (2016), are a set of interrelated computational and data-handling tasks designed to achieve a specific goal.

1.3 Lessons from the case study

The lycopene case-study from Chapter 2 highlighted some problems that various industries have encountered when transitioning to the adoption of automation. Industries such as the semiconductor industry or the automobile industry have encountered similar problems when bringing automation technologies (mechanisation) in their assembly lines and employing robotic process automation (computerisation) and automating their workflows. We show, with the problems that we have encountered and identified while developing our case-study application, that what has been true for other industries is also valid for engineering biology automation.

Biotechnologies are one of the technological breakthroughs, among others like artificial intelligence, that are *'*now heralding a Fourth Industrial Revolution, with the power to reshape almost every sector in every country.' (Clark, 2019b). The UK positions its industrial strategy to make the most of this global transformation (Clark, 2019b).

Industry 4.0 is a buzzword (Bongomin et al., 2020) popularised by Klaus Schwab from the World Economic Forum (Schwab, 2015) which has been introduced by a team of scientists developing an industrial strategy for the German government (Kagermann, Lukas & Wahlster, 2011). The first industrial revolution started in 1760 with the invention of the steam engine followed a century and a half later by the second industrial revolution which began in 1900 with the invention of the internal combustion engine. This led to an era of rapid industrialisation using oil and electricity to power mass production (Xu, David & Kim, 2018). The third industrial revolution started in 1960 and was characterised with the implementation of electronics and information technology to automate production (Xu, David & Kim, 2018). The fourth Industrial Revolution is building on the digital revolution characteristic of the third industrial revolution. It is characterised by a fusion of technologies that is blurring the lines between the physical, digital, and biological spheres (Xu, David & Kim, 2018).

The fourth industrial revolution (Industry 4.0) is happening and is expected to bring major changes in the future manufacturing and production processes which will benefit from Industry 4.0 main design principles: interoperability, virtualisation, decentralisation, real-time capability, service orientation, and modularity (Indri, Grau & Ruderman, 2018).

Robotic Process Automation (RPA) is the automation of tasks that reproduce the work that humans do (van der Aalst, Bichler & Heinzl, 2018). Ribeiro et al. (2021) define robotic process automation as a 'technique that results in the automatic execution of administrative, scientific or industrial tasks'. Robotic process automation gives many advantages in terms of automating organisational and business processes. The automation of the tasks is done with the help of software robots or AI workers that are able to perform, accurately, repetitive tasks. Human robotics collaboration (Baratta et al., 2023) and robotics process automation (Axmann & Harmoko, 2020) both play a crucial role in Industry 4.0.

When it comes to the field of engineering biology, Gurdo et al., (2023) have recently written that automation plays a central role in synthetic biology and metabolic engineering. Automation not only reduces workflow times but can be exploited as a discovery tool to guide engineering efforts. Other industries have embraced the automation of their workflows and processes. The automotive industry is a good example of an industry that has seen significant changes as a result of introducing automation in the manufacturing process (Hajarnavis, 2018; Raptis, Passarella & Conti, 2019).

Information and communication technologies have penetrated all aspects of industrial and

manufacturing systems and generate very large volumes of industrial data (Raptis, Passarella & Conti, 2019). A notable industry that has faced challenges – when having to deal with an exponential increase of large data volumes, rates, quality, merging, and analytics in the digital space– is the semiconductor industry. Semiconductor manufacturers have been faced with a need for new approaches to data management and use across the Fab (a semiconductor fabrication plant in the semiconductor industry jargon) (Moyne, Samantaray & Armacost, 2016).

The concept of cyber-physical convergence (and the related concept of digital twin which we introduce in Section 3 of this chapter) is a cornerstone of the most disruptive Industry 4.0 innovations. Consecutively, data management is one of the key enablers for the realisation of this concept. Data coming from the real physical world (collected through sensors and other data generating sources) are seamlessly transferred into the cyber world where they are treated to adapt cyber applications and services to the physical context, and then modify the physical world itself (through actuators and robotic elements). Digital twinning – a collection of tools and methodologies to create the virtual models for physical objects in the digital space to simulate their behaviours – leads to the cyber-physical convergence (Qi & Tao, 2018).

Data is what enables the integration of the two worlds (physical and cyber). Data enables digital twins to interact, it is what enables digital twins to represent their physical counterparts and what enables knowledge extraction. The natural evolution of the data enabling industrial technologies and services leads to the generation of huge amounts of data; data of many different volumes and types, increased data traffic and criticality. Data is what allows the transition from machine automation to information automation and then to knowledge automation. Data also enables fast control cycles for applications like zero-defect manufacturing (very strict quality control). It allows information sharing across production sites of the same factory operator, or traversal information sharing across multiple production chains composed by different stakeholders (Raptis, Passarella & Conti, 2019). In the past decades, large amounts of data have been generated in the industrial environments, through the wider use of network controlled systems (NCS) and increased and cheaper data storage. At the very beginning, those large amounts of data were rarely used for detailed analyses, and were instead only used for routinely technical checks and process log fulfillments. Later, awareness of the importance in extracting information from data became a key asset for Industry 4.0 (Ge et al., 2017). There has been an exponential increase in the number of data sources. Furthermore, enhancements in data networks and infrastructure have contributed to the substantial accumulation of data (Wang, 2018). Data is being systematically archived and generated in real-time. However Wang, (2018) points out that data is not equal to value. To create value with data requires analytics. Analytics enables the transformation of data into

actionable insights.

Because our application makes use of combinatorial metabolic engineering techniques and iterating through DBTL cycles and replicating measurement to reduce the error, we need a lot of data.

We aimed to automate most of our processes. We employed mechanisation through the use of robotic equipment to increase throughput and ensure repeatability of the pipetting movement of liquid-handling processes (to remove any human input variability) and decrease costs. When implementing our case-study, we encountered the same problems that other industries have faced when dealing with large volumes of data (i.e. moving data from source (measurement equipment etc..) to storage, to processing data steps, to data analysis, to visualisation). We have observed that data acquisition and processing require strict data structuring. For getting insights from our data analysis we need to be able to link the various data produced along the pipeline and we need to be able to question the data and establish the dependance between data.

The various tasks along the operational pipeline requires us to work on different locations (different workcells or different places of data treatment, i.e. the lab, the biofoundry, an office or the cloud). We need to manage the complexity of the pipeline. Different tasks and positions require expertise and specialisation. We need to manage the access of data-sharing among the different actors along the pipeline. As a consequence, we need automated workflows to deal with the multiple data processing steps. With our case study we have been at the limit of what is manageable manually by a single human operator.

2. Tasks and Workflows

2.1 Tasks

2.1.1 Definition

We define a task as a specific action that needs to be performed. A task itself can be composed of multiple steps of smaller actions to perform.

An example of a task in a wetlab experiment is: *incubate the tube at 37°C*.

In our framework tasks rarely stand by their own and are part of workflows which we will define in the next section (Section 2.2). Tasks are essentially the atomic elements of workflows and successive tasks compose workflows.

A task is a step in the overall process of completing an experiment or an assay as seen in Figure 5-1A.

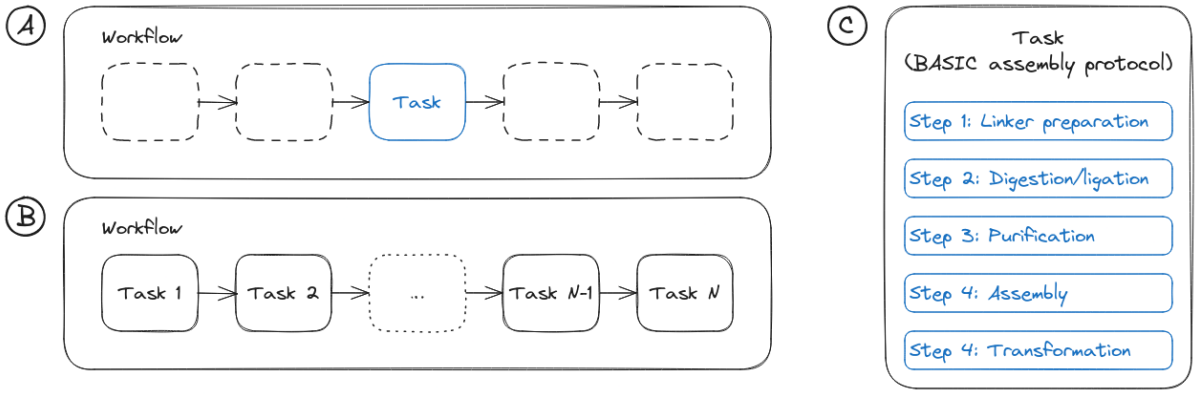


Figure 5-1 – Representation of tasks. (A) Representation of a task (in blue) within a workflow. **(B)** Diagram of a workflow with different tasks defined in a specific order ranging from 1 to N. **(C)** An example of a task with different steps (in blue) of action to perform.

The order in which the different tasks are performed, is defined in the workflow. There are different steps of actions to perform within a task. The diagram in Figure 5-1B represents a succession of N number tasks which make a workflow.

Within a workflow, an example of a task is a laboratory protocol comprising many steps. Tasks are an essential part of a workflow, as they help to ensure that all necessary steps are taken to complete an experiment. Tasks can be of various types: manual wet lab tasks, operated by a scientist, automated wet lab tasks, operated by some robotic equipment (these tasks are usually found in biofoundries workflows) and computational (dry lab) tasks.

Figure 5-1C illustrates the example of a wetlab task (the BASIC DNA assembly protocol)

with the different steps of action to perform in order to complete the task.

We define a task as an operation having one or more inputs and one or more outputs.

Figure 5-2A shows a task with an input and an output. Tasks can have more than one input and more than one output. Figure 5-2B illustrates a task with multiple inputs and outputs and Figure 5-2C shows a real-world wetlab example of a task – a PCR setup mix – with its different inputs and outputs of labware, chemical reagents and DNA parts.

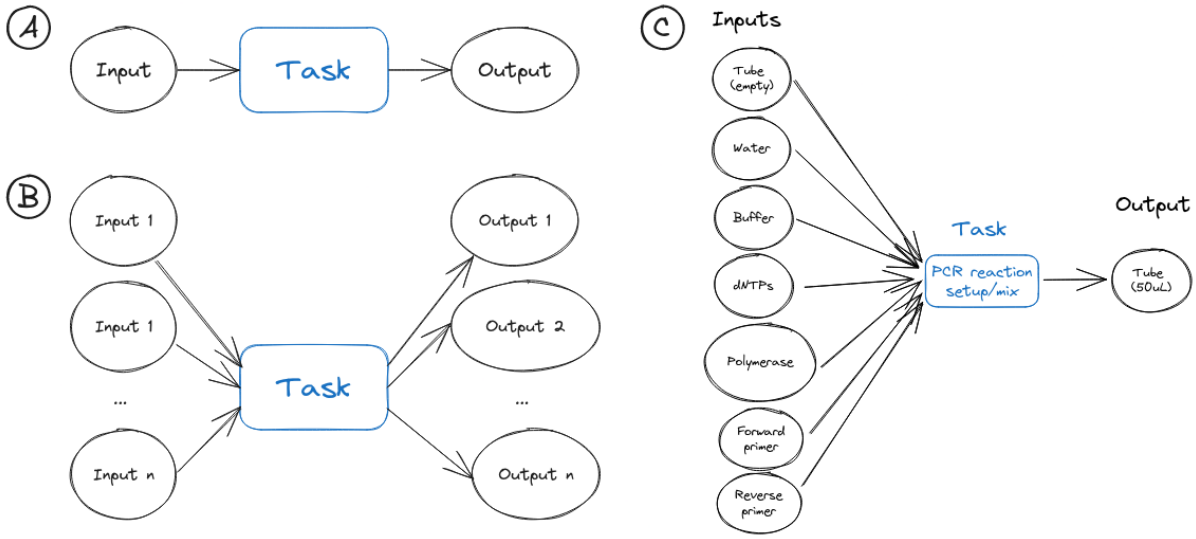


Figure 5-2 – Tasks with inputs and outputs. (A) A task with one input and one output. **(B)** A task with multiple inputs and outputs. **(C)** Example of a real-world wetlab task (a PCR setup mix) with its inputs (labware, chemical reagents and DNA) and outputs (the tube containing the PCR mix).

2.1.2 Dependency

Task dependencies refer to the relationships between tasks in a workflow. These dependencies determine the order in which tasks need to be executed in order to complete the workflow successfully.

In Figure 5.3-A we illustrate the concept of dependencies of 3 successive tasks: Task A, Task B and Task C. Task C depends on Task B which in turn depends on Task A. Task A is upstream of Task B which is upstream of Task C. Task C is downstream of Task B which is downstream of task A.

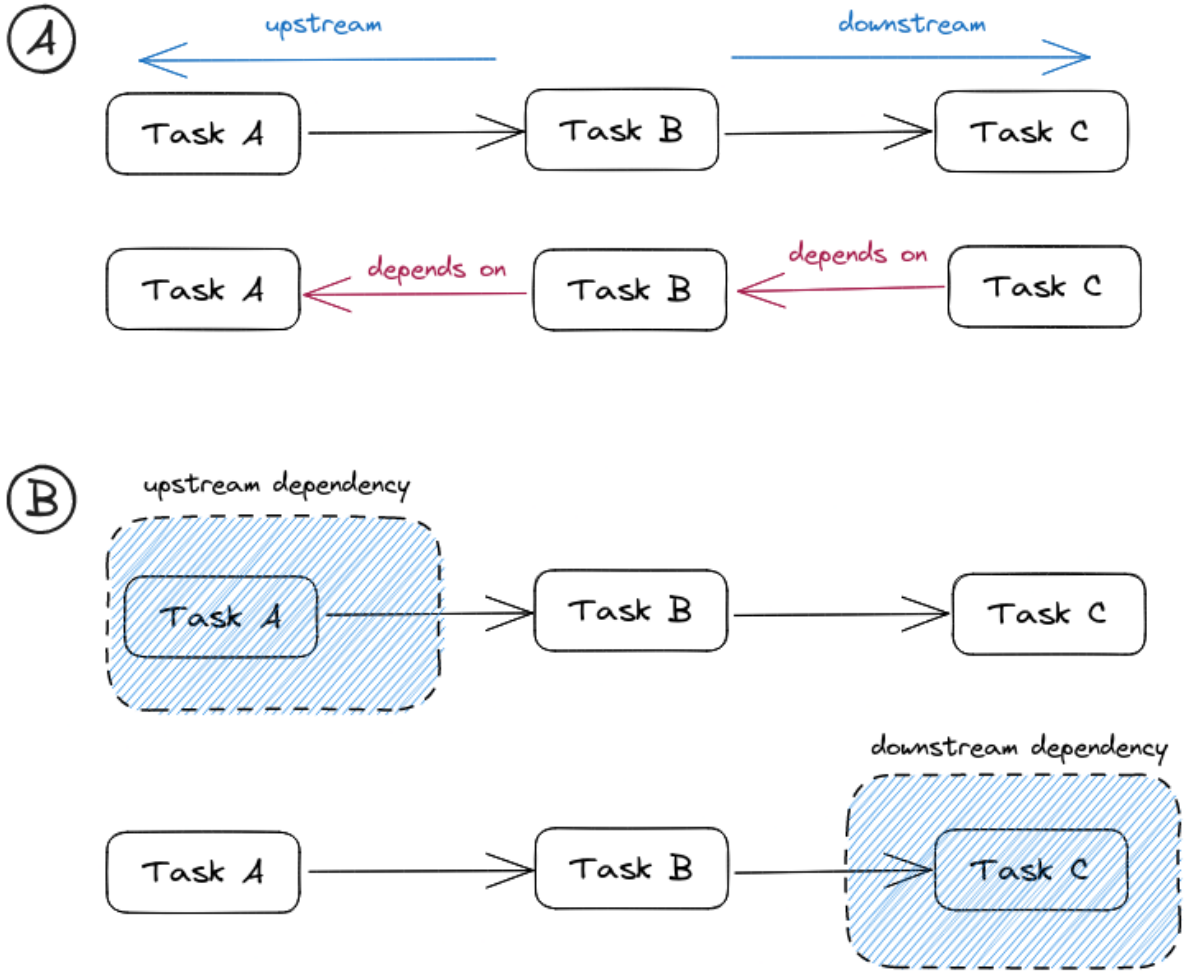


Figure 5-3 – Task dependencies. (A) The red arrows show the dependency relationship between the tasks. Blue arrows represent the upstream and downstream direction from Task B. (**B)** Illustration of upstream and downstream task dependencies.

There are two types of task dependencies as illustrated in Figure 5.3-B:

- Upstream (or predecessor) dependencies: An upstream dependency means that one task must be completed before another task can begin. In other words, the second task is dependent on the completion of the first task.
- Downstream (or successor) dependencies: A downstream dependency means that one task cannot be completed until another task has been completed. In other words, the first task is dependent on the completion of the second task.

For example – as illustrated in Figure 5.4-A – if Task 1 is ‘Write content’ and Task 2 is ‘Edit content’, Task 2 cannot start until Task 1 is completed. Task 1 is an upstream dependency of Task 2

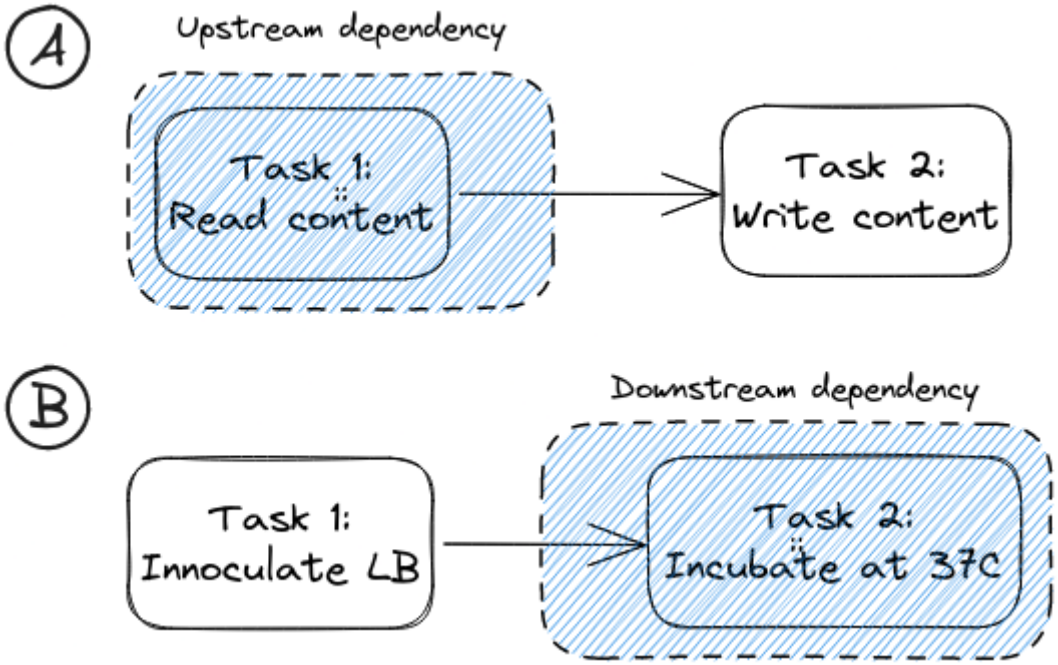


Figure 5-4 – Examples of Upstream and Downstream dependencies. (A) Example of a computational workflow for reading and writing a file. **(B)** Example of a wetlab workflow for inoculation and incubation of a culture.

In the example of dependencies shown in Figure 5.4-B, if Task 1 is 'Innoculate LB media with *E. coli*' and Task 2 is 'Incubate at 37°C and grow the *E.coli* liquid culture', Task 2 cannot be completed until Task 1 is completed. Task 2 'Incubate at 37°C' is a downstream dependency for Task 1

Tasks can be dependent on one other. The output of a task can be necessary as the input of one of its downstream tasks. Some tasks cannot be accomplished without their upstream dependencies being completed. We define task dependencies to know which task is dependent on which. Having task dependencies defined allow us to understand which output of a task generates which input of another task.

2.2 Workflows

2.2.1 Definition

As defined by IBM (IBM, 2023): 'A workflow is a system for managing repetitive processes and tasks which occur in a particular order. They are the mechanism by which people and enterprises accomplish their work, whether manufacturing a product, providing a service, processing information or any other value-generating activity.'

Liew et al. (2016) identify three components of a workflow: a list of tasks, a set of

dependencies between the interconnected tasks (the flow), and the set of data resources used to generate or terminate the flow.

A workflow can be visualised as a diagram or a flowchart which contain tasks as seen in Figure 5-5, it illustrates the workflow diagram of the BASIC DNA assembly protocol seen in Chapter 2 (Storch et al., 2015).



Figure 5-5 – Workflow diagram of the BASIC DNA assembly protocol.

2.2.1 Two kinds of workflow structures

There are two classes of workflow structure: control flow and data flow (Shields, 2007). Shields (2007) describes the similarities between the two classes as they both specify the interaction between the individual activities that comprise the workflow but points out the differences in the methods of implementation of their interaction.

- Control-driven workflows, or control flows, describe the connections between the tasks in the workflows as the transfer of control from one preceding task to the one that follows. They consist of control structures such as conditionals and loops.
- Data-driven workflows, or data flows, are designed to support data-driven applications. Dependencies in the workflow represent the flow of data between the tasks. Tasks in data-driven workflows are data producers or data consumers or both.

Some workflows have a hybrid representation and are based on both control flow and data flow (Shields, 2007).

2.2.2 Workflow architectures

A wide range of Workflow Management Systems (WMSs) have been developed and (Wratten, Wilm & Göke, 2021) have reported more than 150 workflow managers currently in use and under development. (the listing repository contains 340 entries at the time of writing) (Github topic, 2023. Notable scientific workflow management systems are: Pegasus (Deelman et al., 2015), Kepler (Ludäscher et al., 2006), Taverna (Oinn et al., 2006), Galaxy (Goecks et al., 2010) which are well established and widely used in multiple domains (Liew et al., 2016). Additional WMSs have been reviewed in (Liew et al., 2016) but ‘a review of all WMSs is not feasible’ (Liew et al., 2016) given the large number of workflow systems that have emerged in the past decade.

As a result of various workflow studies (Liew et al., 2016) identified different roles of scientific workflows: workflows support collaborative research and enable scientists to share automated processes such as data analysis, workflow enable the automation of steps to allow the repetition of *in silico* experiments, workflows can integrate resources from various experimental or computational platforms and workflow can handle large volumes of data and complex computations.

In the previous two decades, workflow technologies have been used by scientific communities to automate computational experiments running on distributed high-performance computing infrastructures such as grids, cloud and high-performance computing (HPC) resources (Liew et al., 2016).

(Liew et al., 2016) give an interesting taxonomy of workflows which can be split in five main categories: processing elements, coordination methods, workflow representations, data processing models, and optimization stage.

2.2.3 Patterns

The different patterns of task dependencies can be modelled in a workflow:

- one-to-one task dependency
- one-to-many task dependencies
- many-to-one task dependencies
- conditional loops (also called conditional repeats)

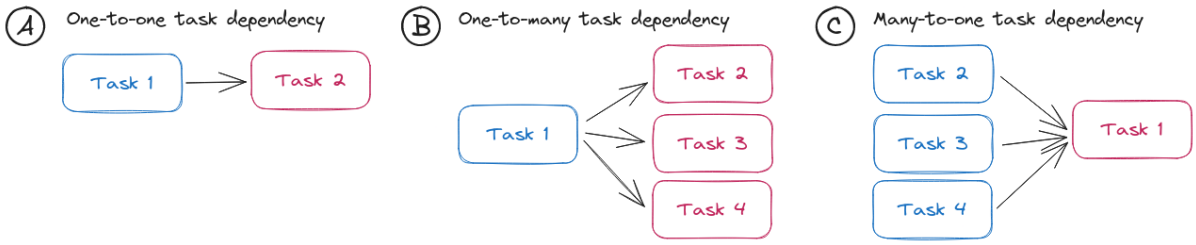


Figure 5-6 – Task dependency patterns to model workflows. (A) One-to-one task dependency. **(B)** One-to-many task dependency. **(C)** Many-to-one task dependency.

(i) One-to-one task dependency

This task dependency is the most direct task dependency where one task is dependent on one upstream task. In the example in Figure 5-6A Task 2 is dependent on Task 1.

(ii) One-to-many task dependency

This task dependency is when multiple tasks are dependent on one upstream task. In the

example shown in Figure 5-6B, Task 2, Task 3 and Task 4 are all directly dependent on Task 1.

(iii) Many-to-one task dependency

This task dependency is when on task is dependent on multiple upstream tasks. In the example in Figure 5-6C, Task 1 is directly dependent on Task 2, Task 3 and Task 4.

(iv) Conditional loops

As represented in Figure 5-7, workflows can contain conditional repeats which introduce a loop in the workflow. The condition decides if the task gets repeated or not.

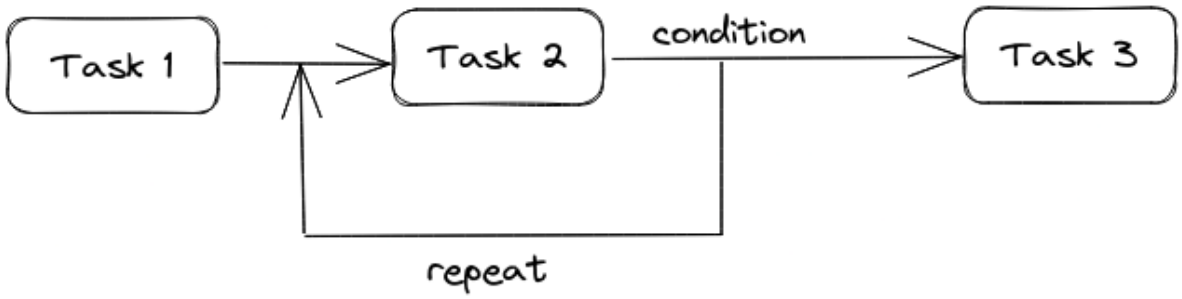


Figure 5-7 – A workflow with a conditional loop.

Figure 5-8 illustrates the example of a workflow with an incubation task and a measurement of OD_{600}, incubation is repeated until a certain threshold value of OD_{600} is met.

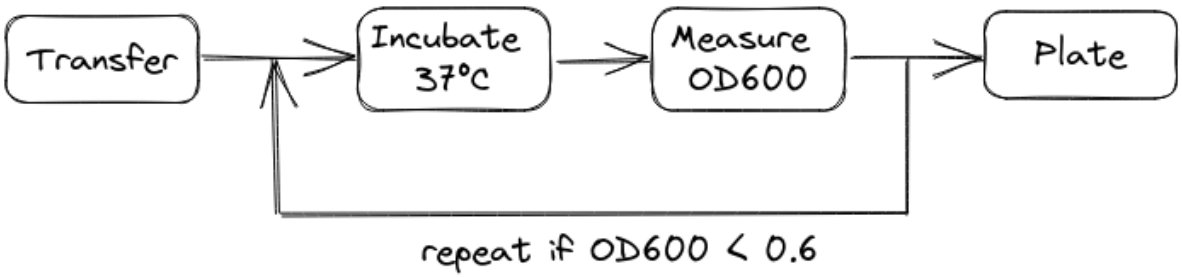


Figure 5-8 – Example of a wetlab workflow with a conditional loop. The workflow loop comprises an incubation task and a measurement of OD600. Incubation is repeated until the threshold value of 0.4 is reached when measuring OD600

2.3 Tasks, workflows and composability

2.3.1 Granularity

It is important to define tasks and workflows at an appropriate level of granularity because it is required to have a level of reading of the workflow that conveys the main steps of an application (or an experiment).

Defining tasks at too high a level will lead to tasks that are too broad, and won't give an appropriate and human comprehensible overview of the application. Too many details in the workflow makes it difficult to split it into tasks that can be delegated to a unique resource. A level of too few tasks will define tasks that are either too broad and won't allow enough QC between the tasks. If QC processes are too sparse (i.e. too many tasks of the workflow are left between two QC processes), the detection of an error (i.e. the failure of a QC process) would have led to additional task work being wasted.

Becker & zur Mühlen (1999) define three levels of granularity for application frameworks (and make level naming analogies with geology) – the Application level (Rocks), the Module level (Stones) and the Function level (Sand) – defining three levels of granularity is enough to cover the various applications and workflows that we have encountered in the biofoundry. However, to anticipate future scaling of experiments in a biofoundry and the possibility of a hierarchical or specialised separation of work, we give the possibility in our framework to define as many levels of granularity as desired.

We define levels of granularity both at the task level and at the workflow level.

Figure 5-9 illustrates the concept of the granularity of the representation of a task. Figure 5-9A shows two tasks: Task 1 and Task 2 each being composed of a certain number or steps (3 steps for Task 1 and 4 steps for Task 2).

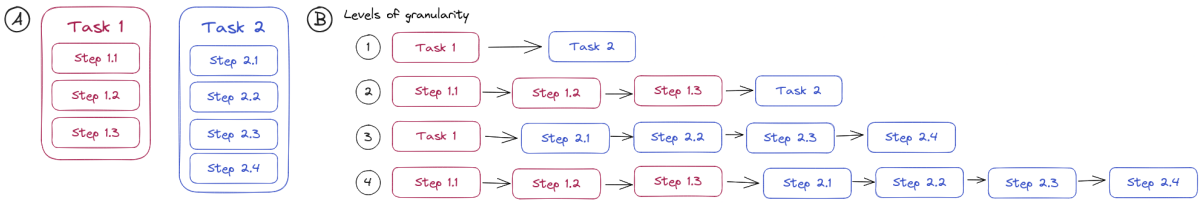


Figure 5-9 – Task level of granularity. (A) Two different tasks: Task 1 and Task 2 each composed of different steps. **(B)** Different levels of granularity of tasks for equivalent workflows.

In Figure 5-9B, each of the workflows, numbered from 1 to 4, are equivalent but are represented at different levels of task granularity. The graph of Workflow 1 has two nodes:

Task 1 and Task 2 whereas the graph of Workflow 4 has 7 nodes. Each of these four workflows would produce a similar output but they vary in their granularity of task representation.

When defining a workflow to structure an experiment, the question comes down to how many individual tasks should the experiment be split into. There are no rules to how small or large a task should be but we tend to split tasks into atomic processes that consume an input and produce an output. A convenient way of splitting tasks is one where one resource (wherever it is human, automated or computational) can be assigned to one task at a time. Some pieces of equipment such as liquid handling robots or computers can take advantage of parallelisation and execute multiple similar tasks at the same time and process multiple parallel inputs.

We define a sub-workflow as a subset of connected tasks of a workflow (see Figure 5-10 for an example of sub-workflow). By extension, because workflows are composed of tasks, workflows can also be represented at different levels of granularity.

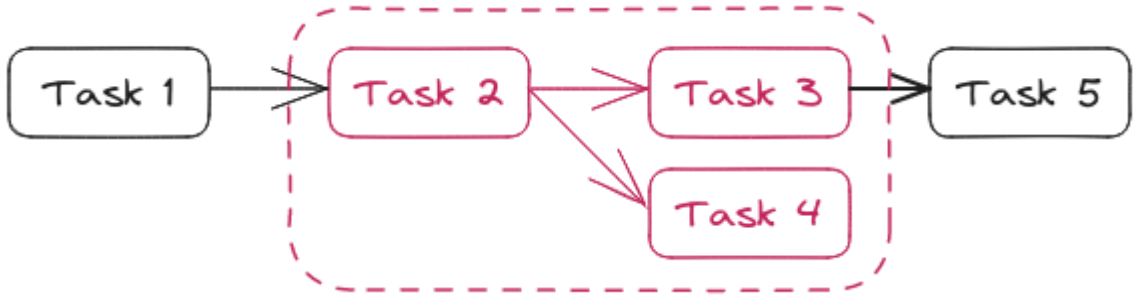


Figure 5-10 – Sub-workflow. The workflow comprises five tasks (Task 1 to Task 5), the subset of tasks (Task 2, Task3, Task 4) highlighted in red constituted a sub-workflow of the workflow.

The wetlab workflow illustrated in Figure 5-11 shows the concept of sub-workflows and workflow levels of granularity. The first level shows a workflow that comprises three tasks. Task 2 is defined by a sub-workflow comprised of five tasks (Task A, Task B, Task C, Task D and Task E). The second level represents the same workflow with Task 2 expanded as a sub-workflow. At the lowest level of granularity represented in this example (Level 3), Task B can be expanded to another sub-workflow comprising six tasks (Task a, Task b, Task c, Task d, Task e, Task f and Task g).
A same workflow can be defined or represented at different levels of granularity.

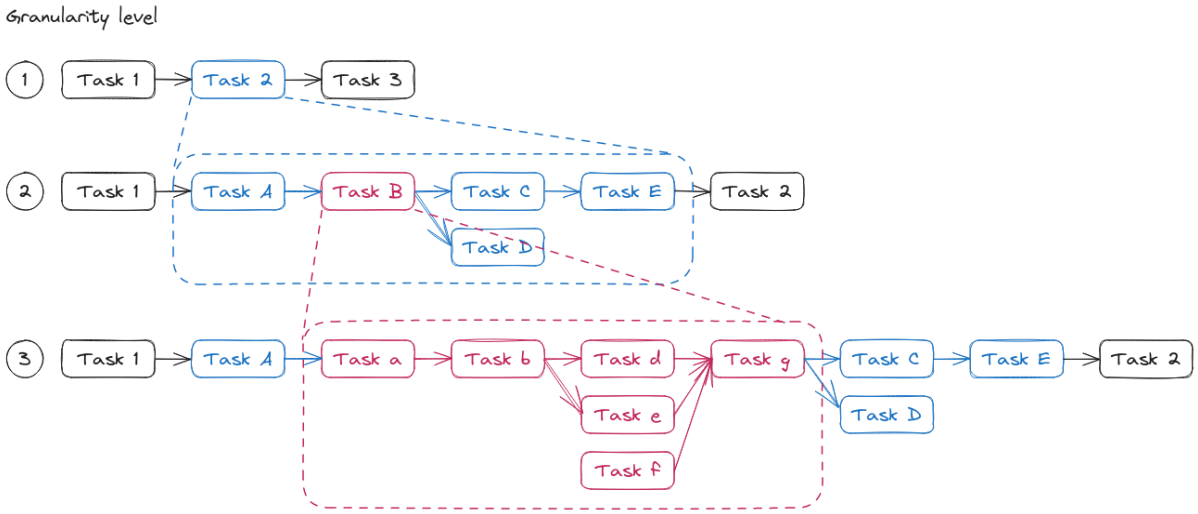


Figure 5-11 – A workflow represented at different levels of granularity. This example of a workflow illustrates a simple workflow comprising three tasks at the highest level of granularity (1). At a lower level (2), 'Task 2' can be expanded to an equivalent sub-workflow (illustrated in blue). The lowest level of granularity represented (3) shows the same workflow with its 'Task B' expanded to its equivalent sub-workflow (illustrated in red). All three levels of representation of the workflow are equivalent functional workflows.

2.3.2 Modularity

The industrial revolution was a time when factories were built and machines were created to do more work than people could do manually. This allowed for goods to be made faster and in larger quantities, which helped the economy grow. The division of labour involved different people doing different tasks in factories so that each worker could focus on their specific job. This allowed products to be made quickly and with less error.

Modularity allows to build complexity. We can replace manual tasks with automated tasks. We can replace one task with another equivalent task. As long as we define interfaces, the tasks becomes independent of its execution or implementation. Modularity allows the execution of a task independently of its executor resource. Modularity of tasks in an automated lab can be achieved by designing the system to perform each task independently. This allows for tasks to be performed in any order, or even simultaneously, with minimal interaction between them. The individual components of the system should also have standardised interfaces so that they can easily communicate with one another. Finally, it is important to ensure that all processes are well-documented and understood by those who will be using the automation system.

For example, the same liquid transfer task can be done on a traditional liquid handling robot such as the Cybio Felix or on an Opentrons OT-2, the function of transferring remains the same and both tasks result in the same volumes being transferred. As such, a pipeline designed with modularity in mind will be equivalent no matter what the underlying equipment performing the same functionality is.

It can be argued that certain piece of equipment perform some tasks better than other, but on an equivalent level of requirement.

Computational tasks are, at their essence, programmable tasks and thus the nature of composable code applies to the modularity of computational tasks. Programmable tasks can be broken into sub-programs that can be grouped and defined in reusable libraries and modules. Even if the different tasks are written in different programming languages or built on different technology stacks, the definition of common interfaces and common communication protocols (e.g. HTTP) allows for their interoperability.

Task modularity

We consider two tasks as equivalent if they serve the same function and share the same inputs and outputs. In the example illustrated in Figure 5-12 we illustrate the concept of task modularity by showing two equivalent tasks that are interchangeable. The sequence of tasks comprise four tasks: Task 1, Task 2, Task 3 and Task 4. Task 3 (FIgure 5.12-A) has an equivalent task Task 3'.

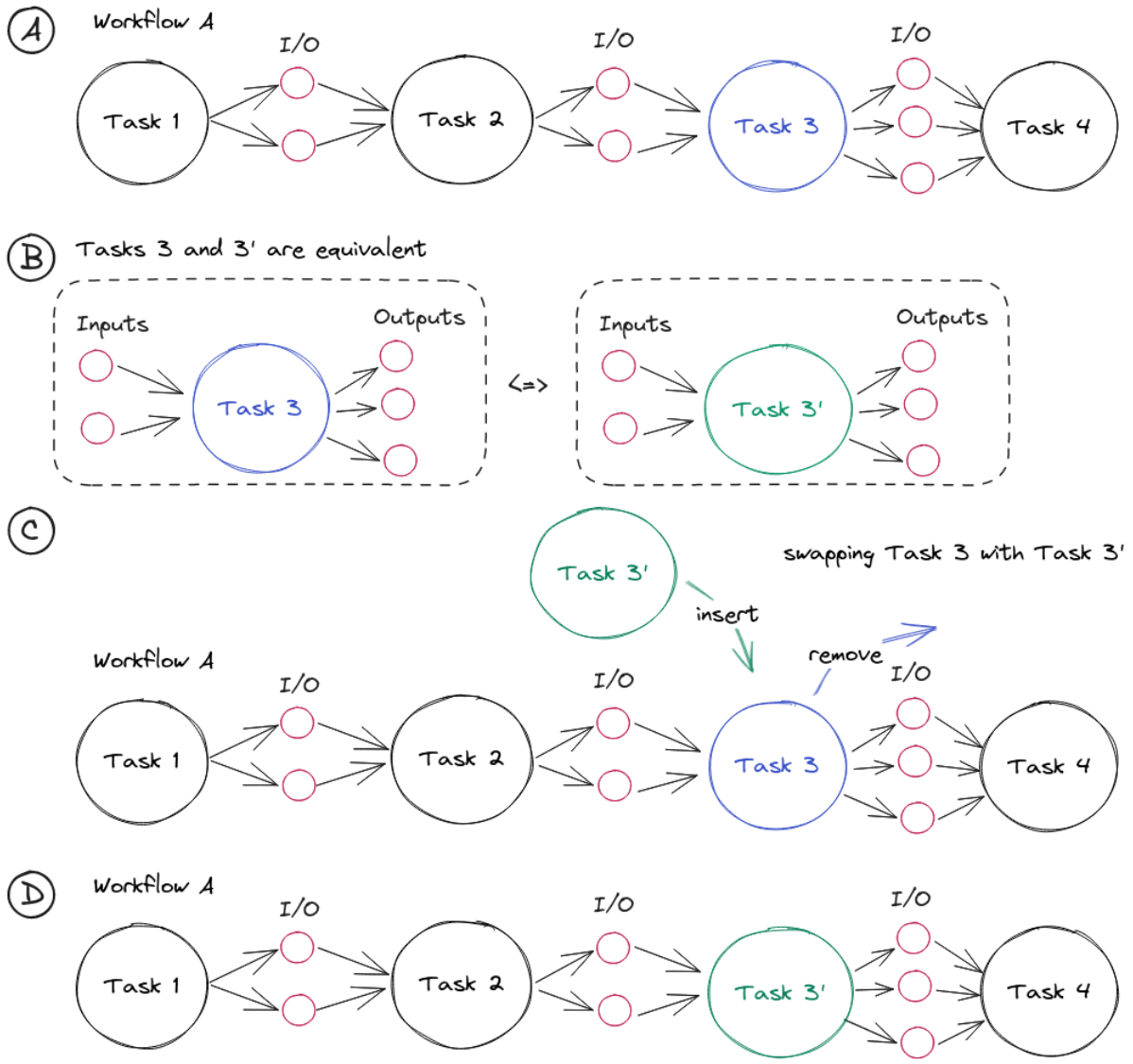


Figure 5-12 – Task modularity. (**A)** Successive tasks with inputs and outputs (I/O) represented in red. **(B)** Two tasks (Task 3 and Task 3') are equivalent with the same functionality and the same inputs and outputs. **(C)** Swapping two equivalent tasks. **(D)** The resulting succession of tasks with Task 3 swapped for Task 3' keeps the same functionality of the succession of tasks.

The two equivalent tasks Task 3 and Task 3' can be swapped in the workflow, resulting in an equivalent workflow shown in Figure 5-12D.

Workflows modularity

In the same way as tasks which can be modular (as seen in Section 2.1.4) and as illustrated in Figure 5.13, the concept of modularity extends to workflows. Here we illustrate the concept of task modularity with a simple example of two different workflows, 'Workflow 1' (in black) and 'Workflow A' (in blue) that both use 'Task 2' (in green) in a modular manner.

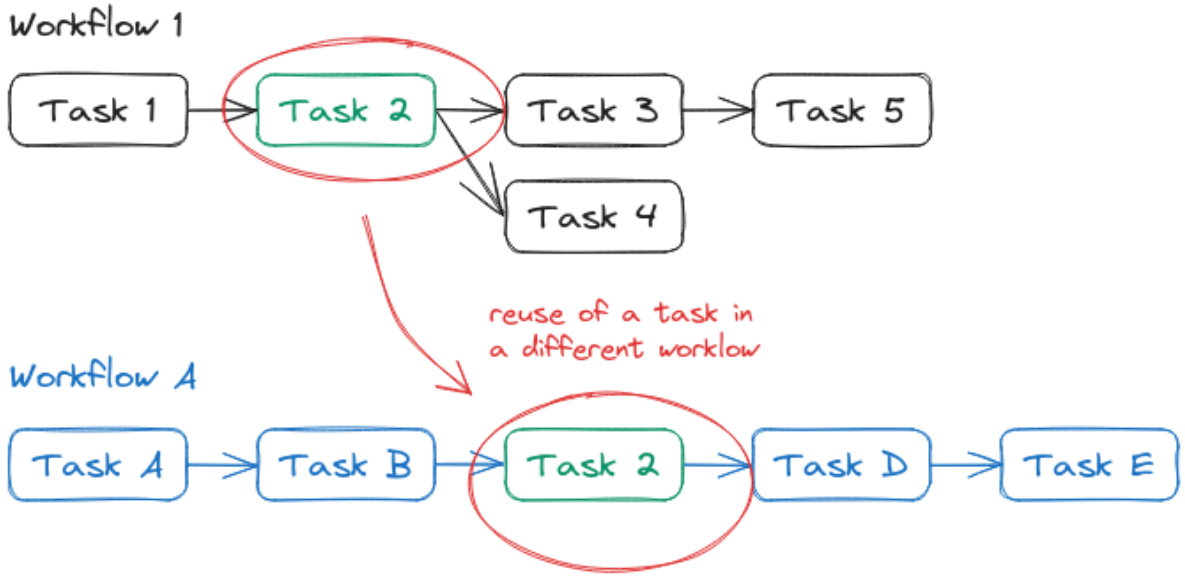


Figure 5-13 – Example of the modularity and the reuse of a task in a different workflow. To illustrate the example of modularity of a task within a workflow, the same task 'Task 2' in green is being used both in 'Workflow 1' (in black) and in 'Workflow A' (in blue).

Figure 5.14 illustrates the concept of task modularity between different workflows. The example illustrates how different workflows can be assembled together to create new workflows.

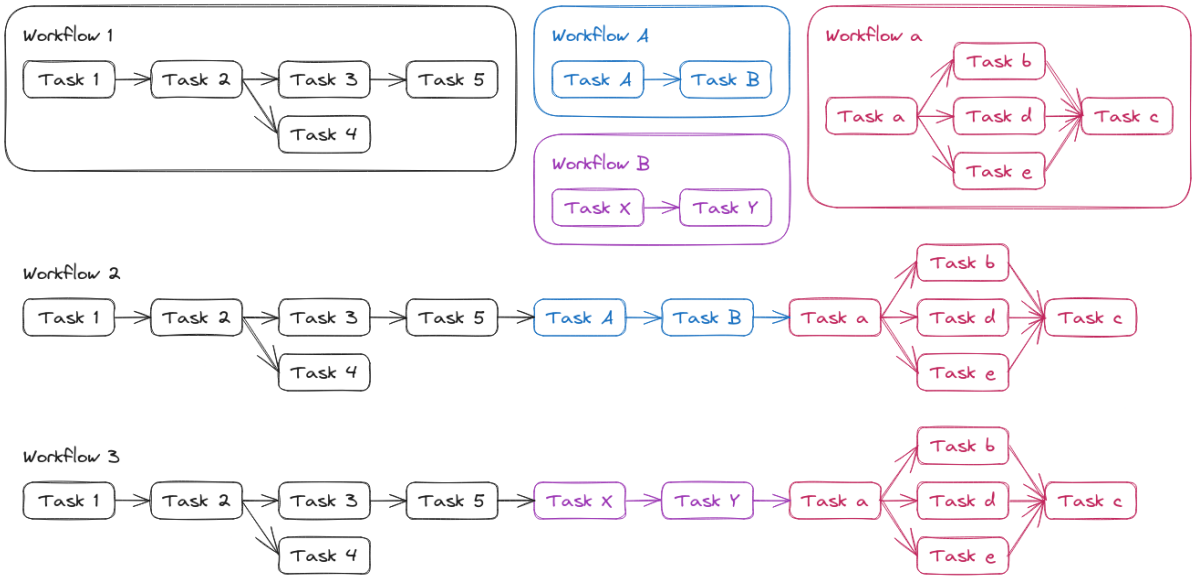


Figure 5-14 – Example of workflow modularity. This example illustrates the modularity of workflows. Given four different workflows ('Workflow 1' in black , 'Workflow A' in blue, 'Workflow B' in pink and 'Workflow a' in red) and assuming that 'Worklow A' and 'Workflow B' are equivalent, It is possible to use the four workflows in a modular manner to create new workflows that comprises them, Workflow 2' and 'Worklfow 3'.

2.3.3 Composability

Finally, workflows can be composed. It is possible to insert a workflow into another as long as one task in a workflow is equivalent to another workflow – same functionality and same input/output interface. This concept is illustrated in an example in Figure 5.15.

Given two workflows: 'Workflow 1' and 'Workflow 2' (FIgure 5-15A) if a task from 'Workflow 2' is equivalent to 'Workflow 2', in the example 'Task 2' is equivalent to 'Workflow 2' (Figure 5-15B), then 'Workflow 2' can be composed (or inserted) in 'Workflow 1'.

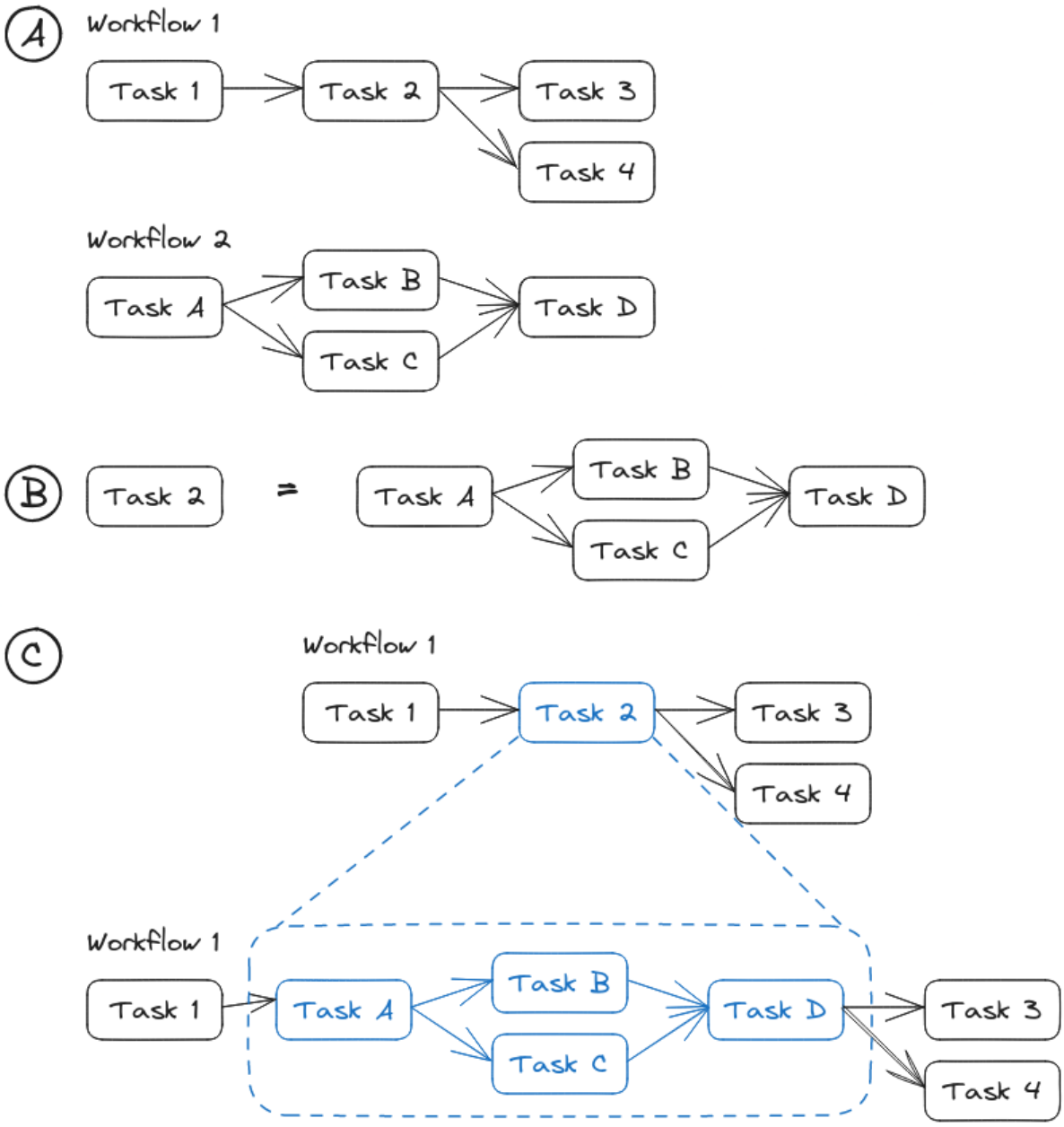


Figure 5.15 – Workflow composability. (**A**) Two different workflows (Workflow 1 and Workflow 2) comprising different tasks. (**B**) We define Task 2 as being Workflow 2. **(C)** We can compose Workflow 2 in Workflow 1 (through Task 2).

2.4 Protocols

2.4.1 Definition

‘A protocol describes how to carry out some form of laboratory or research process.’ (Bartley et al., 2021).

We define a protocol as a set of written instructions to execute a task in the context of synthetic biology wet lab work.

2.4.2 Representation and abstraction

Protocols have various forms of representation. The most common representation is a written set of instructions written in natural language that the experimentalist will follow. Protocols are commonly exchanged within a lab in the form of electronic documents in various formats (DOCX, PDF etc..). Some protocols are shared through the web, sometimes on the lab web page, sometimes on websites that curate protocols. Some searchable databases of protocols offer a dedicated visualisation (Teytelman & Stoliartchouk, 2015; Teytelman et al., 2016) and protocol editors as well as a search function.

Example of a PCR protocol exported as a PDF file

The image in Figure 5-16 illustrates a representation of a commercial protocol from New England Biolabs (NEB) for a Polymerase Chain Reaction (PCR) with the Phusion® High-Fidelity DNA Polymerase (M0530) (New Englands Bioloabs, 2020) exported from protocols.io as a PDF file. The picture represents a montage of the pages of the PDF document. Protocols are mainly written as instructions for human execution in the wetlab. Although, to port these executions of manual human tasks to machines, protocols have to be adapted from a description of tasks for humans to task descriptions understandable by robotic equipment. In the context of a biofoundry where equipment is programmable for the automation of tasks, protocols are a set of programmable instructions for the machines.

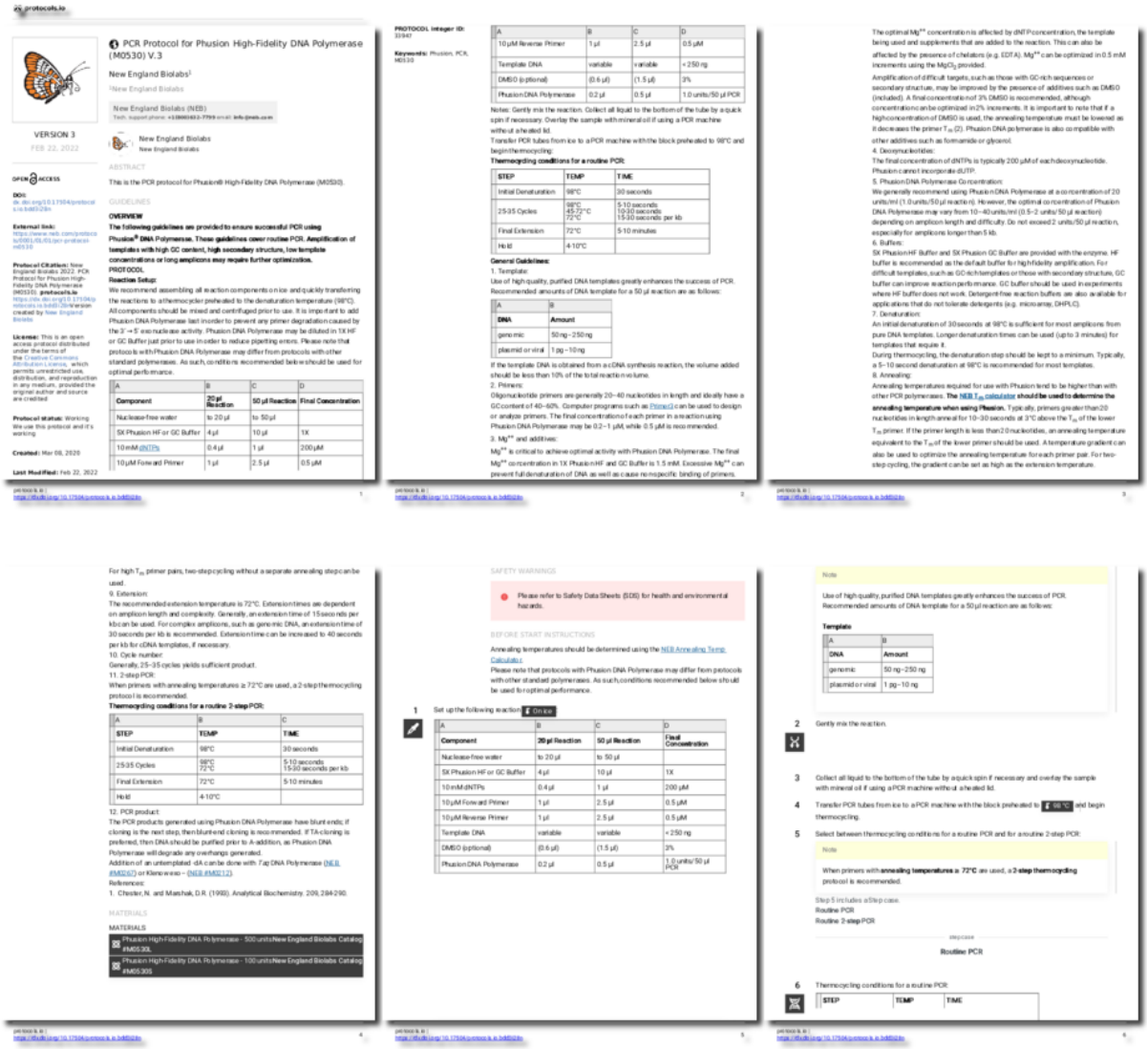


Figure 5-16 – Representation of a PCR protocol. (The text in this screenshot is **not meant to be readable**, it has been added here to illustrate an example of a representation format of a protocol)**.**

These set of instructions are defined by an application programming interface (API) accessible through some domain specific language (DSL) or a general-purpose language (example: C or Python). Some of the APIs are open and fully specified, some are open-source (Opentrons, n.d). Whereas some other commercial equipment software require the definition of a protocol within the software itself and through a specific graphical user interface (GUI).

General hand-written and defined protocols can be used and adapted for any purpose whereas a programmable protocol is defined for a certain API or a certain piece of equipment and is thus not portable without extra additional portage work. There have been

efforts to define a flexible and platform independent protocol language (Autoprotocol, 2014; Bates et al., 2017) that can be executed on different machines when appropriate drivers exist. More recently the The Bioprotocols Working Group (2022a) has been working on specifying the LabOp protocol language (formerly known as PAML) (The Bioprotocols Working Group, 2022b) that can output protocols through different drivers (also called ‘specialisation’ in the LabOp terminology). A protocol written in LabOp can be exported as a text protocol formatted as Markdown or as a PDF document but it can also be directly translated or executed as instructions or code for a specific machine.

Portability of protocols across machines is desirable as it enables one to write the protocol once and execute it across multiple platforms. Traditional hand-written protocols are a series of steps that can be performed in any conventional wet lab or adapted to any piece of equipment. As soon as we move to automated protocols, the representation of the steps of the protocol gets tied to the API of the equipment. This makes protocols that can't be moved directly from one piece of equipment to another one. Automated protocols become non-directly-portable. Projects like Autoprotocol and its successor LabOp (The Bioprotocols Working Group, 2022b) both overcome this problem by defining a protocol language that can be implemented over any piece of equipment. Protocols written in this language can be written once and re-used in multiple environments as long as a driver exists for the equipment. This is parallel to computer higher-level languages which can be executed on machines built with totally different hardware components and microprocessor architectures. There has been tentative work to bring automation languages that define higher expression and then get compiled to lower level robot instructions (Whitehead et al., 2018).

2.5 Examples of real world workflows

In the following section we show three examples of workflows that we have implemented for the lycopene case-study (presented in Chapter 2). The first example of a workflow is a computational workflow for generating the combinatorial constructs of the lycopene operon with the BASIC DNA assembly method. The second example is a semi-automated wetlab workflow of the lycopene extraction pipeline (described in Chapter 4) and the third example is the workflow of automated serial dilutions to create a standard curve (described in Chapter 4).

2.5.1 Example of a computational workflow: combinatorial BASIC DNA assembly of the lycopene operon

The following workflow example illustrated in Figure 5.17 is a computational workflow which precedes the BASIC DNA assembly processes in the wetlab. Given the parts that we wish to assemble to create our constructs, these computational tasks produce the final *in-silico*

constructs as well as the instructions for the DNA parts assembly for both a manual or an automated assembly process. This computational process produces all the constructs resulting from the combinatorial assembly of the different DNA parts of the lycopene operon, the three CDS: crtE, crtI and crtB as well as the different genetic regulators (promoter, RBS, terminator)

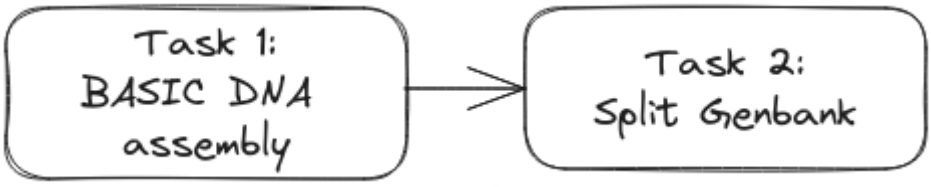


Figure 5-17 – Example of a computational workflow. The two-step computational BASIC assembly workflow.

Task 1: BASIC DNA assembly

The different DNA parts are gathered from a DNA parts repository. These parts, along with a JSON construct template, are then processed by the BASIC assembly Python script that outputs one Genbank file containing all the combinatorial constructs. The Python script, which calls the basicsynbio Python module (Haines et al., 2022) uses the construct template and the parts collections to generate the constructs. The script also generates the BASIC linkers list, as well as the clips list, the assemblies list and an Echo source to destination (well-to-well) transfer list.

We generate all the possible combinatorial combinations of collections of genetic parts (mostly RBS and promoters) that we express in order to find the best construct(s) according to some criteria (lycopene yield in our case).

Task 2: Split Genbank

This task splits the unique Genbank multi-record in individual DNA construct individual sequence files. This is the separation of one multi-construct Genbank file into individual files containing the DNA sequence. This is primarily used by our cMatch constructs reconstruction and identification algorithm (See Chapter 3) combined with sequencing data.

3.3.2 Example: automated serial dilution

The measurement in the testing step of the lycopene application (see Chapter 2) is decisive for the following step of the application, the measurement interpretation will tell if we have reached a point in our design space that is ‘good enough’ or if we continue exploring the design space. A final absorbance measurement (the full spectra) follows lycopene extraction

from the cells. In order to get the exact quantity of lycopene, a standard curve is required. Quantification of the measurement of lycopene with an UV-vis spectrography measurement requires the lookup on a standard curve (also known as a calibration curve). The concentration of the substance of interest (lycopene) in unknown samples is determined from a curve extrapolated from a set of standard samples of known concentration. We automate the creation of standard curves. A standard curve is obtained through measuring the UV-vis absorption of different samples of standard grade lycopene crystals diluted in DMSO at various known concentrations.

The following worklfow example is the Automated Serial Dilution workflow presented in Chapter 4. The workflow – presented in Figure 5-18 – which is a biofoundry workflow mixing wetlab, automated and computational tasks, consists of a 3-task workflow:

1. Automated dilution on a robotic liquid-handler
2. Absorption (UV-vis spectrum) measurement in a spectrophotometer
3. Curve fitting (computational)

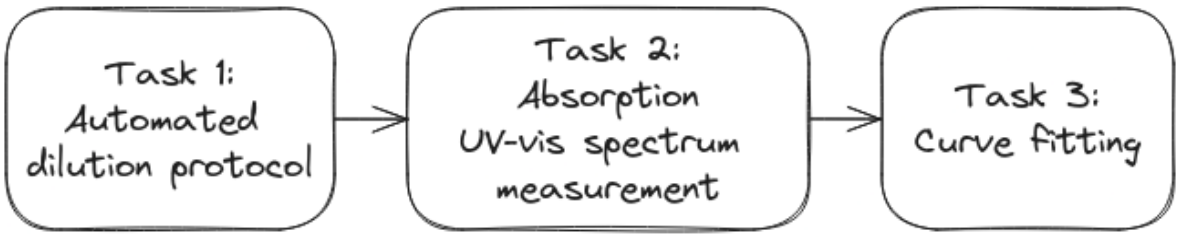


Figure 5-18 – Example of a biofoundry workflow. The workflow for automated serial dilution as seen in Chapter 4.

The initial input of the workflow is the lycopene standard at an initial known concentration C0 in the dilutant (DMSO) and the final output of the workflow is the standard curve.

Task 1: Automated dilution

A standard automation dilution method such as the one described in (Ng, Gately & Ooi, 2021) consists of different steps as follows: a dilution method sequentially aspirates a specified volume from a specified column of wells, dispense to a right adjacent column, and mix, until the end column is reached, allowing for serial dilution of multiple plates designated by sequence position. In which various aspiration settings can be adjusted such as the aspiration level, the distance to retract after aspiration, and triturations to mix can be adjusted. Dispensing settings such as the dispense height and retraction distance can be adjusted. The automated dilution protocol was implemented on an Analytik Jena CyBio FeliX liquid-handling platform (Analytik Jena AG, n.d.a; Suckling, McClymont & Hirst, 2017) using

a variable 8-channel pipette (independent volumes per channel).

Task 2: Absorption UV-Vis spectrum measurement
This tasks consists of measuring the 96-well plate containing the different dilutions of the lycopene standard with a UV-Vis spectrum measurement assay in a spectrophotometer (BMG Clariostar).

Task 3: Curve fitting
This task is a computational task. The values collected from the previous task are used to fit a curve to create the standard curve.

3. Conceptual Framework and Digital Twins

3.1 Digital twins

As defined by Grieves & Vickers (2017) a digital twin (DT) is 'a set of virtual information constructs that fully describes a potential or actual physical manufactured product from the micro atomic level to the macro geometrical level. At its optimum, any information that could be obtained from inspecting a physical manufactured product can be obtained from its Digital Twin.'

In essence, a DT is a computer program that takes real-world data about a physical object or system as inputs and produces as outputs (predications or simulations of how that physical object or system will be affected by those inputs) (Raj & Surianarayanan, 2020).

Digital twins have emerged from the fields of aerospace (Grieves, 2014; Grieves & Vickers, 2017; Xiong & Wang, 2022) and industrial manufacturing (Qi & Tao, 2018) and has extended to many others such as materials (Kalidindi et al., 2022), the energy sector (Yu et al., 2022), or to the automotive sector (Mendi, 2022). The concept of digital twins (DT) has more recently been brought up in field of life sciences, in areas such as health care (Bruynseels, Santoni de Sio & van den Hoven, 2018; Sun, He & Li, 2023; Venkatesh, Brito & Kamel Boulos, 2024), drug discovery (Subramanian, 2020), viral infection (Laubenbacher, Sluka & Glazier, 2021) or biomedical patient organs and tissues modelling (Barbiero, Viñas Torné & Lió, 2021).

The idea of digital twins has started to appear in engineering biology as ways to make the link between an engineered bacterial strain and its digital representation, as an example (Tellechea-Luzardo et al., 2020) present a version control system (VCS) for physically linking engineered cells to their digital footprint (i.e. their digital twin). In their paper calling for defining synthetic biological chassis more rigorously (de Lorenzo, Krasnogor & Schmidt, 2021) raise the topic of barcoding and digital twinning of engineered strains and point out

that one important aspect of standardised chassis is their digital twinning that can be implemented through DNA barcoding (de Lorenzo, Krasnogor & Schmidt, 2021). Barcoding of synbio chassis allows traceability as barcodes are unique identifiers that allow to directly link items to their digital twins and ensure availability of information (de Lorenzo, Krasnogor & Schmidt, 2021). De Lorenzo, Krasnogor & Schmidt (2021) write that synthetic biology would benefit from adopting digital twinning technologies which have had an enormous and positive impact on other industries.

3.2 Modelling the physical world

The biological work is being done in the physical world: in the wetlab and in the biofoundry. Observations of the biology gets recorded in the information world through measurement equipment, computational systems, software, files and databases. Engineering of biology is a combination of both actions in the physical world and a series of instructions to act in the physical world, that derive from observations. Our framework shown in Figure 5-19 lies at the intersection of both worlds and acts as an interface between them.

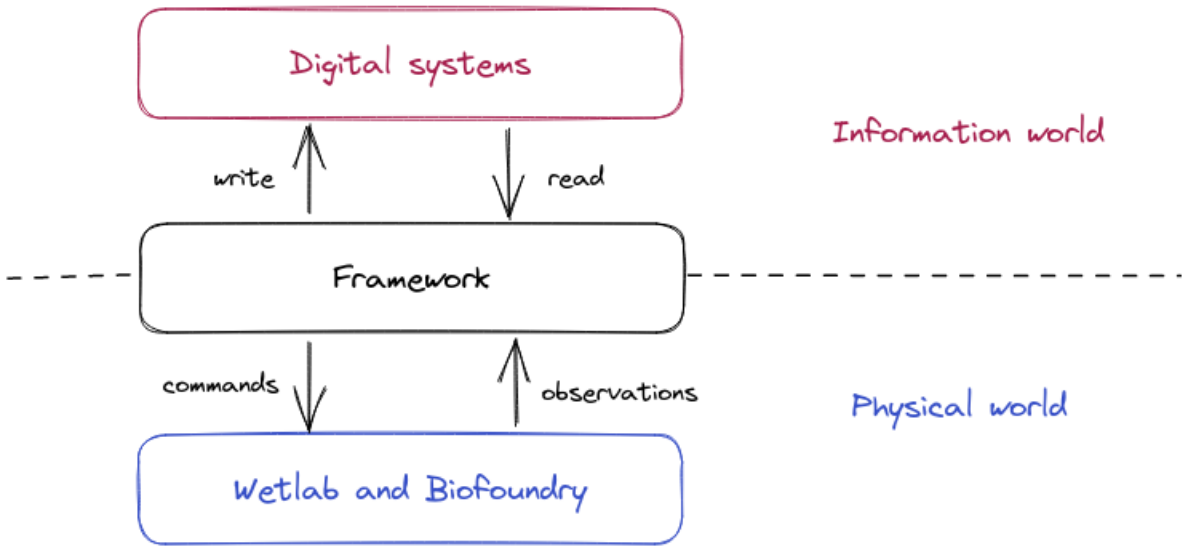


Figure 5-19 – Our Framework. The framework links the information world and the physical world.

The framework introduces a set of abstract objects to model reality. The framework that we introduce provides structures that allow the modelling of the work done in the physical world. This framework also provides structures to store information from the observation of the physical world and links both the modelling of the physical world and the information world allowing to move back and forth between information reading and retrieval and actions in the physical world.

4. Experiments in Synthetic Biology

In this section we will see how we define experiments in synthetic biology as a pipeline of treatments on a biological cohort that we observe along the pipeline. We will start by defining a biological cohort, then see what is the pipeline that the cohort flows through.

4.1 Studying a biological cohort

We define a synthetic biology experiment as the treatment and observation of a biological cohort (of bacterial cells) in a controlled environment. Figure 5-20 illustrates the succession of treatment and observation of a biological cohort.

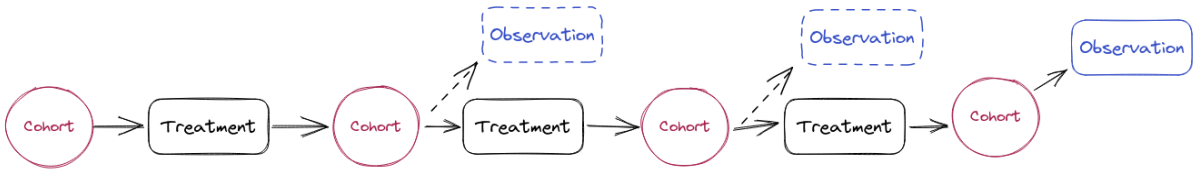


Figure 5-20 – Treatment of a biological cohort. Successive treatment of a biological cohort along an experiment with an observation at the end and some optional observations (represented with dashed contours) along the pipeline.

Cohorts of cells are bacterial strains of interest that have been genetically modified (genomic modification or transformation with heterologous DNA). The cohort of cells has different genetic properties.

Figure 5.21 shows the example of a biological cohort: A 96-well plate with cultures of *E.coli* BL21 cells that have been transformed with plasmids containing various genetic constructs designs expressing a protein of interest.

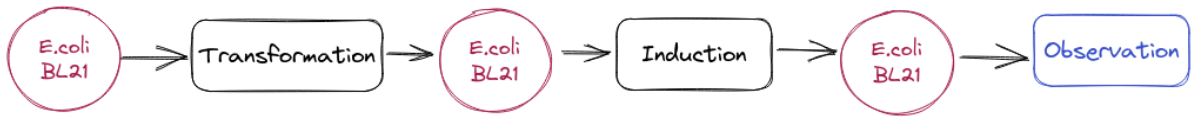


Figure 5-21 – Example of a biological cohort treatment in a workflow. The biological cohort of *E.coli* BL21 cells is being transformed and induced along the worfklow before observation.

We represent an experiment as a workflow, a succession of procedures (protocols) that modify the cohort or its environment (chemical change, temperature change, light change etc..). The biological cohort moves along the pipeline of the experiment. Along this pipeline

various procedures are applied to the biological cohort in an order that is defined in a workflow.

4.2 The operational pipeline

We define a procedure as an ordered sequence of instructions that does a particular task.

A combination of a software system and resources follows the workflow and runs the different procedures (that we also call operations, abbreviated Op). Along the pipeline multiple outputs are being generated by the various procedures. Some of the outputs are digital, some are physical. A digital record is kept whether the output is digital or physical, in the case of the digital output this is straightforward, we keep the digital output and a track of it (e.g. in a database), when the output is physical (e.g. a plasmid, some molecule) we keep a digital track of it. We keep a digital record of all the outputs being produced along the pipeline and structure their relationship in a model that lies within a database.

Additionally to keeping a digital record of the output we collect and store any metadata that is associated to the various executions of operations, thus capturing the context of the experiment. The operational overall operational pipeline is illustrated in Figure 5-22.

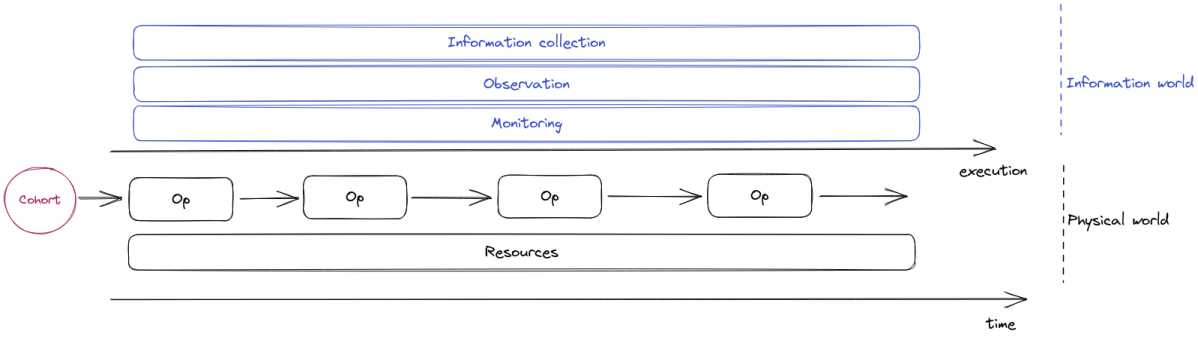


Figure 5-22 – The operational pipeline. A biological cohort comes in and many successive operations (Op) are being applied by resources. The execution of these resources is either driven manually or by a software layer. The execution is being monitored, observations of the cohort are made and information gets collected.

A data-driven point of view of the physical world

An experiment is a succession of procedures in both the wet lab and the dry lab. Some of these procedures are computational tasks, some others are wetlab tasks. These procedures can be manual or automated. The order of the procedures are defined in a workflow. We execute these procedures within a framework that we present in this chapter.

The framework is supported by a software system that orchestrates the execution of all these procedures and captures the context of the execution as well as the data that is

produced along the experiment. The data and metadata produced, as well as the recording of the execution gets structured and stored in a way that allows retrieval of data at any point in time (during and after the execution) of the experiment.

Our interest is in the data that gets produced for further data analysis, thus we structure our experimental workflows around data and metadata flow and storage. We have a data-driven point of view of the physical world.

5 Summary

In this chapter we have set the basis of our conceptual framework that once implemented as computational tools, allows us to structure experiments and their resulting data.

Our framework introduces the concept of workflows that are composed of tasks which can represent these protocols.

We have introduced our conceptual framework which acts as a bridge between the physical world and the information world (digital twinning). We have shown how we represent experiments in synthetic biology (the real world model) and how we apply an operational pipeline that treats our objects for observation.

Chapter 6
Biofoundry Information Manager

Experiments run in a biofoundry make use of various pieces of equipment and measurement instruments. Each experiment generates data and the volume of data can be large. All the data produced needs to be organised for further analysis. Additionally to measurement data, there is metadata generated of which the collection helps organising the data for an easier recollection later down the experimental pipeline. It is thus necessary to have an information system to collect and organise all the data and metadata generated during the experiments. We call this system the Biofoundry Information Manager.

In this chapter we introduce two data structures that allow us to model experiments and their execution as well as to structure the collection of their resulting data and metadata.

In the first section we introduce a first structure, directed graphs (DGs) on top of which workflows can be built. Directed graphs (DGs) allow us to structure the workflow and the dependencies between the different tasks. We introduce the model graph, a directed graph (DG) that links the dependencies of all the data being produced along the workflow pipeline.
In a second section we explain how workflows can be executed and how the result of the execution can be recorded for further usage. A second data structure is the introduced, called directed acyclic graphs (DAGs) – and the execution graph, which structures the information resulting from the execution of the workflow. Workflows can be executed as runs and the result of these runs is stored in an execution graph.
In a third section we introduce the notion of data provenance and explain its importance in experimental runs.
In a fourth section we then show how we can technically implement and build those DAGs with computational libraries and store the DAGs in a database that can be queried. Structuring the data in a DAG and storing it in a graph database allows us to build data lineage and query the graph database to find data provenance.

This chapter is ended with exemplars of workflows, the computation workflow of BASIC DNA assembly and the workflow of the Automated Dilution of a Standard Curve.

Laboratory Information Management Systems (LIMS) have been used since the early seventies. The roots of LIMS were in laboratory automation (Gibbon, 1996). Prasad & Bodhe (2012) review developmental stages of LIMS and give the definition of a management system. A management system is the framework of processes and procedures used to ensure that an organisation can fulfil all tasks required to achieve its objectives. Historically a LIMS is a software utility that was developed in-house by organisations (laboratories) who wanted to improve the data acquisition and reporting processes. An information system is based on the continual availability of the data which is further sorted and organised into various reports based on the requirement (Prasad & Bodhe, 2012). More recently LIMS have started to include the component of biosecurity (Sun, Wu & Fan, 2021). LIMS has been widely used to facilitate laboratory activities. Sun, Wu & Fan (2021) argue that current LIMS do not contain functions to improve the safety of laboratory work, which is the major concern of biosafety laboratories (BSLs) and propose a set of guidelines to improve safety in LIMS.

In the context of synthetic biology and biofoundries the Klavins lab has developed Aquarium, an open-source, web-based software application that integrates experimental design, inventory management, protocol execution and data capture (Vrana et al., 2021). Aquarium has supported a biofoundry at the University of Washington, the UW BIOFAB. Aquarium supports flexible development and deployment of standardised workflows, composed of modular protocols that drive on-screen, step-by-step instructions for human technicians. In this regard, the system presented in this thesis also includes the human-in-the-loop to enable tracking of manual work but differs from Aquarium which has not been designed to orchestrate a fully automated workflow in a biofoundry.

One notable commercial system developed by Synthace Ltd., a UK based technology company, is the Antha platform (Quaglia, 2017). Antha provides graphical interface into experimental workflows to execute on their equipment regardless of the brand and provides drivers to CyBio and Gilson instruments. The system presented in the thesis allows the execution of experimental workflows similarly to Antha – although not through a graphical interface editor – but does not yet provide individual drivers to commercial platforms and equipment. One of the limiting restrictions for writing drivers to access commercial platforms is quite often the lack of open and well documented APIs.

LIMS have been software systems that have enabled an early form of computerised information organisation and reporting automation in laboratories. Some more recent works

have built software systems to manage and organise information at the scale of a biofoundry. They are worth discussing with regards to the BIM.

As stated by Hillson et al. (2019) biofoundries benefit from the use of software and other new workflows and tools. In addition to computer-aided design tools (to enable biofoundries to to support large-scale, reproducible, and formalised manufacturing processes required by Synthetic Biology), new software tools and systems are required to support a paradigm shift to a biofoundry-based manufacturing ecosystem (Densmore et al., 2023). Winston, Willsey & Ceze (2023) and Bartley et al. (2023) provide two frameworks (and APIs) to standardise, respectively, fluidic liquid-handling workflows and protocols (LabOP). Similar to these two frameworks, the programmatic API to access the Biofoundry Information Manager (BIM) and Biofoundry Automation Manager (BAM) presented in this thesis, is written in the Python programming language. However, the system that we implemented is not restricted to solely liquid-handling automation or protocol description but also enables both computational and mechanised automation workflow orchestration and automated collection and organisation of data.

In their work entitled ‘Building a Biofoundry’, Holowko et al. (2021) mention the need for expertise in LIMS and the need for data management tools and state that ‘standardised’ biofoundry solutions do not currently exist. Many researchers in the field of engineering biology agree that solutions for managing biofoundries are needed. Hence, in the thesis reference is made to recent work of Berezin et al. (2023), offers ten simple rules for managing laboratory information. Most of the rules apply to the management of biofoundries information and are amenable to computerised automation such as using computer-generated sample identification numbers, or data management guidelines (using databases and linking data to protocols) and standardising workflows. All the ten rules described in (Berezin et al., 2023) directly apply to or can be supported by the Biofoundry Information Manager presented in this chapter (and the Biofoundry Automation Manager, see Chapter 7).

Similar to the work presented in this thesis, Peccoud et al. (2022) present a research data management system which uses custom data structures to document the execution of experimental workflows. In their work, data were managed in the GenoFAB LIMS. GenoFAB organises data in a customisable database of objects using three layers of abstraction (‘catalog type’, ‘catalog entries’ and ‘catalog items’). The system is presented in the thesis also makes use of a database (the BIM makes use of a graph database system). But is not restricted to a predefined number of layers of abstraction and allows the user to define any number of object types which enables working and organising data with an arbitrary levels of granularity. Peccoud et al. (2022) demonstrate their work with two experimental workflows for plasmid preparation and cell culture. My work also demonstrates the use of the BIM with

experimental workflows with the examples of DNA assembly and an automated serial dilution (see Chapter 6 Section 5 and Chapter 7 Section 4).

Habich & Beutel (2024) provide an overview of various aspects of digitisation and how academic laboratories can digitise their working environment. Through digital integration of laboratory devices, usage of laboratory assistants (digital such as electronic laboratory notebooks (ELN) and what they call ‘physical assistants’ which are automated mechanised devices such as liquid handling stations and robots), adoption of standard communication protocols such as Standardisation in Lab Automation 2 (SiLA2) (Bromig et al., 2022; Juchli, 2022) and Laboratory and Analytical Device Standard (LADS) based on Open Platform Communications Unified Architecture (OPC UA) (Brendel et al., 2022) and storing the data and metadata according to the FAIR (Findable, Accessible, Interoperable, Reusable) data principles (Wilkinson et al., 2016). The BIM and the BAM differ from ELNs because they do not require the active input of digital notes but automatically collect (‘log’ in the computer jargon) information of the execution of the workflow. The BIM and the BAM do not currently integrate any specific communication standards such as the ones mentioned by Habich & Beutel (2024), but are not restricted to any specific as the BAM allows the execution of any computation process (which could talk to any SiLA2 driver for example). The BIM respects FAIR principles. All the data and metadata generated or collected through the execution of workflows is accessible in a graph structure in a database system which can be queried computationally through the BIM API.

In the context of biofoundries information organisation systems to manage workflows are necessary. Additionally the large amount of data generated by biofoundries necessitates the adoption of data management solutions as well as laboratory resources and reagent tracking systems.In their recent review on ‘Physical Laboratory Automation in Synthetic Biology’, Stephenson et al. (2023) emphasise how high-throughput physical automation when combined with computational tools and data management systems – makes closed-loop iterative experimentation possible, with proven gains in discovery time and productivity. Stephenson et al. (2023) write that humans are still responsible for some manual tasks and all experimental design, data management, and analysis (Stephenson et al., 2023). The BIM offers a data management solution that aims to organise data automatically according to the designed workflow and thus free the user from the burden of manual data management.

Bartley et al. (2020) offer a perspective on very large scale (gigabase) genome engineering. Although not referring directly to the topic of biofoundries, Bartley et al. advocate the adoption of an ecosystem of tools, services, standard, frameworks and automation technologies to manage projects of such a large complexity without massive teams, which

resembles very closely to what Holowko et al. (2021) what highlights for building a biofoundry. Bartley et al. (2020) highlight the need for integrating informational tasks, computational workflow engines, reproducible execution, and exchange of complex workflows involving multiple software programs and computing environments such as common workflow tools (and cite the Common Workflow Language (CWL) (Crusoe et al., 2022), Dockstore (O'Connor et al., 2017) and bioinformatics engines such as Cromwell (Broad Institute, 2024), Galaxy (Goecks et al., 2010; Hérisson et al., 2022), Nextflow (Di Tommaso et al., 2017) and Toil (Vivian et al., 2017)). Bartley et al (2020) also advocate for the usage of LIMS to improve both reproducibility and efficiency and help track workflows and reagent stocks. To this end, the BIM (and the BAM, see Chapter 7) offers a system that enables the execution of computational workflows in a reproducible manner and act both as a workflow engine and a LIMS enabling the tracking and usage of reagents (usage of resources and reagents can be queried in the graph database, see Chapter 6 Section 4.1).

1. Model graph - Representing Workflows

1.1 Structure: directed graphs

A directed graph is a graph (vertices connected by directed edges) in which edges have orientations (Bang-Jensen & Gutin, 2018). We define a directed graph as a representation of the succession of different tasks that constitute a workflow.

The nodes (also defined as 'vertices') of the graph represent the various processes. The directed edges connecting the nodes represent the relationships between the nodes. We define the dependency relationship between two nodes as the relationship that subordinates the existence of one node to another. For any node in the graph, any upstream nodes would be dependencies of the node and any nodes downstream are implicitly made dependent on the former node. Figure 6.1 is an example of a simple directed graph with 4 nodes connected by edges.

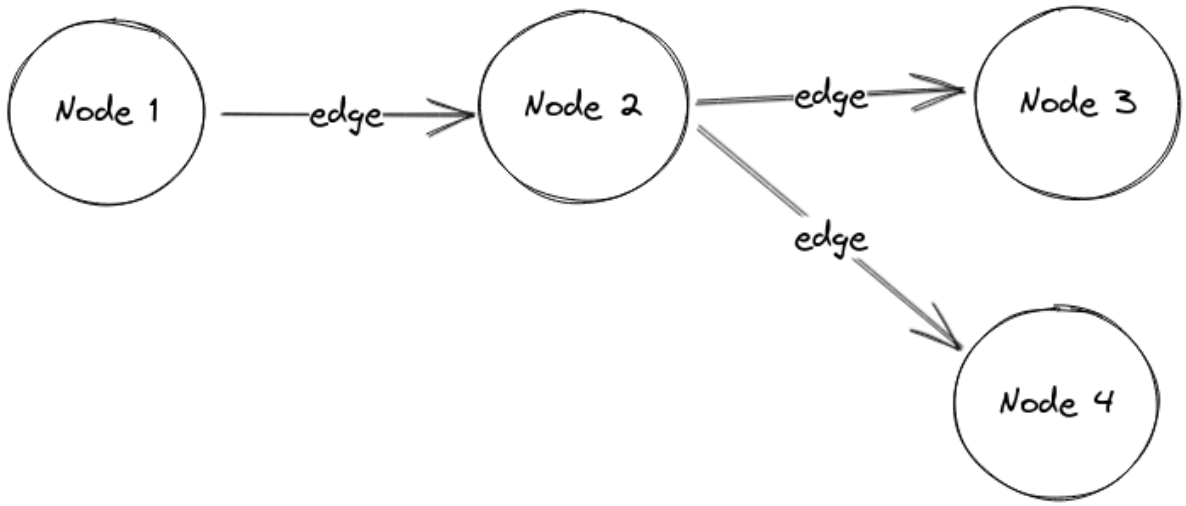


Figure 6-1 – Example of a directed graph with 4 nodes connected by edges.

1.1.1 Representing workflows as graphs

In order to plan and execute an experiment workflow, we need to model it with an appropriate structure that enables its execution. A workflow can be represented as a directed graph because such a graph can be evaluated by following the graph's edges and executing its vertices which represent tasks.

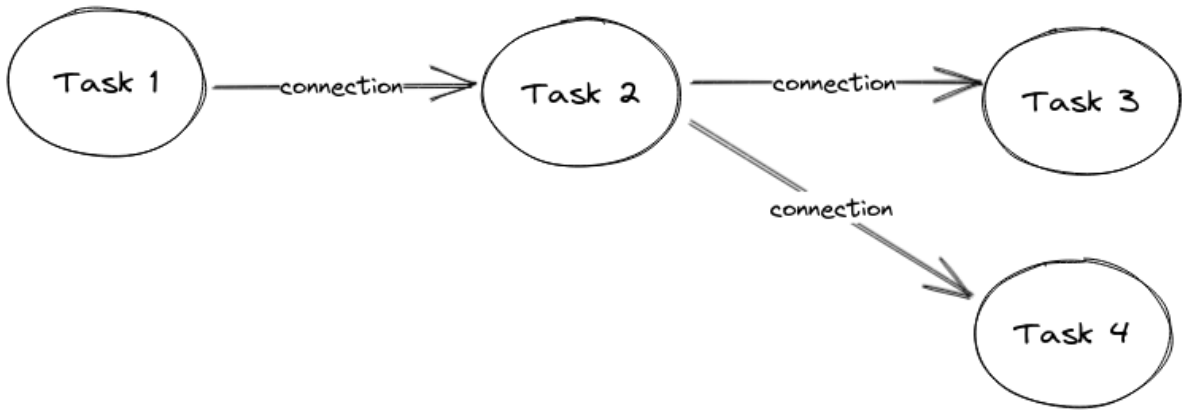


Figure 6-2 – Example of a workflow represented in a directed graph.

1.2 Model graph

The model graph is a graph structure that represents the workflow and all the components it

comprises – all the inputs and outputs, as well as the resources (physical or digital). This model contains a representation of all the workflow tasks and all the objects that the workflow uses, interacts with or creates, such as the system resources, the biofoundry resources, the labware, the biological cohorts, the DNA constructs (both *in silico* and *in vivo*), the chemical reagents, the digital files, the protocols, the measurements.

Everything that is being used in the workflow is represented in this graph structure. In the model graph, the different nodes represent the different tasks and objects and the edges represent the different relationship between these tasks and objects.

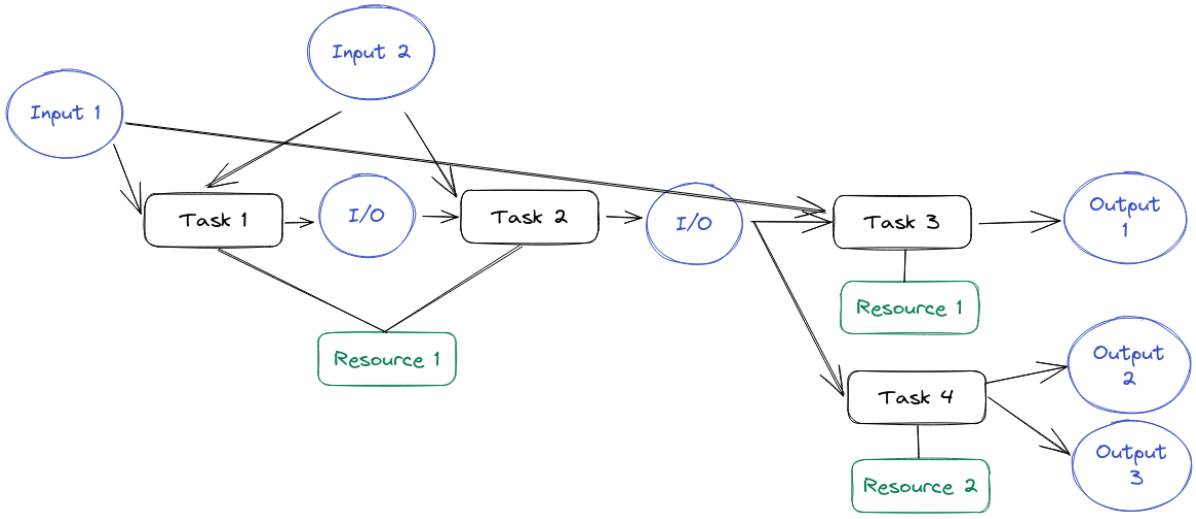


Figure 6-3 – Example of a model graph with four tasks.

As can be seen in the simple model graph in Figure 6.3 – with only four tasks, three resources and a few inputs and outputs – the complexity of the graph can increase rapidly as nodes can share various connections. For slightly more complex workflows the visualisation of the data graph quickly becomes visually unreadable. Thus it is necessary to visualise a small sub-part of the entire model graph after filtering.

In the example in Figure 6.4, we see a resource-centric view – where we can see one resource: `Resource 1` in the centre of the graph with the connecting nodes of the Tasks that the resource is being used in.



Figure 6-4 – Example of a resource centric view.

2. Execution graph: Representing Execution

2.1 Workflow execution and runs

2.1.1 Definitions: workflow execution and runs

The workflow is represented with a directed graph (DG). When we '*execute'* a workflow we move from tasks to tasks along the edges of the graph according to conditionals and execute each of the tasks. Executing a task means executing the action defined in the task.

We define a run as a single application of a workflow under listed constraints on our infrastructure. As can be seen from in Figure 6.5, a workflow can be executed multiple times and will create a certain multiple runs.

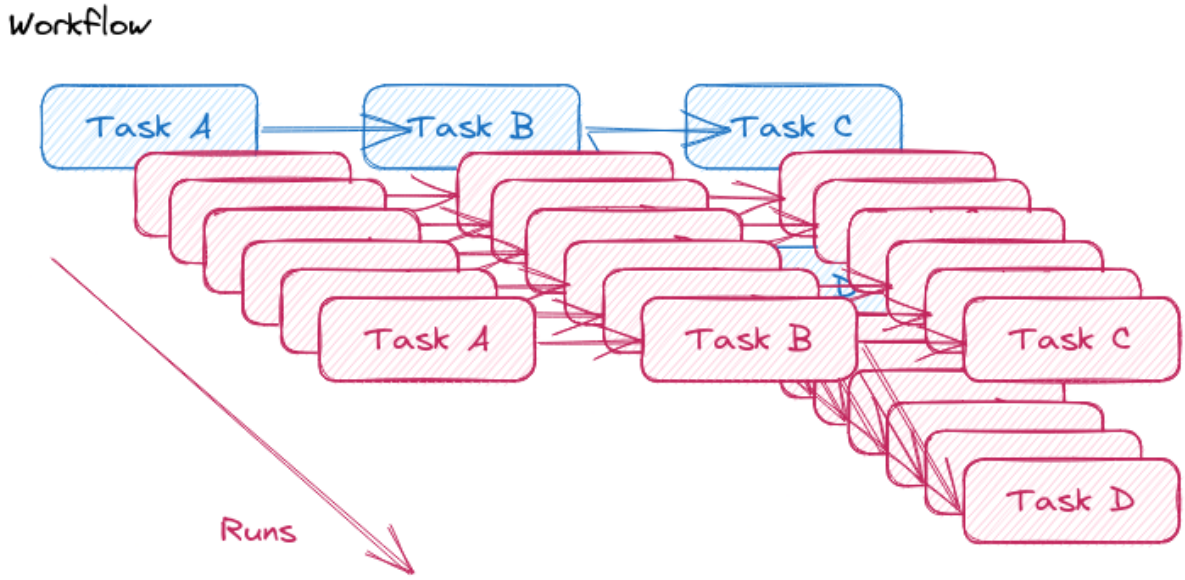


Figure 6-5 – A workflow and its executions. A workflow (in blue) and multiple executions resulting in multiple runs (in red).

2.1.2 Task execution patterns

(i) Conditional execution of a task

This pattern involves the evaluation of a condition to determine which tasks get executed after the condition. In the example in Figure 6.6-A, after Task 1 has been executed, the condition is evaluated; the evaluation of the condition can result in either condition 1 or condition 2 or condition 3; if condition 1 is satisfied, Task 2 is executed; if condition 2 is satisfied, Task 3 is executed; if condition 3 is satisfied, Task 4 is executed.

(ii) Parallel execution of a task

This pattern defines parallel execution of multiple tasks In the example in Figure 6.6-B, after Task 1 is executed, the graph branches out into branch 1 and branch 2. Both branch 1 (which contain Task 2 and Task 3) and branch 2 (which contain Task 4 and Task 5) are then executed in parallel.

(iii) Repeated execution of a task

This pattern defines the repeated execution of a task (or multiple tasks). In the example in Figure 6.6-C, after Task 1 and Task 2 have been executed, Task 3 gets executed and a condition is evaluated, the evaluation of the condition will either redirect the flow of execution back to Task 2 and repeat the execution of Task 2 or will follow to the execution of Task 4.

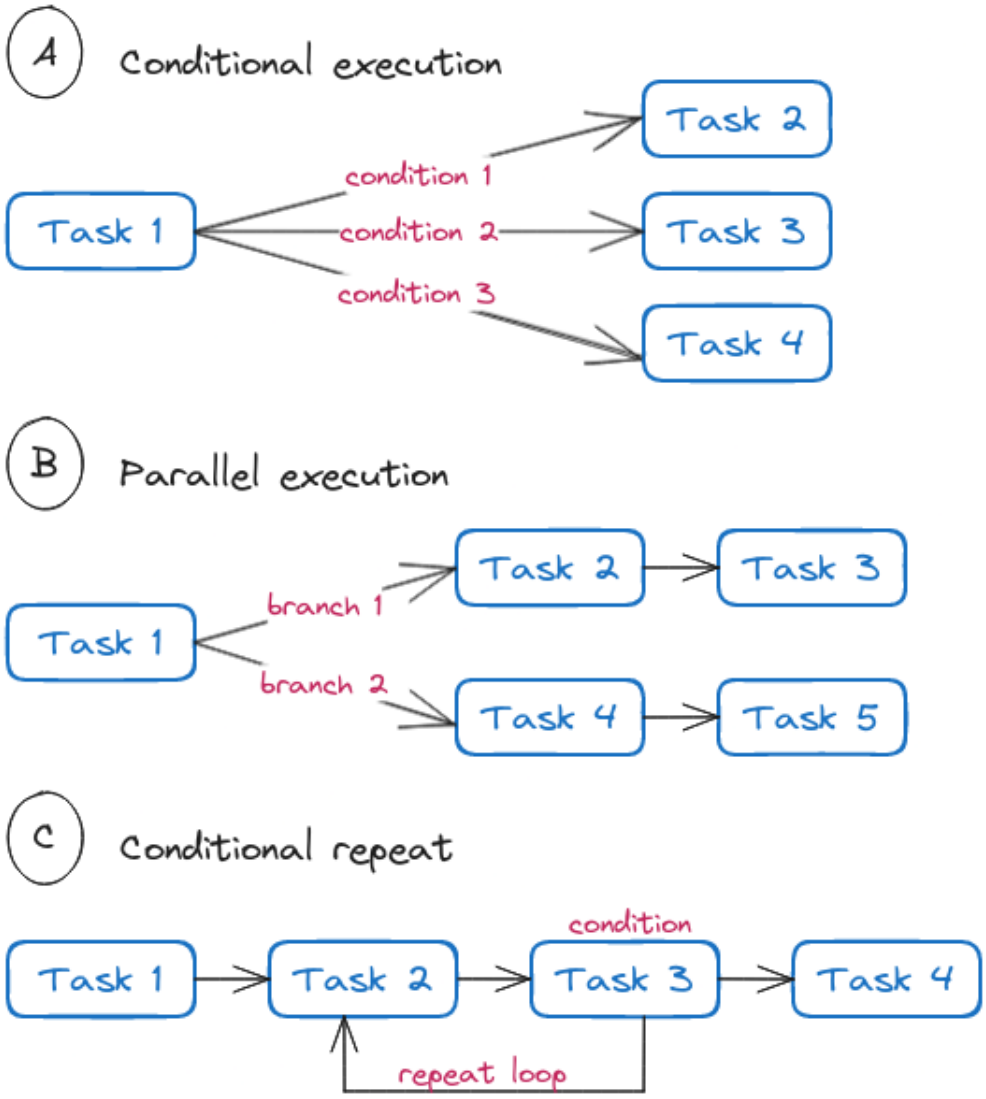


Figure 6-6 – Task execution patterns. (**A**) Conditional execution of a task. (**B**) Parallel execution of tasks. (**C**) Conditional repeat of a task.

(iv) Upstream hierarchical conditional execution

In addition to the patterns described in the previous section, execution of tasks can be conditional to the execution of previous tasks in the dependency hierarchy. Such conditions can be met if:

- all upstream tasks have succeeded
- all upstream tasks have failed
- at least one upstream task has failed
- at least one upstream task has succeeded

Figure 6.7 illustrates the four aforementioned cases of upstream hierarchical conditional execution.

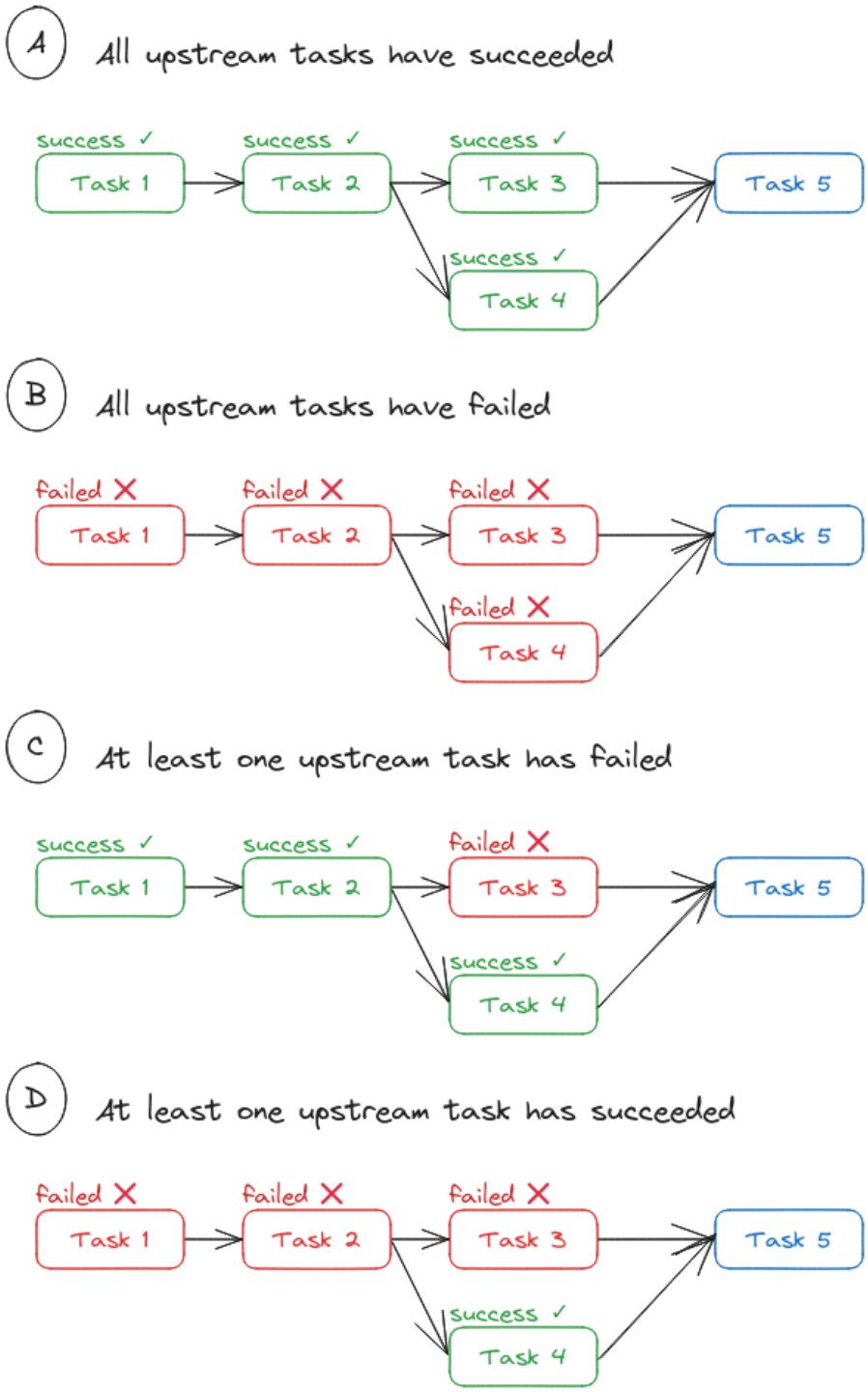


Figure 6-7 – Task failures. Example of a worklow with upstream conditionals on 'Task 5' (in blue). **(A)** all upstream tasks have succeeded. **(B)** all upstream tasks have failed. **(C)** at least one upstream task has failed. **(D)** at least one upstream task has succeeded.

2.2 Execution graph

2.2.1 Definition

During each run, the workflow is applied and each task gets executed (according to the conditions specified by the workflow). There is a record of the execution of a task, including metadata such as timestamps, resources metadata, operational metadata and collected generated data (outputs) and data twinned to physical objects. The resulting history (log) can then get cast onto a graph that we call the execution graph.

The workflow gives the overall structure and the dependencies of various tasks composing the workflow, but doesn't tell us about what happens when we execute the workflow. We want to get a representation of the execution of the workflow so that we get a trace and history of the various tasks that got executed during a run.

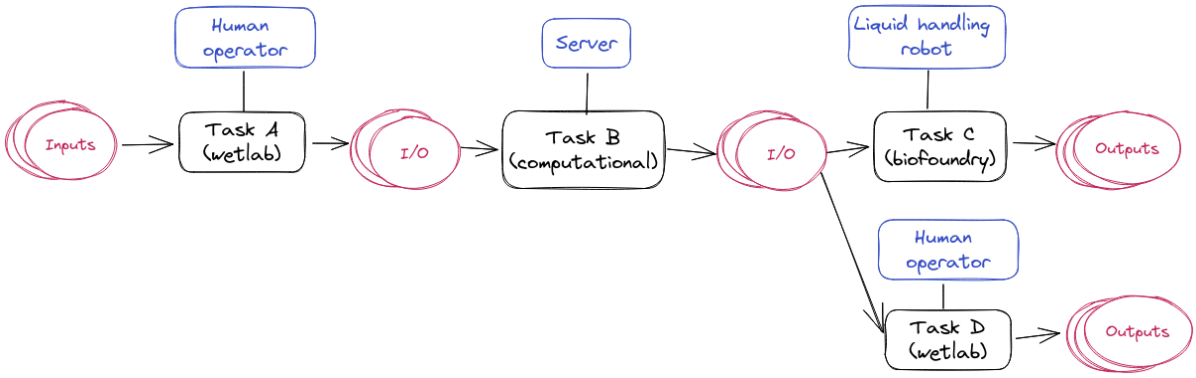


Figure 6-8 – Example of an execution graph with resources assigned to the different tasks. Three different resources: a human operator for wetlab tasks, a robotic liquid handler for automated biofoundry tasks and a computer server for computational tasks.

The resulting graph inherits its directedness from the workflow and because the execution is indexed by time, the execution graph is a directed acyclic graph.

A directed acyclic graph (DAG) is a directed graph (as seen Section 1.1) with no directed cycles. An acyclic graph is a graph having no graph cycles (Weisstein, 2023a). A cycle of a graph G, also called a circuit if the first vertex is not specified, is a subset of the edge set of G that forms a path such that the first node of the path corresponds to the last (Weisstein, 2023b)

2.2.2 Generation of execution graphs from the model graph

Figure 6-9 on the next page shows the example of the generation of an execution graph from a model graph. The model graph contains seven tasks (T1-T7) with a conditional

branching on T2. The two resulting possible branches are [T3-T5] and [T6-T7]. This example in Figure 6-9 decomposes the generation of the execution graph in four steps detailed below.

- **Step 1.** T1 is evaluated and creates two outputs. The execution generates a first task T1 written in the execution graph as well as the two outputs.
- **Step 2.**T2 is evaluated and creates one output. The execution generates T2 which is written in the execution graph with its output. The conditional evaluation on T2 leads to the branch [T6-T7].
- **Step 3.** T6 is evaluated and creates one output. This writes a task T6 in the execution graph with its output.
- **Step 4.** T7 is evaluated and creates two outputs. T7 gets written in the execution graph with its two outputs.

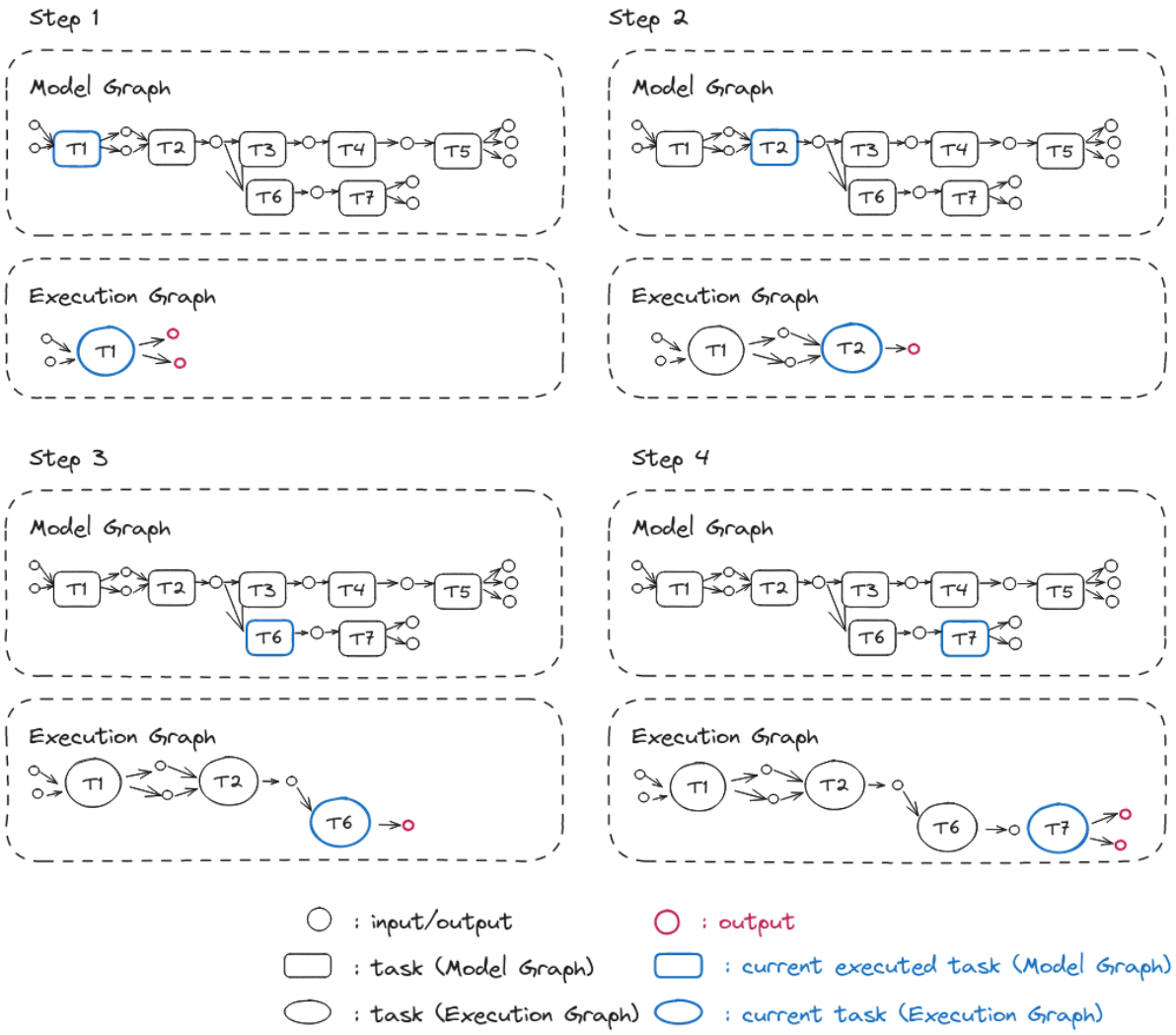


Figure 6-9 – Example of the execution of a workflow step by step. Each of the steps shows the model graph and the currently executed task is highlighted in blue. The executed task gets written in the Execution graph as well as its outputs (highlighted in red). Step after step we can see the execution graph being written as the workflow gets executed.

The workflow (DG) acts as a generator of execution graphs (DAGs). The example shown in Figure 6.10 illustrates the generation of multiple execution graphs from many runs of the workflow (run 1 to run 6).

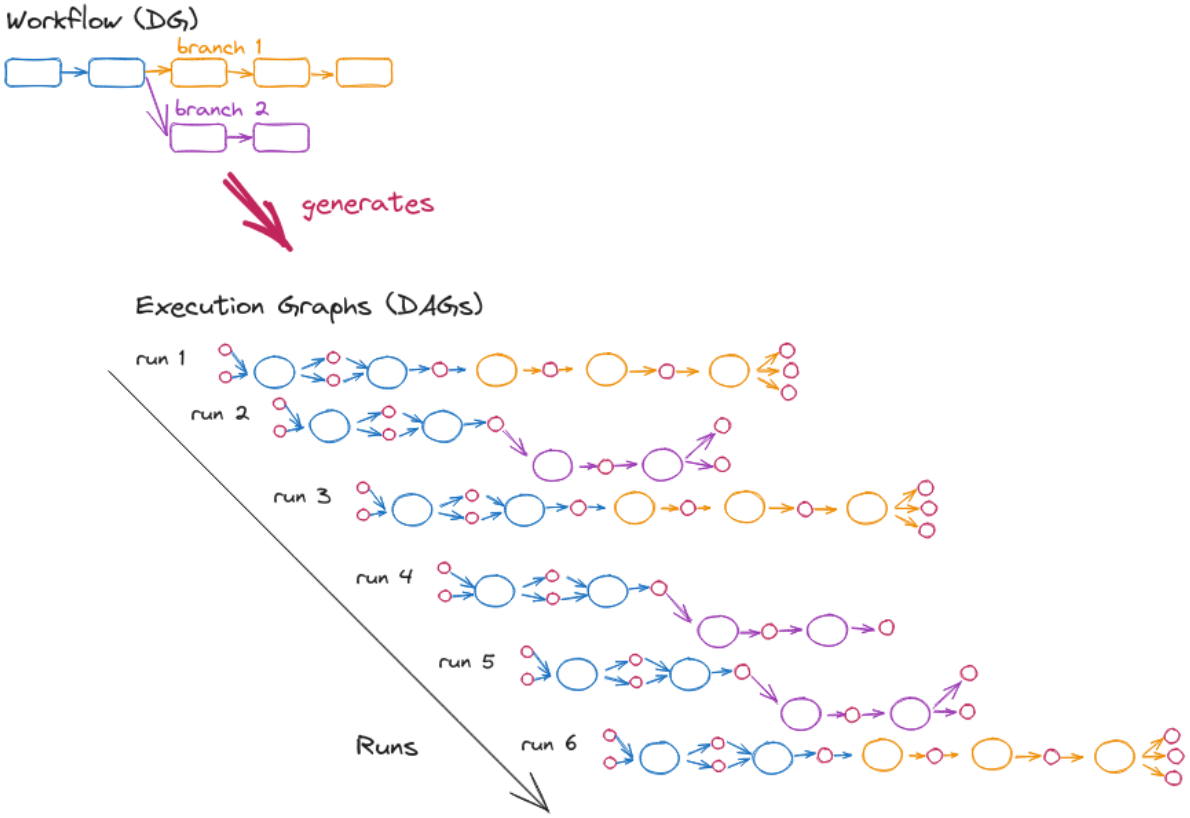


Figure 6-10 – Workflow to runs. Example of a Worklow (DG) with a conditional on the second task and two possible branches (branch 1 (yellow) and branch 2 (purple)) which acts as a generator of execution graphs (DAGs). The example illustrates six different runs (from run 1 to run 6) each taking different branches and generating various outputs (red). If we compare run 3 and run 4, run 3 takes branch 1 whereas run 4 takes branch 2. Not only do run DAGs reflect conditional executions but they also reflect also their outputs – i.e. when comparing run 4 and run 5, both runs take branch 2 (purple) but the output of the last task of run for only generates one output whereas the output of the last task of branch 2 in run 5 generates two outputs.

Special cases

(i) Case of conditional branching

Figure 6-11 illustrates the case of conditional branching with an example of a conditional repeat on Task 2 with two possible branches: branch [Task3-Task4] and [Task5-Task6]. The execution graph show a first run (Run 1) which takes branch 1 and a second run (Run 2) which takes branch 2.

(ii) Case of conditional repeats

Figure 6-12 illustrates the case of conditional repeats (a loop) with an example. The

workflow shown in the example has a condition on Task 3, if the condition is met the execution moves on to Task 4 otherwise the execution loops back to Task 2. The execution graphs show two runs:

- Run 1 with no repeat, resulting in one and only execution of [Task2 and Task3]
- Run 2 with one repeat, resulting in one repeat of [Task2 and Task3]

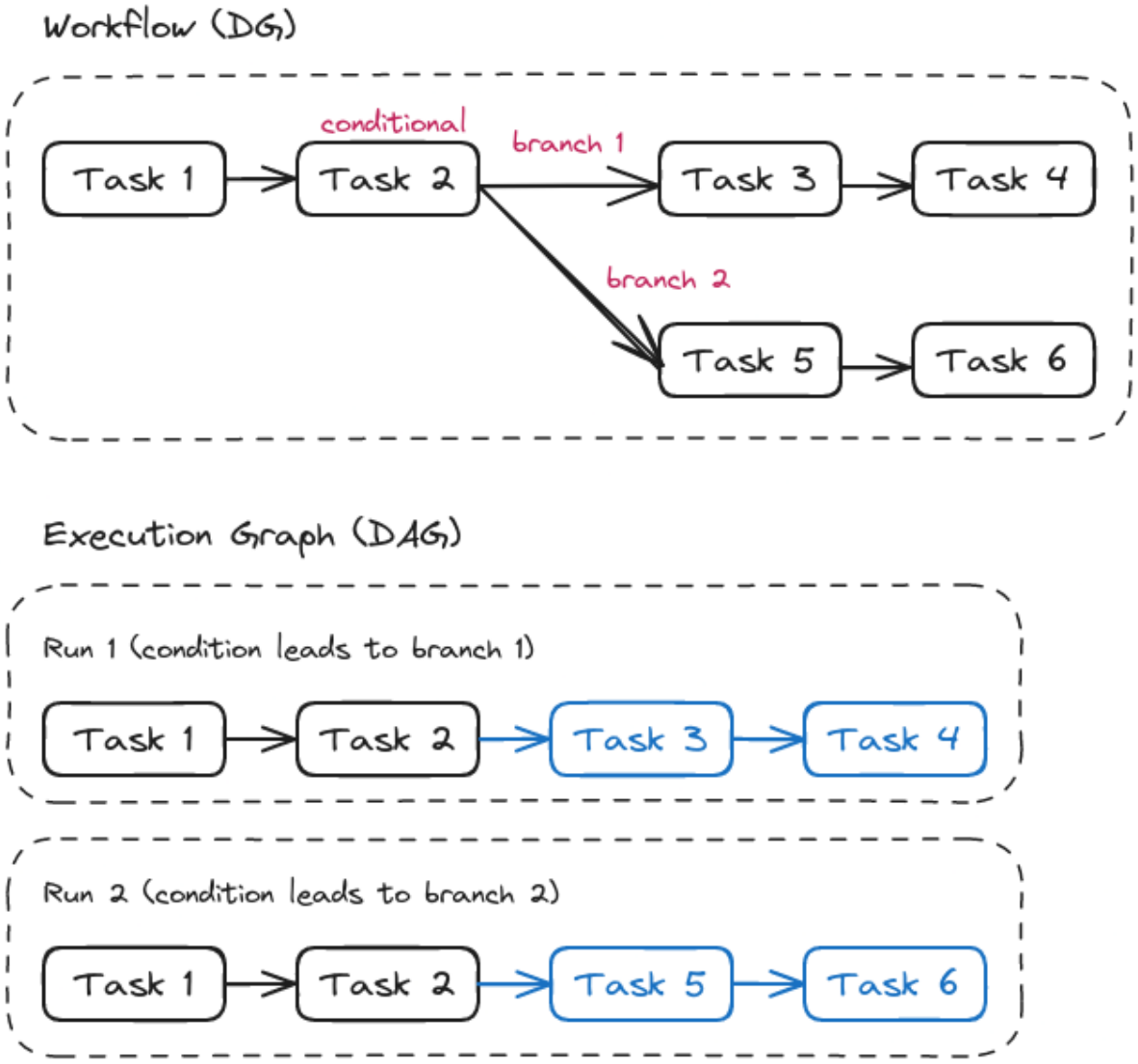


Figure 6-11 – Example of the case of conditional branching. Conditional branching happens in Task 2. The execution graphs show a first run (Run 1) which takes branch 1 and a second run (Run 2) which takes branch 2.

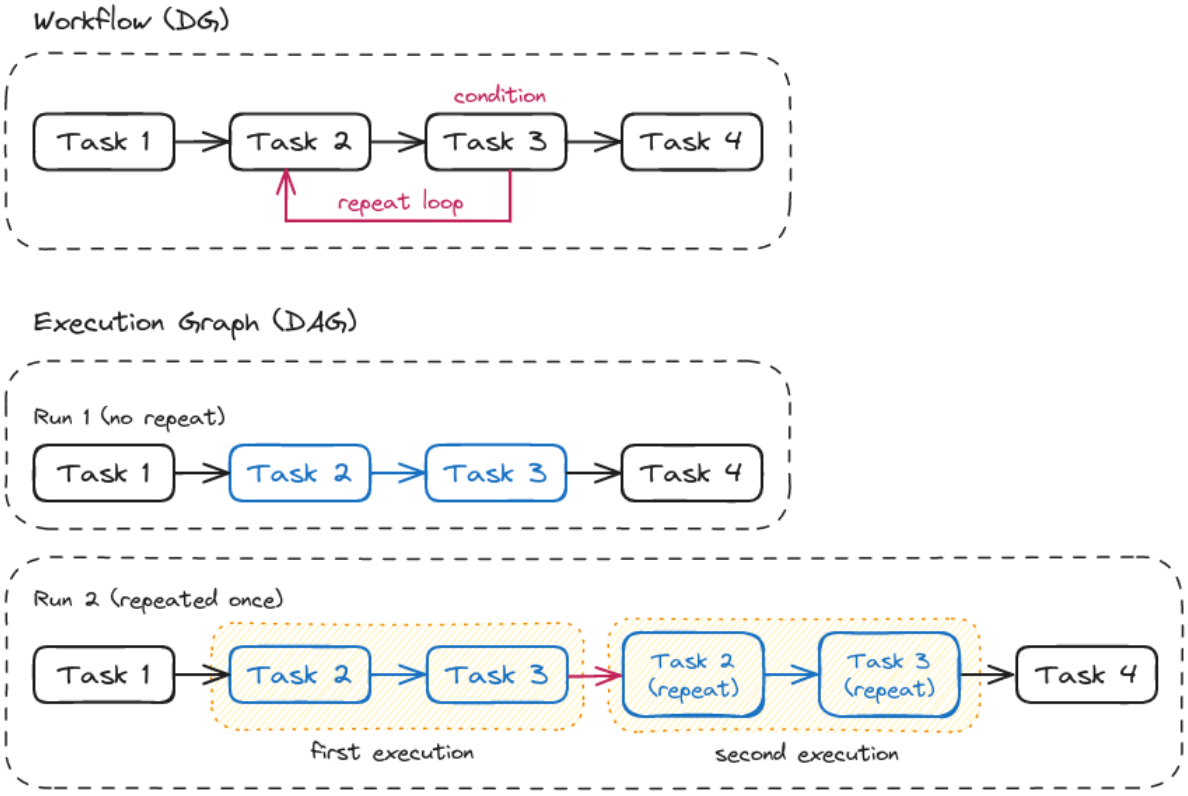


Figure 6-12 – Example of the case of conditional repeats (a loop). In this example Task 3 has a condition, if the condition is met the execution moves on to Task 4 otherwise the execution loops back to Task 2. The execution graphs show two runs, Run 1 with no repeat and Run 2 with one repeat.

3. Data Dependency Structuring and Data Provenance Querying

3.1 The importance of data provenance in experiments

Provenance means ‘lineage’. In relation to arts and antiques, provenance deals with how a piece came to be in its current condition and/or ownership. Data provenance is information about the origin of a data item, as well as the transformations used to produce this data item (Glavic & Alonso, 2009).

Data provenance is the process of tracking and verifying the origin, authenticity, and integrity of data as it moves through a system. The process involves capturing information about how a particular piece of data was created or collected, where it has been stored, who accessed it and what changes were made to it over time. This type of traceability can be used to ensure accuracy in research results and compliance with regulations such as GDPR. Data

provenance also helps organisations identify problems that could arise due to incorrect or missing data. In the thesis we will use the term ‘data provenance’ or ‘data lineage’ interchangeably and take them as equivalents.

During the runs various data gets created such as quality control (QC) data and measurement data. It is necessary to link this data to the run to be able to get the correspondence between the data files and the run later on in the data analysis pipeline. There is a need to know which particular set of data at the end of the pipeline corresponds to which biological inputs, which cohort, which constructs?

In scientific experiments, provenance helps the interpretation and understanding of results. By examining the sequence of steps that led to a result. It is possible to gain insights into the chain of reasoning used in its production, verify that the experiment was performed according to acceptable procedures, identify the experiment’s inputs, and, in some cases, reproduce the result (Freire et al., 2008). Laboratory notebooks have been the traditional mechanism for maintaining such information, but because the volume of data manipulated in computational experiments has increased along with the complexity of analysis, provenance becomes hard to manage. As stated by Freire et al. (2008), the manual determination provenance and writing detailed notes is ‘no longer an option – in fact, it can have serious limitations’. I agree with Freire’s statement and have experienced the limitation of manually capturing provenance in the wetlab and in the biofoundry during the experimental laboratory work of the lycopene case-study.

Provenance can be defined as a form of structured meta-data that describes the execution of processes in which one artefact is transformed into another. This record is essential for understanding where data comes from, deciding whether it should be trusted, and integrating it with other information sources (Bartley et al., 2019).

Clifford et al. (2008) define two forms of provenance: prospective provenance and retrospective provenance.

- Prospective provenance is the ‘recipe’ for a procedure. It defines the steps that should be taken to accomplish some goal. This contrasts with retrospective provenance.
- Retrospective provenance is the information on what has happened in a particular experiment.

Prospective and retrospective provenance are neither mutually exclusive nor mutually dependent; some systems accommodate one or the other, and some accommodate both (Vicknair, 2010). The BIM offers both prospective and retrospective provenance through the

model graph (prospective provenance) and the execution graph (retrospective provenance).

As reviewed by Magagna et al. (2020) there exist several provenance models used in specific settings promoted by different international initiatives. The main basic standard (widely-used and referred to as the World Wide Web Consortium (W3C) PROV recommendation) evolved from the Open Provenance Model (OPM). PROV is a specification to express provenance records, which contains descriptions of the entities and activities involved in producing and delivering or otherwise influencing a given object (W3C Working Group, 2013c, 2013a). PROV is a DAG consisting of nodes (“entity”, “activity”, and “agent”). The Open Provenance Model (OPM) is a model for provenance where provenance of objects (whether digital or not) can be represented by an annotated causality graph, which is a directed acyclic graph, enriched with annotations capturing further information pertaining to execution (Moreau et al., 2008, 2011). In addition to the two provenance data model cited above, a systematic review by Pérez, Rubio & Sáenz-Adán (2018). These authors identified 25 major provenance systems which they categorise according to six dimensions: general aspects, data capture, data access, subject, storage and non-functional requirements such as security. Interestingly all the systems reviewed in the work of (Pérez, Rubio & Sáenz-Adán, 2018) use Relational databases or XML files as storage systems – with the exception of BURRITO (Guo & Seltzer, 2012) which uses a NoSql database (MongoDB). Unlike the provenance systems listed by Pérez et al. (2018), the BIM uses a graph database.

One notable project in biomedical research which emphasises on visualisation of data provenance is AVOCADO (Stitz et al., 2016). Although we haven’t developed a dedicated graphical user interface for the BIM, we display the BIM execution graphs in Neo4j’s browser which offers a dynamic view to explore data provenance graphs. We noted the difficulty of visually navigating large graphs (see the example of the ‘Generic workflow’ in Chapter 7 Section 4.14) and acknowledge the necessity of using dedicated visual tools such as AVOCADO which reduces complexity by aggregation strategies. Another interesting piece of work regarding the visualisation of provenance graphs is Visionary (de Oliveira et al., 2022), an application domain-free framework adapted to any system that uses the PROV provenance data model (PROV-DM). Visionary consists of five distinct steps: capture, ontology, data transformation, data analysis and visualisation.

The BIM could support the Visionary framework with the integration through a PROV Ontology (PROV-O) python API which provides a set of classes and properties (W3C Working Group, 2013b).

In synthetic biology, the Synthetic Biology Open Language (SBOL) leverages the World Wide Web Consortium (W3C) Provenance Ontology (PROV-O) (Bartley et al., 2019).

In PROV-O, workflows are represented as a DAG linked by ‘Activities’ executed by ‘Agents’ (e.g., persons, robotics, and/or software tools) according to a ‘Plan’ upon ‘Usages’ of other artefacts. The PROV-O data model is deliberately generic. Although it has previously been used primarily for describing computational workflows, it has been adapted for use in SBOL to describe experimental workflows as well.

The provenance model of the BIM builds upon a DAG, similarly to OPM, but does not specify node objects (types) and can accommodate more than OPM’s three types of nodes (‘Artifacts’, ‘Process’ and ‘Agents’) or PROV-O’s three base classes (‘Entity’, ‘Activity’ and ‘Agent’). BIM can be extended to be compatible with the standards mentioned above, as the nodes of the DAG of the execution graph can be assigned a type freely. The execution graph in BIM is similar to OPM’s provenance graph which is defined as a record of a past execution (or current execution); it is not a description of something that may happen in the future, nor a general recipe (workflow) that could be used to derive future data (Moreau et al., 2008). BIM’s general recipe of the workflow is defined in the model graph.

Jauer & Deserno (2020) analysed five provenance data standard recommendations from DataCite (Brase, 2009), ECRIN (Canham & Ohmann, 2016), Working Group Data Citation (WGDC) (Rauber et al., 2015) and Data Quality Collaborative (DQC) (Kahn et al., 2015). The authors extracted a minimal list of six requirements for provenance models: (1) a persistent identifier (PID) (each data object is assigned a unique, persistently stored identifier. Ideally, a DOI is assigned.), (2) a data origin (the project or event that generated the data), (3) a data creator (a person or institution to be credited for), (4) a data timestamp (the time of dataset creation/modification), (5) data versioning (each transformation result of the data object is stored. Earlier versions are retrievable) and (6) a query PID (if subsets of data are generated or cited, the query is stored with a persistent ID for reproducibility). All these attributes exist and are generated within the BIM (with the exception of a DOI, although the BIM generates a UID). Any number of metadata attributes can be assigned to the nodes in the BIM, thus the BIM can adhere to the PROV data model or any other standard such as the ones cited above.

In the work presented in this thesis, we did not directly implement the PROV data model standard in the BIM. In future work we consider adhering to the PROV-O standard to enable compatibility of the BIM with synthetic biology standards such as SBOL3 (McLaughlin et al., 2020) and LabOP (Bartley et al., 2023).

3.2 Example of the automated dilution standard workflow

To illustrate the notion of data provenance queries we will use the example of the ‘Automated dilution standard workflow’ presented in Chapter 4.

3.2.1 Workflow

Figure 6-13 shows five steps of the 'Automated Dilution Standard Workflow'. It can be noted that this workflow contains two successive Absorbance spectrum steps.



Figure 6.13 – Example of the 'Automated Dilution Standard Workflow'.

3.2.2 Model graph

Figure 6-14 shows the model graph of the 'Automated Dilution Standard Workflow'. It is important to emphasise on the fact that this model graph uses two different plate readers, Clariostar A' and 'Clariostar B' for the two Absorbance spectrum measurement steps but only one single set of Clariostar protocol parameters is used for both plate-readers.

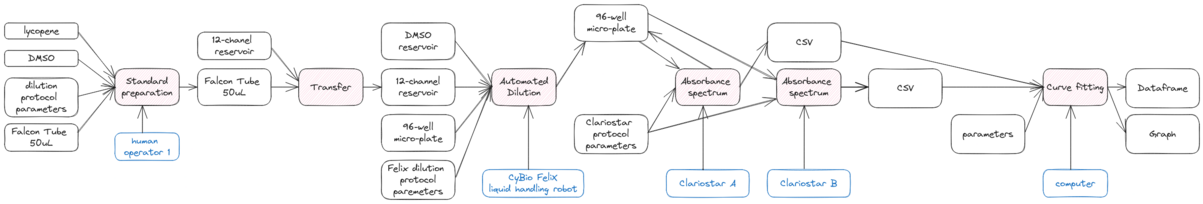


Figure 6-14 – Example of the Automated Dilution Standard model graph. Resources are shown in blue. A full page version of the diagram can be found in the Appendices C6-Figure S1.

3.2.3 Execution graphs from runs

Figure 6-15 and Figure 6-16 show the example of two execution graphs resulting from two different runs: Run 01 and Run 02.

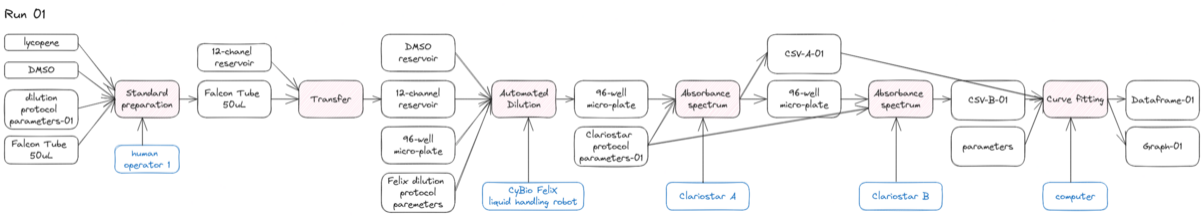


Figure 6-15 – Example of the Automated Dilution Standard execution graph: Run 01. A full page

version of the diagram can be found in the Appendices C6-Figure S2.

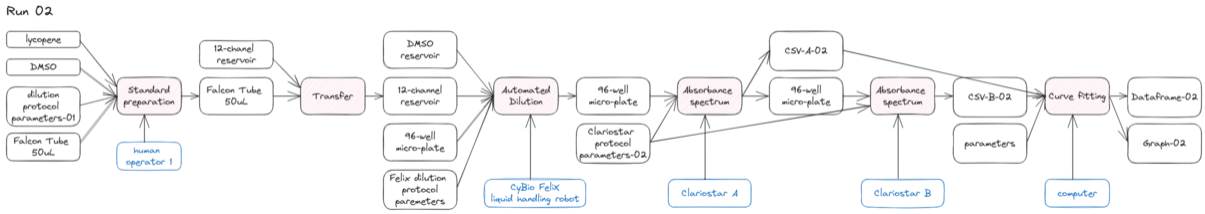


Figure 6-16 – Example of the Automated Dilution Standard execution graph: Run 02. A full page version of the diagram can be found in the Appendices C6-Figure S3.

3.2.4 Data provenance queries

Figure 6-17 and Figure 6-18 illustrate the result of the query of the data provenance, respectively from files called ‘Dataframe-01’ and ‘Dataframe-02’ – which both result from the last step of the workflow. The query on the execution graph asks for:

- The set of parameters that have been used for the measurement protocol in the plate-reader used in the Absorbance measurement steps (i.e. how?)
- The set of parameters that have been used for the standard preparation (i.e. how?)
- The operator who did the standard preparation (i.e. who?)
- The datafiles resulting from the Absorbance measurement (i.e. where from?)
- The equipment used for the measurement steps (i.e. on what ?)
- The run which created the dataframe file (i.e from which run ?)

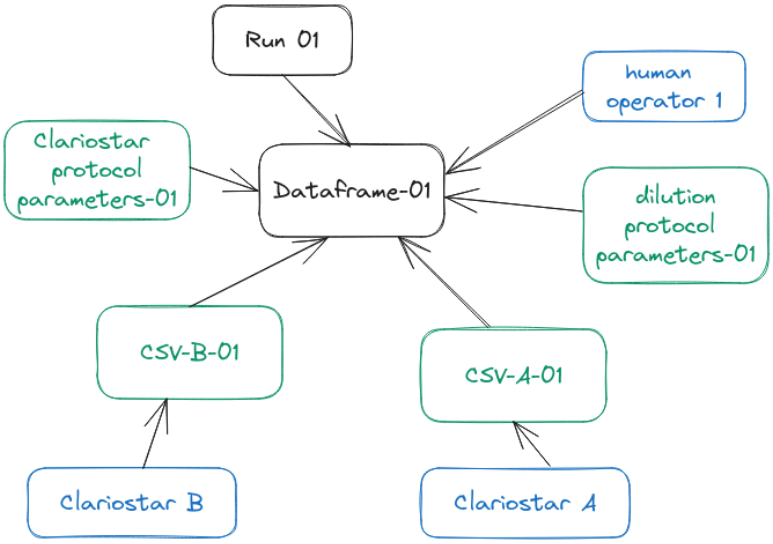


Figure 6.17 – Example of the result of a query for data provenance of the file 'Dataframe-01'.

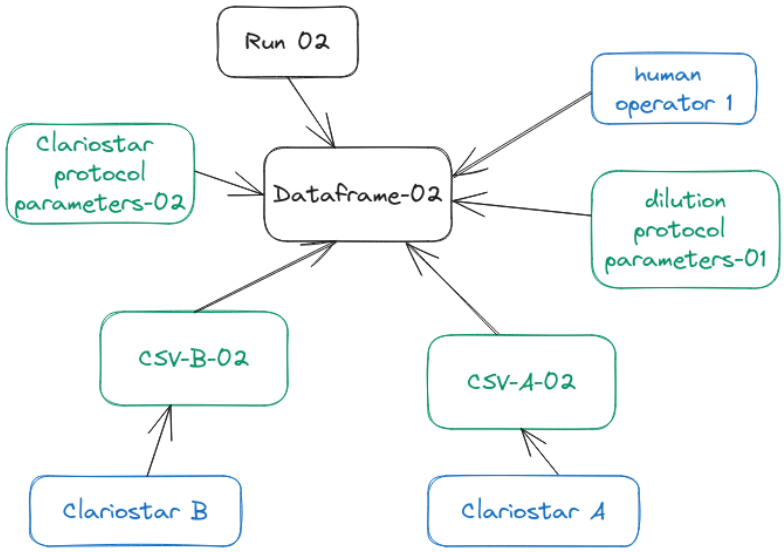


Figure 6.18 – Example of the result of a query for data provenance of the file 'Dataframe-02'.

4. Implementation

In this section we will describe how to store the two different graphs – the model graph and the execution graph that we have seen in the previous sections, in a graph database.

4.1 Using a graph database

We store both the model graph and the various DAG runs of the execution of the workflow in a graph database. Having the graph in a graph database system allows the reading and writing of the workflow as it is being executed – but also allows us to store the data resulting from the execution of the system.

The core graph representing the workflow exists within a database in the graph DataBase Management System (DBMS). Our model graph is implemented over the *Neo4j* database system. Neo4J offers the best functionalities to implement a graph database and stands out for its simplicity. Neo4j offers two architecture options, an intuitive interface and a flexible schema which allows very specific information and very fast retrieval of information (Fernandes & Bernardino, 2018). To interact with this model graph, we created an API implemented as a Python module that interfaces with the Neo4j DBMS. This API allows to create and read any component of the workflow from the system. Because the model is structured as a graph, all the relationships between the components are stored in the model. This graph structure allows various questions about our workflows and their executions to be answered. What is in what ? What depends on what? What creates what? On which resource is executed which task ?

Because this graph can be written in real-time (as the workflow is being executed), it also contains the timestamps of the creation and updates of all the objects. Timestamps allow us to make queries to the DBMS to ask questions such as: When was a component created ? When was a component updated ?

Querying the graph database

Neo4j has a graph query language called *Cypher* (Francis et al., 2018). Cypher is Neo4j's graph query language that lets you retrieve data from the graph. It is like SQL for graphs, and was inspired by SQL (Codd, 1970). Cypher is Neo4j's query language and it has been open-sourced as *openCypher* (openCypher Implementers Group, 2017).

Cypher provides a visual way of matching patterns and relationships. Cypher has a query syntax writing a query is the equivalent of drawing a graph pattern through the data in the form of: `(nodes)-[:ARE_CONNECTED_TO]->(otherNodes)`

The language is almost a direct graphical representation in an ASCII form, with parenthesis

`()` representing nodes and arrows and brackets `-[]->` representing the edges (connections) between the nodes. Figure 6-19 shows the Cypher language query syntax.

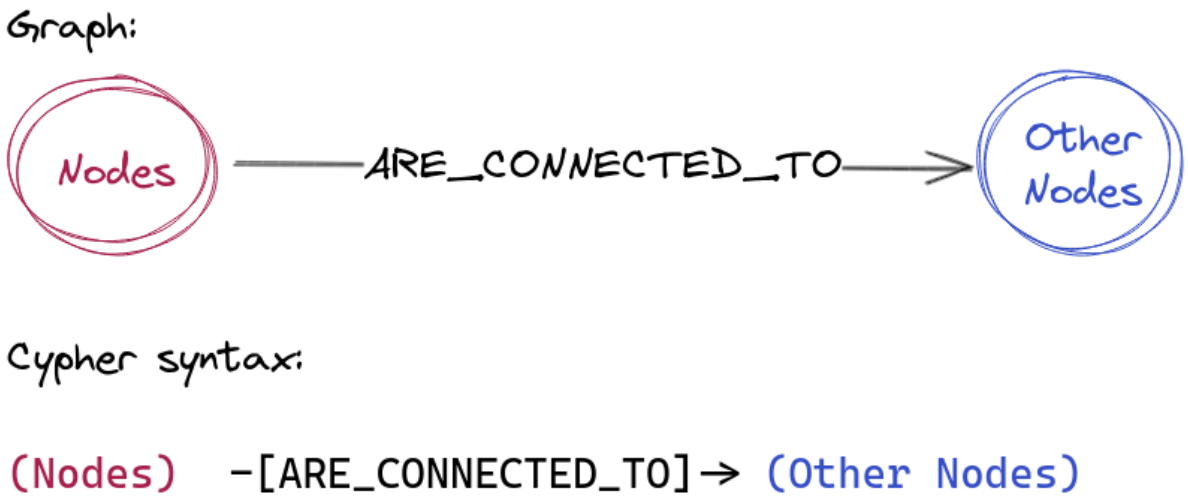


Figure 6-19 – Cypher query syntax. The Cypher query syntax with an analogy between the graph and the ASCII syntax of a query.

Now that we have graph structures which contain the information of our workflows and their executions and that this information is stored in a database, information retrieval is possible. We can write questions (queries in the Cypher graph query language) about what happened during the experiment.

4.2 Workflow libraries

In order to define our workflow as a DG, we have created different software libraries containing all the classes and methods to let us define and build the DG in a programmatic way.

We implemented these functionalities in different libraries:

- Workflow library
- Tasks library
- Input/Output (I/O) classes
- Resource repository

These libraries have been written as python modules. The two different graphs seen in the previous sections, the model graph and the execution graph are stored in a graph database.

The diagram in Figure 6.20 illustrates the technical architecture supporting our framework.

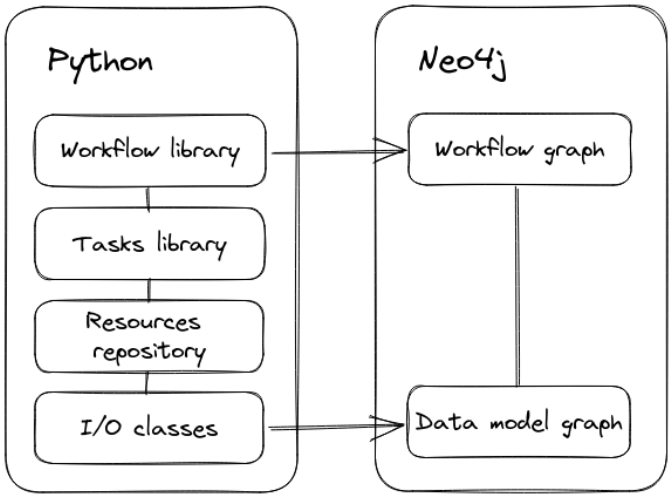


Figure 6-20 – Framework technical architecture. The framework technical architecture with Python libraries and Graph databases.

The workflow library

The workflow library defines different functions that are necessary to define the workflow as a DG.

The task libray

The task library is the core library of reusable task components to build workflows, these tasks are a direct representation of the various computational tasks or wetlab tasks in the biofoundry. This library contains functions which represent the various tasks that build up workflows such as: `` `transfer()` ``, `` `incubate()` ``, `` `vortex()` ``, `` `extraction_DMSO()` `` etc...

I/O classes

The I/O classes represent the virtual components (or objects) of the workflows.

There is a base abstract class representing the I/O objects with has all the common attributes shared among all the types of objects and then sub-classes for each different type of object used within the workflow: `` `files` ``, `` `protocols` ``, `` `reagents` ``, `` `containers` ``, `` `human resources` ``, `` `equipment` ``.

The resources repository

The resource repository contains all the I/O objects that exist in the biofoundry such as the different pieces of equipment and the labware.

The resources repository contains resources of different categories (defined as classes) such as:

- Reagents (chemical reagents, biochemical reagents)
- Equipment (robots, laboratory instruments)
- Containers (tubes, plates)
- Protocols (protocols for automation, manual protocols)

The resource repository can be viewed as a virtual inventory of the whole biofoundry.

Example of reagents: DMSO, lycopene crystals powder.
Example of equipment: Centrifuge, Robotic liquid handler, Incubator, Plate reader, Vortexer.
Example of containers: Falcon tube 50uL, Flat-bottom 96-well plate, 12-channel reservoir.
Example of protocols:

- Lycopene extraction with DMSO protocol for the Felix liquid handling robot
- Serial dilution protocol for the Felix liquid handling robot
- Split-plate transfer protocol for the Felix liquid handling robot

Protocols are versioned. Example:

```
Protocol(id="protocol_dmso_extraction", name="Extraction w/ DMSO",
version="1.0.0")
```

Objects such as equipment (example: a liquid handling robot or a plate-reader) exist in a very small numerable quantity in the lab (usually less than ten) and thus can be referred to by a unique representation (unique name, unique identifier). Since all the metadata associated with them is contained in the object attributes, it is not necessary to provide this additional information everytime the resource is being used.

Example: Anywhere in the workflow code, the equipment of a plate-reader labelled '*Clariostar-A'* from the biofoundry lab 106 can be referred to from its unique name defined in the resources repository: `"clariostar-A"`.

Example: We define a piece of equipment such as the Biofoundry's centrifuge, an Eppendorf 5810R centrifuge, referred to as `centrifuge_epp` as:

```
Equipment(id="centrifuge_epp", name="CentrifugeEpp", model="5810R",
vendor="Eppendorf")
```

The different attributes representing the equipment's metadata information are:

- its unique name: `` `CentrifugeEpp` ``
- a unique identifier: `` `centrifuge_epp` ``
- it's model:`` `5810R` ``

- the vendor's name: `Eppendorf`.

Any additional types of metadata information can be added to any resources as attributes.

Other pieces of labware, (especially consumable plastic containers) exist as classes that can be instantiated an infinite number of times. (For example: to make an analogy with a big box full of hundreds of Eppendorf tubes; there's no need to address each of the tubes individually in a repository, the class acts as the tubes box giving a tube every time it's instantiated.) Only unique pieces of equipment or objects from the biofoundry are given unique names to be referred to; on the contrary throw-away (one user) plasticware gets instantiated from the class directly.

5. Exemplars

We used the novel programmatic framework on two examples of workflow from Chapter 5: BASIC DNA Assembly and Standard Curve.

5.1 Workflow examplar 1 - BASIC DNA assembly (computational)

5.1.1 Workflow presentation

The BASIC DNA Assembly workflow has been presented in detail in the previous chapter (Chapter 5 - Section 3.4.1). We show the diagram of the workflow in Figure 6-21 as a reminder.

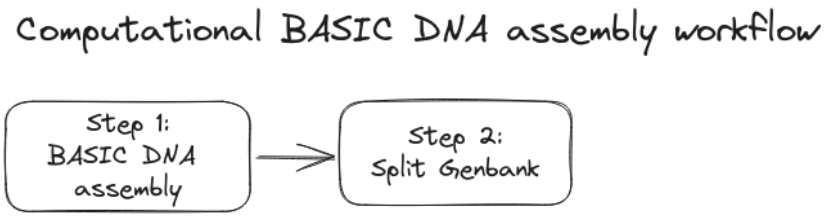


Figure 6-21 – Two-step computational BASIC assembly workflow.

5.1.2 Application of programmatic framework

Figure 6-22 shows the model graph of the BASIC assembly workflow. The diagram contains the tasks of the workflow as detailed in Chapter 5 – Section 2.5.1.

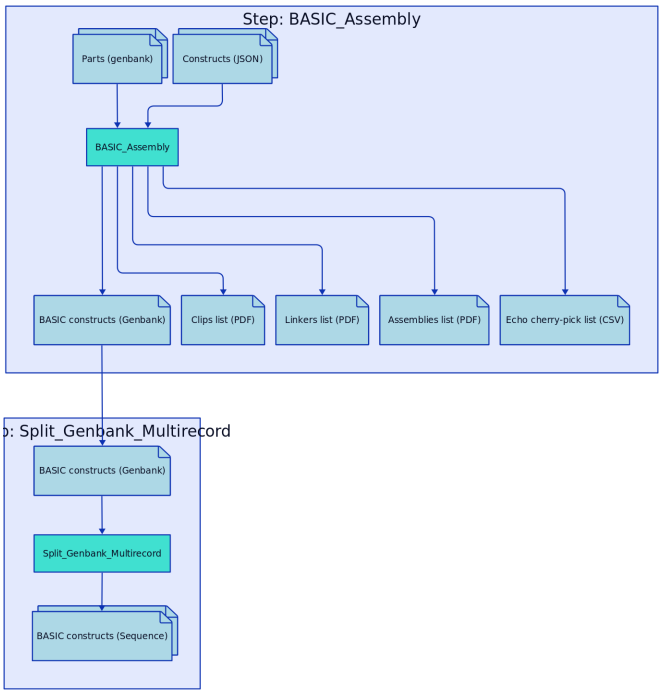


Figure 6-22 – Model graph of the BASIC assembly workflow.

The pseudo-code below shows the tasks functions definitions with their parameters as well as the dependencies represented by the bit-shift operator symbol '>>'.

`'Task A >> Task B'` means that Task B is dependent on Task A.

Workflow pseudo-code:

```
# Tasks definition
      basic_assembly = basic_assembly(parts=parts, constructs=constructs)
      parse_genbank = split_genbank(genbank=basic_constructs)

# Tasks dependencies
      basic_assembly >> parse_genbank
```

Step1. BASIC DNA assembly

The different DNA parts are gathered from a DNA parts repository. These parts, along with a JSON construct template, are then processed by the BASIC assembly Python script that outputs one Genbank file containing all the combinatorial constructs. The Python script, which calls the basicsynbio Python module uses the construct template and the parts collections to generate the constructs. The script also generates the BASIC linkers list, as well as the clips list, the assemblies list and an Echo cherry-pick list.

All the possible combinatorial combinations of collections of genetic parts (mostly RBS and promoters) that we express in order to find the best construct(s) according to some criteria

(lycopene yield in our case).

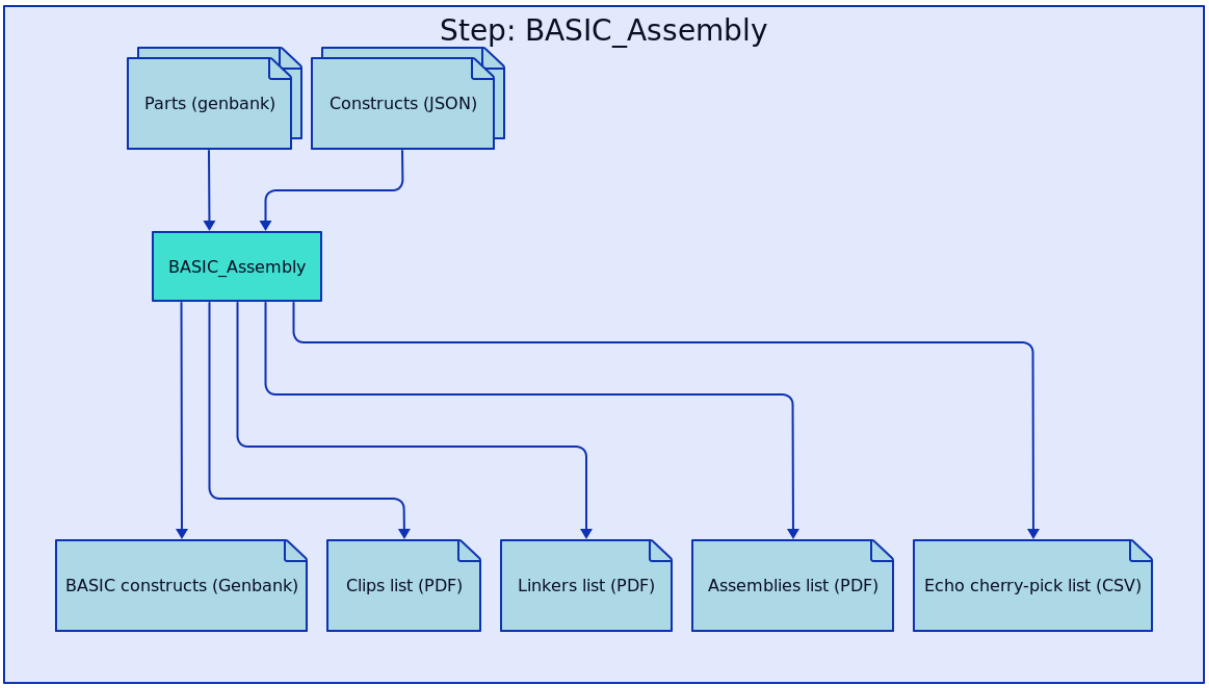


Figure 6-23 – Model graph of step: BASIC_Assembly.

```
FUNCTION DEFINITION:
       BASIC_Assembly(parts, constructs) → basic_constructs, clips_list,
       linkers_list, assemblies_list, cherry_pick_csv

INPUTS:
  - parts (File)
  - constructs (File)

OUTPUTS:
  - basic_constructs (File)
  - clips_list (File)
  - linkers_list (File)
  - assemblies_list (File)
  - cherry_pick_csv (File)

FUNCTION DESCRIPTION:
       BASIC assembly Takes multiple genbank parts and a construct JSON template
       performs a BASIC assembly with the Python basicsynbio module
       and returns a genbank multi-record with the different constructs,
       the clips list, linkers list, assemblieslist and an Echo cherry-pick
       list (CSV).

       Example:     BASIC_Assembly(Parts, Constructs) → BASIC constructs, Clips
                    list, Linkers list, Assemblies list, Echo cherry-pick list

INPUTS:
  - Parts genbank(.gb) (File)
  - Constructs JSON(.json) (File)
```

```
OUTPUTS:
  - BASIC constructs (File)
  - Clips list (File)
  - Linkers list (File)
  - Assemblies list (File)
  - Echo cherry-pick list (File)
```

Step2. Split multi-record

This step splits the unique Genbank multi-record in individual DNA construct individual sequence files. This is the separation of one multi-construct Genbank file into individual files containing the DNA sequence. This is primarily used by our cMatch constructs reconstruction and identification algorithm (See Chapter 3 - Section 5.3) combined if sequencing data.

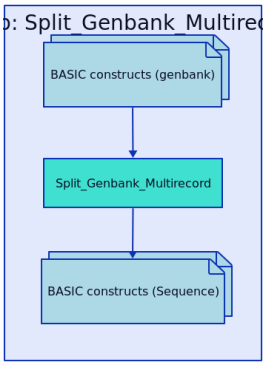


Figure 6-24 – Model graph of step: Split_Genbank_Multirecord.

```
FUNCTION DEFINITION:
       Split_Genbank_Multirecord(genbank) → basic_constructs

INPUTS:
       - genbank (File)
OUTPUTS:
  - basic_constructs (File)
FUNCTION DESCRIPTION:

       Split Genbank Multi-record       This task takes a Genbank multi-
record file as an input and splits it in multiple .seq files

       Example: Split_Genbank_Multirecord(BASIC constructs) → BASIC constructs

INPUTS:
       - BASIC constructs genbank(.gb) (File)
OUTPUTS:
       - BASIC constructs (File)
```

5.1.3 Results

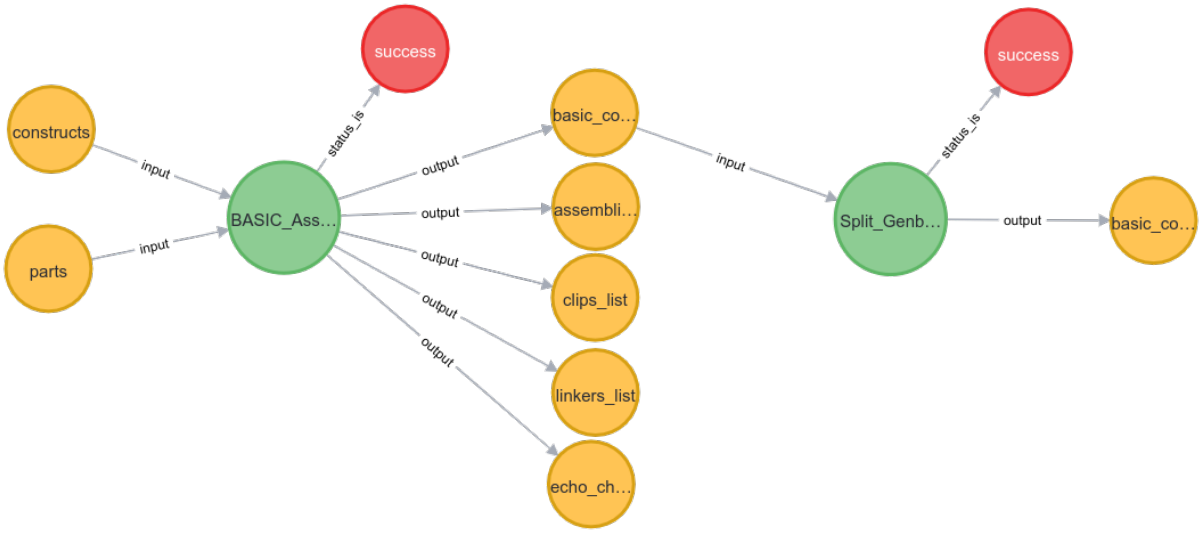


Figure 6.25 – Example of a run of the 'Automated Serial Dilution Workflow'.

5.2 Workflow examplar 2 - automated serial dilution

5.2.1 Workflow presentation

The 'Automated Serial Dilution Workflow' has been presented in details in Chapter 5 - Section 3.4.2. We show the diagram of the workflow in Figure 6-26 as a reminder.

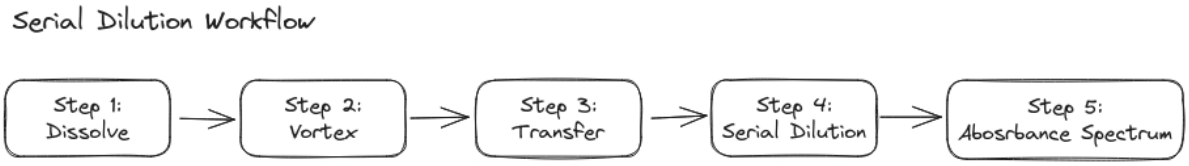


Figure 6-26 – Serial dilution workflow.

5.2.2 Application of programmatic framework

Figure 6-27 shows the model graph of the serial dilution workflow. Figure 6-27 has been generated entirely programmatically. The workflow is defined as python code with the different modules from the BIM. The execution of the workflow code generates a

representation of the model graph as diagram written in D2, a declarative diagramming language, which then gets rendered as an image shown in Figure 6-27.

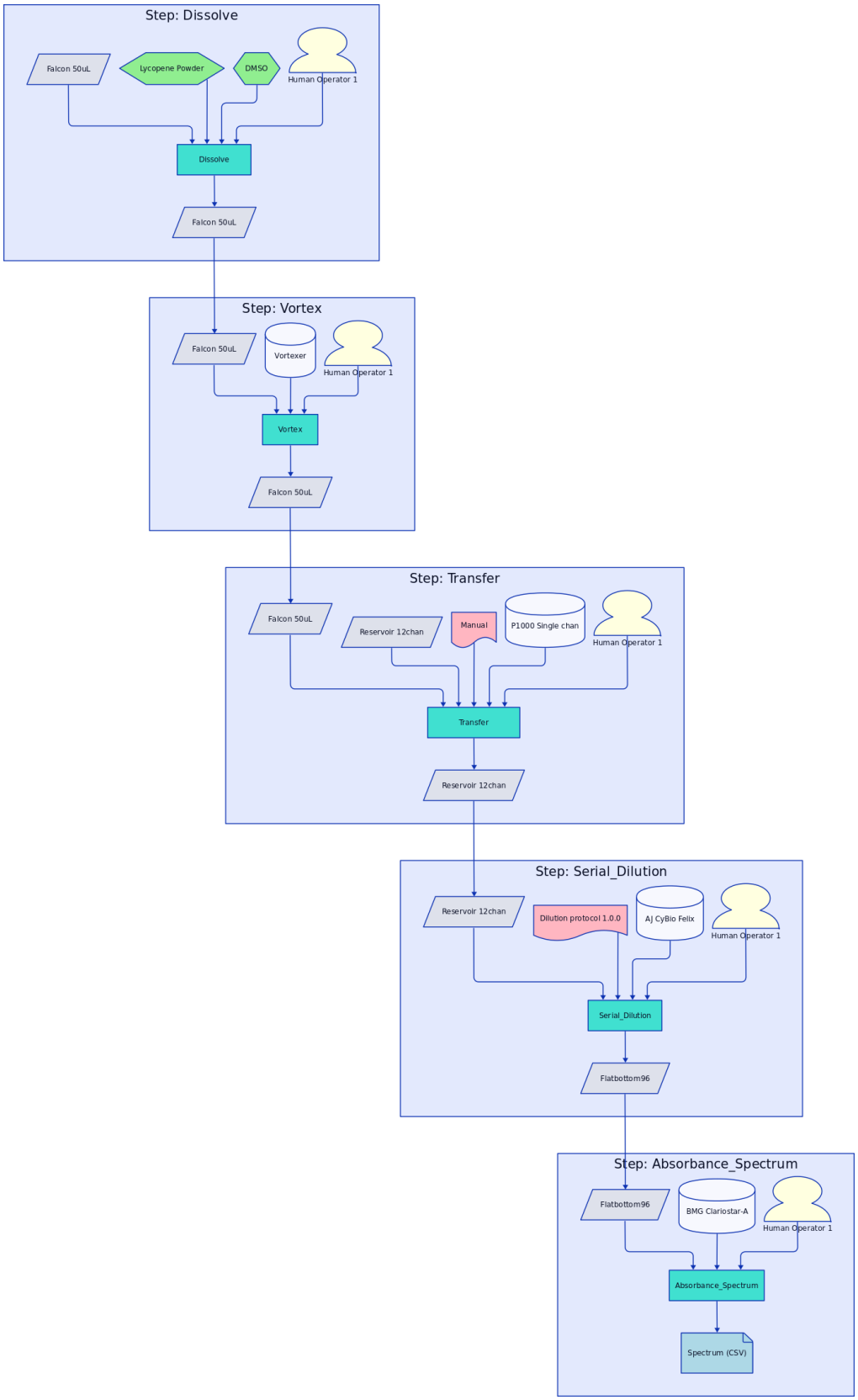


Figure 6-27 – Model graph of the serial dilution workflow.

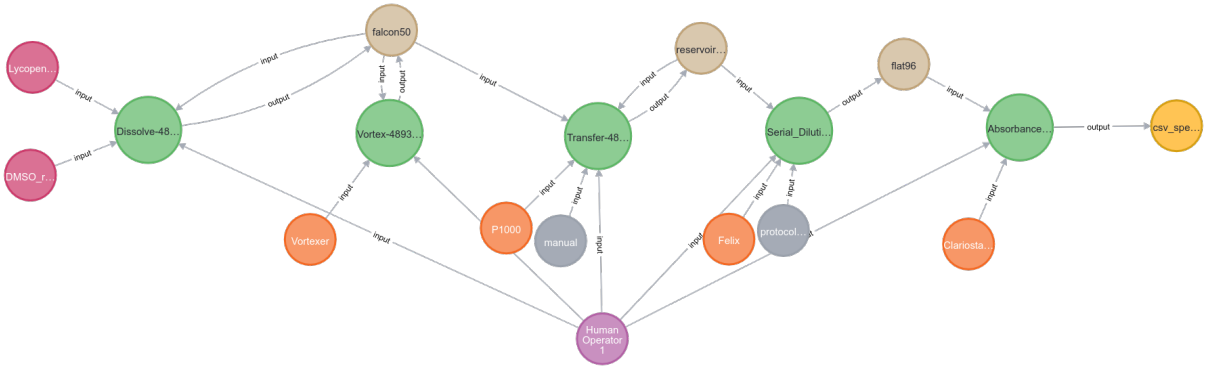


Figure 6-28 – Model graph of the serial dilution workflow as stored in the BIM. The five tasks of the workflow are in green, the equipment resources in orange, the unique human resource is represented in purple, the reagents in magenta, the labware in beige, the protocols in grey and the CSV output file in yellow. Note, there is always a path going from the reagents entry to the final file output passing through labware transfers.

The results are similar to the results in the previous example (see Section 5.1). For a complete overview see C6-Examplar-2.

5.1.3 Results

Figure 6-29 show the example of a run of the automated serial dilution workflow. The input reagents (DMSO+lycopene) are represented in magenta. The CSV output file is in yellow.

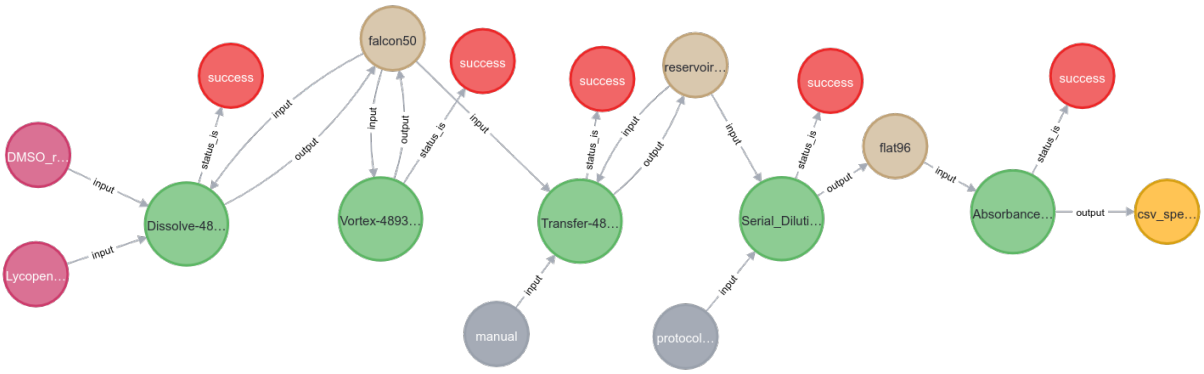


Figure 6-29 – Example of a run of the Automated Serial Dilution Workflow. All the tasks (in green) have been successfully executed and the workflow outputs a CSV file (in yellow) containing the absorbance measurement from the 96 flatbottom well plate. The graph only shows the inputs and outputs (reagents (in magenta), protocols (in grey), labware (in beige), files (in yellow)).

6. Summary

In this chapter we have introduced the structure of directed graphs (DGs) and directed cyclic graphs (DAGs) on top of which workflows can be built. The DAGs allow us to structure the workflow and the dependencies between the different tasks. We also introduced the model graph, a directed graph (DG) that links the dependencies of all the data being produced along the workflow. Workflows can be executed as runs and the result of these runs gets stored in a DAG called an execution graph.

We have shown how, technically, we can implement and build those DAGs with computational libraries and store the DAGs in graph databases. Structuring the data in a DAG and storing it in a graph database allows us to record a history of the runs and query the graph database to find data provenance.

In addition, this way of structuring experiment in workflows built on top of DGs has another advantage that will be described in the next chapter. By following the graph edges of the DG, it is possible to read the order of the task and schedule of the execution of the tasks that it comprises This allows the DG to be used as a blueprint for the orchestration of tasks and automation of the execution of workflows in the biofoundry.

Chapter 7

Biofoundry Automation Manager

In the two previous chapters, we have conceptualised a framework that links the physical world to the information world. We will now see how we can implement such a framework in the context of a working biofoundry. We have shown in the previous chapter that experiments can be represented as workflows. We have also seen that the workflows can be structured in graphs. The idea of workflows and model graphs was introduced and we've seen how we can record the execution of workflows with execution graphs and that we can query data and metadata from the workflow runs.

Wetlab workflows without any supporting system are commonly written in the form of a collection of protocols in paper notebooks, Excel spreadsheets and PDF files. The workflow and its associated tasks and protocols are to be executed on various pieces of wetlab equipment. Figure 7-1 illustrates a typical wetlab configuration.

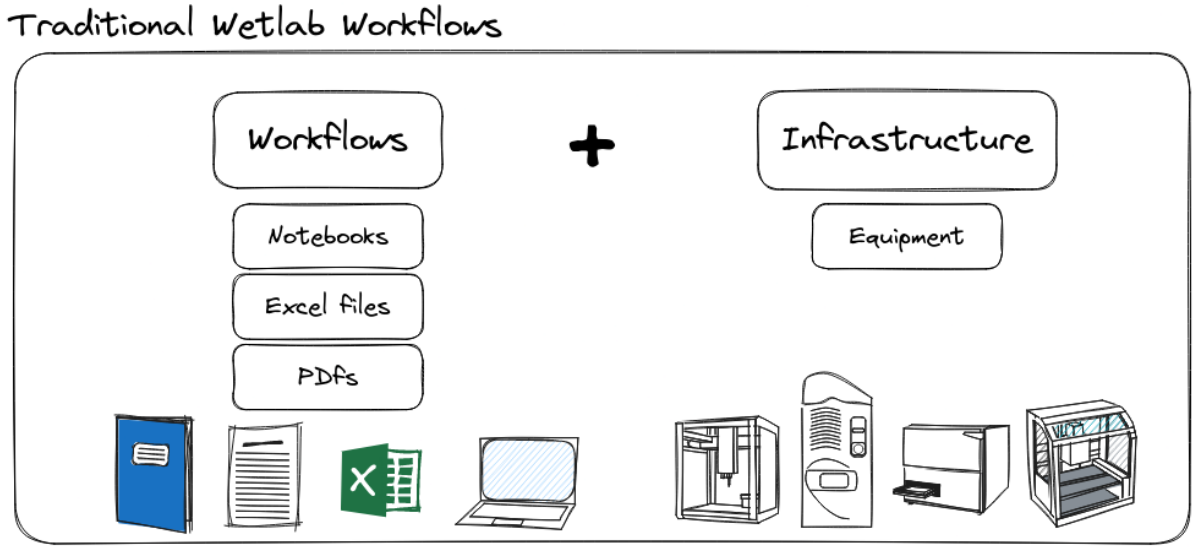


Figure 7-1 – Traditional wetlab workflows. Wetlab workflows where protocols are defined in notebooks and Excel files with their supporting infrastructure equipment.

It is possible to manually follow the workflows and run the various protocols on the supporting infrastructure equipment and collect and organise the data files manually. Figure 7-2 illustrates the traditional manual operations of following a workflow defined as an ordered collection of protocols.

Although this can quickly become a tedious process, it is still possible to proceed manually for relatively simple workflows. More complex workflows that imply conditions, branching and loops and many dozen of tasks can be quite difficult to execute. For the execution of workflows with parallel runs or multiple runs, the data collection and organisation is beyond what is humanly manageable.

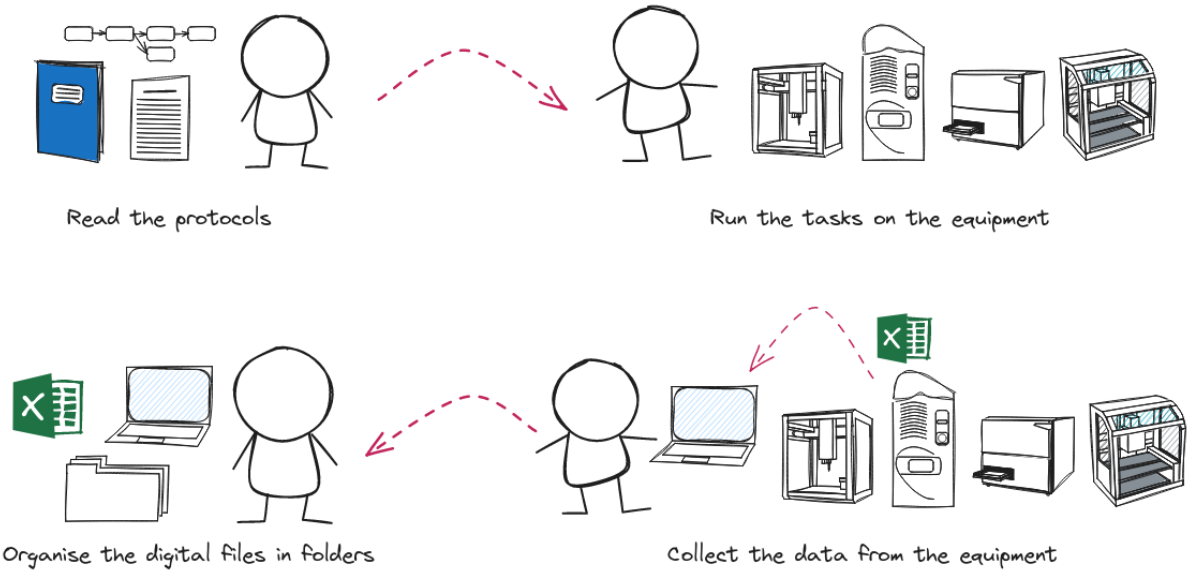


Figure 7-2 –Traditional wetlab workflow. A human operator (HOP) reads the tasks and protocols that are defined in notebooks and PDF files. HOP runs the tasks on the equipment. HOP collects the data files from the equipment. HOP organises the digital files in folders on a laptop.

To be able to manage workflows and collect the resulting data in well structured and a more efficient way, we developed the Biofoundry Information Manager (BIM) that we introduced in the previous chapter (Chapter 6 Section 4.2). The BIM is a system that we built that is supported by a graph database. The BIM allows us to define and store workflows in a graph structure that we call the model graph. This system also allows us to record the execution of the workflows in a structure called an execution graph. Figure 7.3 illustrates the combination of workflows stored in the BIM with the supporting biofoundry infrastructure.

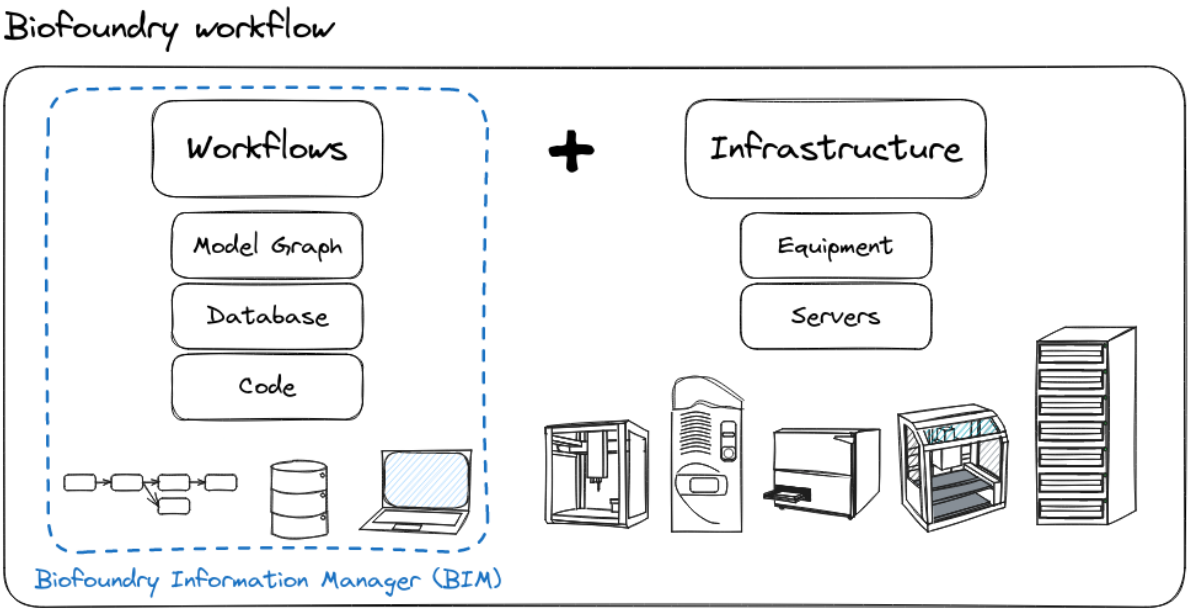


Figure 7-3 – Biofoundry workflow. Biofoundry workflow with workflows defined as model graphs in the Biofoundry Information Manager (BIM) (See Chapter 6) (circled in blue) and their supporting infrastructure equipment.

We built the BIM to be able to store our workflow data in a structured manner. The BIM is built on top of the databases – the workflows can be read from and written to through computational queries. It is possible to read the tasks in the database and manually run workflow tasks on the equipment of the biofoundry and collect the data manually and write it back into the database. Figure 7-4 illustrates a manual workflow supported by the BIM.

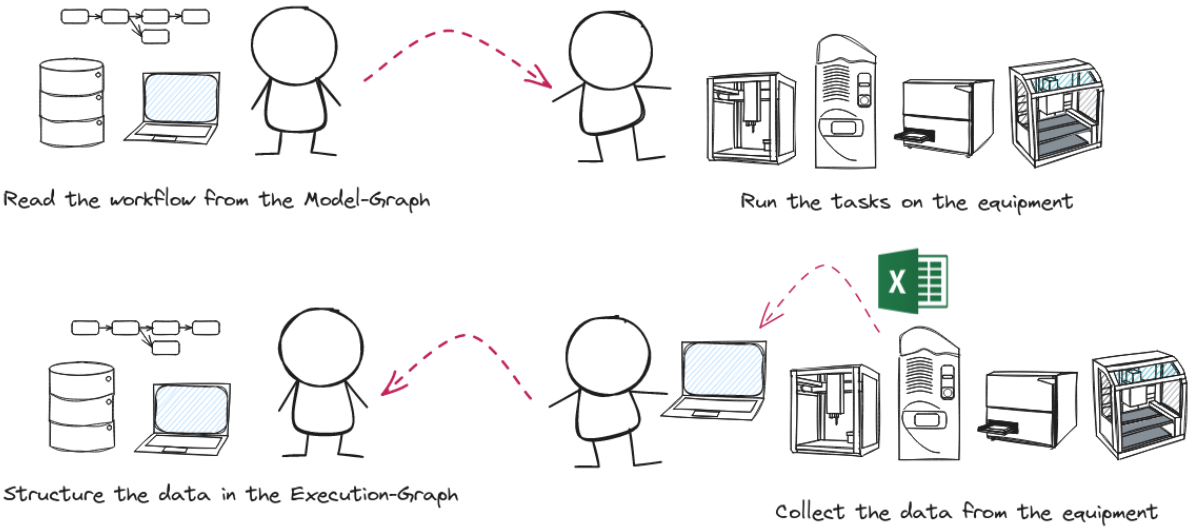


Figure 7-4 – Biofoundry workflow supported by the Biofoundry Information Manager (BIM). A human operator (HOP) reads the workflow from the model graph stored in the Biofoundry Information Manager (BIM). HOP runs the tasks on the equipment. HOP collects the data from the equipment. HOP structures the data in an execution graph with the help of the BIM.

If the data input and output operations – to and from the BIM – are executed manually (e.g. by manually copying the data on a USB key from a piece of equipment and then transferring it to a laptop and then upload it to BIM), the data would be well organised in the BIM but it is a very inefficient way of working. First, data files collection is a tedious manual operation that has to be repeated for each task comprising the workflow and for each run. Second, data file collection requires a very strict file naming convention which is prone to human error. Additionally, such a manual workflow doesn't take any advantage from the graph structure of the model graph (i.e. links and dependencies between the tasks), nor does it take advantage of any automation process.

In this chapter we will describe the use of the structure of the graph and, more particularly the model graph as a template for automating the execution of the workflow.
The model graphs provides a data structure to store workflows. Given the right supporting laboratory infrastructure, if we add a software system called an orchestrator, we can build an automated biofoundry. Figure 7-5 illustrates the combination of workflows plus an Orchestrator and some supporting Infrastructure which results in an automated biofoundry.

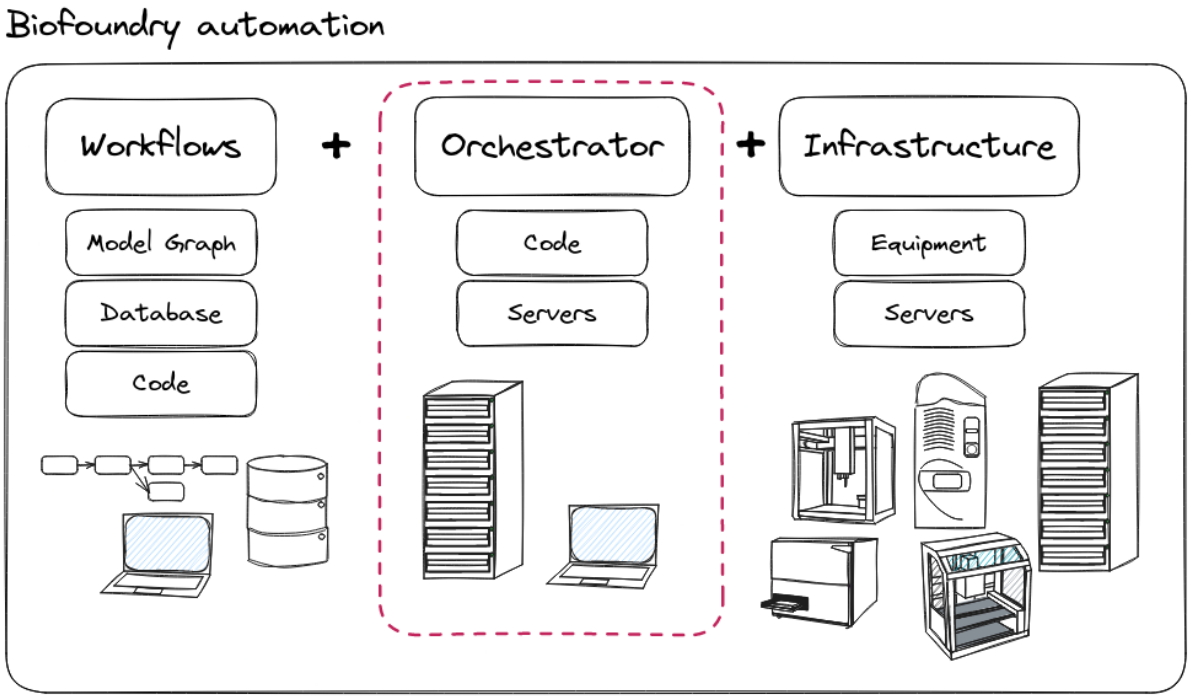


Figure 7-5 – Biofoundry automation. Biofoundry automation is seen as the combination of workflows, infrastructure and an orchestrator (circled in red).

We introduce the Biofoundry Automation Manager (BAM) – a new software system that we developed. In the project, the BAM is built on top of an industry proven orchestrator technology, 'Apache Airflow'. The BAM comprises a number of features:

- Automatically manages the distribution of workflow tasks to the biofoundry's equipment
- Automatically writes the execution of the workflow and collects the data
- Enables the organisation of the data in a structured manner which eases its retrieval in a programmatic way.
- Allows the collection of the history of the execution of workflows as they unfold.

Figure 7.6 illustrates an automated Biofoundry workflow orchestrated by the BAM.

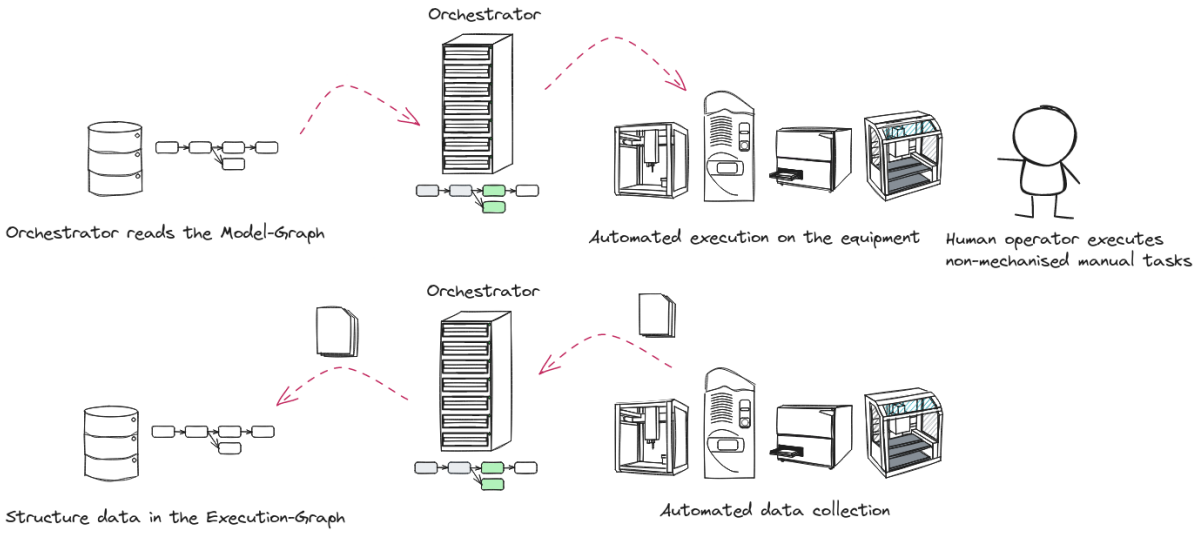


Figure 7-6 – Biofoundry workflow orchestrated by the Biofoundry Automation Manager (BAM). The orchestrator from the BAM reads the model graph and distributes the tasks comprising the workflow on the equipment of the Biofoundry executes them. A human operator is there to assist with manual tasks that are not mechanised. After the completion of each task of the workflow, the orchestrator automatically collects the data and structures it in the execution graph.

The BAM also enables the monitoring of the biofoundry resources and provides a wider, 'cross-experiment' view of the usage of its equipment. The BAM enables the automation of execution of workflows and data collection in the biofoundry in a systematic manner which would otherwise be prone to human error. Additionally, the BAM allows us to run and repeat workflows. The BAM can execute multiple workflows in parallel by scheduling the execution of tasks on shared resources. Automating workflows, manually scheduling the different tasks on shared equipment on the infrastructure and enabling parallel execution of multiple workflows is difficult to manage manually. Managing a Biofoundry manually requires a very strict and rigorous reporting workflow (which is not always done rigorously by the human operators) in addition to constant monitoring of operations which is prone to human distraction and fatigue. The BAM enables the multiplication (scaling) of the number of

concurrent operations in the Biofoundry by automating these tasks.

In the first section of this chapter, we will describe what an orchestrator is and how it allows us to manage the execution of the workflows. In Section 2 we introduce our solution – the Biofoundry Automation Manager (BAM) – for managing workflows and their executions on biofoundries infrastructures and see its architecture. We then describe how this system has been implemented from a technical point of view. In Section 3, we present how we tackle the peculiarities of orchestrating repeats in workflows with the BAM. In the last section of the chapter (Section 4), we demonstrate working examples of workflow executions on the Biofoundry Automation Manager (BAM).

Typically, traditional wetlab work is planned on paper and tasks are executed manually, additionally, the collection and organisation of data is also processed manually (as illustrated in Figure 7-2). In the previous chapter (Chapter 6) we introduced the Biofoundry Information Manager which offers a solution to structure workflows and record and organise their resulting data (as illustrated in Figure 7-4). The BIM adds a level of workflow and data structuring to the traditional way of working in a wetlab. In this chapter we introduce the Biofoundry Automation Manager (BAM) which comprises an orchestrator that distributes and executes tasks on the infrastructure of a Biofoundry and enables automatic collection of data (as illustrated in Figure 7-6). The BAM adds another level of automation of the workflows on the infrastructure of the Biofoundry.

1. Workflow Orchestration

In this Section we describe what comprises an orchestrator and a scheduler, and why an orchestrator is needed to execute workflows.

In a traditional wetlab workflow (and in the context of a shared biofoundry infrastructure), the different researchers, running different experiments (workflows), plan and schedule their work according to the availability of the different resources. Most usually the facility offers a shared calendar, where researchers book a time slot to use the equipment (most of the time resource allocation happens on a first come first serve basis). Such a way of allocating resources and planning is non-optimal as humans tend to book time slots that are longer that the length of the task just to be on the safe side (e.g. when a technical problem occurs or when a repetition of the task is needed but the allocated time on the equipment has ended and the biological process cannot be halted until a future time slot on the equipment becomes available). All the steps of the workflow are time-based and of variable length. Each step needs a certain amount of time for completion. Not all tasks require an equal amount of time for completion. Consequently, manual planning for the execution of a wetlab workflow requires juggling with estimates of the time completion of a task and the availability

of the equipment. For dual computational/wetlab workflow, there is a significant difference between the different computational and wetlab tasks, in terms of their timeframe. Computational tasks can be completed within milliseconds, seconds or minutes; whereas, experimental lab tasks may take many hours and or even days to complete. Tasks can be executed on and by various resources such as computational resources – like laptops, desktops, servers or cloud resources. Some tasks can be completed manually in the wetlab and some are executed by automated equipment in the biofoundry. Manual coordination between all those tasks is often sub-optimal (e.g. some computations run faster than the time that it takes for a researcher to notice that a particular task on some piece of wetlab equipment has ended and to transfer the data to the computer that runs the task).
When automating a workflow there is a need for a system capable of handling the role of task distribution among the resources and task scheduling (i.e. when to launch which task and on what resource). A workflow task orchestrator – whose role is to automatically distribute and schedule the various tasks of the workflow – is needed. The difference in time length between the tasks and the variety of resources for their execution justifies the need for a workflow task orchestrator.

1.1 Definitions

An orchestrator is a software that can execute a workflow and assign tasks to resources.

An orchestrator schedules, runs and monitors the tasks. A workflow orchestrator is a software tool or service that manages and automates the execution of a series of tasks, processes, or services to achieve a specific goal or outcome. It provides a way to create, schedule, monitor, and manage complex workflows that involve multiple tasks, dependencies, and actors. Figure 7.7 illustrates how a workflow distributes tasks over the supporting infrastructure.

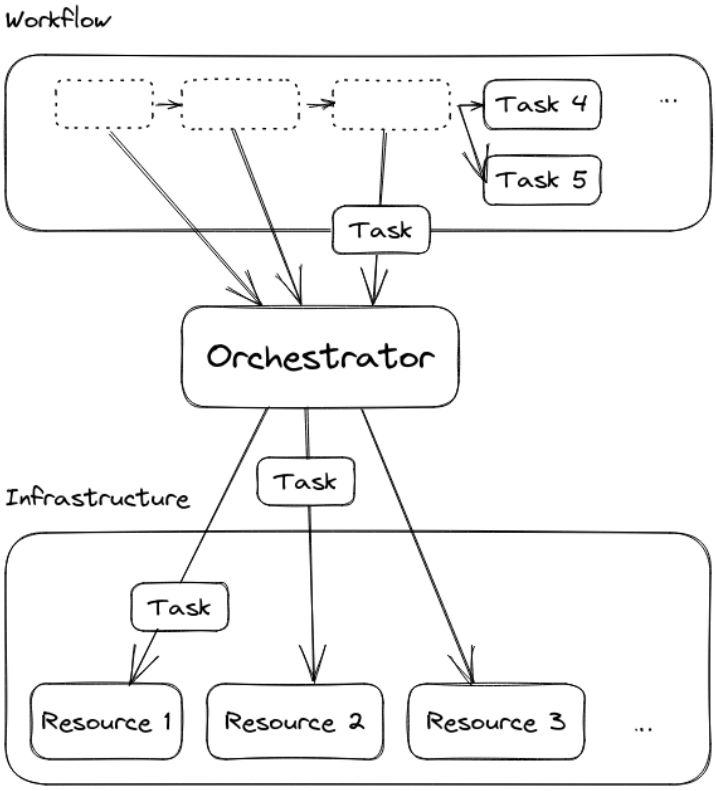


Figure 7-7 – Task distribution with the orchestrator. The orchestrator manages the execution of the tasks from the worklow and distributes them over the resources of the supporting infrastructure.

Hence a workflow orchestrator typically comprises:

1. **Workflow Designer**: A graphical interface or programming language that allows users to define and design workflows. This can include drag-and-drop interfaces, code editors, or visual programming languages.
2. **Workflow Schedule**r: A system that schedules workflows to run at specific times or intervals, taking into account dependencies and constraints.
3. **Workflow Execution Engine**: A component that executes the individual tasks or services that make up the workflow, ensuring that they are run in the correct order and with the right inputs and outputs.
4. **Workflow Monitoring and Management**: A user interface or API that allows users to monitor the progress of workflows, track errors and exceptions, and manage workflows that are in progress or completed.

A workflow orchestrator is used in a variety of industries and applications, such as data processing (Barika et al., 2020), machine learning (Zhong et al., 2022), scientific computing (Singh et al., 2019) and web development (Peltz, 2003). It helps streamline complex

processes and ensures that tasks are executed in a reliable and efficient manner – allowing companies to focus on higher-level objectives and goals. An orchestrator is required to supervise the execution of the workflow. The supervision of a workflow consists of scheduling, running and monitoring the different tasks of the workflow, as well as all objects going in and out of the tasks. The orchestrator acts as a scheduler (see the definition below) and manages the execution of tasks in time.

We are now going to define different terms that we will use frequently in the rest of this chapter. These terms are standard terms employed in the literature and the industry jargon when talking about orchestrators.

(i) Scheduler

A workflow scheduler is a software tool or service that schedules and manages the execution of tasks or processes in a predefined sequence or workflow (Masdari et al., 2016). The scheduler ensures that the tasks are executed in the correct order, taking into account dependencies, constraints, and other factors (Ahmad et al., 2021).

The workflow scheduler includes the following features:

- Task scheduling: The scheduler allows users to define and schedule tasks to be executed at specific times or intervals.
- Dependency management: The scheduler allows users to specify dependencies between tasks, ensuring that tasks are executed in the correct order.
- Parallel execution: The scheduler can execute tasks in parallel when possible, taking advantage of available computing resources to speed up the workflow.
- Error handling: The scheduler can detect and handle errors and exceptions that occur during task execution, providing notifications and taking corrective actions if necessary.
- Resource management: The scheduler can manage computing resources such as CPU, memory, and disk space, ensuring that tasks have access to the resources they need to run successfully.

Workflow schedulers are used in a variety of applications, including data processing, scientific computing, and batch job scheduling. They can help automate and streamline complex processes, reducing errors and improving efficiency. Examples of popular workflow schedulers include Apache Airflow, Luigi, Azkaban, and Oozie (Garg, Wang & Ranjan, 2018; Kempf, 2017). In our system, the scheduler we have chosen is the program that assigns tasks to resources according to some predefined rules and resource availability. Once the workflow gets executed it requires the help of the orchestrator to order and launch the various tasks on the various resources available. Most of the resources such as lab

equipment, and in particular automated lab equipment are shared resources among the wet lab. These pieces of equipment are shared among the different processes executed in the biofoundry. Consequently there needs to be an order in which the different tasks can be run on the various equipment – hence the need for a scheduling process.

(ii) Job

A task is a process defined within the workflow but its execution is called a job within the orchestration vocabulary. A job is the process (also called a ‘batch job’ in computing) submitted to a worker to execute a task.

(iii) Workers

A worker is a computational resource (computer server or cloud server) on which the orchestrator launches a task. An example of a worker is a server (bare metal) or a virtual server such as a VPS or a hypervisor. The supporting infrastructure of an orchestrator usually consists of a certain number of servers and/or virtual servers that support the computational load of the orchestrator. As we have defined the different categories of tasks, we've seen that some of the tasks, like the manual wet lab tasks or the automated lab tasks are not computational (dry lab). The workers allow us to interface these concrete tasks with the orchestrator.

The worker can interact with a workflow resource, typically some piece of equipment in the lab in the case of a wetlab workflow, or a computational resource. Here, it is important to make a distinction between the computational resources needed for the orchestrator and the computational resources that serve the execution of the workflow. The workflow execution has its own pool of computational resources and the orchestration has another pool of computational resources. Whether these servers are shared between the pools or not, is not of a concern – we just make the logical separation between these two computational resources: one for the orchestrator, one for the workflow.

In the context of the work in the biofoundry, the need of a worker can be illustrated with the example of the task of executing a liquid-transfer protocol on the CyBio-Felix liquid-handling robot. The task is defined by the liquid handling protocol that is being executed on the liquid-handling device. The task is the liquid-transfer and the physical resource is the CyBio-Felix liquid-handling robot. The worker acts as the computational resource which will interact (directly or indirectly) with the biofoundry resource.

1.2 The different roles of the orchestrator.

The orchestrator has different purposes in the management of workflows and their tasks – it acts as a task runner, a conductor and a monitor. Understanding the different roles of the

orchestrator will allow us to understand the basic design of the Biofoundry Automation Manager (BAM) the next section (Section 2).

(i) The orchestrator as a task runner

Since the tasks are defined executable code, the orchestrator operates as a task runner ensures that the code of the task is launched on a worker.

(ii) The orchestrator as a conductor

Acting as a conductor of the entire workflow orchestration, the orchestrator serves various roles required for the execution of a workflow:

- Triggering the various runs of the workflow.
- Triggering the tasks composing the workflow.
- Resubmitting tasks: the orchestrator retries tasks and recovers from failures.
- The orchestrator passes data between the tasks.

(iii) The orchestrator as a monitor

The orchestrator alerts when a task is complete or when something fails during the execution of the workflow (when a task fails, or if a worker accidentally stops running or fails to execute a task). As a system that monitors the start, execution and completion of tasks, the orchestrator also logs the status of the tasks (started, running, failed, rescheduled etc…). The status of the various tasks of the workflow can be displayed on a monitoring dashboard. Such monitoring dashboards are most of the time web-services that display a web-view.

1.3 Examples of orchestrators

Many orchestrators exist and many of them are available as open-source projects (Github Workflow Engines list, 2023). Some orchestrator tools exist for Big Data such as YARN or Mesos and have been reviewed by Garg, Wang & Ranjan (2018). Recupito et al. (2022) list a set of tools for orchestrating Machine Learning operations (MLOps). Managing workflows has become even more challenging when dealing with Machine Learning (ML) models. ML workflows do not solely consist of the ML code but also data sources, model management, and model deployment (Recupito et al., 2022). The following are orchestrators that we considered and some were tested for evaluation: Apache Airflow, Luigi, Azkaban, Oozie, Metaflow, Prefect, N8n, Node-RED, Camunda, AWS Step Functions, Cadence.

2. Biofoundry Automation Manager

2.1 Introducing the BAM

This section gives an overview of the Biofoundry Automation Manager (BAM) – our solution for managing an automated infrastructure.

Synthetic iology tasks involve a combination of work from multiple disciplines and various digital and experimental techniques. (virtual/digital *vs* real lab work). Some of the work, like genetic construct designs is done entirely *in silico*, some work done entirely in a wetlab (e.g. cellular cultures) and some work, like lab-automation lies at the intersection of both. Because of this dual nature of the work, an automating solution that works for both types of work, wetlab and computational, is needed.

In the context of iterative engineering cycles (DBTL iterations), it is desirable to operate within a framework that oversees the various work tasks, with a supporting system to develop, process, execute the tasks and track the produced data.

Here we present the BAM, a graph workflow system that enables an asynchronous execution of an application (workflow, pipeline) and enables the structuring of the relationship between the application input data and the produced output data. As described in the previous chapter, we have designed a model graph that can be stored in a graph database. An API was built to get experiment data and metadata in and out of this database. The solution we introduce is the technical bridge between the blueprint of the workflow (the model graph) and its implementation in the biofoundry. To achieve this, we introduced a system called an orchestrator that can direct the workflow execution and overview its environment on an infrastructure.

The solution we introduce offers many advantages that are desirable when running different workflows. Below we list the advantages of our solution and explain why our solution is appropriate for the type of work typically conducted in a biofoundry and why it is appropriate for DBTL applications (such as our application presented in Chapter 2):

(i) Physical-world and digital workflows
Our workflows can be both digital (fast compute, virtual) or laboratory operation based (human manual work, automated or semi-automated). Figure 7-8 illustrates the different types of workflows (computational and wetlab) and their respective resource location for execution (biofoundry, cloud servers).

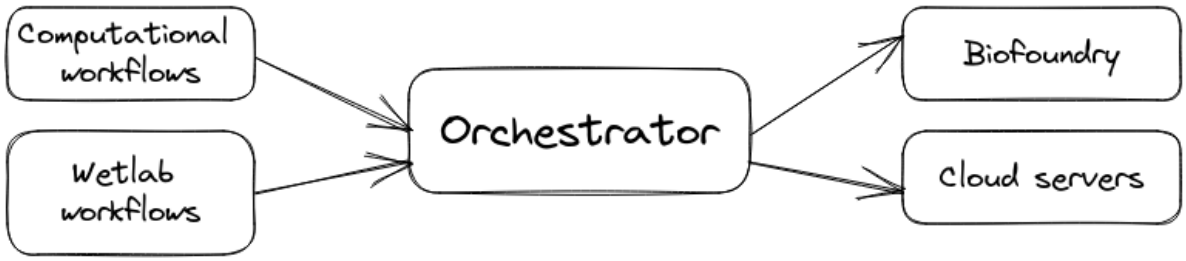


Figure 7-8 – Orchestrating different types of workflows. The orchestrator as a workflow management system fits both computational and biofoundry workflows.

(ii) Time length (asynchronous work)

There is a significant difference between the different virtual and real tasks in their timeframe, computational tasks can be completed within seconds or minutes, whereas experimental lab tasks, mostly due to the biology itself, take hours and days.

(iii) Fits the Design-Build-Test-Learn (DBTL) cycle

Applications built upon the DBTL engineering cycle as presented in Chapter 2, are typical applications run in biofoundries and involve an interplay of digital design tasks, wetlab and biofoundry tasks for the Build and Test steps and computational tasks for the Learn step.Our application framework fits that model as it allows to mix and execute digital and wetlab tasks from within the same workflow. The Biofoundry Automation Manager runs both of these types of tasks.

(iv) Overlapping application runs

Each workflow is run multiple times, either through the different DBTL cycles or to just to collect a sufficient amount of data. Because different tasks in the workflows take different time, in order not to waste time (doing nothing while a tasks takes 24h to execute (e.g. a 24h incubation task)) by simply executing workflows sequentially (run the next workflow only when the previous one has finished) it is desirable to run multiple workflows in parallel and shifted in time.

We will now see in the following section how we have designed the architecture of the solution, how we have implemented it and how we integrated its various components.

2.2 The BAM architecture and software system overview

This section will present an overview of the architecture of the solution and its supporting infrastructure. The four main components upon which the system architecture are described. The overall goal of the infrastructure system is to drive and monitor the various processes that comprise the workflows and collect and structure the resulting data produced by these

various processes. The infrastructure system presented here comprises four different sub-systems. Each of the sub-systems is independent. The various subsystems talk to each other and exchange data between them.

The diagram in Figure 7-9 shows the four different components (model graph, orchestrator, software libraries and the infrastructure) of the architecture of the system.The diagram lists the different technologies used for the sub-systems (the Neo4j graph database, the Apache Airflow orchestrator and the Python programming language). Although some of the subsystems integrate free open-source software (Apache Airflow) or commercially available software (Neo4j). These software are not plug and play and required integration (and a lot of code development) into the Biofoundry Automation Manager (BAM).

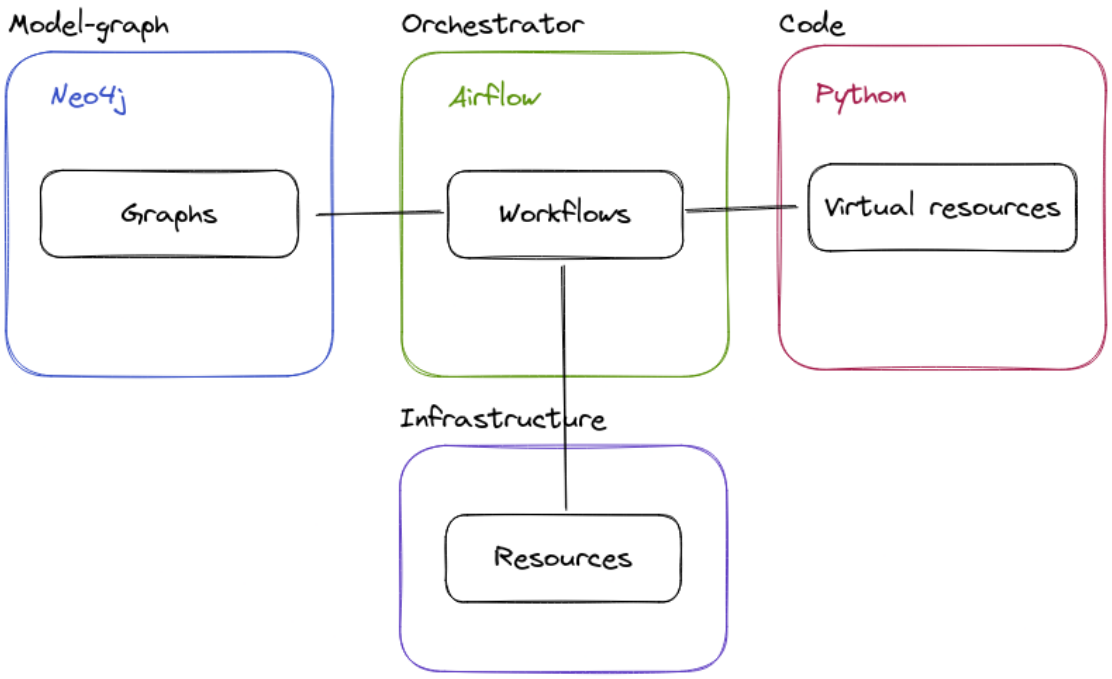


Figure 7-9 – System Architecture. The system architecture and its four components: 1 the model graph (in blue), 2 the orchestrator (in green), 3 the code libraries (in red) and 4 the supporting infrastructure (in purple). The different sub-systems are supported by different technologies (Neo4j graph database, the Apache Airflow orchestrator and the Python programming language).

The four components can be described as follow:

- The first component of the system is a model graph of the workflow. This system comprises an API and a database.
- The second component of the system consists of an orchestrator that can execute the workflow of the experiment processes. The orchestrator comprises an API, a database as well as a pool of workers (computer servers) that can execute independently all the tasks of the workflows.

- The third component is a set of software libraries that support the development of the workflows. These libraries contain the classes and methods necessary to write workflows, some of them contain re-usable tasks that can be shared between workflows as well as classes to describe the various objects that flow between the tasks.
- The fourth sub-system is the supporting infrastructure (the biofoundry and the computational resources)

We will see in the following sections the details of each of the components, (as presented in Figure 7-10). Because the integration of an orchestrator into the Biofoundry Automation Manager required a lot of integration efforts and challenges, will emphasise on the orchestrator from Section 2.4 and onwards.

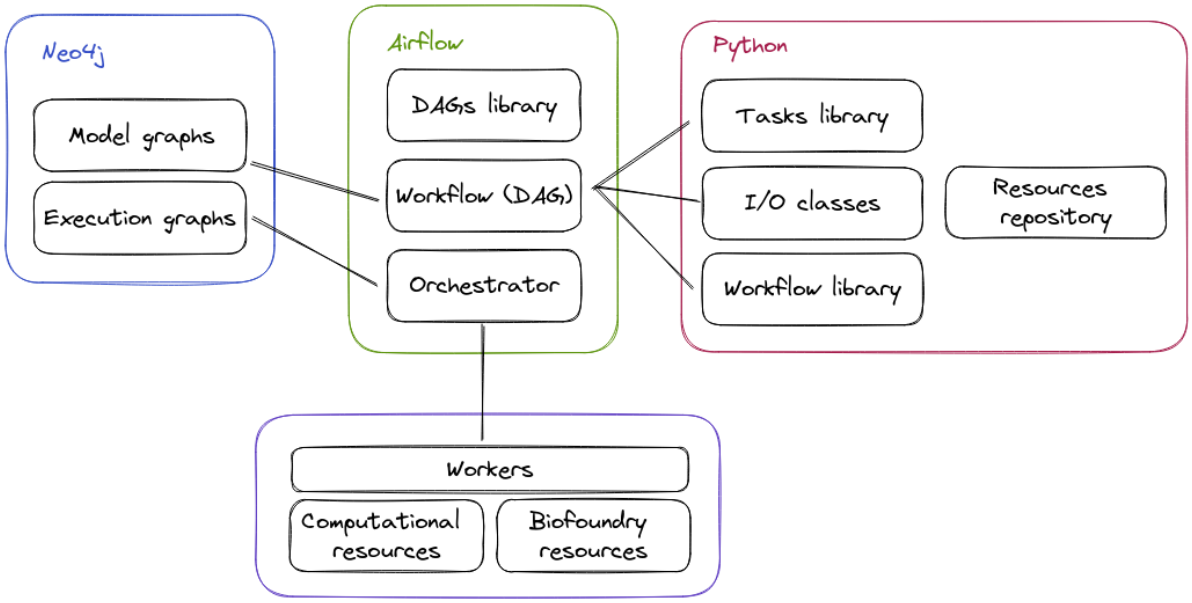


Figure 7-10 – Detailed architecture of the system. The four different sub-systems (shown in blue, green, red and purple) list the different components that comprise them.

The database

We've seen in the previous chapter (Chapter 6 - Section 4.1) that we can store the model graph in a graph database that allows reads and writes of the different objects that comprise the workflow. This graph database also allows us to write queries for information retrieval. The same graph database system (Neo4j) is where we store the execution graphs resulting from the executions of the workflows. Although The Biofoundry Automation Manager (BAM) uses the same graph database (Neo4j) it is not necessary to the same underlying technology base and both systems could be implemented atop two different independent

databases.

The orchestrator

The orchestrator as we will see in the next Section 2.4 is a software system, a workflow management system, that schedules, launches and monitors each of the tasks of the workflow. We have used an orchestrator called ‘Apache Airflow’, that we will shorten to ‘Airflow’ in the rest of this chapter. Airflow's orchestrator is being used for launching the various tasks of the workflow.

In Airflow, the workflow is modelled as a DAG. Airflow uses the word DAG in its terminology. We will call these DAGs: ‘AF DAGs’ (Airflow DAGs) to distinguish them from the execution DAGs seen in the previous chapter (Chapter 6). These AF DAGs are stored in a DAGs library.

Libraries (Python modules)

We have designed four libraries, written as Python modules, that support the system that we will detail later in Section 2.3. These libraries are a task library, a workflow library, some I/O classes (Input/Output) and a resource repository.

The infrastructure

The infrastructure needed for the system combines both computational resources (some servers) and the biofoundry resources (automated equipment, the wetlab and the operators).

2.3 Workflow libraries

In order to define our workflows in the BAM, we developed different software libraries containing all the classes and methods to let us define workflows as AF DAGs. We implemented these functionalities in different independent modules: Workflow library, Tasks library, I/O classes and Resource repository. The four libraries are similar to the libraries used for the Biofoundry Information Manager (BIM) as presented in Chapter 6 - Section 4.2 as they share the same classes. Both systems share exactly the same Input/Ouput classes and the same infrastructure resources (i.e. the biofoundry).

The workflow libraries can be seen in Figure 7-10 on the right side of the diagram highlighted in red. For the sake of simplicity of integration with Airflow, these libraries have been written as python modules: the Workflow library, Tasks library, I/O classes and the Resources repository.

- The **Workflow library** defines different functions that are necessary to define the workflow as code as a DAG in Airflow (AF DAG).
- The **Tasks library** is the core library of reusable task components to build workflows, these tasks are a direct representation of the various computational tasks or wetlab tasks in the biofoundry.

- The **I/O classes** represent the virtual in and out (I/O) components (or objects) of the workflows. There's a base abstract class representing the I/O objects which has all the common attributes shared among all the types of objects and then sub-classes for each different type of object used within the workflow: files, protocols, reagents, containers, human resources, equipment.
- The **Resource repository** contains all the I/O objects that exist in the biofoundry such as the different pieces of equipment and the labware.

The Resource repository can be viewed as a virtual inventory of the whole biofoundry.

Objects such as equipment (example: a liquid handling robot or a plate-reader) exist in a very small numerable quantity in the lab (usually less than ten) and thus can be referred to by a unique representation (unique name, unique identifier). Since all the metadata associated with them is contained in the object attributes, it is not necessary to the information every time the resource is being used. Example: anywhere in the workflow code, the equipment Clariostar-A from lab 106 can be referred to as `"clariostar-A"`.

Other pieces of labware, especially consumable plastic containers exist as classes that can be instantiated an infinite number of times. (Example: to make an analogy with a big box full of hundreds of Eppendorf tubes; there's no need to address each of the tube individually in a repository, the class acts as the tubes box giving a tube every time it's instantiated)

2.4 The choice of Airflow as an orchestrator

To implement our system, we made use of the Apache Airflow orchestrator. In order to implement the Biofoundry Automation Manager (BAM) in a reasonable timeframe we decided to select an orchestrator that we could integrate seamlessly with the rest of the system. The choice was made according to several criteria:

- Integration with the Python programming language. The BAM was written in Python and the orchestrator needed to be able to integrate with Python.
- We wanted an orchestrator that could be both run locally (for the ease of the development) and which could be deployed on servers for a future deployment in the biofoundry.
- We wanted a system with a graphical user interface (GUI) to ease the demonstration of its functionality.
- Good documentation to ease its integration.
- A proven record of usage in a production environment in some other industries. We wanted to rely on a production proven and robust system to avoid potential failures when running in a biofoundry.

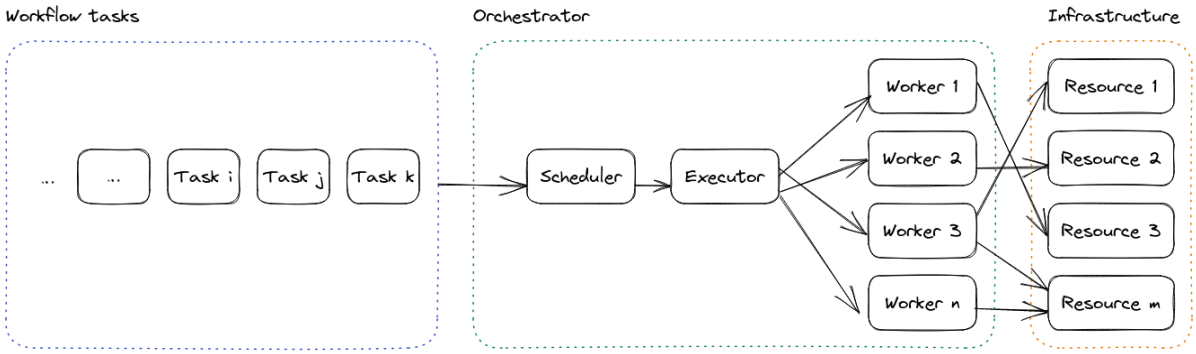


Figure 7-11 – The orchestration process. Detailed view of the orchestration process dispatching tasks through the scheduler and the executor down to the workers which then call the resource to execute the task on.

Airflow is industry proven. Airflow is an open-source workflow management platform that has been around since 2015 (Haines, 2022). Airflow is a project that has been proven in the industry; it originated at the company AirBnB, an online marketplace for short-term homestays.

Airflow is written in Python, and workflows are created via Python scripts. Python is one of the most commonly used programming languages within the biotech industry (Howard et al., 2021) and synthetic biology community (Yeoh et al., 2021). Most of the researchers and engineers at the London Biofoundry have some knowledge of Python. It is the programming language of choice that has been used to create the tools used at the London Biofoundry and for our work. Airflow workflows are expressed in Python. Using Python as a coding language, allows us to import any external libraries (Python modules) to allow us to design custom workflows. Airflow comes with a web interface dashboard to manage workflows and their execution. Airflow displays workflows as graphs and displays the various tasks states with different colour-coded states (scheduled, queued, running, success, failed, skipped etc…). Airflow comes with a variety of predefined re-usable tasks (called '*Operators'* in the Airflow vocabulary) which can be used as is or can be expanded on and customised.

Airflow allows us to both orchestrate and monitor workflows.

Airflow limitations

Airflow as any software tool, has its own limitations. Airflow can be complex for users who are not familiar with Python or programming. Users need to be familiar with Python code to create custom operators, DAGs, and plugins. Airflow can scale horizontally to handle more tasks, but vertically scaling Airflow can be challenging. Airflow's architecture can become a

bottleneck when running workflows with large amounts of data. Although Airflow is open-source and free to use, running it at scale can incur costs for infrastructure, maintenance, and support. Debugging can be challenging in Airflow, especially when dealing with complex DAGs with multiple dependencies. Airflow's UI is limited in terms of functionality, which can make it difficult to track down errors and monitor workflows. Airflow provides limited security features out-of-the-box. Users need to configure security settings manually or use third-party plugins to secure their workflows. Lack of support for real-time workflows: Airflow is not designed for real-time workflows that require sub-second latency.

Despite these limitations, Airflow remains a powerful and widely used platform for managing complex workflows in data engineering, data science, and machine learning (Finnigan & Toner, 2021; Haines, 2022).

Airflow workflows are defined as DAGs and by the intrinsic nature of DAGs: being acyclic, it is not possible to express all the patterns of flows, especially feedback loops (cycles).

2.5 The BAM implementation

In this section we are going to focus on the technical implementation of the Biofoundry Automation Manager and more specifically the challenges of the integration of the Airflow orchestrator with our model of workflows, the model graph. The integration has been challenging because Airflow defines its workflows as DAGs whereas our workflows as being defined as DGs (see Chapter 6 Section 1). We are going to see the first challenge of transferring the workflow tasks – which are defined in the model graph of the BAM – to the Airflow orchestrator by encapsulating a model graph task in an Airflow task (AF task). The second challenge that we are going to discuss is how we overcame the incompatibility between the two models of workflows in the case of cycles in the workflow, in order to accommodate workflows containing loops and repetition of tasks.

2.5.1 Encapsulating model graph tasks in Airflow tasks

Our workflow is being defined by the model graph (as defined in Chapter 6 - Section 1) which is composed of tasks (DG tasks). Airflow uses DAGs (that we will note AF DAGs for the rest of this chapter) to represent workflows. AF DAGs are made of tasks (that we will call **AF tasks**). At a higher level both DG tasks and AF tasks are equivalent but from a technical point of view there is a distinction between them: AF tasks encapsulate DG tasks. This is illustrated in Figure 7-12.

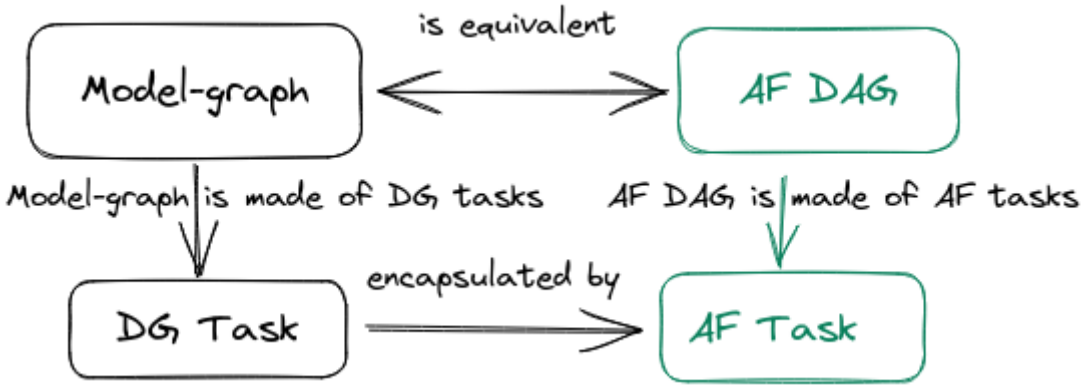


Figure 7-12 – AF task encapsulation.

As illustrated in Figure 7-13 the orchestrator dispatches the AF task on a worker. The worker dispatches the DG task to the resource.

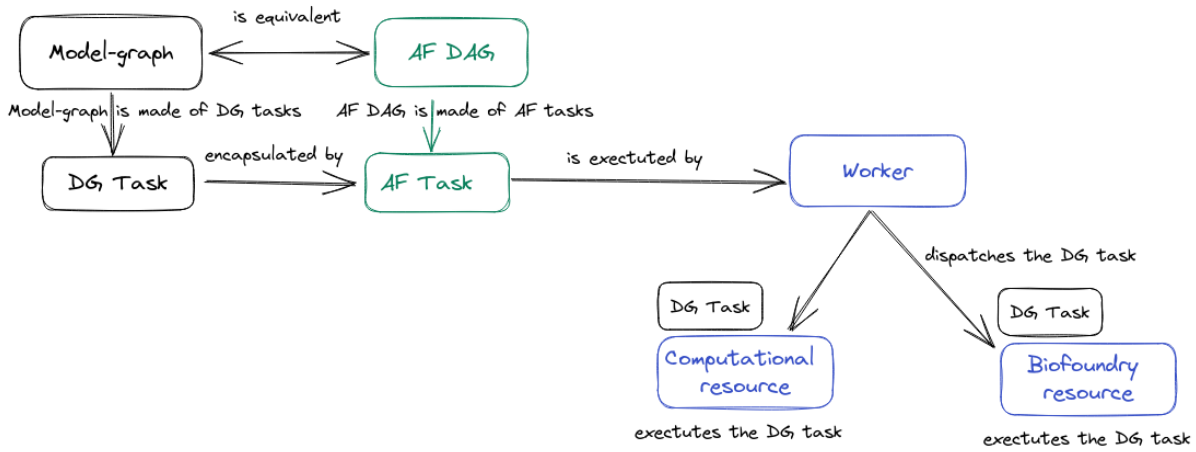


Figure 7-13 – Relationship between the model graph and the DAG in Airflow.

We now illustrate how we implement this encapsulation in practice with the example of a very common wetlab task that we incorporated many times in our work and in our various workflows.

Let’s take the example of a task in a workflow that does a simple liquid transfer from a 96-well plate to another 96-well plate. We will call this task: ‘Transfer’. Such a task is defined in the model graph as a DG task whose function is to drive the liquid transfer in a liquid handling robot as defined by a protocol on the liquid handler. Because we model this task as a AF Task in the AF DAG, the AF Task 'Transfer' is equivalent to the DG task 'Transfer'. The

orchestrator Airflow doesn’t have a direct way to talk to the resources (i.e. the liquid handler), as such Airflow will execute the AF Task ‘Transfer' on a worker. The worker will then drive or trigger the execution of the transfer protocol on the liquid handling robot.

2.5.2 Fitting DG tasks in AF DAGs

We will now explain one of the biggest challenges that we had when integrating Airflow into the Biofoundry Automation Manager (BAM). This challenge has proven difficult because of the intrinsic nature of the two workflow models in both Airflow and BAM. The workflows in the BAM are stored as model graphs which are directed graphs (DGs) whereas Airflow models workflows as DAGs (AF DAGs) – written in Python code. Consequently there is an incompatibility between the two models, the latter can fit the former – a directed acyclic graph (DAG) can fit a directed graph (DG) but the opposite – which is what we need to do – is not always possible.

Our model graphs are stored in a graph database as seen in Chapter 6. To convert the model graphs to python code to create AF DAGs we would need to write a graph parser that outputs python code to create AF DAGs. To avoid writing a graph parser that outputs python code – which is beyond the scope of the thesis – we write our workflows directly in an AF DAG which outputs a model graph. DAGs are directed graphs but the opposite is not true. This is problematic for any cycles that we have in a model graph. If the model graph is a DAG then it fits the Airflow DAG model, if the model graph contains a cycle, it won't fit the AF DAG model as illustrated in Figure 7-14.

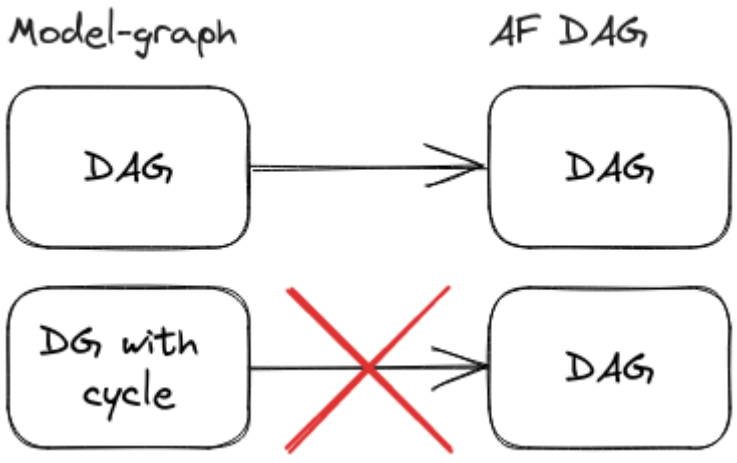


Figure 7-14 – Fitting the model graph in an AF DAG. AF DAGs can only fit model graph DAGs and not cyclic DGs.

This is a restriction that limits the expression of directed graphs and would not allow us to do

any loops in the workflow. Wetlab experiments often require repeats of tasks, whether it is because the task has failed for technical reasons or because the workflow requires a repeat until a condition is met (e.g. incubate until the OD reaches a certain threshold value).

We have seen that we overcame a first challenge of making tasks modelled in the model graph available to the orchestrator Airflow as AF Tasks. We had to integrate tasks defined in the model graph into AF Tasks with encapsulation. The second challenge that we have seen is how to fit workflows with cycles (stored into the model graph as DGs) into workflows executable by Airflow (defined as DAGs which by definition are acyclic); for this we have implemented and tested different strategies.
In the next section we present and discuss the six different strategies that we have tested to allow the orchestrator to do repeats of tasks. We implemented these strategies with different levels off success. We will see the reasons why some implementations of the strategies have failed and which strategy has been our strategy of choice for the implementation in the BAM with Airflow.

3. Orchestration Repeats with the BAM

3.1 Repeats and dealing with repeats in workflows

We call a ‘repeat’, the consecutive repetition of a task (as previously defined in Chapter 6 - Section 2.1.2). Figure 7-15 illustrates the repetition of a task in a workflow and shows an execution graph with a repeat.

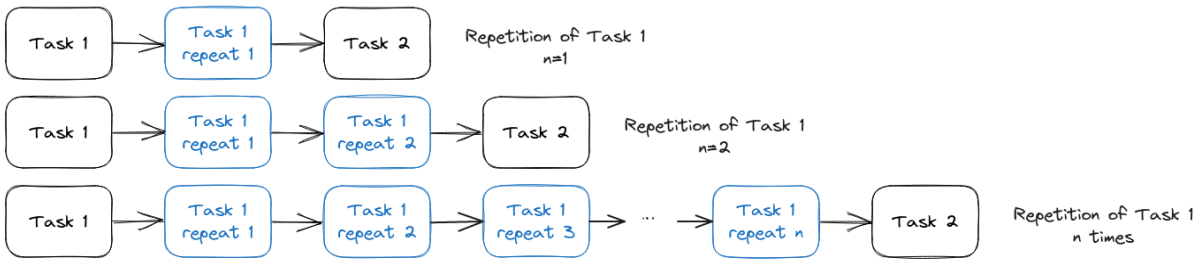


Figure 7-15 – Repetition of a task in a workflow. The repetition of a task (Task1) is shown in the resulting execution graph for n=1, n=2 and n number of repetitions.

There can be different reasons for repeating a task:

1. the task didn't achieve completion
2. the task has an internal condition (set of rules) that triggers its repetition

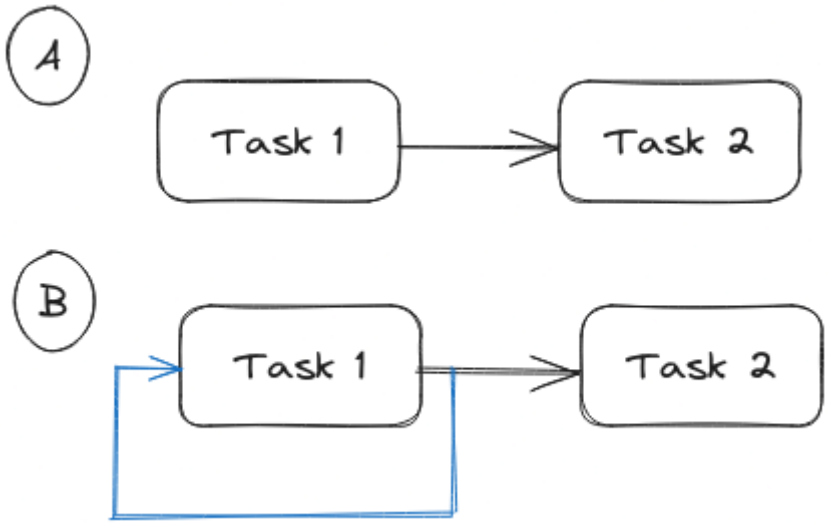


Figure 7-16 – Repetition of a task. A) A simple two-task workflow (Task 1 and Task 2). **B)** The same two-task workflow with a repeat (shown in blue) of Task 1. Task 1 is being executed a certain number of times until a condition is met. Once the condition has been met, the execution of the workflow follows the downstream task and executes Task 2.

Several strategies for solving the problem of fitting a model graph containing a repeat (shown in Figure 7-16) in the AF DAG, are available:

1. Unfold the loop. Chain a pre-defined number of task repeats and skip downstream tasks after the condition is met
2. Do the repeat within the task, implement the logic (loop + condition) inside the task
3. Create a dynamic AF DAG
4. Create a pre-defined number of parallel Tasks, set the upstream task ‘trigger_rule’ to: ‘one_success’
5. Dynamic task mapping (equivalent to a map-reduce)
6. Set the AF task to retry on failure

The implementation of the various strategies has proven to be challenging and difficult (if not impossible for some of them) to do in Airflow. We discuss the implementations of the six different strategies, as well as their pros and cons below:

(i) Strategy 1: ‘Unfolding’ the DAG

When doing repeats in the real physical world, as our experiments occur during a finite amount of time, the number of repetitions is finite and is often within the range of a two-digit number. It is therefore possible to fix the number of repetitions in advance and 'unfold' the DAG by multiplying the task that needs repetition a certain number of times.

At each repeat of the task, we test a condition. If the condition is met, we skip the remaining downstream tasks and move on to the next task of the workflow. Figure 7-17 illustrates the strategy of repeating the task in the workflow.

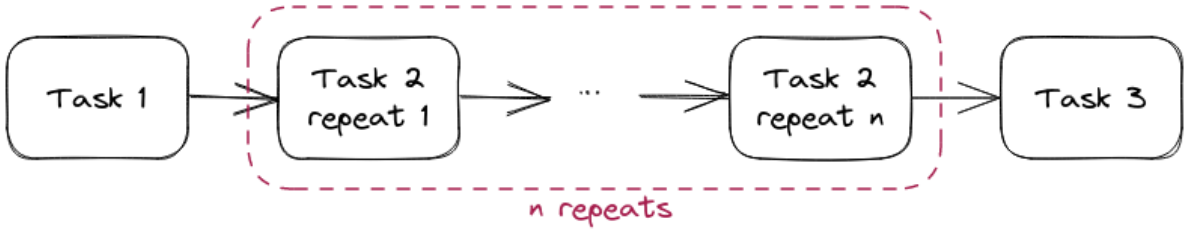


Figure 7-17 – Creating repeats by 'unfolding' the graph.

(ii) Strategy 2: Doing the repeat within the task

This strategy relies on the concept of task granularity as defined in Chapter 5 Section 2.3.1. As illustrated in Figure 7-18 the task (name 'Task 2' in the figure) encapsulates a task and a condition. This strategy has been implemented with success.

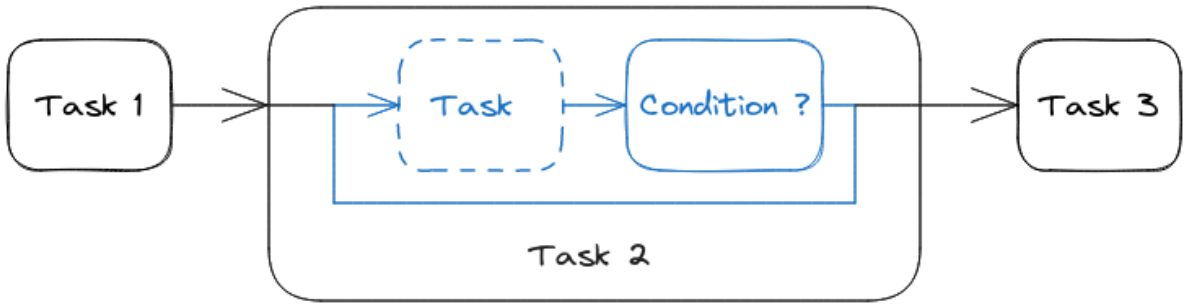


Figure 7-18 – Condition and repeat inside the task. 'Task 2' encapsulates both a task and a condition.

(iii) Strategy 3: Implementing a dynamic AF DAG

Since DAGs in Airflow are implemented in python, it is possible to have dynamic DAG generation. Based on the number of repeats we would generate a number of repeated tasks dynamically at the DAG creation. The implementation of this strategy failed because we cannot loop in an AF DAG and therefore we cannot use the output of a task for the condition.

(iv) Strategy 4: Parallel tasks and upstream trigger rule

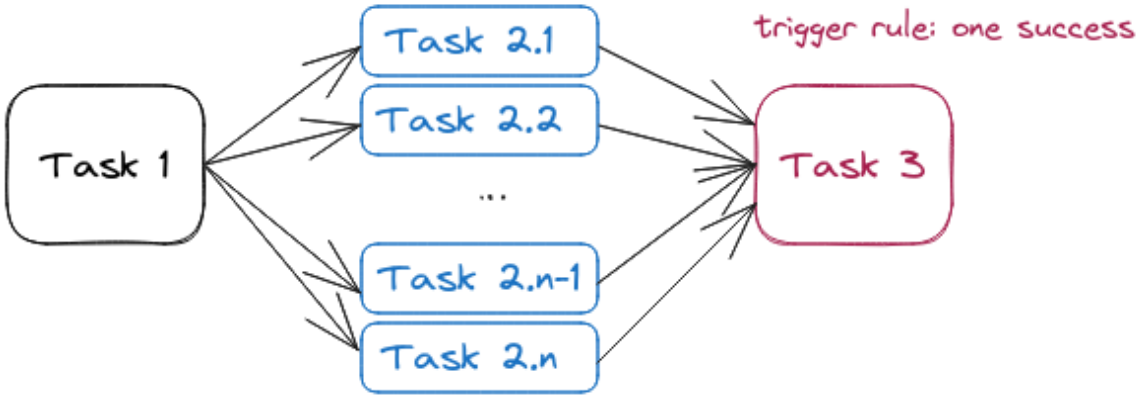


Figure 7-19 – Repetition with parallel tasks and an upstream trigger rule.

The implementation of this strategy would tend to send the execution of the workflow in an apparent stalled state. This state would happen if the trigger rule on the downstream task of the repeat task never got met. Airflow being an orchestrator makes use of a scheduler which decides which tasks will be executed and when. As we fan-out multiple tasks in parallel (see Figure 7-19) we have no control in the order in which the tasks get executed. In addition to not knowing the order of execution of parallel tasks, if there are more parallel tasks than workers, which is often the case. This strategy is not effective because the order of parallel execution of tasks is decided by the scheduler and cannot be guaranteed.

(v) Strategy 5: Dynamic task mapping (equivalent to a map-reduce)

Dynamic task mapping in Airflow allows a way for a workflow to create a number of tasks at runtime based upon current data, rather than the DAG author having to know in advance how many tasks would be needed (as in Strategy 1). This is similar to defining your tasks in a ‘for loop’, but instead of having the DAG file fetch the data and do that itself, the scheduler can do this based on the output of a previous task. Right before a mapped task is executed the scheduler will create n copies of the task, one for each input. It is also possible to have a task operate on the collected output of a mapped task, commonly known as map and reduce. This strategy is similar to Strategy 4 but uses the ‘dynamic task mapping’ feature of Airflow to launch the parallel tasks. The downstream task of the repeat will use some data from each dynamic task to validate the condition.

(vi) Strategy 6: Set the AF task to retry on failure

In this strategy we check the condition at the orchestration level and we set the AF task to retry on failure. The task needs to set the AF DAG to pass or fail depending on a conditional test it has to do itself. The condition inside the task sets the AF task to pass or fail. The

orchestrator repeats the task ('retry') on failure. This creates the equivalent of a 'while condition' loop.

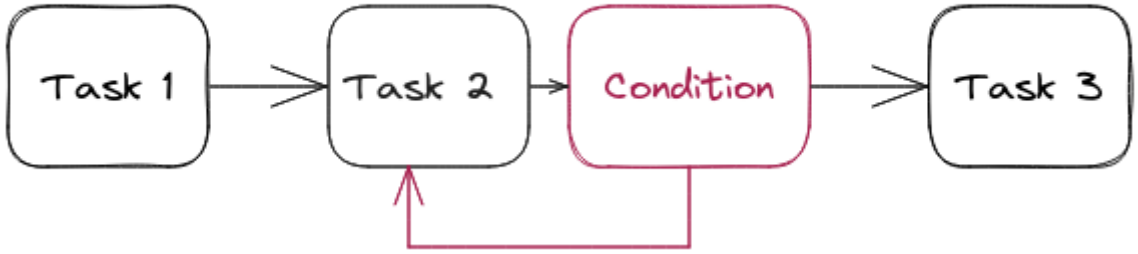


Figure 7-20 – Repetition of a task (Task 2) with a condition at the orchestration level.

The implementation of this strategy works well but is similar to Strategy 2, the disadvantage that this strategy has is that monitoring of the AF task is not possible anymore and the technical failure, at the orchestrator level would not be visible and would appear as an infinite loop.

3.2 Implementing the right strategy for repeats

After trying implementing the strategies with various levels of success, we chose Strategy 2 (doing the repeat within the task) (Figure 7-20).

The first strategy of ‘unfolding’ the DAG works but requires the user to set a maximum fixed number of repeats that would be skipped once the condition is met. This strategy is not appropriate if the number of repeats required exceeds the initial maximum fixed number of repeats. Strategy 2 of doing the repeat within the task itself (by using the granular properties of workflows) is the most practical implementable strategy that we have implemented with success. We discarded Strategy 3 (implementing a dynamic AF DAG) as it is not possible to use the output of a task for the condition as AF DAG**s** cannot loop. Strategy 4 (Parallel tasks and upstream trigger rule) has been implemented with success but is not the most effective strategy because the order of parallel execution of tasks is decided by the scheduler and cannot be guaranteed. Strategy 5 (Dynamic task mapping) has also been implemented with success with the similar drawback as Strategy 4 where the order of parallel execution of tasks cannot be guaranteed. Strategy 6 (Set the AF task to retry on failure) is a solution that we tested and that works but that we discarded because the failure set within the task by the condition would make the failure indistinguishable from a technical failure which defeats the purpose of having the Airflow orchestrator acting as a monitoring system.

To illustrate the successful implementation of Strategy 2, we take the example of a growth-

measurement-induction workflow of transformed bacterial cell cultures. We have used this looping workflow many times in the wetlab. This workflow loop is recurring in almost all synthetic biology workflows that grows bacterial cell cultures after transformation with an inducible construct. Figure 7-21 shows an example of a simple 3-task workflow (step 1:‘culture’, step2: ‘measurement’, step3: ‘induction’) where we repeat the measurement 3 times (initial measurement + 3 repeats). After the execution of the workflow (which includes repeats), we show the result of the run and query the Biofoundry Automation Manager (BAM) – by running a query in the graph database (as shown in Figure 7-22) – to get the execution graph resulting from the run.

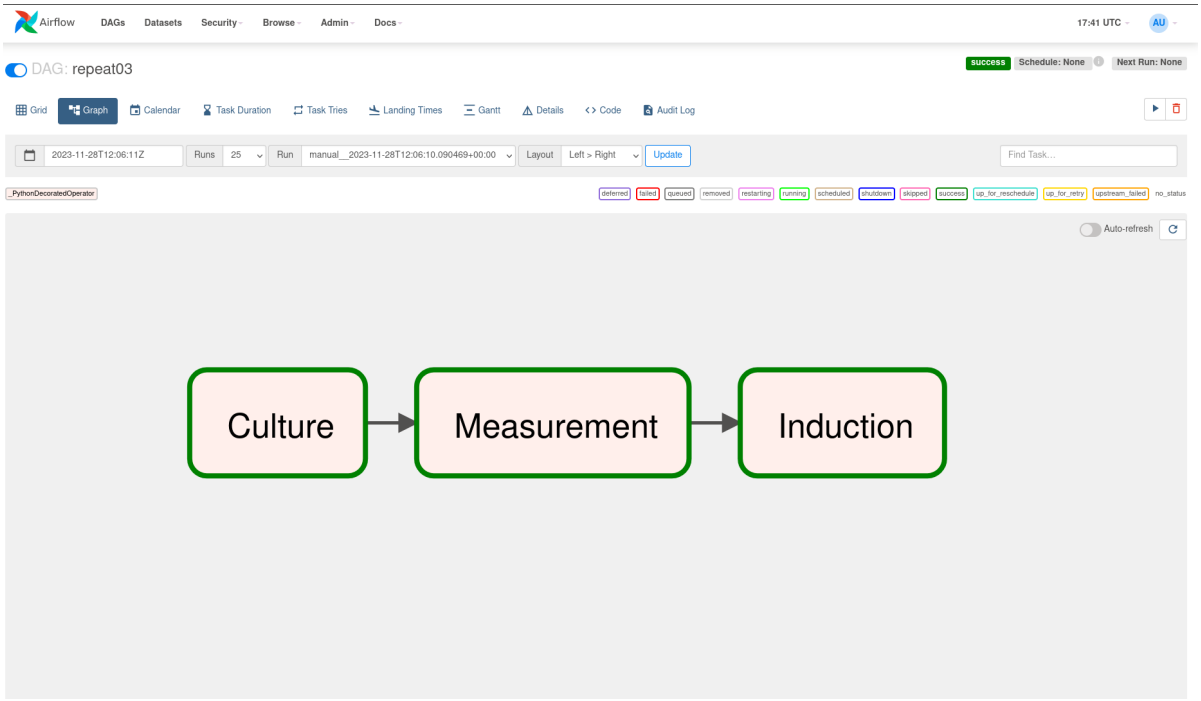


Figure 7-21 – Screenshot of Airflow’s user interface. Repetition of a task. Screenshot of the user interface of Airflow showing a simple 3-task workflow with a repetition on the task called ‘Measurement’ (implemented with Strategy 2).

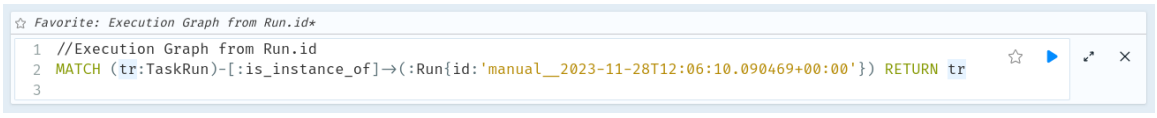


Figure 7-22 – Cypher request to query the execution graph (Screenshot of Neo4j Browser user interface). Screenshot of the user interface of Airflow showing the workflow with a repetition on the task called ‘Measurement’ (implemented with Strategy 2).



Figure 7-23 – Execution graph of the run of the workflow (graph exported from Neo4j Browser user interface). Screenshot of the user interface of Neo4j Browser showing the workflow with a repetition on the task called ‘Measurement’ (implemented with Strategy 2).

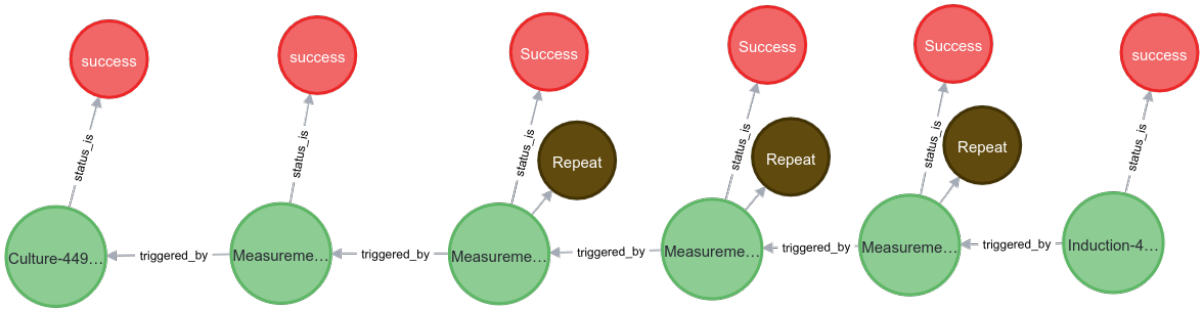


Figure 7-24 – Execution graph of repeats (graph exported from Neo4j Browser user interface). Screenshot of the user interface of Airflow showing the workflow with a repetition on the task called ‘Measurement’ (implemented with Strategy 2).

To our knowledge, the only workflow system that implements repeats with DAGs is the ‘Cylc Workflow Engine’ (Oliver, Shin & Sanders, 2018; Oliver et al., 2019) which needs originate in weather forecast applications and more specifically numerical weather predictions (NWP). In NWP, at regular intervals, *ad infinitum*, vast quantities of meteorological observation data are gathered and processed for assimilation by large atmospheric models and multitude of output data are processed to generate forecast products. Each run of a forecast model has to be partly initialised by a previous forecast because there is not enough information in the observations alone to determine the state of the model.

The workflow, represented as a DAG, when executed repeatedly with any cross-cycle dependence, can be interpreted as a ‘potentially infinite monolithic DAG composed of cycling tasks’ (Oliver et al., 2019). This technically creates a single neverending workflow rather than a never-ending sequence of single-cycle workflows. The authors of Cylc solve the problem by decomposing the DAG and determining the pre-requisite and outputs of each tasks with respect to other tasks and relative to the cycle point, and by creating proxy objects of each cycle task. This strategy for conducting repeats can be seen as a combination of our Strategy 1 (with N being infinite) and Strategy 3 using a file sensor Airflow operator with the addition of proxy tasks. The solution that the authors of Cylc employ requires a scheduling algorithm that allows the dynamic creation of proxy tasks. Such a scheduler is not available in Airflow.

4. Examples of Workflow Execution

We are now going to discuss three different examples of workflows that have been implemented and executed using the Biofoundry Automation Manager (BAM). We have chosen these three examples because they are real examples taken from our experimental workflows.

The first example, even if named ‘Generic Example’ is an example taken from our lycopene application (Chapter 2). The ‘Generic Example’ is a workflow of the growth and induction of transformed bacterial cell cultures. This first example contains repetition of tasks in the workflow which is has been a challenge to implement (see Section 3.2).
The second example is a fully computational workflow taken from the step of BASIC DNA assembly of the DNA constructs of our lycopene case-study (Chapter 2 Section 5.4), this example has been taken to demonstrate that the BAM works for fully computational workflows.
The third example is a wetlab experiment and computational workflow from the Automation Standard Curve (Chapter 4). With these three examples we want to demonstrate that the Biofoundry Automation Manager is a versatile biofoundry automation system that can accommodate fully wetlab, fully computational or wetlab and computational workflows.

4.1 Generic example

The following example illustrates the execution of a simple generic workflow of a bacterial culture that has a conditional repeat with the measurement of the culture at OD600.

4.1.1 Workflow description

Figure 7-25 illustrates the example of a generic workflow where a bacterial culture (transformed with an inducible protein expression construct) is inoculated and the put to incubation for 2h, after incubation a sample of the culture is measured for OD600 and depending on the value measured, the culture is either induced and incubated for 24h or put back into incubation for an additional hour (to reach the right OD) or the culture is discarded if the OD threshold is exceeded.

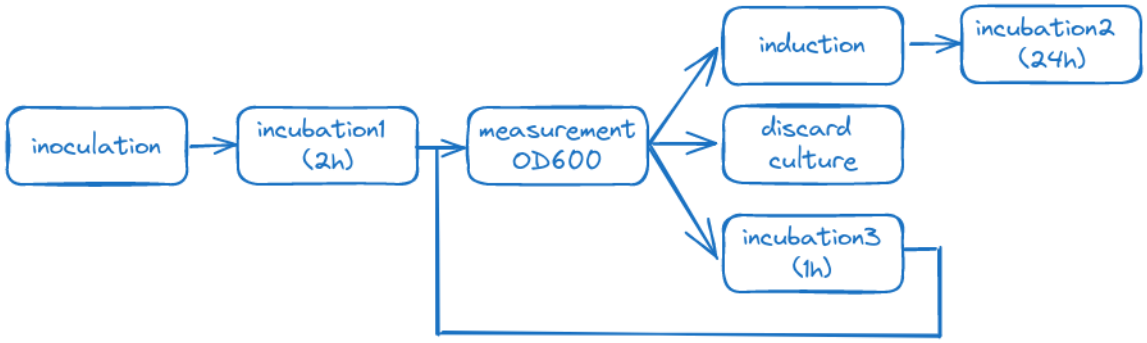


Figure 7-25 – The 'Generic workflow', an example of a workflow which contains a loop. The first two tasks 'inoculation' and 'incubation1 (2h)' grow a bacterial culture that has been transformed with a DNA construct and the culture is measured for OD600. Depending on the result of the value of the measurement, the workflow forks and can take three different paths: the repeat (the culture gets back into incubation ('incubation3 (1h)'), the culture gets discarded, the culture gets induced and incubated for 24h ('incubation2 (24h)').

4.1.2 Workflow implementation

The generic workflow is implemented in the BAM as an AF DAG using the workflow libraries previously described in Section 2.3. Figure 7-26 represents the workflow as it has been implemented. The repeat loop has been implemented using the 'Strategy 2' previously presented in Section 3.2. Both tasks 'measurement OD600' and 'incubation 1h' as well as the conditional loop have been implemented (encapsulated) in one AF DAG task.

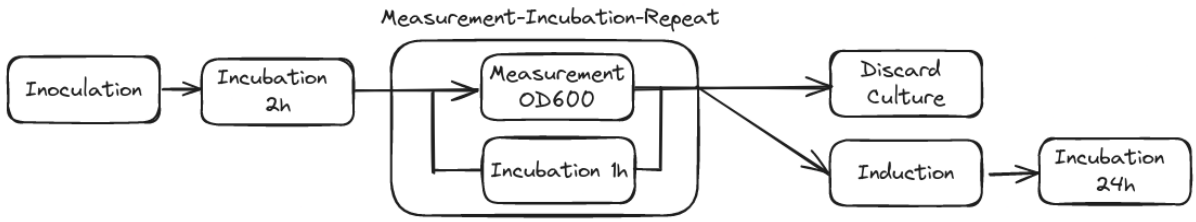


Figure 7-26 – Implemented workflow. The workflow as implemented in BAM. Note the grouping of the repeat tasks (both workflow tasks 'Measurement OD600' and 'Incubation 1h').

The screenshot presented in Figure 7-27 shows the 'Generic workflow' in Airflow's graphical interface (in the Graph view). The repeat loop of the tasks 'measurement OD600' and 'incubation 1h' have been grouped in one AF DAG task called: 'Measurement-Incubation-Repeat'.

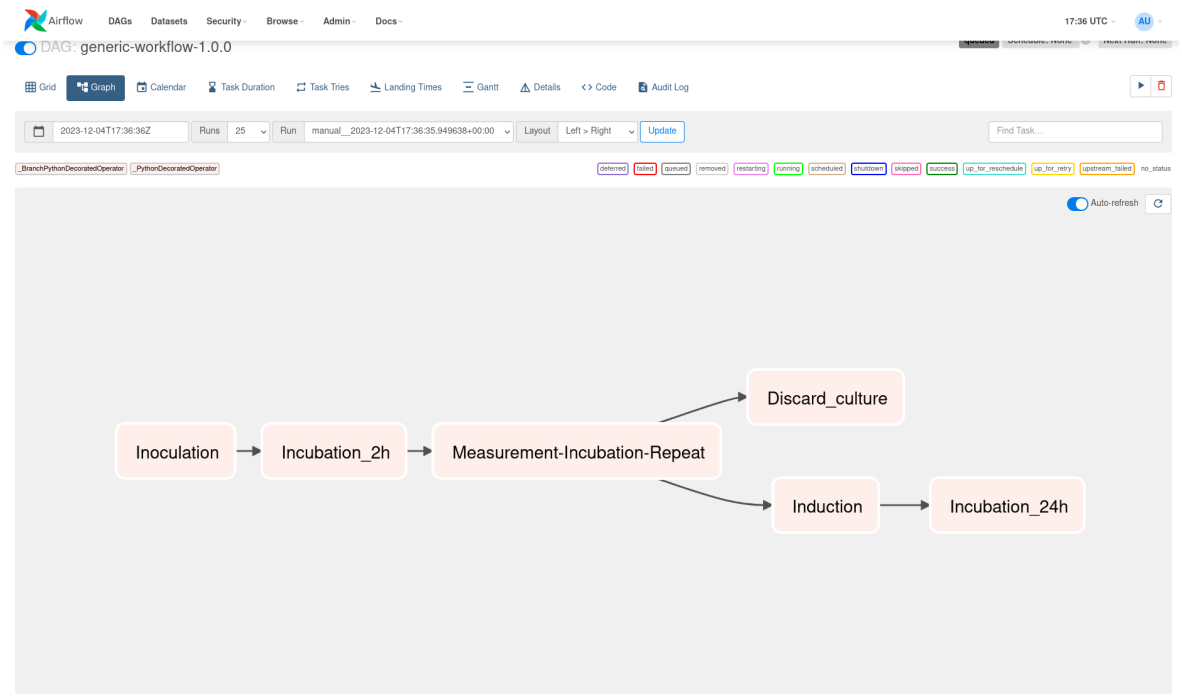


Figure 7-27 – 'Generic workflow' as represented in Airflow's graphical interface. Screenshot of the user interface of Airflow.

4.1.3 Workflow execution (AF DAG)

The workflow, implemented as an AF DAG, is being executed by the orchestrator. Figure 7-28 shows a series of screenshots of the Airflow DAG interface. The successive screenshots represent various stages of the execution of the workflow.

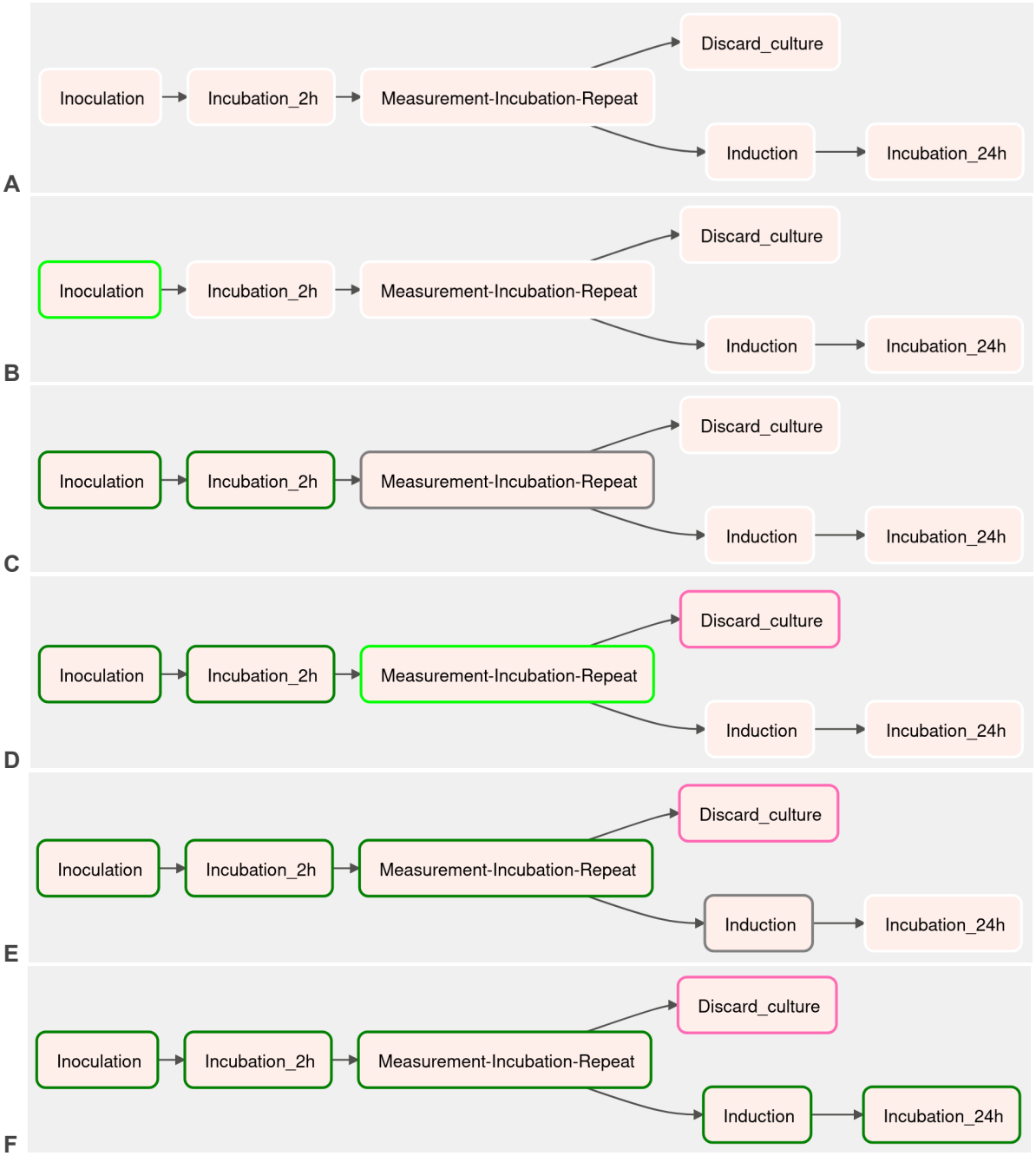


Figure 7-28 – Execution run of the 'Generic workflow' (screenshots from of the user interface of Airflow). The screenshots shows the progress of the workflow at various states of the execution: **(A)** 'Generic workflow' before execution. **(B)** Current execution of the task 'Incubation' (in light green). **(C)** Tasks 'Induction' and 'Incubation_2h' have been successfully executed. **(D)** Current execution of the task 'Measurement-Incubation-Repeat' (in light green), the task 'Discard_culture' has been skipped (in pink). **(E)** Task 'Induction' is being queued (in grey). **(F)** Tasks 'Induction' and 'Incubation_24h' have been successfully executed (in dark green).

4.1.4 Execution graph of the ‘Generic workflow’

The ‘Generic workflow’ has two possible execution graphs topologies (when not paying attention to the repeat nodes). The workflow has two possible branches forking after the repeat: branch 1 ‘Discarding the culture’ and branch 2 ‘Induction and 24h incubation’.

Figure 7-29 shows the execution graph of the ‘Generic workflow’ displayed in the Neo4j browser graph view. The execution graphs represented come from two different runs. Figure 7-29A shows an execution graph with one repeat and the execution of the branch where the culture is being induced and incubated for 24h. Figure 7-29B shows an execution graph with one repeat and the branch that discards the culture.

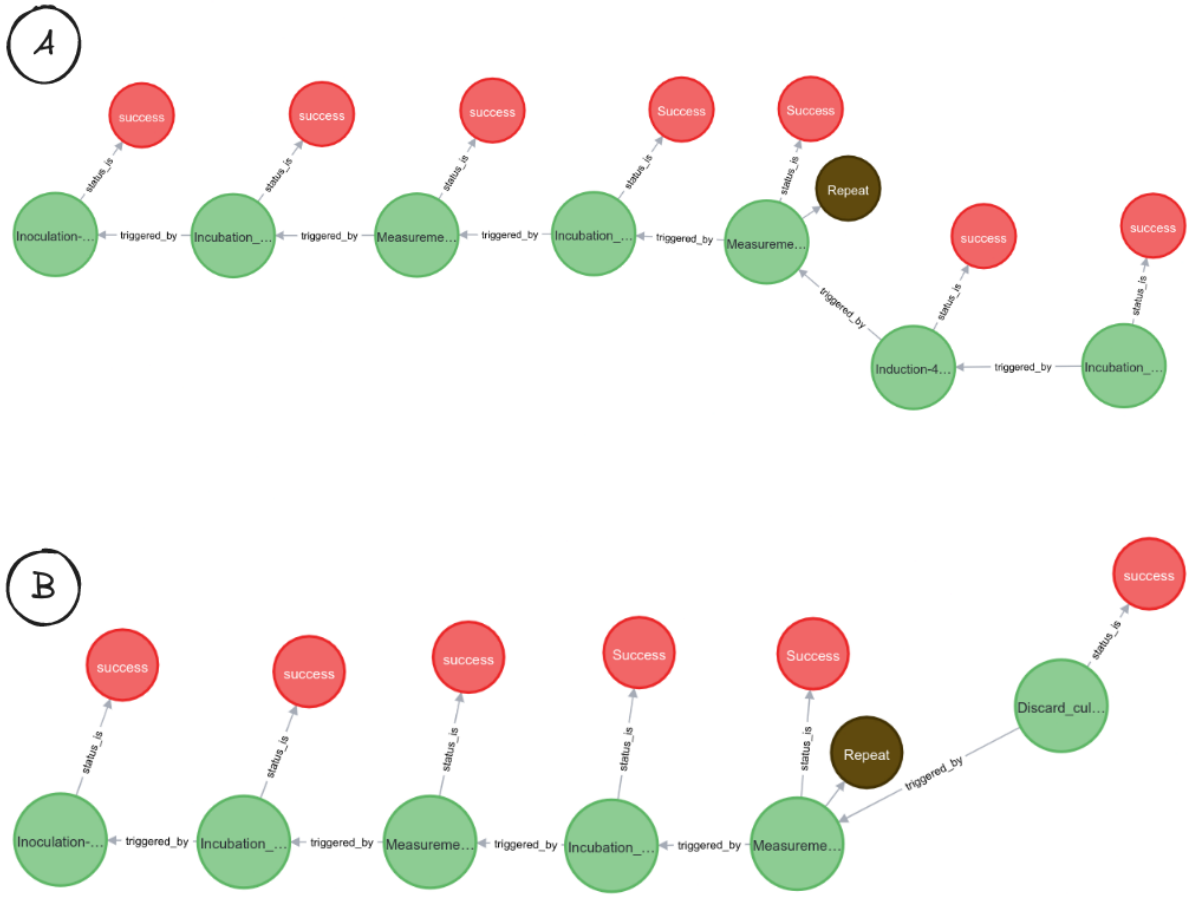


Figure 7-29 – ‘Generic workflow’ as represented in Airflow’s graphical interface (screenshot of Neo4j Browser user interface). **(A)** The execution of the workflow repeats a measurement and proceeds to the induction of the culture. **(B)** The execution of the workflow repeats a measurement and discards the culture.

The execution graph of a run of the ‘Generic workflow’ can be queried from the Graph database (Neo4j) and can be displayed visually as a graph (in Neo4j Browser). Figure 7-30

shows the screenshot of the Neo4j Browser displaying the execution graph of a run queried with a query (in the Cypher language) to the Biofoundry Automation Manager (BAM) database (Neo4j). The whole execution graph of the run can be queried given it's run ID with a Cypher query such as:

```
//Execution graph from Run.id
MATCH (tr:TaskRun)-[:is_instance_of]→(:Run{id:'manual__2024-01-
18T10:24:23.455087+00:00'}) RETURN tr
```

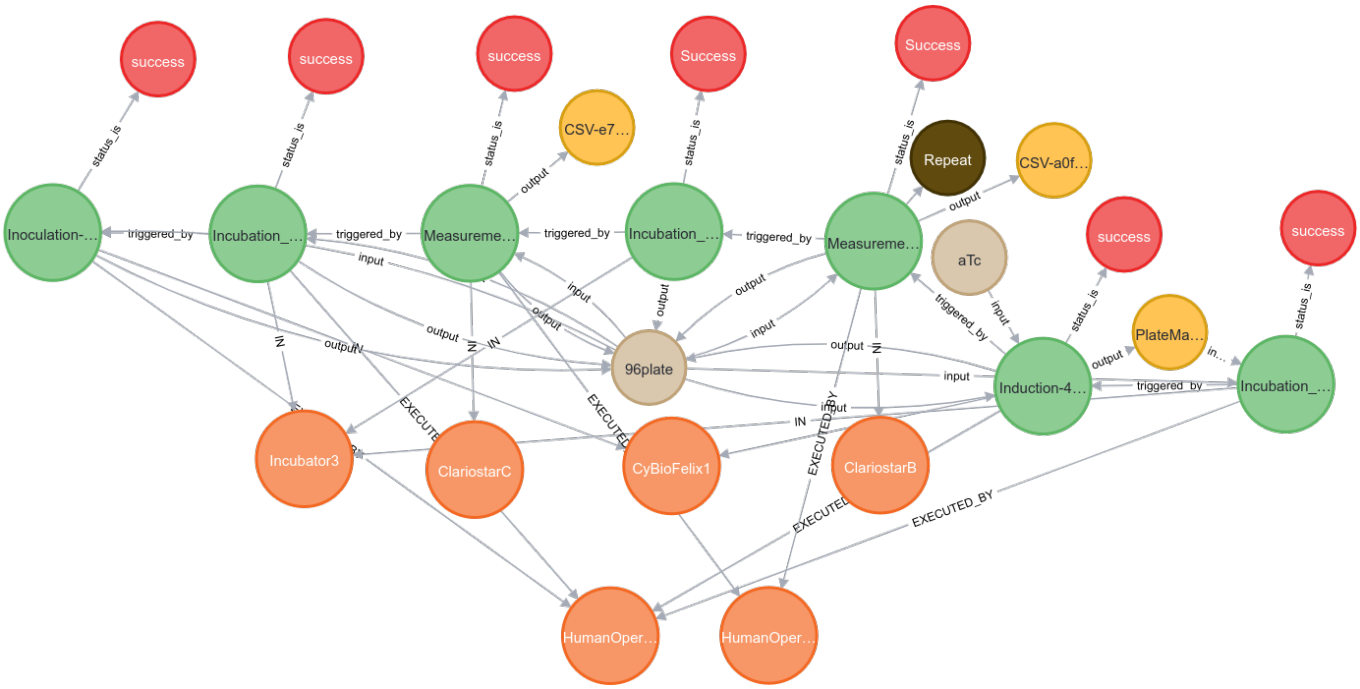


Figure 7-30 – Execution graph of the 'Generic workflow'. Screenshot of the Neo4j browser graph view. The graph view displays the executed Tasks (in green), the status of the task (in red), the resources (in orange), if a task is a repeat (in brown), the output files (in yellow) and the containers (labware) (in beige) of the Execution graph resulting from the run.

The execution graph contains different nodes representing different objects in the graph database such as the Tasks, the status of the task, the input/output (I/O) files. The CSV files produced by the measurement instrument (Clariostar) which contain the OD600 values can be seen in the execution graph in Figure 7-30 displayed as yellow nodes. The resources such as equipment (Incubator, Clariostar, Felix pipetting robot) or human operators can be seen in Figure 7.28 displayed as orange nodes. The containers such as the 96-well plate or the tube containing the anhydrotetracycline (aTc) for the induction of the culture are

displayed as beige nodes.

The complexity of the workflow and the number of objects rapidly reaches an overwhelming visual complexity when displayed as a graph. It is possible to reduce the visual complexity of the graph by displaying only certain types of objects (nodes).

Figure 7-31 shows the screenshot of the execution graph displayed in the Neo4j browser and only displaying the Tasks (green nodes), their status (red nodes), repeats (brown node) and the I/O object such as files(yellow nodes) and containers (beige node). Figure 7-32 goes a step further and displays only the Tasks and their I/O objects of the execution graph.

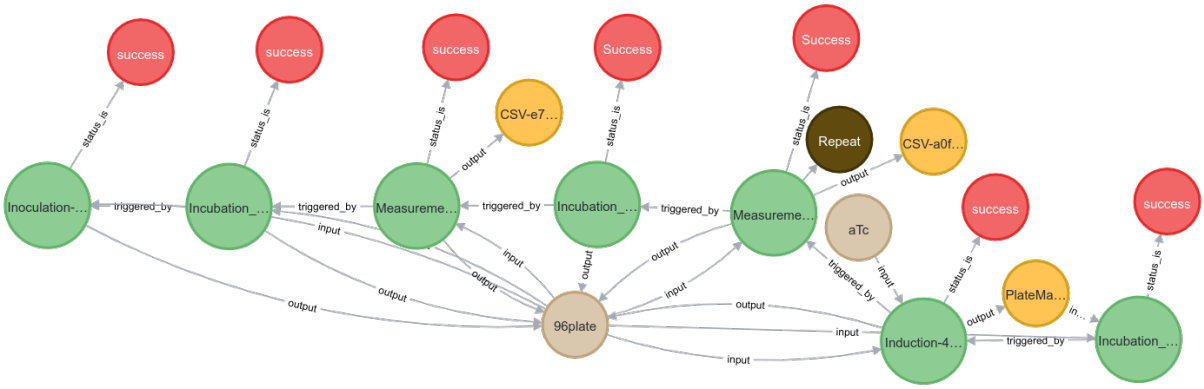


Figure 7-31 – Execution graph of the 'Generic workflow'. Screenshot of the Neo4j browser. The view displays the tasks (in green) and their status (in red and brown) as well as the I/O objects (in yellow and beige) of the execution graph of the run.

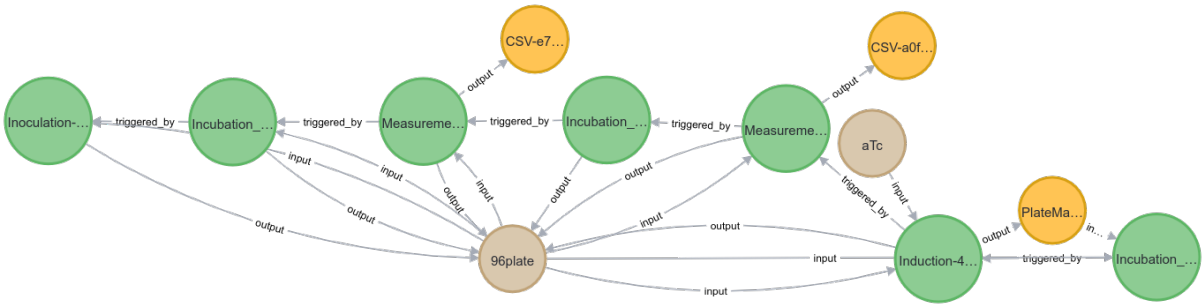


Figure 7-32 – Execution graph of the 'Generic workflow'. Screenshot of the Neo4j browser. The view displays only the tasks (in green) the I/O objects (in yellow and beige) of the execution graph of the run.

4.1.5 Queries of the execution graph

Querying and displaying the entire execution graph of a run can be visually complex even for

the simple ‘Generic workflow’ and that the visual complexity can be reduced by selectively displaying certain types of nodes of the graph.

Because the Biofoundry Automation Manager (BAM) organises information in a graph database, it is not necessary to query the entire execution graph of a run but it is possible to selectively retrieve relevant information. the BAM enables the creation of database queries that fetch the information relevant to different stakeholders in the Biofoundry.
The ability to query the BAM allows different stakeholders to ask different questions. For example the researchers and data scientists can query information about the runs of the experiments whereas personnel of the biofoundry can aggregate information cross experiments (always ask the same question) and observe the usage of the laboratory equipment and labware.

We present here the example of two different use-cases for queries of the execution graph: an example of longitudinal studies which consist of observing some variable over multiple runs of the same experiment and an example of a query cross-experiments.

Example of a query for longitudinal studies:

Longitudinal studies consist of repeated observations of the same variables over multiple runs. The goal of the study for this example is to analyse the cause of a variation in the growth of the bacterial culture in an experiment (‘Generic workflow’).

A typical study consists of observing a parameter (e.g. the ori of a plasmid which could be high-copy or low-copy) which would cause burden to the growth of a culture which in turn would cause the ‘Measurement Task’ to discard the culture according to the value of the Measurement (e.g. discard the culture if the measurement is above a certain threshold).

‘Which runs have cultures that have been discarded after the OD_{600} measurement ?’

The result of this question would return a list of runs which execution graphs could further be queried by the data-scientist conducting the analysis to identify a parameter impacting the growth of the culture.

The above question formulated as a query in Cypher:

```
// All runs that were discarded after Measurement
MATCH (r:Run)<-[:is_instance_of]-(:TaskRun)-[:is_in]-
>(:Task{id:'Discard_culture'}) RETURN r
```

The query returns a list of the run IDs (in a textual representation, run IDs are highlighted in red):

"r"
{"parent":"generic-workflow-1.2.0","created_at":"2024-01-19T18:01:03.775204","id":"manual__2024-01-19T18:00:59.884118+00:00"}
{"parent":"generic-workflow-1.1.0","created_at":"2024-01-19T02:19:33.588428","id":"manual__2024-01-19T02:19:28.925519+00:00"}
{"parent":"generic-workflow-1.1.0","created_at":"2024-01-19T02:19:06.599775","id":"manual__2024-01-19T02:19:02.192715+00:00"}
{"parent":"generic-workflow-1.1.0","created_at":"2024-01-19T00:22:55.567534","id":"manual__2024-01-19T00:22:50.255207+00:00"}
{"parent":"generic-workflow-1.1.0","created_at":"2024-01-19T00:22:30.425148","id":"manual__2024-01-19T00:22:25.916828+00:00"}
…

The query of the database returns data which can be visualised in different view representations such as graph nodes or JSON objects. Figure 7.31-A shows screenshots of the result of the above query as displayed in the Neo4j browser. The screenshots in the context of the thesis are difficult to read (this is not the case on a computer screen where the navigation in the viewer allows zooming in and out of the displayed image). Two thumbnails have been produced. They contained the same information as the above query result.

Example of a query for vertical studies (cross experiments):
Here, a stakeholder (e.g. biofoundry manager) wants to monitor the usage of their equipment across the entire biofoundry facilities to improve (e.g. to better schedule the access to the equipment, to schedule equipment maintenance tasks, to buy an additional piece of equipment if the former equipment exceeds its maximum capacity usage).

For example, to measure the usage of a plate-reader (to better schedule maintenance and re-calibration procedures) the stakeholder might ask the question:

‘Which runs have made use of the Clariostar-C plate-reader ?’

The same query in Cypher:

```
// Runs that used the Clariostar-C plate-reader
MATCH (r:Run)<-[:is_instance_of]-(:TaskRun)-[:IN]-
>(:Resource{id:'ClariostarC'}) RETURN r
```

The query returns a list of the run IDs (in a textual representation, run IDs are highlighted in red):

"r"
{"parent":"generic-workflow-1.0.0","created_at":"2024-01-18T10:24:27.290742","id":"manual__2024-01-18T10:24:23.455087+00:00"}
{"parent":"generic-workflow-1.0.0","created_at":"2024-01-18T10:24:05.418274","id":"manual__2024-01-18T10:24:01.780827+00:00"}
{"parent":"generic-workflow-1.0.0","created_at":"2024-01-18T10:23:44.136083","id":"manual__2024-01-18T10:23:39.652501+00:00"}
{"parent":"generic-workflow-1.0.0","created_at":"2024-01-18T10:23:23.765253","id":"manual__2024-01-18T10:23:19.381264+00:00"}
…

Similarly to the example above and to Figure 7.32-A, the query of the database returns data which can be visualised as graph nodes or JSON objects. Figure 7.32-B shows screenshots of the result of the query respectively displaying graph nodes of the runs and JSON objects of the candidate runs.

Such cross-experiments information which can be retrieved with the same query over and over again can be incorporated in a Biofoundry dashboard that would monitor the usage of the equipment of the entire Biofoundry. Figure 7.32b below represents a mock-up of what a biofoundry dashboard might look like. The dashboard displays the equipment usage and the workflows that are being run on the infrastructure. On the left side of the dashboard, the ‘Equpiment Usage’ widget displays the various pieces of equipment of the biofoundry and their current usage (‘In Use’ or ‘Available’), which workflow and its run number and which tasks is being run, the equipment widget also displays the queued tasks (queued by the scheduler of the BAM) as well as the history logs of the equipment use. Additionally the equipment widget displays the load of the equipment. The three numbers of the load metric (similarly to the CPU load on a computer) represent the equipment usage in the last hour (normalised), in the last 24h and in the last 7 days. The ‘Current Workflows’ widget on the right side of the dashboard displays the various workflows currently being run in the biofoundry; each of the individual workflow widgets display the workflow graph view (similarly to the graph view in Airflow, (see Figure 7.24)) with colour coded task nodes (grey for done tasks, green for current execution, red for failure). Such a biofoundry dashboard can display the information contained in the Biofoundry Automation Manager (BAM) in real-time (or at regular time intervals) by querying the database.

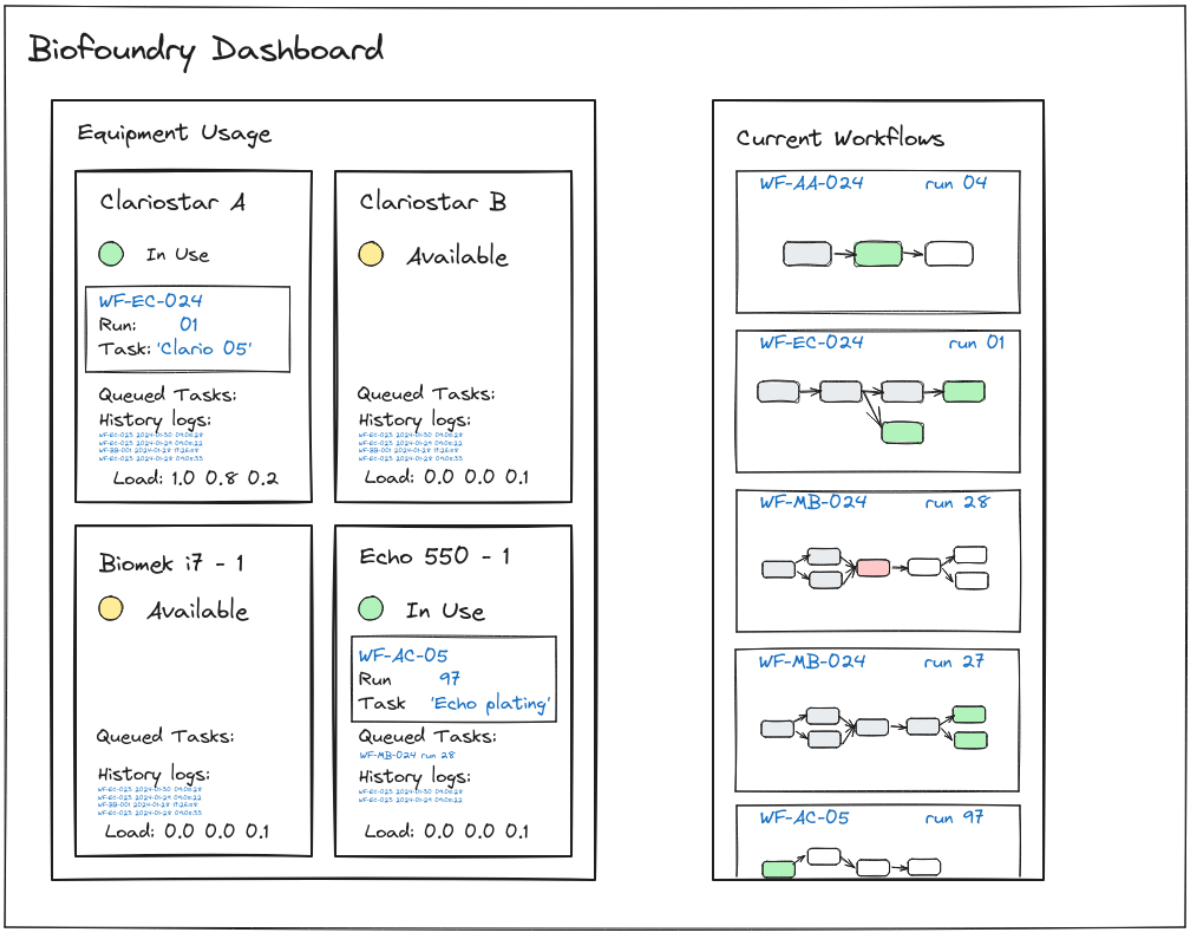


Figure 7-33 – Mock-up of a biofoundry dashboard. Equipment usage is displayed on the left side of the dashboard with the different pieces of equipment and their availability, history of usage and the load. The widget on the right side of the dashboard displays the current workflows being run in the biofoundry (the running tasks are coloured in green on the workflow diagrams, the done tasks in grey and the failed tasks in red).

4.2 Computational example (BASIC DNA assembly)

In this section we present the example of the implementation of a computational workflow for BASIC DNA Assembly (*in silico* DNA assembly).

4.2.1 Workflow description

The 'BASIC DNA Assembly' workflow is a computational workflow which creates an *in silico* BASIC assembly (See Chapter 2 Section 5.4) of DNA constructs. Figure 7-34 illustrates the two computational tasks for the workflow:

- Task 1 'BASIC DNA assembly' takes multiple genbank parts and a construct JSON template and performs a BASIC DNA assembly with basicsynbio (Haines et al., 2022) and returns a Genbank multi-record with the different constructs, the clips list,

linkers list, assemblies list and an Echo cherry-pick list (CSV).

- Task 2 ‘Split Genbank’ takes the Genbank multi-record file as an input and splits it in multiple .seq files

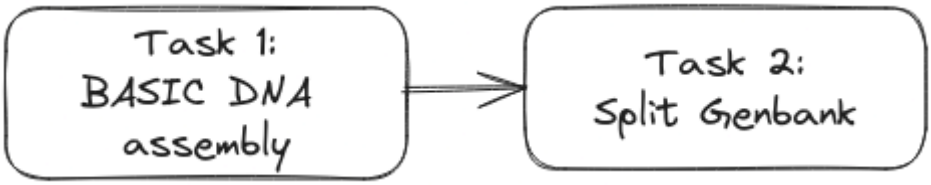


Figure 7-34 – Workflow of BASIC DNA Assembly.

4.2.2 Workflow implementation

Figure 7-35 shows a screenshot of the AF DAG of the workflow displayed in the graph view in Airflow. Additionally, to perform the *in silico* assembly, the two tasks use additional external libraries: the task ‘Task 1: BASIC DNA assembly’ makes use of methods from the basicsynbio python module. The second task of the workflow ‘Split Genbank’ uses Biopython SeqIO objects and parses its input Genbank file multi-record and outputs multiple sequence files (.seq); each file being an individual DNA construct)

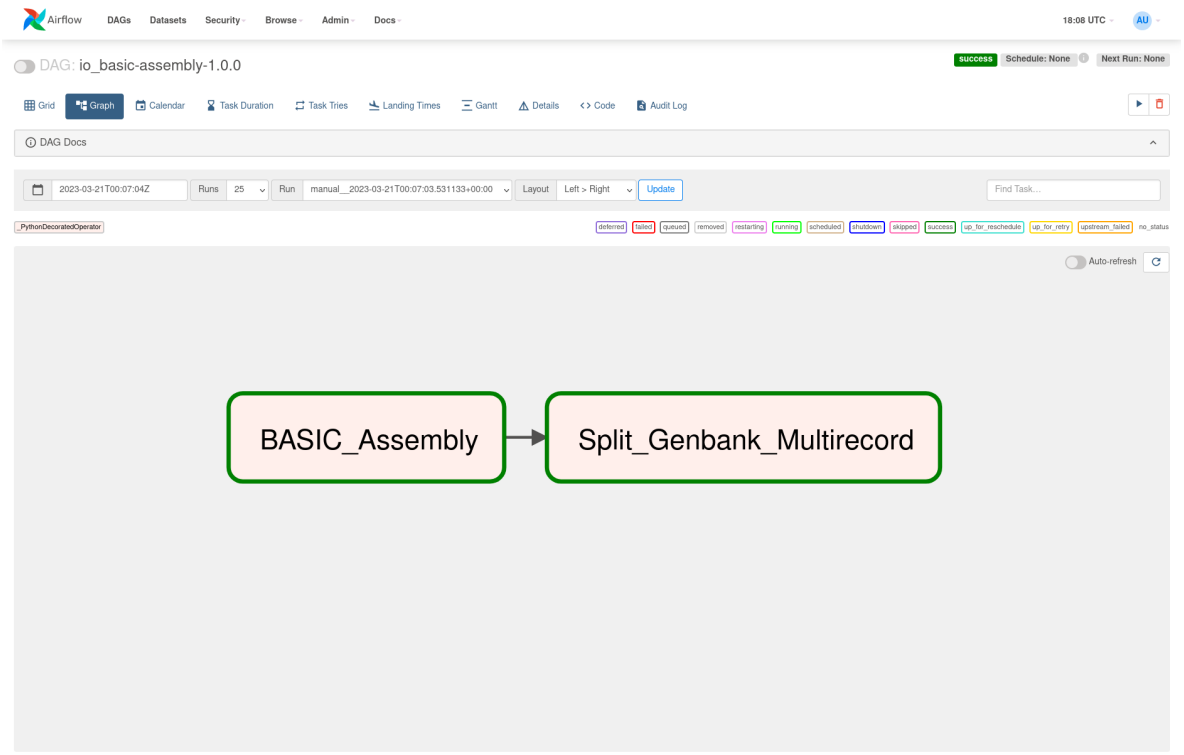


Figure 7-35 – ‘BASIC DNA Assembly’ workflow as represented in Airflow’s graphical interface. Screenshot of the user interface of Airflow with a zoom-in on the graph.

4.2.3 Execution graph of the ‘BASIC DNA assembly’ workflow

Figure 7-36 shows the screenshot of the Neo4j Browser displaying the execution graph of a run queried with a query (in the Cypher language) to the Biofoundry Automation Manager (BAM) database (Neo4j).

The whole execution graph of the run from Figure 7-36 has been queried given it’s run ID with the following Cypher query:

```
//Execution graph from Run.id
MATCH (tr:TaskRun)-[:is_instance_of]→(:Run{id:'manual__2023-03-
08T18:56:51.282491+00:00'}) RETURN tr
```



Figure 7-36 – Execution graph of the ‘BASIC DNA Assembly’ workflow. Screenshot of the Neo4j browser graph view. The graph view displays the executed Tasks (in green), the status of the task (in red), the input and output files (in yellow).

The execution graph from figure 7-36 contains different nodes representing different objects in the graph database such as the Tasks (in green), the status of the task (in red), the input/output (I/O) files (in yellow). It is worth noting that in this example the output of the second task ‘Split Genbank’ outputs 10 different sequence files (.seq). Since the ‘BASIC DNA Assembly’ workflow is a fully computational workflow both the input and output nodes represent digital files that have been read and written on disk.

4.3 Experimental example (automated standard curve)

4.3.1 Workflow description

The ‘Automated Standard Curve’ workflow is a wetlab and computational workflow which creates a standard curve by performing a series of automated dilution on a liquid handling robot (See Chapter 4) of DNA constructs. Figure 7.36 illustrates the overall workflow:

A 12-channel reservoir containing the molecule/dilutant mix at an initial known concentration comes in the workflow and gets diluted on a CyBio Felix liquid handling robot, then the plate containing the dilutions is measured with UV-vis on a Clariostar plate-reader, finally the absorbance measurement output comes out a CSV export from the plate-reader software and is fed into a Python script (Jupyter notebook) that processes the data and outputs a standard curve (values and graphic file).

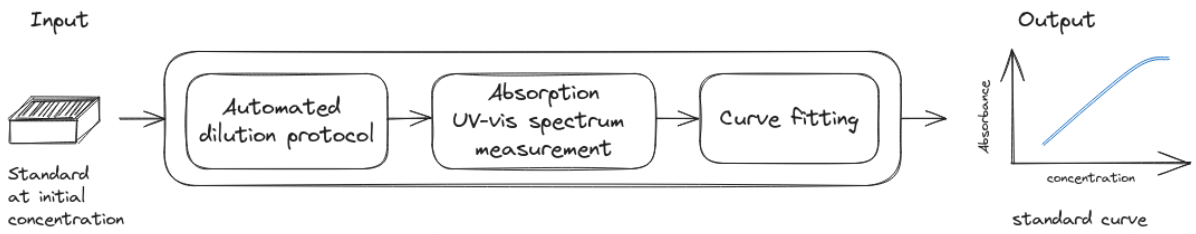


Figure 7.38 – Overall workflow of the Automated Standard Curve.

The workflow is being executed by different actors (human operators): a wetlab scientist that performs the automated dilution and the measurement on the micro-plate reader, both operating in the facilities of the biofoundry, and a second actor a data scientist that receives the measurement file and processes it through a script to produce the final standard curve.

The ‘Automated Standard Curve’ workflow is a semi-automated workflow. Some tasks such as moving the reagents and labware in and out of the equipment (the CyBio Felix liquid handling robot and the Clariostar micro-plate reader) is being done manually.

4.3.2 Workflow implementation

The workflow comprises three tasks shown in Figure 7-39:

- Task 1: ‘Automated dilution protocol’ is executed on the Cybio Felix and takes three inputs: the 12-channel reservoir containing the reagents mix for the standard curve and the dilution protocol for the Felix liquid-handler. A human operator moves the labware in and out of the robot.
- Task 2: ‘Absorption UV-vis spectrum measurement’ is being executed on the Clariostar and takes the 96-well plate with the diluted standard created by the previous task and the Absorption UV-vis spectrum measurement protocol as inputs and outputs a CSV file. The CSV file is written by the Clariostar software. Task 2 in the AF DAG contains a PythonSensor from the Airflow Sensor module which waits for a python callable to return True. Sensors in Airflow are a type of operator that checks if a condition is met at a specific interval. The callable function monitors the directory for a CSV file, once the CSV file is written (either automatically by the Clariostar software when the measurement is finished or when the human operator exports it manually from the Clariostar software), it allows the execution of the workflow to proceed to the downstream task.
- Task 3: ‘Curve fitting’ calls a python script implemented in a Jupyter notebook that takes the CSV file from the measurement and fits a curve to produce the standard curve (the points values in a CSV file and a graphic SVG file).

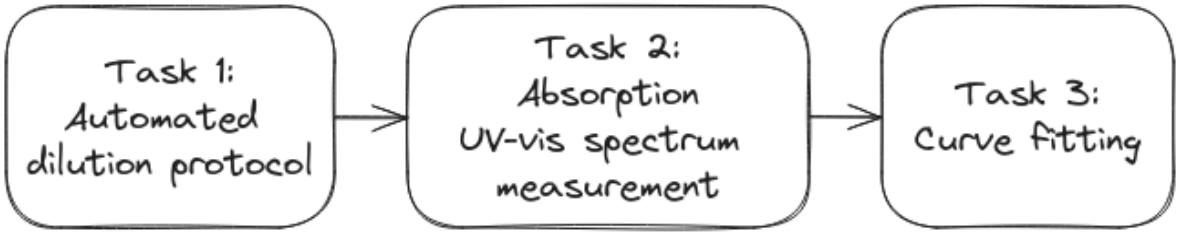


Figure 7-39 – Workflow of the ‘Automated Standard Curve’.

Figure 7-40 shows a screenshot of the AF DAG of the workflow displayed in the graph view in Airflow.

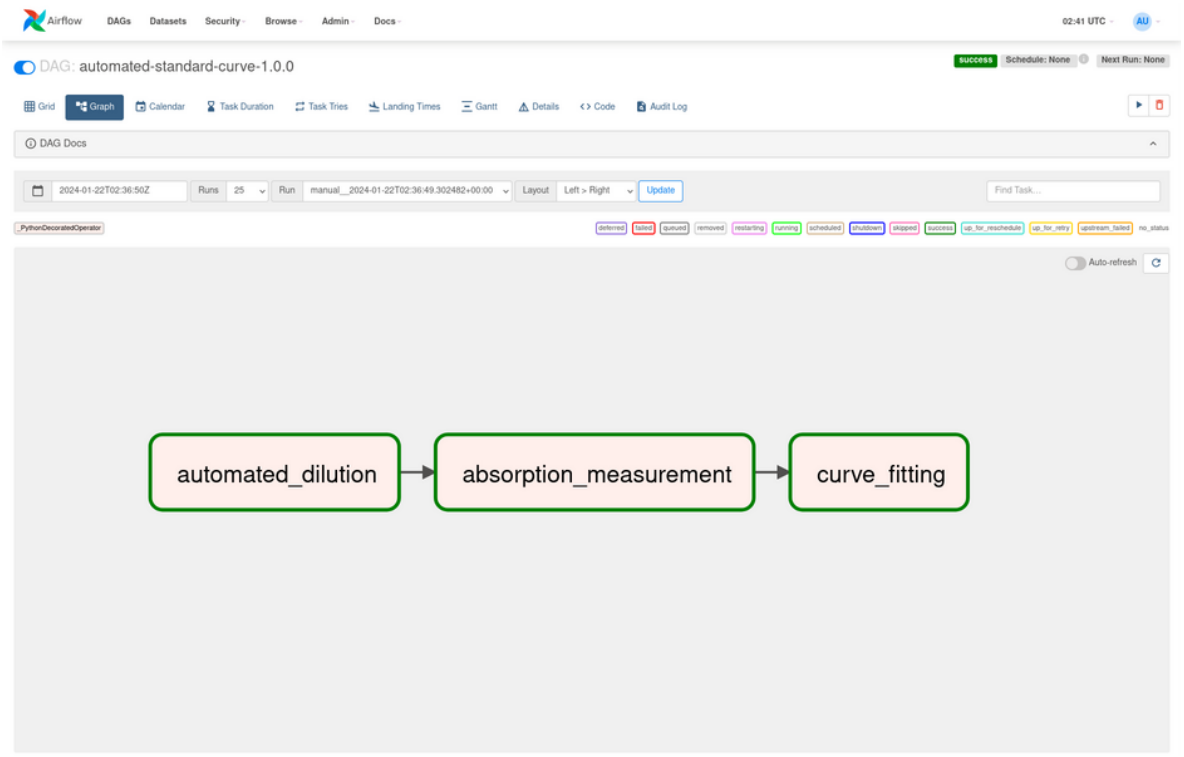


Figure 7-40 – 'Automated Standard Curve' workflow as represented in Airflow's graphical interface. Screenshot of the user interface of Airflow with a zoom-in on the graph.

4.3.3 Workflow execution (AF DAG)

In the lycopene study (presented in Chapter 4) we have done multiple runs of the 'Automated Standard Dilution' with lycopene/DMSO with two different dilution schemes (geometric and linear) which correspond to a parameter for the 'automated dilution protocol' task on the liquid-handling robot.

An analyst would query the system in several steps. First they would query the systems for all runs meeting a certain requirement, then they would import for each run what files they need for their analysis. For instance, they would run the following set of queries:

Step 1: Get all runs that used the 'geometric' dilution scheme

Query 1:

```
//Retrieve all the runs that that used the geometric dilution scheme
MATCH (:Job{id:'automated-standard-curve-1.0.0'})<-[:is_instance_of]-
(r:Run)<-[:is_instance_of]-(:TaskRun)<-[:input]-
(p:Protocol{id:'Felix_dilution'}) WHERE p.dilution_scheme='geometric'
RETURN r
```

Here is an example of a result of Query 1 returned in a text format (with the attribute of interest highlighted in red):

"r"
{"parent":"automated-standard-curve-1.0.0","created_at":"2024-01-22T02:36:53.612726","id":"manual__2024-01-22T02:36:49.302482+00:00", "dilution_scheme":"geometric"}
{"parent":"automated-standard-curve-1.0.0","created_at":"2024-01-22T02:32:21.924206","id":"manual__2024-01-22T02:32:16.314294+00:00","dilution_scheme":"geometric"}
{"parent":"automated-standard-curve-1.0.0","created_at":"2024-01-22T02:30:11.928453","id":"manual__2024-01-22T02:30:06.374138+00:00","dilution_scheme":"geometric"}
{"parent":"automated-standard-curve-1.0.0","created_at":"2024-01-22T02:24:01.191799","id":"manual__2024-01-22T02:23:52.278856+00:00","dilution_scheme":"geometric"}

Step 2: For a given run get all the files (Query 2) or return the CSV measurement file (Query 3)

Query 2:

```
//Retrieve all the output files from Run.id
MATCH (csv_file:File)<-[:output]-(:TaskRun{})-[:is_instance_of]->(:Run{id:'manual__2024-01-22T02:36:49.302482+00:00'}) RETURN csv_file
```

Figure 7-41 shows the results of the above Cypher query displayed in the Neo4j browser. The files are represented as yellow nodes. We can see that the CSV from the measurement file is there.

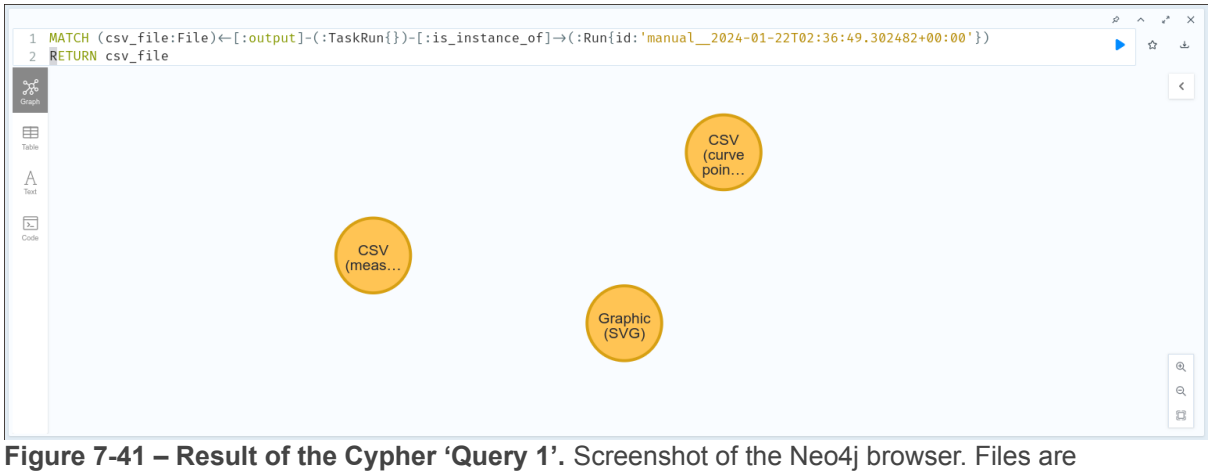


Figure 7-41 – Result of the Cypher 'Query 1'. Screenshot of the Neo4j browser. Files are represented as yellow nodes.

If the data analyst needs to access the CSV file, the location of the file can be deducted from its URI (Universal Resource Identifier) attribute. The text information below is a textual representation of the result of the above query (the file of interest has been highlighted in red):

"csv_file"
{"created_at":"2024-01-22T02:24:15.295860","id":"Graphic (SVG)","uri":"URI://datalake/img/"}
{"created_at":"2024-01-22T02:30:21.465845","id":"CSV (curve points)","uri":"URI://datalake/measurements/csv_out/"}
{"created_at":"2024-01-22T02:30:16.694063","id":"CSV (measurement)","uri":"URI://datalake/measurements/clariostar/"}

If the data analyst wishes to only retrieve the CSV file from the Clariostar measurement the query would be:

Query 3:

```
//Retrieve measurement CSV file from Run.id
MATCH (csv_file:File{id:'CSV (measurement)'})←[:output]-(:TaskRun{})-
[:is_instance_of]→(:Run{id:'manual__2024-01-22T02:36:49.302482+00:00'})
RETURN csv_file
```

The result of the Cypher ‘Query 3’ in text (attribute of interest used by the query highlighted in red):

"csv_file"
{"created_at":"2024-01-22T02:30:16.694063","id":"CSV (measurement)","uri":"URI://datalake/measurements/clariostar/"}

As with the previous examples, it is possible to retrieve and visualise the whole execution graph of a given run. The run from Figure 7-42 has been queried given it’s run ID with the following Cypher query:

```
//Execution graph from Run.id
MATCH (tr:TaskRun)-[:is_instance_of]→(:Run{id:'manual__2024-01-
22T02:36:49.302482+00:00'}) ←(:File{})RETURN tr
```

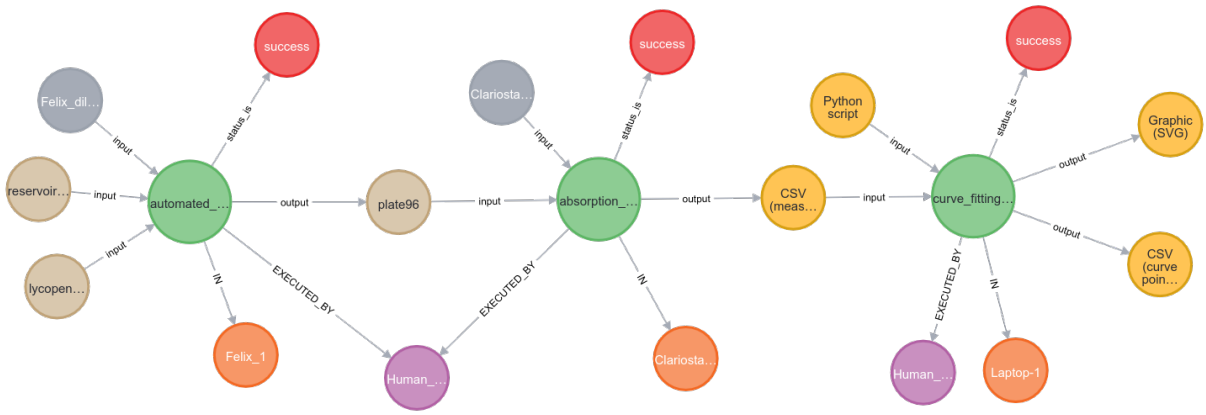


Figure 7-42 – Visualisation of the execution graph of the 'Automated Standard Curve'. Screenshot of the Neo4j browser graph view. The graph view displays the executed Tasks (in green), the status of the task (in red), the input and output objects (files in yellow and labware in beige, protocols in grey, equipment in purple and human resources in orange).

5. Summary

In this chapter we describe how we can use the structure of the graph, and, more particularly, the model graph**,** as a template for automating the execution of the workflow introduced in Chapter 6, We will describe how an automated laboratory can be built when workflows are stored as model graphs (given the right supporting laboratory infrastructure) and coupling the workflowto software called an orchestrator.

Section 1 of the chapter explained an orchestrator and how it allows the management of the execution of the workflows – together with the advantage of the approach.

Section 2 of the chapter introduced our solution to automated workflow – the Biofoundry Automation Manager (BAM). The BAM automatically manages workflows and their execution on a biofoundry's infrastructure. The architecture of the BAM is divided in four main components: the model graph, the orchestrator, the code libraries and a computational/physical infrastructure. An explanation is given as to why the Apache Airflow was chosen as the orchestrator and how it is implemented as a model graph in Airflow.

Section 3, describes the challenges of orchestrating repeats in workflows with the BAM. Different strategies for testing the BAM's functionality and operation are presented, together with an explanation of the choice of strategy for implementing a repeat within a task (Section 3.2 - Strategy 2).

In Section 4 describes the results of worked examples of workflow executions, using the

Biofoundry Automation Manager (BAM). The execution of a generic workflow that contains a loop is demonstrated. In addition, an example of a computational workflow of 'BASIC DNA Assembly' for *in silico* DNA assembly is presented – together with a wetlab workflow for an 'Automated Standard Curve', to automatically dilute a reagent to produce a standard curve.

When implementing automated workflows in a Biofoundry, it is necessary to have a solution that can accommodate any type of experimental workflow -eg automated, semi-automated, manual and both computational, and wetlab workflows. Once a workflow has been defined and structured in a format such as the model graph (as presented in Chapter 6), the execution of a workflow necessitates a task orchestrator. Our solution, the Biofoundry Automation Manager (BAM), accommodates different types of workflows – as illustrated in the user-case examples. The BAM both executes the workflow and records the execution of the runs in an execution graph. The BAM allows different stakeholders to query the information contained in the execution graph**.** The information can be retrieved during multiple runs of the same experiment by researchers involved in the experiment or cross-experiments information can be retried by people managing the infrastructure. The Biofoundry Automation Manager (BAM) is seen as offering an effective solution for managing an automated infrastructure of biofoundries.

Chapter 8

Discussion

In this chapter we include the name of the discussed chapter in the name of the section to enable the reader to relate back to the original chapter.

1. Strategy Overview - Goal and Application

The goal of the work presented in the thesis was to develop a framework and software to support workflow automation in a Biofoundry. The thesis has been presented in two parts. First an exemplar (Lycopene Case Study) an application for the production of a metabolite: lycopene to identify the problems, second, taking lessons from the case study, building an infrastructure framework to support automation in a Biofoundry.

Chapter 2 introduced the concept of metabolic engineering and presented an overview of the different strategies to combinatorial engineering. The problems introduced by large combinatorial pathway approaches were discussed – i.e the search space is very large, a lot of data gets generated and it requires many rounds of experiments. The advent of high-throughput automation and modular DNA assembly techniques have made the combinatorial pathway approach feasible and engineering methods, such as DoE, help reduce the number of experiments to run to converge to an optimal solution in a reduced amount of time. Finally, an engineering workflow to investigate combinatorial metabolic pathways was presented. To illustrate this workflow, the case study of an application for the production of a metabolite: lycopene, was chosen.

The lycopene application presented in Chapter 2 is considered to be a good case study because lycopene has a strong industrial interest. The search space is very large; and the lycopene operon is a model pathway that has a size that is large enough but not too large to explore the combinatorial space. In addition, combinatorial metabolic engineering application generates a lot of data and the application requires many rounds of experiments.

2. PASIV and cMatch - Discussion (Chapter 3)

2.1 Overview

Chapter 3 describes a method called PASIV developed to overcome the encountered problem of initiating the application presented in Chapter 2. Two problems were identified. The first problem of constructs viability that prevented us from initiating a DoE cycle, and second, a problem of construct identification. We undertook a replication study previously conducted in the lab (Exley et al., 2019) and couldn't observe growth for some of the constructs. Consequently, the absence of growth would prevent the measurement of lycopene produced by the cells. From our own observations and from the literature, we concluded that toxicity and burden from lycopene and the construct are responsible for this absence or slowdown of growth. For this reason (and to enable us to bootstrap the DoE cycle) we designed the PASIV method. Instead of drawing random samples (similarly to a classic DoE approach and inferring regions of interest from them) PASIV identifies a viable region of the design space through a viability assay.

Experimental results resulting from the PASIV approach applied to the lycopene operon construct were presented:

- The pooled approach building phase of PASIV resulted in 99 colonies.
- 87 of these constructs were successfully sequenced.
- 73 out of 87 isolates were identified with sufficient reliability (See Chapter 3 - Section 4.2).
- Most of the identified isolates proved to be duplicates and, in the end, only 23 constructs were unique. (See Chapter 3 - Section 4.4)
- The visualisation phase identified a viable region of the design space that enables the intitiation of a DoE cycle. (See Chapter 3 - Section 4.5)

The second problem identified, which arose in the Identification phase (Phase 3) of PASIV, is the need to identify the constructs made by the pooled-approach used in the Pooled Approach (Phase 1) – all the combinations of parts are randomly being assembled in a one-pot reaction, all simultaneously. Consequently, it is impossible to keep track of which construct is which – (such as in a traditional approach of building constructs where we know what we build). Therefore, the PASIV method requires a robust construct identification step. To this end, a novel, automatic software tool, called **cMatch**, was developed. cMatch exploits the structures of the constructs and operates at the functional (component) level. cMatch

utilises the information contained in a modular template of the constructs. cMatch is designed to identify long constructs from fragmented sequencing results (i.e. Sanger sequencing) by utilising split-apply-combine strategy. We demonstrated that cMatch is very efficient by running benchmarks on very large synthetic constructs (See Chapter 3 - Section 6.1.4). cMatch can automatically identify constructs in a matter of seconds. By comparison, a construct identification operation performed quasi-manually by a human is tedious, extremely time consuming and error prone.

2.2 PASIV - discussion

PASIV is based on a multiplex construction phase via a pooled approach in conjunction with a viability assay. This experimental phase is coupled to a software phase, where the isolated colonies are sequenced, their genetic content (construct) is identified, to finally yield an estimated location for one (or several) region(s) of interest. We believe that this form of interplay between biology and software methodology provides a flexible, time-efficient solution. Even at the relatively modest scale of this study, identification of the viable region of the design space would have been a cumbersome, lengthy and unreliable process without these tools and methods.

2.2.1 Extending PASIV to other pathways

Although PASIV has been illustrated with the lycopene case study, it is worth emphasising that the workflow could relatively easily be developed for a broader range of applications. The modular assembly method, BASIC, is also known to perform well for longer pathways – thus extending the range of pathways PASIV can be applied to. The screening phase is also not dependent on the presence of a visual reporter (be it a colourful metabolite, or through the addition of a dedicated sensor (Zhang, Carothers & Keasling, 2012). PASIV is based on a viability assay – viable colonies are to be picked regardless of the visual stimulus they emit. The viability assay is followed by a titration assay, which is specific to the metabolite of interest. For instance, in the lycopene exemplar (see Appendices C3 - Methods), lycopene was first extracted with DMSO (Takehara et al., 2014) from the liquid culture and its concentration was estimated from absorption reading at 471 nm (Takehara et al., 2014). Visual reporters could, in theory, be used to steer towards high producing regions in the picking phase. No attempts to do so were conducted in our work, as too few viable colonies were present on the plates – and authors were concerned with importing possible biases in the screening phase. As for the software phases of the workflow, they are agnostic to the chosen pathway.

2.2.2 The viable region

In the lycopene case study, a homogeneous, contiguous region of space, containing strong performers could be identified. The region, as could be expected for a problem where toxicity plays a major role, corresponds to the weakest constructs. The region was located close to the origin with the minimal set of coordinates (the enzyme synthesis rates) used by PASIV. Although extrapolating from the data yielded with PASIV should be done with appropriate care, there are good reasons to consider the identified region for further investigation.

The identified region had an enviable set of properties. Collected data show a continuous, viable region of space that contains healthy and productive colonies: some constructs yield colonies with high multiplicity, with a clear growth advantage and performing well. In the context of metabolic engineering, these constructs would be good choices as they are, most likely, reliable performers. Conversely constructs that fail the pooled approach, while possibly capable of good production, are more likely to be subject to the toxicity effects and be unreliable performers.

Higher producers were also very heavily clustered inside the identified viable region close to the origin, while lower producers were spread further from the origin. This fact, coupled with our expectations regarding the effect of toxicity and burden on production, and the results of the viability assay (very few viable colonies) convinced us not to look for further regions of interest in the lycopene case. Instead, the region containing the most promising isolates was expanded (as shown in the Chapter 4 Section Results) – as it clearly was the region to favour for the screening phase. In parallel, the larger region containing all isolates was retained for a possible separate screening, as it contained more constructs but had been much more sparsely sampled. In general, not possible to tell with certainty if the whole of the viable region has been identified without several repeats of the workflow – especially for more complex pathways where several viable regions may exist. One or more rounds of PASIV should therefore be run – in order to validate the identified viable region. Figure 8-1 illustrates the process. In Figure 8-1-A, an initial round of PASIV yields a viable region, which is then confirmed by a second round. In Figure 8-1-B, on the contrary, the second run yields an additional region – and a third confirmatory round needs to be run.
A stopping condition can easily be constructed from the standard classification performance metrics (recall and F1, since the scoping round aims to avoid false negatives). When there are several sub-regions, clustering metrics (silhouette score, Davies-Bouldin index, Calinski-Harabasz index) can also be used to quantify the evolution of the clustering results. In parallel, as more data are collected, estimation of the production landscape (mean, confidence interval) can be refined, as can the low-producer/high-producer clustering.

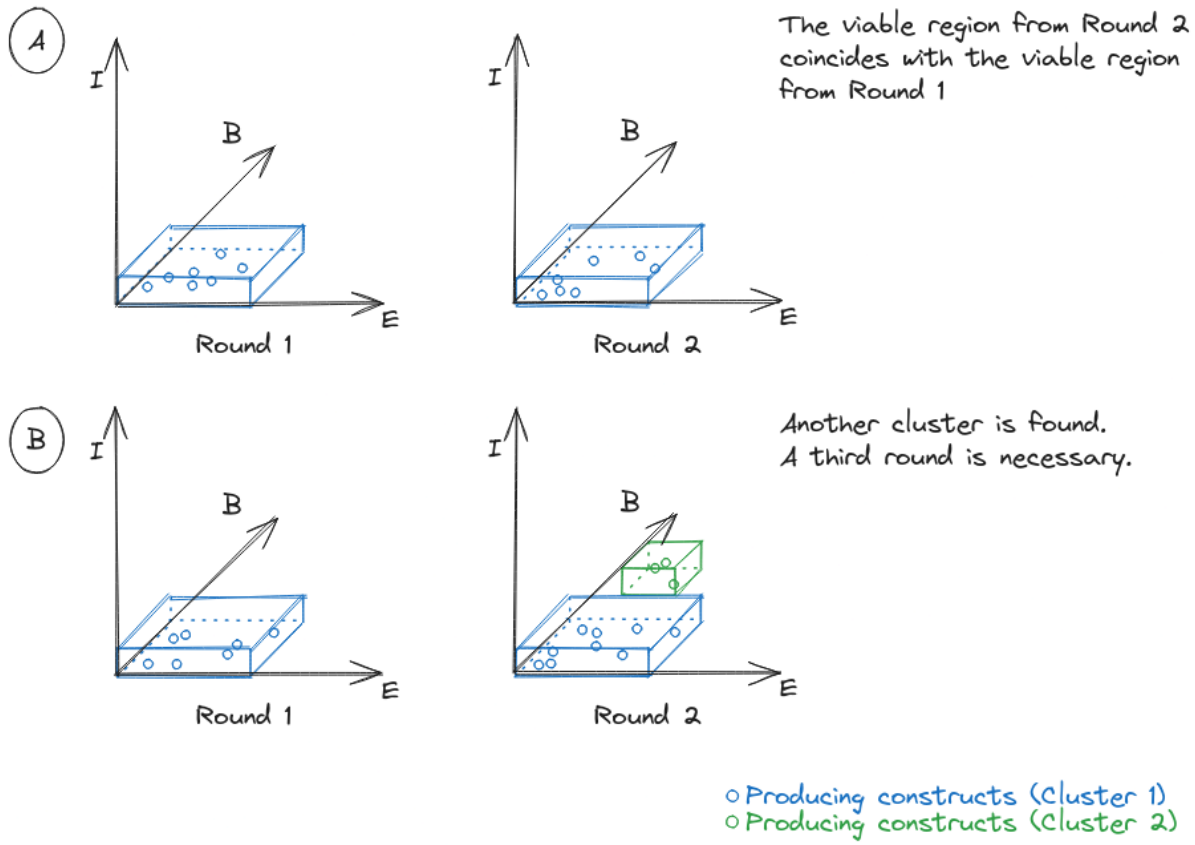


Figure 8-1 – Illustration of repeats of rounds of PASIV. (A) After a first round, a first viable region of producing constructs (Cluster 1 in blue) is identified, after a second round, the same viable region is found, confirming the result of round 1. **(B)** After a first round, a first viable region of producing constructs (Cluster 1 in blue) is identified, after a second round, an additional viable region is found (Cluster 2 in green). A third round is needed to confirm the additional viable region found in round 2.

Even in the case where multiple viable regions exist, the advantages of using PASIV remain. The multiplex construction remains fast and simple as all components are mixed at once in a one pot reaction – with identical experimental conditions. More importantly, each new round of PASIV will yield constructs that accumulate in the viable regions – which remains a better outcome than the outcome of random sampling rounds (with a lot of failures).

2.2.3 Titration assay

The benefits of conducting a proper titration assay in the second phase, and not just a viability assay, extend beyond collecting data that will be of use in the subsequent rounds of the DoE cycle: the collected data can be useful in identifying for the location of the region of interest, as shown in the lycopene example. Clustering identified two clusters (one of the low producers, one of the higher producers) – the stronger producers being located very close to

the origin – and corresponded to the lowest values of E synthesis. Also, yield data hint that it is advantageous to have stronger RBS than RBS3 in front of crtI, and that regions with strong producers should be expanded along the I dimension.

A benefit of using coordinates to define a region of interest is to enable resampling of the region to make it less sparse. This would be of clear benefit to generate constructs even closer to the E-axis. In practice it means identifying more constructs using the specified operon design, but outside the original design space as they use new components. This can be achieved, for instance, by extending the RBS libraries to include all the members of the Biolegio collection and drawing from the constructs that fall in the newly identified region of interest. Likewise, more suitable constructs can be generated by adjusting the promoter in the operon and using weak promoters.

Next, another benefit of operating at region level (rather than individual construct level) and adding margins of error to the location of the region of interest is to be found in the final visualisation step in PASIV. The visualisation phase relies on a minimal set of coordinates related to enzyme synthesis, and for these coordinates to be estimated for each element in the design space. Transcription values were derived from promoter characterization data in a similar context. For translation, it was decided to rely solely on a bioinformatic tool, the RBS calculator, to estimate the translation rates of all the RBS involved in the study and for all possible upstream and downstream contexts. The decision was taken for several practical reasons. First, The RBS calculator is a popular tool that is consistently improved. Second, experimental characterization of the RBS in context (to determine translation rates) was cumbersome:

- First, common reporters such as GFP cannot be used outright since their first 60bp differ from the crtE, crtI and crtB enzymes. Modifying the reporters at their 5’ ends was rejected as too liable to modify the reporters’ properties. Using a fusion protein was also rejected as it was liable to alter too much the metabolic burden (and could lead to folding problems).
- His-tag characterization of protein concentration was also investigated but protein purification considerations, concerns about folding, and concerns about potential over-expression of the recombinant proteins (Park et al., 2015) led to its rejection.

Finally, although there are other methods – most notably targeted proteomics (Borràs & Sabidó, 2017; Sinha & Mann, 2020) – that are well suited to directly assess protein levels, they were rejected for the scoping phase, as these modern capacities are not widely accessible yet, and PASIV relies on growth information first (multiplicity and OD) and production information second. These methods should of course be considered for the subsequent targeted rounds of the DoE cycle.

Although it is tempting to use calculators (i.e. RBS calculator, EFM calculator), when possible, due their simplicity of use, it is important to bear in mind that their use may lead to some features of some of the constructs being estimated with a significant margin of error – making individual construct predictions risky. While this is a potential issue in the screening and optimisation phases (protein measurements should then be conducted), we argue that it is less of a problem in the scoping phase as conducted in PASIV. PASIV's purpose is the identification of a region that is worthy of further investigation. PASIV is not so much concerned with minimising false positives (including constructs that should not be) than maximising true positives (making sure the region covers the right area of the design space). This is reflected in the simple manner the region of interest is reconstructed from the data (clustering the identified isolates into boxes with added margins for errors).

2.2.4 Conclusion

The PASIV approach presented in Chapter 3 offers an effective solution to a particular issue in DoE cycles: what can be done if the scoping phase is severely hampered by burden and toxicity issues. It is also a showcase for a systematic approach to synthetic biology based on a few principles:

- As far as possible, the biology should be allowed to behave naturally – the first two phases of PASIV harness biology, so the viable region reveals itself.
- Reproducibility, reliability and robustness are key. In PASIV, they featured in the design of the workflow – to account for possible failure modes – as well as the conservative approach adopted to identify the viable region.
- Automation and software are to be deployed as much as possible to operate at scale and make it possible to reliably collect and process large amounts of data. PASIV would not work without its software identification phase (Phase 3).
- Synthetic biology should move further into the direction of data science – and the biology be abstracted as much as possible. The visualisation phase (Phase 4) is but a small example of such an application.

As a consequence of these principles, several computational tools, as well as a set of problem-specific experimental methods, have been developed. Although the most obvious application area for these tools and methods is metabolic engineering and large combinatorial optimisation problems, we are confident that they can be applied to a wider range of problems.

2.3 cMatch - discussion

In Chapter 3 Section 5, we have presented a novel, automatic tool, cMatch, to reconstruct

and identify a synthetic genetic construct from its sequence, or more practically, a set of sub-sequences. The method is based on two pieces of information: knowledge of the genetic construct modular structure, and its libraries of components. Although first intended and developed for combinatorial pathway engineering problems and lifting their QC bottleneck, cMatch is not restricted to these applications. The algorithms can indeed be applied to any synthetic, modular construct, provided a template and a list of elements for every position of the template are available.

2.3.1 Applications of cMatch

cMatch has been designed to perform an operation that we have called ‘construct-matching', and which is more thorough than simple sequence-matching or alignment. Construct-matching is concerned with matching sequence data to constructs at the functional level, practically searching for its components, checking if their order matches a given template, and then quantifying for each component how they differ from expected components. Working at component level is important in order to deal with small regulatory components, and eliminate what biases matching methods have towards large sequences. For each component a matching (homology) score is derived. These scores and their weighted average are then used to quantify the reliability of the predictions.

Construct-matching also goes further than automated annotation and sequence alignment, as these matching scores are used to make decisions regarding the matching (reconstruction does not proceed unless all components are identified with sufficient precision, and fails if some components are missing or no component combination can be found that matches the template), and to rank the possible combinations.

To our knowledge no software has been designed to exploit the advantages of modularity in construct design to perform such a task – despite the centrality of the concept in synthetic biology, and the level of friction quality control and similar forms of verification introduce into project workflows when performed at large scale, and the obvious reliability and reproducibility issues associated with these tasks.

cMatch has been applied to the problem of identification of synthetic constructs lying in a specified design space. A typical example of such need is encountered when pooled construction workflows are used. These workflows yield little prior information about the genetic material – except that it matches a given template, was assembled from a known set of components and therefore lies in a, possibly large, space of possibilities. This is insufficient in general, and the link between colonies and their genetic material must be re-established, so phenotypic data may be correlated with genotypic data.

As previously stated, cMatch was first intended and developed for combinatorial pathway

engineering problems. These applications are subject to a particular set of constraints that require the development of precise, reliable, automated quality control for their constructs – namely the need to deal with the large amount of data generated by these applications and the need for precision in the identification of the small regulatory elements. cMatch is, however, not restricted to these applications. The software can be employed for quality control (QC) for any synthetic, modular construct, provided a template and a list of elements for every position of the template are available.

Because of the way cMatch exploits construct modularity, it is also capable of performing two forms of quality control of opposite complexity (called minimal and maximal QC in this work):

- Minimal QC: This is the lowest and most common form of quality control. It aims to check that the sequencing data correspond to a given construct, or to what degree they differ.
- Maximal QC: Sequencing data are compared to all the constructs in the entire design space at once. This is a more thorough form of QC. It offers some insurance against practical errors, such as mislabelling. It also offers estimates of how more likely the best match is compared to the best other possibilities. This is a useful feature when data cannot be assumed to be completely reliable, or to deal with the shortest components, for which changes of a few base-pairs (due to mutations or sequencing errors) may lead to the identification of different library elements.

2.3.2 Reproducibility, reliability and robustness

Reproducibility, reliability and robustness are at the core of the design of cMatch and its algorithms. Automating quality control (regardless of the way it is accomplished, and what sub-tasks it performs) prevents analysis from being affected by considerations affecting human operatives such as fatigue, waning focus and general human inconsistency. Reproducibility was improved by making the algorithms deterministic – with our choice of the Smith-Waterman algorithm as the foundation for cMatch to build on. Only computation times vary for a given input, and search parameters (template, libraries), and do so according to available computing resources – benchmark (Chapter 3- Section 6.1) with the violacein example show limited variation in that regard. Synthetic results from the violacein example show that the algorithms correctly identify the ground truth and return a score of 1.0 for all its components. The algorithms are therefore considered to be reliable.

Robustness, defined as ‘the quality of being strong, and healthy or unlikely to break or fail’ (Cambridge Dictionary, 2017) is, however, more than just reliability and reproducibility. It extends to other parts of the construct-matching process – the most important being decision-making in the face of uncertainty. Unlike the tests with synthetic data, it is

impossible in practice to be sure that an input sequence is itself reliable and can be fully trusted. Errors can be either due to sequencing or be indicative of a genuine alteration of the genetic content caused by a few mutations, some more likely to alter construct output than others, or at the other end of the spectrum proof of assembly failure).

cMatch's algorithms offer several mechanisms to deal with uncertainties in input data. The first mechanism helps deal with poor data. Users can lower the constraints for a successful analysis – practically, the constraints on component-matching (matching threshold) and reconstruction (admissible overlap). Benchmarking results show the cost of relaxing these constraints to be minimal thanks to the efficient step-by-step reconstruction.

The second mechanism is through quantification. Any non-perfect matching score can be used as a warning indicator warranting further investigation of the construct or some of its components (a level of analysis unobtainable without a component-based software). Further investigation can take several forms. It can be a simple repetition of the sequencing process (in case the best match is close to the input sequence and it is legitimate to suspect a sequencing error because discrepancies are located in regions with low phred score) or a more drastic decision to repeat the construction. In either case, construct-matching can be repeated on new sequence data and the best prediction, as measured by the overall matching score (or any user-defined function on the component scores), can be retained as the most reliable output. Previous estimates of construct properties may then require adjustment – for instance a new estimate of the translation rate can be calculated if mutations are found in a UTR region and it is concluded the construct has mutated.

Quantification allows the estimation of the distance between the best match and all the other constructs in the same design space – thus offering a method to rate how likely the best match is compared to these other possibilities. This is the idea behind what we call maximal QC. Depending on the choice of elements in libraries, some search spaces will yield more conclusive identifications than others – since their constructs are spread further from each other. This should come as no surprise, since manual sequence alignment is notoriously harder and error-prone when libraries contain elements close to each other, for instance differing by a few base pairs after they were generated by random-PCR.

2.3.3 Practical considerations

cMatch's algorithms have been developed to be fast enough for common applications, but also so they do not require special hardware to run on. The benchmarks (in Chapter 3 - Section 6.1) show, analysing a sequence for a typical problem (constructs with a few genes, search spaces made of a few thousand constructs) only takes a few minutes. Processing a batch of sequences (a few hundreds for QC), will in no way represent an obstacle for a

project. Algorithm inputs have also been chosen to be as intuitive as possible, consequently, little data conversion must be undertaken before analysis is run. Input sequences are in the .seq format typical of Sanger sequencing, or simple txt format. The search parameters are accessible to the user and encoded in the human-readable format JSON. The corresponding file has a simple structure that can be easily modified so data analysis may start soon after the sequences for the components are gathered. Finally, the common txt, Genbank, and SBOL formats are supported for the sequences of the elements in the component libraries.

The most important consideration for the choice of the construct-matching algorithms relates to the nature of the input – whether a single or multiple sequences are used. This choice is in general, a direct consequence of the sequencing method. CM_2 is the default algorithm for multiple inputs. It is worth noting that because sequences used for inputs will be short, some limitations of the Smith-Waterman algorithm will have limited impact on performance. CM_1 remains the algorithm to use when a single sequence is used as input, as would be the case with NGS sequencing.

An important feature of both CM_1 and CM_2 is that they were built on top of the Smith-Waterman algorithm for pairwise matching. This algorithm is extremely reliable, but its performance suffers with long sequences. When possible, shorter component sequences should be used – for instance with CDS or hybrid components. Benchmark tests have shown that the performance of CM_1 degrades with input sequences longer than 10kb – so would CM_2 if such a long sequence featured in its inputs. Splitting strategies are also being investigated to reduce the size of the inputs and speed up computations at the component-matching step of the algorithms.

Both CM_1 and CM_2 rely on the availability of two specific pieces of information.

They first rely on the availability of the construct template. Algorithms for the purpose of sequence investigation (and template identification) are outside the scope of this work – construct investigation being of little immediate use to the Kitney Lab in its everyday workflows. They offer useful capacities however, for instance as part of an audit of resources in freezer storage. Variants of CM_1 and CM_2, that modify the reconstruction phase so all combinations are pruned not against a set template but under a set of positional and grammatical rules, will be investigated in our future work.

They also need a full list of the elements in the component libraries, since they are based on pairwise-matching of known sequences. In case of missing elements, a possible strategy is to use a very large library of components from an external database, lower the pruning thresholds for these components, and match the large libraries against the target sequence. The best matches can then be used to locate the missing elements, providing they have

enough in common. Because of the high degree of parallelism in the algorithm, we are confident that analysis of a sequence will remain tractable. For longer components, the faster (but less reliable) BLAST search may be recommended.

2.3.4 Conclusion

cMatch is a lightweight tool to perform quality control of modular synthetic constructs at speed and scale. Although originally developed for the application case of combinatorial pathway engineering and making QC frictionless in optimisation workflows, cMatch can be used in many different settings. The use of cMatch has made a significant difference to the operation of the lab – and given a significant boost to our productivity and to the confidence in the reproducibility of the results. We are confident other adopters of the tool will enjoy similar benefits.

3. Development of Automated Standard – Discussion (Chapter 4)

3.1 Overview

In Chapter 4 we presented the need for a calibration curve. The lycopene application (see Chapter 2) requires the measurement of the concentration of the lycopene produced by the cell expressing the genetic constructs. In the application pipeline, lycopene is extracted from the cells with DMSO. An absorbance measurement of lycopene in DMSO is carried in a plate-reader and the lycopene concentration value is derived from a lycopene standard curve. The construction of a standard curve requires the controlled generation of a set of samples of known concentrations on measurement microplates – and for the process to be repeated over a series of assays – which is achieved by using automated liquid handling. We developed an automated protocol to create a standard curve because automation offers many advantages: automation allows to implement any arbitrary dilution pattern over a plate in a programmatic way, automation reproduces the same mechanised liquid transfer movements when repeating the protocol, thus ensuring reproducibility of the curve, automation removes mechanised human induced errors.

Automatic liquid handling and the challenges that come with the automation of liquid handling were introduced at this stage. We presented many issues that had been identified in the literature as obstacles encountered during automated liquid handling operations and result in errors. These liquid-handling errors have been categorised (Albert, 2007) and their

consequences need to be taken into consideration by protocol designers.

In Chapter 4 we presented the challenges of automating liquid handling of a lycopene and DMSO mix, which qualifies as an atypical liquid known to be difficult to work with from a mechanised point of view. Additionally, the combination of lycopene and DMSO has physio-chemical properties that affects concentration and absorption measurement.Two dilution schemes were introduced and were subsequently implemented and studied. The geometric scheme and the linear scheme.The different sources of errors that might be induced by one scheme or another were described.

The study was presented. We developed an automation method to create a standard curve. The study was conducted in two parallel tracks, one for each dilution scheme (the geometric dilution scheme and the linear dilution scheme) and each track included a preliminary study (Phase 1) followed by a study of lycopene and DMSO mix (Phase 2) and a final study that produces a lycopene standard curve (Phase 3).

After conducting the preliminary study (Phase 1) of the geometric dilution scheme we concluded that iterative dilutions introduce a class of correlated errors. Additionally, a geometric dilution of lycopene and DMSO introduced even larger errors, consequently we dropped the geometric dilution scheme and pursued the second track with the linear dilution scheme.

A workflow to assess dilution errors based on a regression modelling was presented. We then ran many experiments to tune the automated linear dilution protocol and to test the error assessment workflow. Additionally, a study of the spectrum of the lycopene and DMSO mix was conducted to ensure that the mix is stable – to make comparison between different runs possible. A degradation study was then run over many hours (12h). Surprisingly, the age of the mix had a very strong effect on the spectrum. We concluded that all assays would then use fresh lycopene and DMSO preparations of less than an hour old to remove all ageing effects.

In a third phase we conducted a series of runs to investigate outliers to understand if outliers could be explained by instrument issues or by a temporary phenomena that would disappear (or not) over some time. We concluded that inhomogeneities appeared in the mix (especially at higher concentrations) and that some particles moving were interfering with the absorbance measurements. We compensated for sample fluctuations by aggregating repeating measurements. Finally (repeat) measurement data from low and high concentration dilutions was aggregated and a final regression with all the combined data to produce a lycopene and DMSO standard curve. Additionally, the conducted studies provided learnings for practical recommendations regarding automation of liquid-handling.

3.2 Discussion

The use of automation to construct a standard curve of lycopene in DMSO is a good example of a deceptively simple problem, but is, in fact, more complex. No biological variation was involved in the study. Measurements were conducted on well-calibrated devices. Industry-grade liquid-handling platforms were used. Assays on simple mixes such as food colourant with water or DMSO, yielded the intended outcomes (smooth curves, whose intensities correlated very strongly with the desired dilution ratios) – thus validating the implementation of the dilution protocols on our platform. But lycopene and DMSO combine into what Alber (Albert et al., 2006) euphemistically refers to as a complex liquid and for which there are few established recommendations for pipetting.

3.3 Design and methodologies

In this section, we only discuss methodologies developed in the second phase of the study for the dilution protocols based on the direct transfer method, and using the plate design named ‘linear dilution scheme’. Due to the transfer issues with the sequential transfer methods, which are discussed in the ‘Automation of Liquid Handling’ section, the iterative transfer method was considered unsuitable for our purpose.

Ideally, dilution and measurement are decoupled. The content of wells can be assumed to be known to a sufficient precision – either because liquid-handling makes small errors and can be trusted (which was not the case with lycopene in DMSO mixes), or because the dispensed volumes can be measured accurately (which was not possible). Measurement errors are also often assumed to be randomly distributed (they possibly depend on the content of the well), but do not depend on time or any external factors. Our data showed that this was not the case with our mixes, and consequently aggregation of successive measurements had to be carefully conducted in order to minimise the overall errors. These challenges are severe (but are not exclusive to lycopene in DMSO). We believe our solutions, discussed below, are valid beyond our choice of solute and solvent.

Two sets of methodologies were developed. The first methodology yields (for all wells) an estimate of the dilution ratio. It utilises the plate design containing on a single plate a large set of dilution ratios (the independent source of variability), and builds a regression model for these samples. Dilution ratios are then estimated by inverting the regression model. An error metric (quantifying how much a data point deviates from the trend of the dataset) returns a measure of the overall dataset consistency. Individual residuals are used to detect outliers. Such error metrics are popular in process control (variance control especially), but are not without their limitations. Their use is blind to systematic biases in the process (e.g. a

consistent 5% bias in volume transfer cannot be detected). Also, they fail if too many samples in the same dataset are unreliable (a case never encountered even with the highest concentration mixes).

The second methodology consisted of a set of rules (protocol) to ensure data quality and consistency. The protocol comprised three levels – each dealing with an identified , confounding source of variation in the data: variation between batches of lycopene mixes (preparation, chemical state), variation between plate treatment and sample processing and finally variation during measurements. When data were collected according to the protocol (as was the case in the final phase of the study), all measurement repeats can be considered either technical repeats or measurements carried out on similar objects (the study equivalent to biological repeats). It is noteworthy that even with such precautions, some wells still exhibited transient fluctuations (which we attributed to severe solubility issues), and that led us to develop a novel measurement aggregation method – based again on dataset consistency.

3.4 Lycopene in DMSO

The mix of lycopene and DMSO proved to be challenging to pipette and, unexpectedly, also challenging to measure (due to a range of solubility issues). This resulted in the need to deal with outlier measurements (especially for the highest concentrations).

Our absorbance measurements confirmed that the absorbance spectrum of lycopene in DMSO had the three-peak shape described by other authors, that is distinctive of carotenoids – although we found peaks that were shifted by a few nanometers from previous reports (Chemat-Djenni et al., 2010; Zhao et al., 2020). The degradation studies also yielded results consistent with previous studies – they confirmed that there was little degradation in the first hour at room temperature and that it was safe to run several technical repeats over that time scale. By combining mixes of lycopene and DMSO, we could construct a standard curve exhibiting a clear linear phase, and then saturation. This standard can be used to estimate yield for lycopene-production assays after extraction in DMSO – as was our initial goal.

At the same time important sources of variability were identified that are largely ignored in the published literature. In the published literature (Zhao et al., 2020; Saeid et al., 2016; Shi et al., 2003; Srivastava & Srivastava, 2015) much is made of denaturation by light and heat – so in our study great care was taken in minimising these effects when storing the mixes. But even with such precautions, the mixes exhibited a large range of absorbance spectra – some being so flat their distinctive three peaks would disappear. Our data led us to conclude that the observable time dependence was caused by another process affected by the age mix and concentration, and that some aggregation process was the main driver of these

results. Only oblique references to the time-dependence could be found in the literature (Zang, Sommerburg & van Kuijk, 1997). This is not uncommon, as few negative results or warnings are published – a common complaint in the scientific community (Matosin et al., 2014; Mlinarić, Horvat & Šupak Smolčić, 2017).3.5 Automation of Liquid Handling

Our study has also revealed three interesting methodological insights into the automation of the liquid handling of lycopene in DMSO. These are: (a) the unsuitability of the iterative transfer method; (b) the comparison of manual and automated protocol implementations; (c) the suitability of the direct transfer method. In our view, some of the conclusions are valid beyond our choice of solute and solvent – and that it's important to share them with the wider community interested in automation of liquid handling. Consequently, we expand on these points in this section

The first point to make is that the geometric dilution scheme (built on iterative transfers) proved to be unsuitable for the construction of the standard of lycopene in DMSO. This is due to the combination of two separate factors: (a) iterative dilution introducing non-negligible, correlated errors; and (b) the mix of lycopene in DMSO making these errors significantly worse. The first part of this result was unexpected – the second part was not.

Initially we believed that using the mix of lycopene and DMSO (especially at higher concentrations) would show that iterative liquid transfers are error prone. The underlying reasons were considered to be: (a) the one-size-fits-all nature of the protocol in its assignment of aspiration and dispensation parameters would cause issues when dealing with columns of liquid with varying viscosity (as is the case with the lycopene mix); and (b) the successive transfers would propagate the errors, and potentially compound them. Our data showed large errors in the transfers from adjacent columns, especially when the operation was conducted on mixes of lycopene in DMSO. No evidence of contamination by leakage could be observed, which can be attributed to a good choice of equipment (mainly the tips). We did not expect to find any issues with standard liquids. However, our data show clear jumps from column to column caused by transfers of too large volumes. If replicated, this result would imply that the popular sequential transfer method should not be used for any accurate form of measurement – only for very coarse assays. It is worth noting that these jumps would not have been identified without the linear spread of concentrations within columns – proving again the importance of plate design.

One of the greatest advantages of the iterative transfer method is how easy it is to use manually – making it a perfect testbed to compare automated and manual protocol implementations. With the training mixes (that were easy to work with), the data generated with the manual protocols had comparable errors caused by the iterative transfer from column to column to automation data. But with difficult mixes (such as lycopene in DMSO),

the differences were very noticeable – in the favour of the manual protocol. We postulate that the cause of the differences is the existence of a control feedback mechanism in liquid-handling among experienced experimentalists. This helps them adapt to the liquid being handled (probably through some form of hand-eye coordination). Such feedback is absent in automated applications where the robotic platform rigidly implements a series of instructions. We believe that the results will apply to more cases (ill-adapted automated protocols can be expected to generate worse data than manual protocols, and sometimes significantly worse data).

In contrast to iterative schemes, schemes built on what we have called ‘the direct transfer approach’ (such as the linear dilution scheme in this study) proved well-suited to automation. Minor variations related to parameters, such as the aspiration and dispensation speeds, were observed. Mixing ‘in the well’ proved an issue with the high-concentration mixes of lycopene and DMSO, which had outliers that required removal. Mixing in the wells is, in general, more difficult with direct transfer as the wells vary in content and are therefore liable to exhibit separate mixing issues. A multiple-repeat measurement assay as conducted in this study can be used to detect well-specific issues and optimise the mixing parameters.

We consider the suitability of the direct transfer method is of great importance in the context of automation of liquid handling. Direct transfer is conceptually the simplest approach to liquid-handling, as it relies on one set of instructions only: go to reservoir R, aspire volume VA, go to well Wij, dispense volume VD. Fewer parameters need to be assigned and tuned: for each of the reservoirs, aspiration and dispensation parameters can be set up globally – although mixing parameters may need to be well-specific or concentration-specific. But its greatest quality is how flexible, systematic and programmatic friendly it is.

One of the most immediate consequences of this flexibility is the capacity to generate almost any plate pattern – a major advantage in relation to subsequent statistical analysis. In addition, any pattern can be randomised to protect against local factors. It is worth emphasising that this is only practical with automated protocols. In regards to achievable concentrations, our study was restricted to the transfer of safe volumes (multiples of 10µl) from one reservoir of solvent and one reservoir of solute. This was enough to evenly sample a space of possible dilutions. With the direct transfer method, dilution schemes can be designed that not only sample dilution spaces unevenly, but also adapt the sampling rate to the portion of the dilution space (possibly achieving very fine rates) – all the while ensuring only safe volumes are transferred. All that is required is to add a handful of reservoirs of different concentrations, and dispatch from these reservoirs combinations of safe volumes that achieve the target dilutions – the same way as one would pay with coins of different denominations. The approach can also be extended to more than one solute and/or solvent.

3.6 Conclusion

Chapter 4 presented a study that approaches liquid-handling and its reliable automation from the standpoint of the construction of the standard curve. The study involved a mix of lycopene and DMSO. Very practical recommendations regarding automation of liquid-handling have been shared.

The study also found evidence that automation is not a panacea. Automation removes to a large extent the influence of the human operators, but it does not, necessarily, produce better data and it introduces new classes of errors. Yet, it is not difficult to be excited by the power and promises of automation. Our study highlighted several aspects that are worth repeating :

- The power of (mechanised) automation: Once the protocols were settled, they could be reliably and straightforwardly repeated over dozens of plates. Such a scale (and higher scales) is very difficult to reach with manual protocols and human operators.
- The power of analytics: the analysis techniques used in the study were standard statistical techniques (despite the amount of data collected, there were too few for most machine learning approaches). Nonetheless the methodology developed in the study, which related all measurements to the trend of the dataset, allowed insights into several issues related to automated liquid-handling. More generally the methodology facilitated the monitoring, ranking and optimisation of the automated protocols.
- The importance of coupling analytics and plate and experimental design: The statistical methodology would not have been possible without the plate designs used in our study, as well as the experimental designs decoupling of the various effects. In many respects the finding that the direct transfer method of liquid handling (which is the most systematic, the most programmatic-friendly and the most amenable to mathematical predictions) is to be recommended is, we believe, of great importance, as direct transfer creates a seamless bridge between analytics, plate design (however complex), and practical implementation.

4. Automation of Workflows in the Biofoundry - Discussion (Chapter 6-7)

4.1 Overview

Chapter 5 set the basis of our conceptual framework for the study. The framework implemented as computational tools allowing us to structure experiments and their resulting data. Our framework introduces the concept of workflows that are composed of tasks which can represent protocols. We have introduced our conceptual framework which acts as a bridge between the physical world and the information world (digital twinning). We have shown how we represent experiments in synthetic biology (the real world model) and how we apply an operational pipeline that treats our objects for observation.

In Chapter 6 we have introduced the structure of directed graphs (DGs) and directed cyclic graphs (DAGs) on top of which workflows can be built. The DAGs allow us to structure the workflow and the dependencies between the different tasks. We introduced the model graph, a directed graph (DG) that links the dependencies of all the data being produced along the workflow. Workflows can be executed as runs and the result of these runs gets stored in a DAG called an execution graph.

We presented results of the implementation of this framework:

- We have shown how, technically, we can implement and build those DAGs with computational libraries and store the DAGs in graph databases.
- Structuring the data in a DAG and storing it in a graph database allows us to record a history of the runs and query the graph database to find data provenance.

Such a way of structuring experiment in workflows built on top of DGs has another advantage: by following the graph edges of the DG, it is possible to read the order of the task and schedule of the execution of the tasks that it comprises. This allows the DG to be used as a blueprint for the orchestration of tasks and automation of the execution of workflows in the biofoundry.

In Chapter 7 we describe how we can use the structure of the graph, and, more particularly, the model graph**,** as a template for automating the execution of the workflow. We described how an automated laboratory can be built when workflows are stored as model graph**s** and coupling the workflow to software called an orchestrator.

We introduced our solution to automated workflow – the Biofoundry Automation Manager (BAM). The BAM automatically manages workflows and their execution on a biofoundry’s

infrastructure. The architecture of the BAM is divided into four main components: the model graph, the orchestrator, the code libraries and a computational/physical infrastructure. An explanation is given as to why the Apache Airflow was chosen as the orchestrator and how it is implemented as a model graph in Airflow.

We presented results of examples of workflow executions, using the Biofoundry Automation Manager (BAM) (See Chapter 7 - Section 4):

- Example 1: The execution of a generic workflow that contains a repetition loop.
- Example 2: a computational workflow of 'BASIC DNA Assembly' for *in silico* DNA assembly.
- Example 3: a wet lab workflow for an 'Automated Standard Curve', to automatically dilute a reagent to produce a standard curve.

4.2 Discussion

Separation of the model graph and the execution graphs

In the Biofoundry Information Manager (BIM) both the model graph and the execution graph are stored in the same graph database (Neo4j). Storing both graphs in one database has been done for the technical convenience of using one unique database for the work presented in this thesis. One advantage of keeping both graphs in the same database is to keep links between the Tasks and I/O objects of the execution graphs and their counterparts in the model graph. There is no reason to keep both graphs in the same database as long as a correspondence between the objects in the execution graphs and the model graphs share a corresponding attribute (the same name, a common unique identifier or a parenting link).

Serialisation of the execution graph for data sharing

The goal of keeping the execution graphs in the BIM is to be able to query the runs. When data from an experiment is shared, our observation of the most common practice in the field shows that sharing is being done via a large unstructured data dump (e.g. zipping all the content of a directory) – in the best cases, the dump comes with an accompanying document describing its content. The addition of an execution graph to the data export of an experiment (on run or multiple runs) provides the global context of each of the runs and a history of all the successive tasks and their statuses (success or failure). Historical context sharing of runs can easily be done from the BIM with an export of a serialisation of the execution graphs in various formats such as CSV, JSON and GraphML (an XML-based file format for graphs) – which are all being supported by the chosen graph-database Neo4j.

Interaction with equipment (need interface for control)

To achieve full automation in the Biofoundry, The Biofoundry Automation Manager (BAM)

requires a cyber-physical infrastructure in the Biofoundry where all equipment is interconnected. The requirement of interconnection between the equipment exists for two reasons: first, the need to send commands to the equipment, second, to retrieve the status of the execution of the command and the measurement data.
The Biofoundry Automation Manager orchestrates tasks from the workflows and dispatches them to all the worker resources (i.e. equipment that executes the tasks) of the Biofoundry. Unless the equipment adheres to a standard with a common interface such as SiLA (Standardization in Lab Automation) it becomes very difficult to achieve an interconnected infrastructure in a Biofoundry. Fortunately, industrial standards for device integration and communication have appeared in the industry with standards such as 'Standardization in Lab Automation' SiLA (Bär, Hochstrasser & Papenfuß, 2012) and more recently SiLA2 (Bromig et al., 2022). Work from Porr et al. has been done to integrate laboratory devices with a central lab server (Porr et al., 2021) and demonstrated that it is also possible to integrate legacy devices (Porr et al., 2020) that existed before the emergence of the SiLA/SiLA2 standard.

Error from infrastructure – querying the state from the infrastructure

The execution of the tasks comprising a workflow necessitates an infrastructure with functioning equipment. Such a statement might seem obvious, but in order to avoid failure during the execution of a workflow there needs to be an infrastructure validation procedure – i.e. making sure that: 1. the equipment is available, 2. the equipment can execute the task. For the worfklow and the tasks comprising it to be able to get executed, multiple states of validation of the resources of the infrastructure need to be reached. In the same way as a wetlab scientist walks around the lab in the morning before starting running their experiments (e.g. making sure that the equipment that is going to be used in there and available, but also making sure that the state of the equipment is valid before running a task (e.g. waiting for an incubator to reach a certain temperature)) – the Biofoundry Automation Manager (BAM) needs to be able to query the state of its underlying infrastructure. The first obvious validation state answers the question: Is the piece of equipment there ? But many other questions from the BAM need to be answered by the resource such as: is the equipment available ? Can we establish communication with the equipment ? Is the equipment in the right state to execute the task ?

Different states of validation need to be verified before executing a workflow or a task on a Resource:

- Existing Resource
- Resource availability
- Is communication with the resource possible ?

- Are conditions for the execution of the Task on the Resource met ?

A hierarchy of states of validation can be established starting with resource existence at the top of the hierarchy, all the way down to the lower level of the hierarchy with the state of the different specific states of the resource needed to meet conditions required by a Task.

If the equipment of the Biofoundry has the right interface (such as SiLA) and if each piece of equipment offers to communicate the information that is necessary for validation, the state of validation of the whole infrastructure is possible. Otherwise monitoring systems of the equipment need to be put in place in the Biofoundry or human intervention to verify the state of the equipment becomes necessary.

Need for a resource manager

The BAM has been implemented with the Airflow orchestrator. Airflow contains a scheduler and distributes tasks to workers (in the Airflow terminology). Computational workers are simply computers that run a job but the equipment of the Biofoundry cannot necessarily communicate directly with an orchestrator and thus an additional computer is required to act as a worker and communicate with the piece of equipment.

The workers need to be managed by a resource manager. A resource manager is a system that manages the computing nodes in a cluster of machines. There exist various resource managers such as Kubernetes (an open-source container orchestrator) (Burns et al., 2016) or HTCondor (Erickson et al., 2018). Airflow has native support for Kubernetes.

Lehman et al. (2023) propose a common workflow scheduling interface which is a simple RESTful (the term REST stands for representational state transfer, a software architectural style, this is the term originally used by the authors in their paper, but the term RESTful is employed here to denote an HTTP based API) interface between workflow managers and resource managers. Lehman et al. (2023) propose an API to exchange scheduling information between workflow managers and resource managers. An adoption of such a standard would allow interoperability between workflow managers and resources managers and avoid a custom integration for each workflow/resource manager possible combination.

Workflow as code

Describing a workflow as code is the most versatile way of describing a workflow that can be directly executed. Other methods of describing workflows (for example: as data structures) require the interpretation of the structure to then be executed by a program. Describing a workflow in pure code (not limiting ourselves to a DAG) allows complete freedom in the nature of the workflow. Workflow as code offers many advantages: executable workflow, freedom of flow, extension of the workflow with external libraries and easy workflow versioning. Workflow as code can be executed directly and there is no need for a specific workflow engine to interpret a data structure representing the workflow. We have seen in

Chapter 7 that the BAM uses Airflow as an orchestrator and because we used Airflow, we implemented workflows as Airflow DAGs which are written in python. Real-world workflows in a Biofoundry require repeats of tasks and repetition loops in the workflows and because we have been restricted by the structure of DAG which is by nature acyclic we have been forced to find a workaround solution (Chapter 7 - Section 3.1). Representing a workflow as code allows easily modularisation of different parts of the code and allows the refactorisation of code with reusable components libraries. As shown in Chapter 7 - Section 2.3 that we can make extensive use of external libraries to extend the capabilities of our workflows. This is not as easily doable if we do not express workflows as codes. Code listings can be versioned very easily. Many distributed collaborative versioning systems (DCVS) such as git (Git, n.d.) exist with a huge software tools ecosystem supporting it and source forges such as Github (GitHub, n.d.) ease collaborative development. As such collaboration on workflow development, versioning, release and sharing is simplified.

4.3 Conclusion

When implementing automated workflows in a Biofoundry, it is necessary to have a solution that can accommodate any type of experimental workflow – e.g. automated, semi-automated, manual and both computational, and wetlab workflows. The solution offered by both the BIM and the BAM accommodate different types of workflows – as illustrated in the user-case examples. The BAM executes the workflow and records the execution of the runs in an execution graph in the BIM. The BIM allows different stakeholders to query the information contained in the execution graph**.** The information can be retrieved during multiple runs of the same experiment by researchers involved in the experiment or cross-experiments information can be retried by people managing the infrastructure. The Biofoundry Information Manager (BIM) coupled with the Biofoundry Automation Manager (BAM) is seen as offering an effective solution for managing an automated infrastructure of biofoundries.

Chapter 9

Future Work

In this chapter we discuss the further steps of the work presented in this thesis. cMatch, presented in Chapter 3, can be extended to become a more versatile tool and its implementation can be optimised. Additionally, to make the tool accessible to non-technical users, cMatch will be enhanced with a user-friendly graphical user interface. The Biofoundry Information and Automation Managers in their current implementation share a single storage system and will in the very near future, be separated into two distinct systems. Furthermore, additional work to interface the equipment of the Biofoundry to the BAM will be undertaken. In addition, a graphical user interface will be added to the BAM to enable manual and semi-manual tasks data transfer in and out of the system.

1. Improvements to cMatch

cMatch has been developed with an initial goal of being a construct identification tool. The aim is now cMatch to be used to perform quality control of modular synthetic constructs at speed and scale.

Optimising the implementation of the algorithm

The next possible steps to make cMatch run faster is to optimise the implementation of the algorithms by using the advantages of parallelising tasks and making use of multiple cpu cores. This will be done with Python's multiprocessing module, which achieves parallelism by creating separate processes – each with its own Python interpreter and memory space. This effectively bypasses the Global Interpreter Lock (GIL), making it suitable for CPU-bound tasks. Multiprocessing can be employed to process multiple constructs in parallel – this works for CM_0, CM_1 and CM_2. Multiprocessing can also be employed for the library parts matching steps which compute the score and positions of the components as the operations are independent of each other.

A graphical user interface for the cMatch tool

cMatch has been developed as software to run batch jobs – processing hundreds of sequences and returning the candidates with the best scores. cMatch will benefit from a graphical user interface (GUI) – as shown in a graphics GUI mockup in Figure 9-1. In the new version cMatch, as a visualisation tool, will take a DNA sequence as an input – e.g. the result of sequencing – and display its content in a similar manner to a graphical DNA editor. Similar to the way annotations are displayed over the nucleotide sequences, cMatch GUI displays the matching parts along with their sequence and the matching score – additionally, the non-matching base-pairs (for matching scores < 1.0) are highlighted.

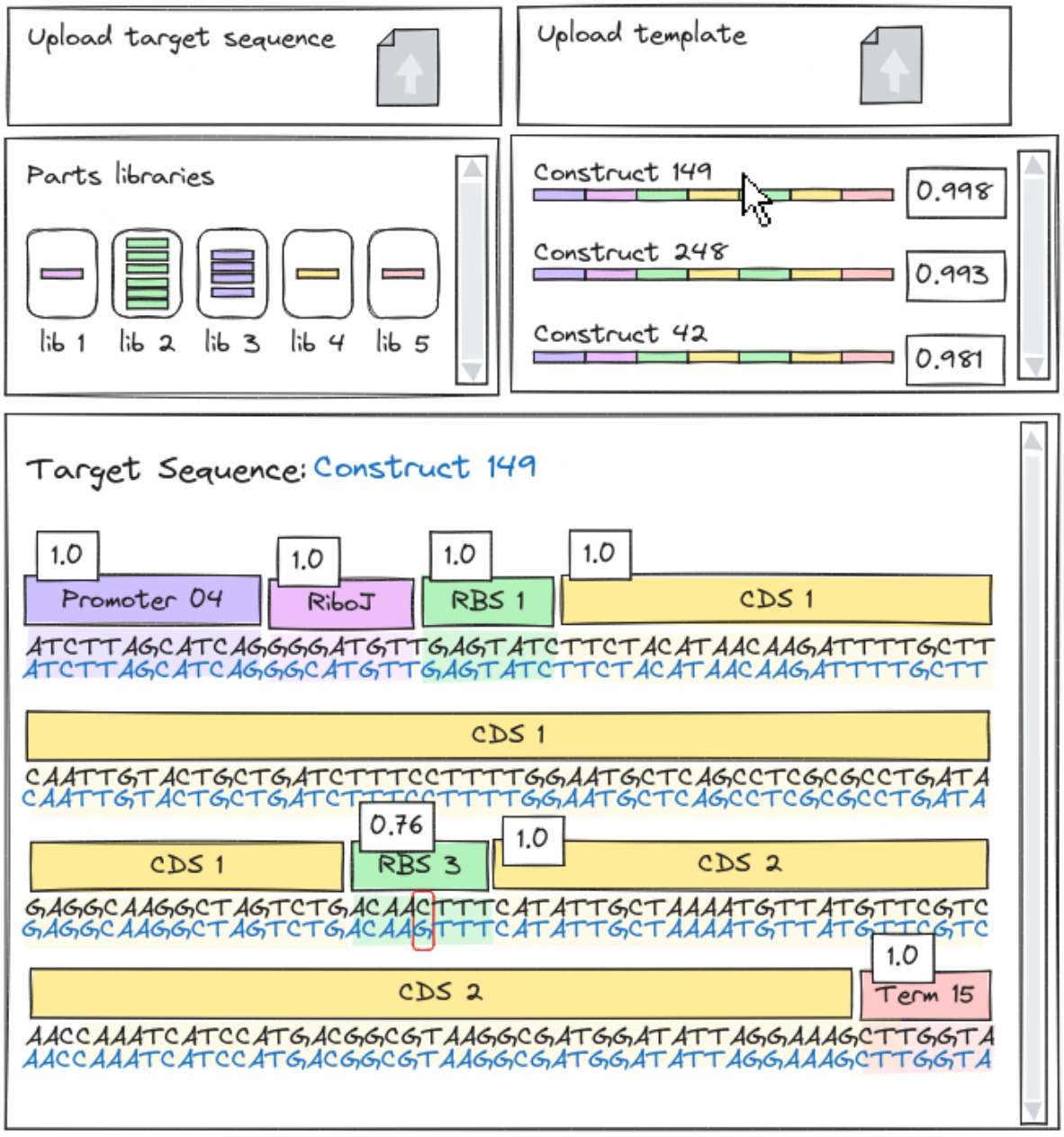


Figure 9-1 – cMatch Graphical User Interface mock-up. The user interface allows the user to upload sequences, templates and parts libraries. The target sequence can be browsed with the matching sequences displaying the matching parts and their scores and highlight (in red) the disparities between the target sequence and the matching parts.

2. The Biofoundry Information and Automation Manager

Storing the model graph and the execution graph in different databases

Chapter 8, (Section 4.2) discussed the separation of the model graph and the execution graphs in two different systems. However, in order to keep a link between nodes from both graphs (e.g. Tasks from a run in the execution graph have a corresponding Task in the model graph), the work presented in the thesis implements both systems (the BIM and the BAM) using the same graph database. For separation of concerns – a general principle in software engineering introduced by Dijkstra (1982) and Parnas (1972) – it is more appropriate to store the model graphs and the execution graphs in two different storage systems. The separation-of-concerns (SoC) principle is one of the essential principles in software engineering which says that software should be decomposed in such a way that different ‘concerns’ or aspects of the problem are solved in well-separated modules or parts of the software (De Win et al., 2002). Consequently, the next step in the work on the BIM and the BAM is to fully separate the two systems and the storage of the model graphs and the execution graphs into two distinct databases.

Exporting execution graphs

Having access to readable files of the execution graphs for data analysis – and thus implementing serialisation of execution graphs – is a very convenient feature of the BIM. Straight serialisation can be done directly through the Neo4j graph database with common formats such as CSV, JSON, GraphML. The next step will be to develop a hierarchical format that allows us to store the relationship between the I/O, the containers and their maps and the transformal operations. This will, in turn, allow us to create digital twins of all the objects and all the operations done in the workflows.

Bioufoundry equipment drivers

The interaction between the Biofoundry’s equipment and the BAM is probably the biggest bottleneck – and the biggest fight to have with commercial vendors that do not give an open operational access to the equipment. (An open API to send commands to operate the equipment (set parameters and operation instructions) and to retrieve data produced by the equipment (operational data and measurement data.)

Graphical user interface for the Biofoundry Automation Manager

Seamless integration of the BAM and the wetlab equipment (such as automated liquid handlers and micro-plate readers) reduces additional manual tasks undertaken by a human operator in the Biofoundry (e.g. exporting and copying data files out of the measurement equipment, renaming the files, importing the data files in the BAM etc). Seamless integration necessitates equipment drivers that the BAM can talk to over a standardised communication

protocol. As shown in Chapter 8 (Section 4.2) the SiLA2 protocol can be used as an exchange protocol for automation equipment.

When it is not possible to integrate the BAM with the Biofoundry's wetlab equipment (e.g because of the lack of equipment with an open API), the transfer of the data between the BAM and the wetlab equipment (e.g. transfering a protocol written for the AJ Felix Automated Liquid Handler to the device or transfering measurement data files back from a BMG Clariostar microplate reader) has to be done manually by the operator. This requires the development of a graphical user interface (GUI) for the BAM. (The BAM GUI displays the tasks of the workflow and the I/O objects with placeholders for I/O proxy objects that the operator would fill by dragging (copying) the data files in the GUI.)
Because the BAM enables combinations of fully manual, semi-automated and fully automated tasks and workflows, a GUI is necessary for the operator to validate manual and semi-automated tasks.

Deploying a resource manager

As described in Chapter 8 (Section 4.2) the BAM distributes the tasks to the equipment of the Biofoundry through a pool of workers that drive the equipment. Consequently, there is the need for a system, the resource manager, to manage the availability of the equipment. To fully integrate the BAM with all the equipment in the Biofoundry requires the deployment of 'workers' along with a resource manager, such as Kubernetes, which enables the management of a cluster of workers.

Defining workflows with a nodal interface

The workflows are written as AF DAGs. This requires the use of Python. Writing and editing a workflow with Python is not the most convenient interface for operators in a wet lab. A more convenient way to define workflows can be achieved by creating a nodal editor, where I/O objects and tasks can be linked in a visual editor by drag-and-drop. Therefore, a near term objective is to create a nodal editor. The ability to create a workflow 'on the fly', while working in a wetlab, can be achieved by creating task nodes and linking them through a graphical user interface. This will facilitate improvisation when building workflows, especially for research projects. The ability to modify a workflow by editing the graph directly through an interface gives the operator far greater flexibility.

Porting all the protocols to Laboratory Open Protocol (LabOP) language

As seen in Chapter 5 (Section 2.4.2), protocols come in various forms of representation. (This is also true for the protocols used in the Biofoundry and in the work presented in this thesis.) In order to have a uniform standardised protocol format, we are planning to write all our protocols in LabOP. Writing all the protocols in a standardised programmatic language forces the protocol to be fully specified – this avoids ill-defined parameters such, as

‘overnight’ (as in ‘incubate overnight).

LabOP has a specification and a reference implementation (in Python), which makes it convenient and safe to use. Protocols written in LabOP can be translated into written instructions in a PDF or simple text format (Markdown) document – which doesn’t alter them from how they are currently . Additionally, the language and its reference implementation enable the automatic execution of the protocol. The drawback is that there are at present cvery few drivers for current automation platforms – Opentrons being one of them. This limits the ability to directly deploy the protocols on the Biofoundry’s equipment and execute them. However, because LabOP ing is open-source, it is possible to write specific drivers – especially if the equipment adheres to the SiLA2 laboratory automation standard.

Having all of the Biofoundry’s protocols written in a standard programmatic language (such as LabOP) allows the specification of the workflows and protocols at an abstract level that makes the workflows independent of the underlying infrastructure. An abstract specification of protocols enables a modular architecture to the infrastructure of the Biofoundry and allows any individual piece of equipment to be readily interchanged with one of equivalent functionality. Additionally, this allows replicability of workflows on different Biofoundry infrastructures. The current state of laboratory automation does not fully permit seamless workflows and protocol transfers, but we consider the framework introduced in this thesis to be a step towards the portability and replicability of workflows over different Biofoundries.

Bibliography

van der Aalst, W.M.P., Bichler, M. & Heinzl, A. (2018) Robotic Process Automation. Business & Information Systems Engineering. 60 (4), 269–272. doi:10.1007/s12599-018-0542-4.

Abt, V., Barz, T., Cruz-Bournazou, M.N., Herwig, C., Kroll, P., Möller, J., Pörtner, R. & Schenkendorf, R. (2018) Model-based tools for optimal experiments in bioprocess engineering. Current Opinion in Chemical Engineering. 22, 244–252. doi:10.1016/j.coche.2018.11.007.

Adrio, J.L. & Demain, A.L. (2006) Genetic improvement of processes yielding microbial products. FEMS Microbiology Reviews. 30 (2), 187–214. doi:10.1111/j.1574-6976.2005.00009.x.

Ahmad, Z., Jehangiri, A.I., Ala'anzy, M.A., Othman, M., Latip, R., Zaman, S.K.U. & Umar, A.I. (2021) Scientific Workflows Management and Scheduling in Cloud Computing: Taxonomy, Prospects, and Challenges. IEEE Access. 9, 53491–53508. doi:10.1109/ACCESS.2021.3070785.

Ajikumar, P.K., Xiao, W.-H., Tyo, K.E.J., Wang, Y., Simeon, F., Leonard, E., Mucha, O., Phon, T.H., Pfeifer, B. & Stephanopoulos, G. (2010) Isoprenoid Pathway Optimization for Taxol Precursor Overproduction in Escherichia coli. Science. 330 (6000), 70–74. doi:10.1126/science.1191652.

Albert, K.J. (2007) Minimizing Liquid Delivery Risk: Automated Liquid Handlers as Sources of Error. 39 (13).

Albert, K.J., Bradshaw, J.T., Knaide, T.R. & Rogers, A.L. (2006) Verifying Liquid-Handler Performance for Complex or Nonaqueous Reagents: A New Approach. JALA: Journal of the Association for Laboratory Automation. 11 (4), 172–180. doi:10.1016/j.jala.2006.06.003.

Alper, H., Jin, Y.-S., Moxley, J.F. & Stephanopoulos, G. (2005) Identifying gene targets for the metabolic engineering of lycopene biosynthesis in Escherichia coli. Metabolic Engineering. 7 (3), 155–164. doi:10.1016/j.ymben.2004.12.003.

Alser, M., Rotman, J., Deshpande, D., Taraszka, K., Shi, H., Baykal, P.I., Yang, H.T., Xue, V., Knyazev, S., Singer, B.D., Balliu, B., Koslicki, D., Skums, P., Zelikovsky, A., Alkan, C., Mutlu, O. & Mangul, S. (2021) Technology dictates algorithms: recent developments in read alignment. Genome Biology. 22 (1), 1–34. doi:10.1186/s13059-021-02443-7.

Amer, B. & Baidoo, E.E.K. (2021) Omics-Driven Biotechnology for Industrial Applications. Frontiers in Bioengineering and Biotechnology. 9. https://www.frontiersin.org/articles/10.3389/fbioe.2021.613307.

Analytik Jena AG (n.d.) CyBio FeliX – Pipetting Robot Liquid Handling. https://www.analytik-jena.com/products/liquid-handling-automation/liquid-handling/flexible-benchtop-liquid-handling/cybio-felix-series/ [Accessed: 9 September 2023].

Anarjan, N. (2020) Evaluation the effects of ultrasonic parameters on simultaneously extraction and size reduction of lycopene from tomato processing waste. Waste and Biomass Valorization. 11 (5), 1929–1940.

Anderson, Jc., Dueber, J.E., Leguia, M., Wu, G.C., Goler, J.A., Arkin, A.P. & Keasling, J.D. (2010) BglBricks: A flexible standard for biological part assembly. Journal of Biological Engineering. 4 (1), 1–12. doi:10.1186/1754-1611-4-1.

Anon (n.d.) CyBio FeliX SELECT Head.

A. Rinaldi, M., A. Ferraz, C. & S. Scrutton, N. (2022) Alternative metabolic pathways and strategies to high-titre terpenoid production in Escherichia coli. Natural Product Reports. 39 (1), 90–118. doi:10.1039/D1NP00025J.

Arnold, F.H. (2018) Directed Evolution: Bringing New Chemistry to Life. Angewandte Chemie International Edition. 57 (16), 4143–4148. doi:10.1002/anie.201708408.

Autoprotocol (2014) Autoprotocol. 2014. https://www.autoprotocol.org/ [Accessed: 15 January 2021].

Axmann, B. & Harmoko, H. (2020) Robotic Process Automation: An Overview and Comparison to Other Technology in Industry 4.0. In: 2020 10th International Conference on Advanced Computer Information Technologies (ACIT). September 2020 pp. 559–562. doi:10.1109/ACIT49673.2020.9208907.

Azubuike, C.C., Edwards, M.G., Gatehouse, A.M.R. & Howard, T.P. (2020) Applying Statistical Design of Experiments To Understanding the Effect of Growth Medium Components on Cupriavidus necator H16 Growth. Applied and Environmental Microbiology. 86 (17), e00705-20. doi:10.1128/AEM.00705-20.

Babaei, M., Borja Zamfir, G.M., Chen, X., Christensen, H.B., Kristensen, M., Nielsen, J. & Borodina, I. (2020) Metabolic Engineering of Saccharomyces cerevisiae for Rosmarinic Acid Production. ACS Synthetic Biology. 9 (8), 1978–1988. doi:10.1021/acssynbio.0c00048.

Bailey, J.E. (1991) Toward a science of metabolic engineering. Science. 252 (5013), 1668–1675.

Baker, I. (2018) Polypropylene. In: I. Baker (ed.). Fifty Materials That Make the World. Cham,

Springer International Publishing. pp. 169–173. doi:10.1007/978-3-319-78766-4_32.

Bang-Jensen, J. & Gutin, G. (2018) Classes of Directed Graphs. Springer Monographs in Mathematics. Cham, Springer International Publishing. doi:10.1007/978-3-319-71840-8.

Bär, H., Hochstrasser, R. & Papenfuß, B. (2012) SiLA: Basic Standards for Rapid Integration in Laboratory Automation. Journal of Laboratory Automation. 17 (2), 86–95. doi:10.1177/2211068211424550.

Baratta, A., Cimino, A., Gnoni, M.G. & Longo, F. (2023) Human Robot Collaboration in Industry 4.0: a literature review. Procedia Computer Science. 217, 1887–1895.

Barba, A.I.O., Hurtado, M.C., Mata, M.C.S., Ruiz, V.F. & Tejada, M.L.S. de (2006) Application of a UV–vis detection-HPLC method for a rapid determination of lycopene and β-carotene in vegetables. Food Chemistry. 95 (2), 328–336. doi:10.1016/j.foodchem.2005.02.028.

Barbiero, P., Viñas Torné, R. & Lió, P. (2021) Graph Representation Forecasting of Patient's Medical Conditions: Toward a Digital Twin. Frontiers in Genetics. 12. https://www.frontiersin.org/articles/10.3389/fgene.2021.652907.

Barika, M., Garg, S., Zomaya, A.Y., Wang, L., Moorsel, A.V. & Ranjan, R. (2019) Orchestrating Big Data Analysis Workflows in the Cloud: Research Challenges, Survey, and Future Directions. ACM Computing Surveys. 52 (5), 95:1-95:41. doi:10.1145/3332301.

Bartley, B., Beal, J., Bryce, D., Goldman, R.P., Keller, B., Lee, P., Nowak, J., Rogers, M. & Weston, M. (2021) Protocol Activity Markup Language (PAML) Version 1.0.0-alpha.

Bartley, B., Beal, J., Rogers, M., Bryce, D., Goldman, R.P., Keller, B., Lee, P., Biggers, V., Nowak, J. & Weston, M. (2023) Building an Open Representation for Biological Protocols. ACM Journal on Emerging Technologies in Computing Systems. 19 (3), 28:1-28:21. doi:10.1145/3604568.

Bartley, B.A., Atallah, C., Humphries, A., Kulkarni, V., Madsen, C., Misirli, G., Goñi-Moreño, Á., Nguyen, T., Oberortner, E., Roehner, N., Samineni, M., Zundel, Z., Beal, J., Myers, C., Sauro, H. & Wipat, A. (2019) The Synthetic Biology Open Language Supports Integration of the Engineering Life-Cycle for Synthetic Biologists. In: 2019 p. https://www.semanticscholar.org/paper/The-Synthetic-Biology-Open-Language-Supports-of-the-Bartley-Atallah/20a4657a2aae8d81143ab3c3d340c5ca76f7c165.

Bartley, B.A., Beal, J., Karr, J.R. & Strychalski, E.A. (2020) Organizing genome engineering for the gigabase scale. Nature Communications. 11 (1), 689. doi:10.1038/s41467-020-14314-z.

basic-assembly.org (2023) basic-assembly.org. 2023. BASIC Assembly. https://www.basic-assembly.org [Accessed: 11 September 2023].

Bates, M., Berliner, A.J., Lachoff, J., Jaschke, P.R. & Groban, E.S. (2017) Wet Lab Accelerator: A Web-Based Application Democratizing Laboratory Automation for Synthetic Biology. ACS Synthetic Biology. 6 (1), 167–171. doi:10.1021/acssynbio.6b00108.

Batista, E., Filipe, E. & Mickan, B. (2008) Volume calibration of 1000 µl micropipettes. Inter-laboratory comparison. Accreditation and Quality Assurance. 13 (4), 261–266. doi:10.1007/s00769-008-0362-1.

Beal, J., Farny, N.G., Haddock-Angelli, T., Selvarajah, V., Baldwin, G.S., et al. (2020) Robust estimation of bacterial cell count from optical density. Communications Biology. 3 (1), 512. doi:10.1038/s42003-020-01127-5.

Beal, J., Haddock-Angelli, T., Baldwin, G., Gershater, M., Dwijayanti, A., Storch, M., de Mora, K., Lizarazo, M., Rettberg, R., & with the iGEM Interlab Study Contributors (2018) Quantification of bacterial fluorescence using independent calibrants D. Olson (ed.). PLOS ONE. 13 (6), e0199432. doi:10.1371/journal.pone.0199432.

Becker, J. & zur Mühlen, M. (1999) Towards a Classification Framework for Application Granularity in Workflow Management Systems. In: M. Jarke & A. Oberweis (eds.). Advanced Information Systems Engineering. Lecture Notes in Computer Science. 1999 Berlin, Heidelberg, Springer. pp. 411–416. doi:10.1007/3-540-48738-7_30.

Ben-David, A. & Davidson, C.E. (2014) Estimation method for serial dilution experiments. Journal of Microbiological Methods. 107, 214–221. doi:10.1016/j.mimet.2014.08.023.

Berezin, C.-T., Aguilera, L.U., Billerbeck, S., Bourne, P.E., Densmore, D., et al. (2023) Ten simple rules for managing laboratory information. PLOS Computational Biology. 19 (12), e1011652. doi:10.1371/journal.pcbi.1011652.

Biermann, F., Mathews, J., Nießing, B., König, N. & Schmitt, R.H. (2021) Automating Laboratory Processes by Connecting Biotech and Robotic Devices—An Overview of the Current Challenges, Existing Solutions and Ongoing Developments. Processes. 9 (6), 966. doi:10.3390/pr9060966.

Biolegio (2023) Biolegio. 2023. Biolegio. https://www.biolegio.com/ [Accessed: 11 September 2023].

Bird, J.E., Marles-Wright, J. & Giachino, A. (2022) A User’s Guide to Golden Gate Cloning Methods and Standards. ACS Synthetic Biology. doi:10.1021/acssynbio.2c00355.

Björksten, M.G., Almby, B. & Jansson, E.S. (1994) Hand and shoulder ailments among laboratory technicians using modern plunger-operated pipettes. Applied Ergonomics. 25 (2), 88–94. doi:10.1016/0003-6870(94)90069-8.

Blaize, J., Suter, E. & Corb, C. (2023) Serial Dilutions and Plating: Microbial Enumeration |

Microbiology | JoVE. 2023. https://www.jove.com/v/10507/serial-dilutions-and-plating-microbial-enumeration [Accessed: 13 September 2023].

Blazeck, J. & Alper, H.S. (2013) Promoter engineering: Recent advances in controlling transcription at the most fundamental level. Biotechnology Journal. 8 (1), 46–58. doi:10.1002/biot.201200120.

Blazeck, J., Garg, R., Reed, B. & Alper, H.S. (2012) Controlling promoter strength and regulation in Saccharomyces cerevisiae using synthetic hybrid promoters. Biotechnology and Bioengineering. 109 (11), 2884–2895. doi:10.1002/bit.24552.

Bogue, R. (2012) Robots in the laboratory: a review of applications. Industrial Robot: An International Journal. 39 (2), 113–119. doi:10.1108/01439911211203382.

Bongomin, O., Yemane, A., Kembabazi, B., Malanda, C., Mwape, M.C., Mpofu, N.S. & Tigalana, D. (2020) Industry 4.0 disruption and its neologisms in major industrial sectors : https://ir.busitema.ac.ug/handle/20.500.12283/698.

Borkowski, O., Ceroni, F., Stan, G.-B. & Ellis, T. (2016) Overloaded and stressed: whole-cell considerations for bacterial synthetic biology. Current Opinion in Microbiology. 33, 123–130. doi:10.1016/j.mib.2016.07.009.

Borràs, E. & Sabidó, E. (2017) What is targeted proteomics? A concise revision of targeted acquisition and targeted data analysis in mass spectrometry. PROTEOMICS. 17 (17–18), 1700180. doi:10.1002/pmic.201700180.

Brase, J. (2009) DataCite-A global registration agency for research data. In: 2009 IEEE. pp. 257–261.

Brayton, C.F. (1986) Dimethyl sulfoxide (DMSO): a review. The Cornell Veterinarian. 76 (1), 61–90.

Brendel, A., Dorfmüller, F., Liebscher, A., Kraus, P., Kress, K., Oehme, H., Arnold, M. & Koschitzki, R. (2022) Laboratory and Analytical Device Standard (LADS): A Communication Standard Based on OPC UA for Networked Laboratories. In: S. Beutel & F. Lenk (eds.). Smart Biolabs of the Future. Advances in Biochemical Engineering/Biotechnology. Cham, Springer International Publishing. pp. 175–194. doi:10.1007/10_2022_209.

Broad Institute (2024) broadinstitute/cromwell. https://github.com/broadinstitute/cromwell.

Bromig, L., Leiter, D., Mardale, A.-V., von den Eichen, N., Bieringer, E. & Weuster-Botz, D. (2022) The SiLA 2 Manager for rapid device integration and workflow automation. SoftwareX. 17, 100991. doi:10.1016/j.softx.2022.100991.

Bruynseels, K., Santoni de Sio, F. & van den Hoven, J. (2018) Digital Twins in Health Care: Ethical Implications of an Emerging Engineering Paradigm. Frontiers in Genetics. 9.

https://www.frontiersin.org/articles/10.3389/fgene.2018.00031.

Bultelle, M., de Murieta, I.S. & Kitney, R. (2016) Introducing SynBIS - the synthetic biology information system. In: IET/SynbiCITE Engineering Biology Conference. 2016 London, UK, Institution of Engineering and Technology. p. 2 .-2 . doi:10.1049/cp.2016.1254.

Burns, B., Grant, B., Oppenheimer, D., Brewer, E. & Wilkes, J. (2016) Borg, Omega, and Kubernetes: Lessons learned from three container-management systems over a decade. Queue. 14 (1), 70–93. doi:10.1145/2898442.2898444.

Cambray, G., Guimaraes, J.C. & Arkin, A.P. (2018) Evaluation of 244,000 synthetic sequences reveals design principles to optimize translation in Escherichia coli. Nature Biotechnology. doi:10.1038/nbt.4238.

Cambridge Dictionary (2017) Cambridge advanced learner's dictionary & thesaurus. 2017. https://dictionary.cambridge.org/dictionary/english.

Cameron, D.C. & Tong, I.-T. (1993) Cellular and metabolic engineering. Applied Biochemistry and Biotechnology. 38 (1), 105. doi:10.1007/BF02916416.

Canham, S. & Ohmann, C. (2016) A metadata schema for data objects in clinical research. Trials. 17, 1–11.

Carbonell, P. (2019) Metabolic Pathway Design: A Practical Guide. Learning Materials in Biosciences. Cham, Springer International Publishing. doi:10.1007/978-3-030-29865-4.

Carbonell, P., Jervis, A.J., Robinson, C.J., Yan, C., Dunstan, M., et al. (2018) An automated Design-Build-Test-Learn pipeline for enhanced microbial production of fine chemicals. Communications Biology. 1 (1), 66. doi:10.1038/s42003-018-0076-9.

Carbonell, P., Radivojevic, T. & García Martín, H. (2019) Opportunities at the Intersection of Synthetic Biology, Machine Learning, and Automation. ACS Synthetic Biology. 8 (7), 1474–1477. doi:10.1021/acssynbio.8b00540.

Cardinale, S. & Arkin, A.P. (2012) Contextualizing context for synthetic biology – identifying causes of failure of synthetic biological systems. Biotechnology Journal. 7 (7), 856–866. doi:10.1002/biot.201200085.

Carlson, R. (2003) The Pace and Proliferation of Biological Technologies. Biosecurity and Bioterrorism: Biodefense Strategy, Practice, and Science. 1 (3), 203–214. doi:10.1089/153871303769201851.

Casas, A., Bultelle, M., Motraghi, C. & Kitney, R. (2022) PASIV: A Pooled Approach-Based Workflow to Overcome Toxicity-Induced Design of Experiments Failures and Inefficiencies. ACS Synthetic Biology. doi:10.1021/acssynbio.1c00562.

Casini, A., Christodoulou, G., Freemont, P.S., Baldwin, G.S., Ellis, T. & MacDonald, J.T.

(2014) R2oDNA Designer: Computational Design of Biologically Neutral Synthetic DNA Sequences. ACS Synthetic Biology. 3 (8), 525–528. doi:10.1021/sb4001323.

Casini, A., Storch, M., Baldwin, G.S. & Ellis, T. (2015) Bricks and blueprints: methods and standards for DNA assembly. Nature Reviews Molecular Cell Biology. 16 (9), 568–576. doi:10.1038/nrm4014.

Cassiano, M.H.A. & Silva-Rocha, R. (2020) Benchmarking Bacterial Promoter Prediction Tools: Potentialities and Limitations. mSystems. 5 (4), e00439-20. doi:10.1128/mSystems.00439-20.

Cazier, A.P. & Blazeck, J. (2021) Advances in promoter engineering: Novel applications and predefined transcriptional control. Biotechnology Journal. 16 (10), 2100239. doi:10.1002/biot.202100239.

Chai, S.C., Goktug, A.N., Cui, J., Low, J., Chen, T., Chai, S.C., Goktug, A.N., Cui, J., Low, J. & Chen, T. (2013) Practical Considerations of Liquid Handling Devices in Drug Discovery. In: Drug Discovery. IntechOpen. p. doi:10.5772/52546.

Chao, R., Mishra, S., Si, T. & Zhao, H. (2017) Engineering biological systems using automated biofoundries. Metabolic Engineering. 42, 98–108. doi:10.1016/j.ymben.2017.06.003.

Chao, R., Yuan, Y. & Zhao, H. (2014) Recent advances in DNA assembly technologies. FEMS Yeast Research. n/a-n/a. doi:10.1111/1567-1364.12171.

Chapman, B. & Chang, J. (2000) Biopython: Python tools for computational biology. https://doi.org/10.1145/360262.360268.

Chemat-Djenni, Z., Ferhat, M.A., Tomao, V. & Chemat, F. (2010) Carotenoid extraction from tomato using a green solvent resulting from orange processing waste. Journal of Essential Oil Bearing Plants. 13 (2), 139–147.

Ciriminna, R., Fidalgo, A., Meneguzzo, F., Ilharco, L.M. & Pagliaro, M. (2016) Lycopene: Emerging Production Methods and Applications of a Valued Carotenoid. ACS Sustainable Chemistry & Engineering. 4 (3), 643–650. doi:10.1021/acssuschemeng.5b01516.

Clark, G. (2019) Regulation for the Fourth Industrial Revolution. June 2019. GOV.UK. https://www.gov.uk/government/publications/regulation-for-the-fourth-industrial-revolution/regulation-for-the-fourth-industrial-revolution [Accessed: 9 September 2023].

Clarke (2012) A Synthetic Biology Roadmap for the UK.

Clarke, L. & Kitney, R. (2020) Developing synthetic biology for industrial biotechnology applications. Biochemical Society Transactions. 48 (1), 113–122. doi:10.1042/BST20190349.

Clifford, B., Foster, I., Voeckler, J.-S., Wilde, M. & Zhao, Y. (2008) Tracking provenance in a

virtual data grid. Concurrency and Computation: Practice and Experience. 20 (5), 565–575. doi:10.1002/cpe.1256.

Clifton, K.P., Jones, E.M., Paudel, S., Marken, J.P., Monette, C.E., Halleran, A.D., Epp, L. & Saha, M.S. (2018) The genetic insulator RiboJ increases expression of insulated genes. Journal of Biological Engineering. 12 (1), 23. doi:10.1186/s13036-018-0115-6.

Codd, E.F. (1970) A relational model of data for large shared data banks. Communications of the ACM. 13 (6), 377–387. doi:10.1145/362384.362685.

Cohen, S.N., Chang, A.C., Boyer, H.W. & Helling, R.B. (1973) Construction of biologically functional bacterial plasmids in vitro. Proceedings of the National Academy of Sciences of the United States of America. 70 (11), 3240–3244. doi:10.1073/pnas.70.11.3240.

Cole, S.D., Beabout, K., Turner, K.B., Smith, Z.K., Funk, V.L., Harbaugh, S.V., Liem, A.T., Roth, P.A., Geier, B.A., Emanuel, P.A., Walper, S.A., Chávez, J.L. & Lux, M.W. (2019) Quantification of Interlaboratory Cell-Free Protein Synthesis Variability. ACS Synthetic Biology. 8 (9), 2080–2091. doi:10.1021/acssynbio.9b00178.

Committee on Realizing Opportunities for Advanced and Automated Workflows in Scientific Research, Board on Research Data and Information, Board on Mathematical Sciences and Analytics, Committee on Applied and Theoretical Statistics, Computer Science and Telecommunications Board, Division on Engineering and Physical Sciences, Policy and Global Affairs, & National Academies of Sciences, Engineering, and Medicine (2022) Automated Research Workflows for Accelerated Discovery: Closing the Knowledge Discovery Loop. Washington, D.C., National Academies Press. doi:10.17226/26532.

Committee on Reproducibility and Replicability in Science, Board on Behavioral, Cognitive, and Sensory Sciences, Committee on National Statistics, Division of Behavioral and Social Sciences and Education, Nuclear and Radiation Studies Board, Division on Earth and Life Studies, Board on Mathematical Sciences and Analytics, Committee on Applied and Theoretical Statistics, Division on Engineering and Physical Sciences, Board on Research Data and Information, Committee on Science, Engineering, Medicine, and Public Policy, Policy and Global Affairs, & National Academies of Sciences, Engineering, and Medicine (2019) Reproducibility and Replicability in Science. Washington, D.C., National Academies Press. doi:10.17226/25303.

Crusoe, M.R., Abeln, S., Iosup, A., Amstutz, P., Chilton, J., Tijanić, N., Ménager, H., Soiland-Reyes, S., Gavrilović, B., Goble, C. & Community, T.C. (2022) Methods included: standardizing computational reuse and portability with the Common Workflow Language. Communications of the ACM. 65 (6), 54–63. doi:10.1145/3486897.

De Win, B., Piessens, F., Joosen, W. & Verhanneman, T. (2002) On the importance of the

separation-of-concerns principle in secure software engineering. In: 2002 Citeseer. pp. 1–10.

Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P.J., Mayani, R., Chen, W., Ferreira da Silva, R., Livny, M. & Wenger, K. (2015) Pegasus, a workflow management system for science automation. Future Generation Computer Systems. 46, 17–35. doi:10.1016/j.future.2014.10.008.

Delépine, B., Duigou, T., Carbonell, P. & Faulon, J.-L. (2018) RetroPath2. 0: a retrosynthesis workflow for metabolic engineers. Metabolic engineering. 45, 158–170.

Densmore, D., Hillson, N.J., Klavins, E., Myers, C., Peccoud, J. & Stracquadanio, G. (2023) Introduction to the Special Issue on BioFoundries and Cloud Laboratories. ACM Journal on Emerging Technologies in Computing Systems. 19 (3), 1–2. doi:10.1145/3609485.

Design-Expert | Stat-Ease (2023) Design-Expert | Stat-Ease. 2023. https://www.statease.com/software/design-expert/ [Accessed: 17 September 2023].

Di Tommaso, P., Chatzou, M., Floden, E.W., Barja, P.P., Palumbo, E. & Notredame, C. (2017) Nextflow enables reproducible computational workflows. Nature biotechnology. 35 (4), 316–319.

Dietrich, J.A., McKee, A.E. & Keasling, J.D. (2010) High-Throughput Metabolic Engineering: Advances in Small-Molecule Screening and Selection. Annual Review of Biochemistry. 79 (1), 563–590. doi:10.1146/annurev-biochem-062608-095938.

Dijkstra, E.W. (1982) On the role of scientific thought. Selected writings on computing: a personal perspective. 60–66.

Dong, H., Ouyang, Z., Liu, J. & Jemal, M. (2006) The Use of a Dual Dye Photometric Calibration Method to Identify Possible Sample Dilution from an Automated Multichannel Liquid-handling System. JALA: Journal of the Association for Laboratory Automation. 11 (2), 60–64. doi:10.1016/j.jala.2006.02.005.

Duval, C. (1951) Francois Descroizilles, the inventor of volumetric analysis. Journal of Chemical Education. 28 (10), 508. doi:10.1021/ed028p508.

EFM Calculator (2021) EFM Calculator. 2021. https://barricklab.org/django/efm/.

Ellis, T., Adie, T. & Baldwin, G.S. (2011) DNA assembly for synthetic biology: from parts to pathways and beyond. Integrative Biology. 3 (2), 109–118. doi:10.1039/c0ib00070a.

Elowitz, M.B. & Leibler, S. (2000) A synthetic oscillatory network of transcriptional regulators. Nature. 403 (6767), 335–338. doi:10.1038/35002125.

Eppendorf (2021) 60 Years Pipette History. 2021. https://www.eppendorf.com/uploads/media/60Years_Pipette_History_Poster_EN_Web_10.pdf [Accessed: 16 August 2023].

Espah Borujeni, A., Channarasappa, A.S. & Salis, H.M. (2014) Translation rate is controlled by coupled trade-offs between site accessibility, selective RNA unfolding and sliding at upstream standby sites. Nucleic Acids Research. 42 (4), 2646–2659. doi:10.1093/nar/gkt1139.

Espah Borujeni, A., Zhang, J., Doosthosseini, H., Nielsen, A.A.K. & Voigt, C.A. (2020) Genetic circuit characterization by inferring RNA polymerase movement and ribosome usage. Nature Communications. 11 (1), 5001. doi:10.1038/s41467-020-18630-2.

Ewald, K. (2015) Influence of physical parameters on the dispensed volume of air-cushion pipette.

Exley, K., Reynolds, C.R., Suckling, L., Chee, S.M., Tsipa, A., Freemont, P.S., McClymont, D. & Kitney, R.I. (2019) Utilising datasheets for the informed automated design and build of a synthetic metabolic pathway. Journal of Biological Engineering. 13 (1), 8. doi:10.1186/s13036-019-0141-z.

Farzaneh, T. & Freemont, P.S. (2021) Biofoundries are a nucleating hub for industrial translation. Synthetic Biology. 6 (1), ysab013. doi:10.1093/synbio/ysab013.

Fernandes, D. & Bernardino, J. (2018) Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB: In: Proceedings of the 7th International Conference on Data Science, Technology and Applications. 2018 Porto, Portugal, SCITEPRESS - Science and Technology Publications. pp. 373–380. doi:10.5220/0006910203730380.

Fernández-Castané, A., Fehér, T., Carbonell, P., Pauthenier, C. & Faulon, J.-L. (2014) Computer-aided design for metabolic engineering. Journal of biotechnology. 192, 302–313.

Finnigan, L. & Toner, E. (2021) Building and Maintaining Metadata Aggregation Workflows Using Apache Airflow. https://scholarshare.temple.edu/handle/20.500.12613/6955.

Fleur, T.L., Hossain, A. & Salis, H.M. (2021) Automated Model-Predictive Design of Synthetic Promoters to Control Transcriptional Profiles in Bacteria.p.2021.09.01.458561. doi:10.1101/2021.09.01.458561.

Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., Plantikow, S., Rydberg, M., Selmer, P. & Taylor, A. (2018) Cypher: An Evolving Query Language for Property Graphs. In: Proceedings of the 2018 International Conference on Management of Data. SIGMOD '18. 27 May 2018 New York, NY, USA, Association for Computing Machinery. pp. 1433–1445. doi:10.1145/3183713.3190657.

Freire, J., Koop, D., Santos, E. & Silva, C. (2008) Provenance for Computational Tasks: A Survey. Computing in Science & Engineering. 10 (3), 11–21. doi:10.1109/MCSE.2008.79.

Frohm, J., Lindström, V. & Bellgran, M. (2005) A Model For Parallel Levels Of Automation Within Manufacturing.

Frohm, J., Lindström, V. & Winroth, M. et al (2008) Levels of Automation in Manufacturing.

Gallego-Jara, J., de Diego, T., del Real, Á., Écija-Conesa, A., Manjón, A. & Cánovas, M. (2015) Lycopene overproduction and in situ extraction in organic-aqueous culture systems using a metabolically engineered Escherichia coli. AMB Express. 5 (1), 65. doi:10.1186/s13568-015-0150-3.

Gallup, O., Ming, H. & Ellis, T. (2021) Ten future challenges for synthetic biology. Engineering Biology. 5 (3), 51–59. doi:10.1049/enb2.12011.

Gander, M.W., Vrana, J.D., Voje, W.E., Carothers, J.M. & Klavins, E. (2017) Digital logic circuits in yeast with CRISPR-dCas9 NOR gates. Nature Communications. 8 (1), 15459. doi:10.1038/ncomms15459.

Gardner, T.S., Cantor, C.R. & Collins, J.J. (2000) Construction of a genetic toggle switch in Escherichia coli. Nature. 403 (6767), 339–342. doi:10.1038/35002131.

Garenne, D., Haines, M.C., Romantseva, E.F., Freemont, P., Strychalski, E.A. & Noireaux, V. (2021) Cell-free gene expression. Nature Reviews Methods Primers. 1 (1), 1–18. doi:10.1038/s43586-021-00046-x.

Garg, S., Wang, S. & Ranjan, R. (2018) Orchestration Tools for Big Data.

Ge, Z., Song, Z., Ding, S.X. & Huang, B. (2017) Data Mining and Analytics in the Process Industry: The Role of Machine Learning. IEEE Access. 5, 20590–20616. doi:10.1109/ACCESS.2017.2756872.

Gerngross, D., Beerenwinkel, N. & Panke, S. (2022) Systematic investigation of synthetic operon designs enables prediction and control of expression levels of multiple proteins.p.2022.06.10.495604. doi:10.1101/2022.06.10.495604.

Gibbon, G.A. (1996) A brief history of LIMS. Laboratory Automation & Information Management. 32 (1), 1–5. doi:10.1016/1381-141X(95)00024-K.

Gibson, D.G., Young, L., Chuang, R.-Y., Venter, J.C., Hutchison, C.A. & Smith, H.O. (2009) Enzymatic assembly of DNA molecules up to several hundred kilobases. Nature Methods. 6 (5), 343–345. doi:10.1038/nmeth.1318.

Gilman, J. (2021) Statistical Design of Experiments for Synthetic Biology. ACS Synthetic Biology. 18.

Git (n.d.) Git. https://git-scm.com/ [Accessed: 26 February 2024].

GitHub (n.d.) GitHub: Let’s build from here. GitHub. https://github.com/ [Accessed: 26

February 2024].

Github topic (2023) Existing Workflow systems. 2023. Existing Workflow systems. https://github.com/common-workflow-language/common-workflow-language/wiki/Existing-Workflow-systems [Accessed: 1 September 2023].

GitHub topics design-of-experiments (2023) https://github.com/topics/design-of-experiments. 2023. https://github.com/topics/design-of-experiments. https://github.com [Accessed: 17 September 2023].

Github Workflow Engines list (2023) awesome-workflow-engines. 2023. Github Workflow Engines. http://meirwah.github.io/awesome-workflow-engines/ [Accessed: 27 November 2023].

Glavic, B. & Alonso, G. (2009) The perm provenance management system in action. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data. SIGMOD '09. 29 June 2009 New York, NY, USA, Association for Computing Machinery. pp. 1055–1058. doi:10.1145/1559845.1559980.

Glökler, J., Schütze, T. & Konthur, Z. (2010) Automation in the High-throughput Selection of Random Combinatorial Libraries—Different Approaches for Select Applications. Molecules. 15 (4), 2478–2490. doi:10.3390/molecules15042478.

Goecks, J., Nekrutenko, A., Taylor, J., & Galaxy Team team@ galaxyproject. org (2010a) Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome biology. 11, 1–13.

Goecks, J., Nekrutenko, A., Taylor, J., & The Galaxy Team (2010b) Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biology. 11 (8), R86. doi:10.1186/gb-2010-11-8-r86.

Goldman, R.P., Moseley, R., Roehner, N., Cummins, B., Vrana, J.D., et al. (2022) Highly-automated, high-throughput replication of yeast-based logic circuit design assessments. Synthetic Biology. 7 (1), ysac018. doi:10.1093/synbio/ysac018.

Grieves, M. (2014) Digital Twin: Manufacturing Excellence through Virtual Factory Replication. https://www.3ds.com/fileadmin/PRODUCTS-SERVICES/DELMIA/PDF/Whitepaper/DELMIA-APRISO-Digital-Twin-Whitepaper.pdf.

Grieves, M. & Vickers, J. (2017) Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In: F.-J. Kahlen, S. Flumerfelt, & A. Alves (eds.). Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches. Cham, Springer International Publishing. pp. 85–113. doi:10.1007/978-3-319-38756-7_4.

Guerra, A.S., Hoyos, C.G., Molina-Ramírez, C., Velásquez-Cock, J., Vélez, L., Gañán, P., Eceiza, A., Goff, H.D. & Zuluaga, R. (2021) Extraction and preservation of lycopene: A review of the advancements offered by the value chain of nanotechnology. Trends in Food Science & Technology. 116, 1120–1140. doi:10.1016/j.tifs.2021.09.009.

Guo, P.J. & Seltzer, M. (2012) BURRITO: Wrapping Your Lab Notebook in Computational Infrastructure. In: 2012 Boston, MA. p. https://www.usenix.org/conference/tapp12/workshop-program/presentation/guo.

Gurdo, N., Volke, D.C., McCloskey, D. & Nikel, P.I. (2023) Automating the design-build-test-learn cycle towards next-generation bacterial cell factories. New Biotechnology. 74, 1–15. doi:10.1016/j.nbt.2023.01.002.

Habich, T. & Beutel, S. (2024) Digitalization concepts in academic bioprocess development. Engineering in Life Sciences. 24 (4), 2300238. doi:10.1002/elsc.202300238.

Haines, M.C. (2021) LondonBiofoundry/basicsynbio. https://github.com/LondonBiofoundry/basicsynbio.

Haines, M.C., Carling, B., Marshall, J., Shenshin, V.A., Baldwin, G.S., Freemont, P. & Storch, M. (2022) basicsynbio and the BASIC SEVA collection: software and vectors for an established DNA assembly method. Synthetic Biology. 7 (1), ysac023. doi:10.1093/synbio/ysac023.

Haines, S. (2022) Workflow Orchestration with Apache Airflow. In: S. Haines (ed.). Modern Data Engineering with Apache Spark: A Hands-On Guide for Building Mission-Critical Streaming Applications. Berkeley, CA, Apress. pp. 255–295. doi:10.1007/978-1-4842-7452-1_8.

Hajarnavis, V. (2018) Process automation in the automotive industry. In: Instrument Engineers' Handbook, Volume 3. CRC Press. pp. 869–880.

Hamed, D., Keddari, S., Boufadi, M.Y. & Bessad, L. (2022) Lycopene Purification with DMSO anti-solvent: Optimization using Box-Behnken's experimental design and evaluation of the synergic effect between lycopene and Ammi visnaga.L essential oil. Chemical Papers. doi:10.1007/s11696-022-02302-0.

HamediRad, M., Chao, R., Weisberg, S., Lian, J., Sinha, S. & Zhao, H. (2019) Towards a fully automated algorithm driven platform for biosystems design. Nature Communications. 10 (1), 5150. doi:10.1038/s41467-019-13189-z.

Hammer, K., Mijakovic, I. & Jensen, P.R. (2006) Synthetic promoter libraries – tuning of gene expression. Trends in Biotechnology. 24 (2), 53–55. doi:10.1016/j.tibtech.2005.12.003.

Hanson, G. & Coller, J. (2018) Codon optimality, bias and usage in translation and mRNA

decay. Nature Reviews Molecular Cell Biology. 19 (1), 20–30. doi:10.1038/nrm.2017.91.

Harding, D., Bradford, D. & Brown, G. (2002) Transferring Industrial Automation Technology to the Laboratory. JALA: Journal of the Association for Laboratory Automation. 7 (2), 84–88. doi:10.1016/S1535-5535-04-00189-3.

Hérisson, J., Duigou, T., du Lac, M., Bazi-Kabbaj, K., Sabeti Azad, M., Buldum, G., Telle, O., El Moubayed, Y., Carbonell, P., Swainston, N., Zulkower, V., Kushwaha, M., Baldwin, G.S. & Faulon, J.-L. (2022) The automated Galaxy-SynBioCAD pipeline for synthetic biology design and engineering. Nature Communications. 13 (1), 5082. doi:10.1038/s41467-022-32661-x.

Hernández-Almanza, A., Montañez, J., Martínez, G., Aguilar-Jiménez, A., Contreras-Esquivel, J.C. & Aguilar, C.N. (2016) Lycopene: Progress in microbial production. Trends in Food Science & Technology. 56, 142–148. doi:10.1016/j.tifs.2016.08.013.

Hey, T., Tansley, S. & Tolle, K. (2009) Jim Gray on eScience: A Transformed Scientific Method.

Hillson, N., Caddick, M., Cai, Y., Carrasco, J.A., Chang, M.W., et al. (2019) Building a global alliance of biofoundries. Nature Communications. 10 (1), 2040. doi:10.1038/s41467-019-10079-2.

Holland, I. & Davies, J.A. (2020) Automation in the Life Science Research Laboratory. Frontiers in Bioengineering and Biotechnology. 8. https://www.frontiersin.org/articles/10.3389/fbioe.2020.571777.

Holowko, M.B., Frow, E.K., Reid, J.C., Rourke, M. & Vickers, C.E. (2021) Building a biofoundry. Synthetic Biology. 6 (1). doi:10.1093/synbio/ysaa026.

Hoshino, T. (2011) Violacein and related tryptophan metabolites produced by Chromobacterium violaceum: biosynthetic mechanism and pathway for construction of violacein core. Applied Microbiology and Biotechnology. 91 (6), 1463–1475. doi:10.1007/s00253-011-3468-z.

Howard, A.P., Slaughter, L.S., Carey, K.M. & Lillard, J.W. (2021) Bridges to biotechnology and bioentrepreneurship: improving diversity in the biotechnology sector. Nature Biotechnology. 39 (11), 1468–1474. doi:10.1038/s41587-021-01110-3.

Hua, S., De Matos, M.B. & Metselaar, J.M. (2018) Current trends and challenges in the clinical translation of nanoparticulate nanomedicines: pathways for translational development and commercialization. Frontiers in pharmacology. 9, 403086.

IBM (2023) What is a Workflow? | IBM. 2023. https://www.ibm.com/topics/workflow [Accessed: 9 November 2023].

IDT Codon Optimization Tool (2021) Codon Optimization Tool | IDT. 2021. Integrated DNA

Technologies. https://eu.idtdna.com/pages/tools/codon-optimization-tool.

iGEM Engineering Committee (2023) iGEM 2022 Distribution Handbook. 2023. https://technology.igem.org/distribution/2022-distribution.

Indri, M., Grau, A. & Ruderman, M. (2018) Guest editorial special section on recent trends and developments in industry 4.0 motivated robotic solutions. IEEE Transactions on Industrial Informatics. 14 (4), 1677–1680.

IUPAC Codes (n.d.) IUPAC Codes. https://www.bioinformatics.org/sms/iupac.html [Accessed: 28 September 2021].

Iverson, S.V., Haddock, T.L., Beal, J. & Densmore, D.M. (2016) CIDAR MoClo: Improved MoClo Assembly Standard and New E. coli Part Library Enable Rapid Combinatorial Design for Synthetic and Traditional Biology. ACS Synthetic Biology. 5 (1), 99–103. doi:10.1021/acssynbio.5b00124.

Jack, B.R., Leonard, S.P., Mishler, D.M., Renda, B.A., Leon, D., Suárez, G.A. & Barrick, J.E. (2015) Predicting the Genetic Stability of Engineered DNA Sequences with the EFM Calculator. ACS Synthetic Biology. 4 (8), 939–943. doi:10.1021/acssynbio.5b00068.

Jauer, M.-L. & Deserno, T.M. (2020) Data Provenance Standards and Recommendations for FAIR Data. In: Digital Personalized Health and Medicine. IOS Press. pp. 1237–1238. doi:10.3233/SHTI200380.

Jervis, A.J., Carbonell, P., Vinaixa, M., Dunstan, M.S., Hollywood, K.A., Robinson, C.J., Rattray, N.J., Yan, C., Swainston, N. & Currin, A. (2018) Machine learning of designed translational control allows predictive pathway optimization in Escherichia coli. ACS synthetic biology. 8 (1), 127–136.

Jeschek, M., Gerngross, D. & Panke, S. (2017) Combinatorial pathway optimization for streamlined metabolic engineering. Current Opinion in Biotechnology. 47, 142–151. doi:10.1016/j.copbio.2017.06.014.

Jessop-Fabre, M.M. & Sonnenschein, N. (2019) Improving Reproducibility in Synthetic Biology. Frontiers in Bioengineering and Biotechnology. 7, 18. doi:10.3389/fbioe.2019.00018.

Joshi, S.H.-N., Yong, C. & Gyorgy, A. (2022) Inducible plasmid copy number control for synthetic biology in commonly used E. coli strains. Nature Communications. 13 (1), 6691. doi:10.1038/s41467-022-34390-7.

Juchli, D. (2022) SiLA 2: The Next Generation Lab Automation Standard. In: S. Beutel & F. Lenk (eds.). Smart Biolabs of the Future. Advances in Biochemical Engineering/Biotechnology. Cham, Springer International Publishing. pp. 147–174. doi:10.1007/10_2022_204.

Jung, J., Lim, J.H., Kim, S.Y., Im, D.-K., Seok, J.Y., Lee, S.-J.V., Oh, M.-K. & Jung, G.Y. (2016) Precise precursor rebalancing for isoprenoids production by fine control of gapA expression in Escherichia coli. Metabolic Engineering. 38, 401–408. doi:10.1016/j.ymben.2016.10.003.

Kaber, D.B., Stoll, N., Thurow, K., Green, R.S., Kim, S.-H. & Mosaly, P. (2009) Human–automation interaction strategies and models for life science applications. Human Factors and Ergonomics in Manufacturing & Service Industries. 19 (6), 601–621. doi:10.1002/hfm.20156.

Kagermann, H., Lukas, W.-D. & Wahlster, W. (2011) Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution. VDI nachrichten. 13 (1), 2–3.

Kahl, L.J. & Endy, D. (2013) A survey of enabling technologies in synthetic biology. Journal of Biological Engineering. 7 (1), 13. doi:10.1186/1754-1611-7-13.

Kahn, M.G., Brown, J.S., Chun, A.T., Davidson, B.N., Meeker, D., Ryan, P.B., Schilling, L.M., Weiskopf, N.G., Williams, A.E. & Zozus, M.N. (2015) Transparent reporting of data quality in distributed data networks. Egems. 3 (1).

Kalidindi, S.R., Buzzy, M., Boyce, B.L. & Dingreville, R. (2022) Digital Twins for Materials. Frontiers in Materials. 9. https://www.frontiersin.org/articles/10.3389/fmats.2022.818535.

Kelly, J.R., Rubin, A.J., Davis, J.H., Ajo-Franklin, C.M., Cumbers, J., Czar, M.J., de Mora, K., Glieberman, A.L., Monie, D.D. & Endy, D. (2009) Measuring the activity of BioBrick promoters using an in vivo reference standard. Journal of Biological Engineering. 3 (1), 4. doi:10.1186/1754-1611-3-4.

Kelwick, R., MacDonald, J.T., Webb, A.J. & Freemont, P. (2014) Developments in the tools and methodologies of synthetic biology. Frontiers in Bioengineering and Biotechnology. 2, 60. doi:10.3389/fbioe.2014.00060.

Kempf, R. (2017) Open Source Data Pipeline – Luigi vs Azkaban vs Oozie vs Airflow - Bizety: Research & Consulting. https://www.bizety.com/2017/06/05/open-source-data-pipeline-luigi-vs-azkaban-vs-oozie-vs-airflow/.

Khastsaev, B.D. (1993) Single channel and multichannel adjustable micropipettes. Biomedical Engineering. 27 (5), 285–291. doi:10.1007/BF00557230.

Kim, G.B., Choi, S.Y., Cho, I.J., Ahn, D.-H. & Lee, S.Y. (2023) Metabolic engineering for sustainability and health. Trends in Biotechnology. 41 (3), 425–451. doi:10.1016/j.tibtech.2022.12.014.

Kim, G.B., Kim, W.J., Kim, H.U. & Lee, S.Y. (2020) Machine learning applications in systems metabolic engineering. Current Opinion in Biotechnology. 64, 1–9.

doi:10.1016/j.copbio.2019.08.010.

Kim, M.J., Noh, M.H., Woo, S., Lim, H.G. & Jung, G.Y. (2019) Enhanced Lycopene Production in Escherichia coli by Expression of Two MEP Pathway Enzymes from Vibrio sp. Dhg. Catalysts. 9 (12), 1003. doi:10.3390/catal9121003.

Kim, Y.-S., Lee, J.-H., Kim, N.-H., Yeom, S.-J., Kim, S.-W. & Oh, D.-K. (2011) Increase of lycopene production by supplementing auxiliary carbon sources in metabolically engineered Escherichia coli. Applied Microbiology and Biotechnology. 90 (2), 489–497. doi:10.1007/s00253-011-3091-z.

King, R. (2011) Logic, Automation, and the Future of Biology. In: Proceedings of The Sophia-Antipolis Spring School on Modelling Complex Biological Systems in the Context of Genomics. 27 2011 Sophia-Antipolis. p. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ecc5fd9d983410f065d1aee58f518a173c6bc9ae#page=91.

Kitano, S., Lin, C., Foo, J.L. & Chang, M.W. (2023) Synthetic biology: Learning the way toward high-precision biological design. PLOS Biology. 21 (4), e3002116. doi:10.1371/journal.pbio.3002116.

Kitney, R., Adeogun, M., Fujishima, Y., Goñi-Moreno, Á., Johnson, R., Maxon, M., Steedman, S., Ward, S., Winickoff, D. & Philp, J. (2019) Enabling the Advanced Bioeconomy through Public Policy Supporting Biofoundries and Engineering Biology. Trends in Biotechnology. 37 (9), 917–920. doi:10.1016/j.tibtech.2019.03.017.

Klingenberg, M. (2005) When a common problem meets an ingenious mind. EMBO reports. 6 (9), 797–800. doi:10.1038/sj.embor.7400520.

Knight, T. (2003) Idempotent Vector Design for Standard Assembly of Biobricks: doi:10.21236/ADA457791.

Ko, Y.-S., Woong Kim, J., An Lee, J., Han, T., Bae Kim, G., Eum Park, J. & Yup Lee, S. (2020) Tools and strategies of systems metabolic engineering for the development of microbial cell factories for chemical production. Chemical Society Reviews. 49 (14), 4615–4636. doi:10.1039/D0CS00155D.

Kong, F., Yuan, L., Zheng, Y.F. & Chen, W. (2012) Automatic Liquid Handling for Life Science: A Critical Review of the Current State of the Art. Journal of Laboratory Automation. 17 (3), 169–185. doi:10.1177/2211068211435302.

Labcyte (2013) Echo Liquid Handler User Manual.

Laohakunakorn, N., Grasemann, L., Lavickova, B., Michielin, G., Shahein, A., Swank, Z. & Maerkl, S.J. (2020) Bottom-Up Construction of Complex Biomolecular Systems With Cell-Free Synthetic Biology. Frontiers in Bioengineering and Biotechnology. 8.

https://www.frontiersin.org/articles/10.3389/fbioe.2020.00213.

Laubenbacher, R., Sluka, J.P. & Glazier, J.A. (2021) Using digital twins in viral infection. Science. 371 (6534), 1105–1106. doi:10.1126/science.abf3370.

Lavickova, B. & Maerkl, S.J. (n.d.) A simple, robust, and low-cost method to produce the PURE cell-free system. 32.

Lee, M.E., Aswani, A., Han, A.S., Tomlin, C.J. & Dueber, J.E. (2013) Expression-level optimization of a multi-enzyme pathway in the absence of a high-throughput assay. Nucleic Acids Research. 41 (22), 10668–10678. doi:10.1093/nar/gkt809.

Liew, C.S., Atkinson, M.P., Galea, M., Ang, T.F., Martin, P. & Hemert, J.I.V. (2016) Scientific Workflows: Moving Across Paradigms. ACM Computing Surveys. 49 (4), 66:1-66:39. doi:10.1145/3012429.

Linshiz, G., Stawski, N., Poust, S., Bi, C., Keasling, J.D. & Hillson, N.J. (2013) PaR-PaR Laboratory Automation Platform. ACS Synthetic Biology. 2 (5), 216–222. doi:10.1021/sb300075t.

Lipsh-Sokolik, R., Khersonsky, O., Schröder, S.P., de Boer, C., Hoch, S.-Y., Davies, G.J., Overkleeft, H.S. & Fleishman, S.J. (2023) Combinatorial assembly and design of enzymes. Science. 379 (6628), 195–201. doi:10.1126/science.ade9434.

Liu, N. & Stephanopoulos, G. (2021) Metabolic Engineering Perspectives. Metabolic Engineering: Concepts and Applications. 13, 1–21.

Lobban, P.E. & Kaiser, A.D. (1973) Enzymatic end-to-end joining of DNA molecules. Journal of Molecular Biology. 78 (3), 453–471. doi:10.1016/0022-2836(73)90468-3.

Long, M.R., Ong, W.K. & Reed, J.L. (2015) Computational methods in metabolic engineering for strain design. Current opinion in biotechnology. 34, 135–141.

de Lorenzo, V., Krasnogor, N. & Schmidt, M. (2021) For the sake of the Bioeconomy: define what a Synthetic Biology Chassis is! New Biotechnology. 60, 44–51. doi:10.1016/j.nbt.2020.08.004.

Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J. & Zhao, Y. (2006) Scientific workflow management and the Kepler system. Concurrency and Computation: Practice and Experience. 18 (10), 1039–1065. doi:10.1002/cpe.994.

Lux, M.W., Strychalski, E.A. & Vora, G.J. (2023) Advancing reproducibility can ease the ‘hard truths’ of synthetic biology. Synthetic Biology. 8 (1), ysad014. doi:10.1093/synbio/ysad014.

Lv, X., Gu, J., Wang, F., Xie, W., Liu, M., Ye, L. & Yu, H. (2016) Combinatorial pathway optimization in Escherichia coli by directed co-evolution of rate-limiting enzymes and modular pathway engineering. Biotechnology and Bioengineering. 113 (12), 2661–2669.

doi:10.1002/bit.26034.

MacDonald, J.T. & Siciliano, V. (2017) Computational Sequence Design with R2oDNA Designer. In: D. Gould (ed.). Mammalian Synthetic Promoters. Methods in Molecular Biology. New York, NY, Springer. pp. 249–262. doi:10.1007/978-1-4939-7223-4_18.

Magagna, B., Goldfarb, D., Martin, P., Atkinson, M., Koulouzis, S. & Zhao, Z. (2020) Data Provenance. In: Z. Zhao & M. Hellström (eds.). Towards Interoperable Research Infrastructures for Environmental and Earth Sciences: A Reference Model Guided Approach for Common Challenges. Cham, Springer International Publishing. pp. 208–225. doi:10.1007/978-3-030-52829-4_12.

Malo, N., Hanley, J.A., Cerquozzi, S., Pelletier, J. & Nadon, R. (2006) Statistical practice in high-throughput screening data analysis. Nature Biotechnology. 24 (2), 167–175. doi:10.1038/nbt1186.

Markossian, S., Grossman, A., Brimacombe, K. & Arkin, M. (2004) Assay Guidance Manual. Bethesda (MD), Eli Lilly & Company and the National Center for Advancing Translational Sciences. http://www.ncbi.nlm.nih.gov/books/NBK53196/.

Masdari, M., ValiKardan, S., Shahi, Z. & Azar, S.I. (2016) Towards workflow scheduling in cloud computing: A comprehensive analysis. Journal of Network and Computer Applications. 66, 64–82. doi:10.1016/j.jnca.2016.01.018.

Matosin, N., Frank, E., Engel, M., Lum, J.S. & Newell, K.A. (2014) Negativity towards negative results: a discussion of the disconnect between scientific worth and scientific culture. The Company of Biologists Limited.

Matsubayashi, H. & Ueda, T. (2014) Purified cell-free systems as standard parts for synthetic biology. Current Opinion in Chemical Biology. 22, 158–162. doi:10.1016/j.cbpa.2014.09.031.

McLaughlin, J.A., Myers, C.J., Zundel, Z., Mısırlı, G., Zhang, M., Ofiteru, I.D., Goñi-Moreno, A. & Wipat, A. (2018) SynBioHub: A Standards-Enabled Design Repository for Synthetic Biology. ACS Synthetic Biology. 7 (2), 682–688. doi:10.1021/acssynbio.7b00403.

McNutt, M. (2014) Reproducibility. Science. doi:10.1126/science.1250475.

Méndez-Carmona, J.Y., Hernández-Almanza, A. & Aguilar, C.N. (2022) Microbial Production of Lycopene. In: S.M. Jafari & F.D. Harzevili (eds.). Microbial Production of Food Bioactive Compounds. Cham, Springer International Publishing. pp. 1–26. doi:10.1007/978-3-030-81403-8_11-1.

Mendi, A.F. (2022) A Digital Twin Case Study on Automotive Production Line. Sensors. 22 (18), 6963. doi:10.3390/s22186963.

Minitab (2023) Data Analysis, Statistical & Process Improvement Tools | Minitab. 2023.

https://www.minitab.com/en-us/ [Accessed: 17 September 2023].

Misawa, N., Nakagawa, M., Kobayashi, K., Yamano, S., Izawa, Y., Nakamura, K. & Harashima, K. (1990) Elucidation of the Erwinia uredovora carotenoid biosynthetic pathway by functional analysis of gene products expressed in Escherichia coli. Journal of Bacteriology. 172 (12), 6704–6712. doi:10.1128/JB.172.12.6704-6712.1990.

Mlinarić, A., Horvat, M. & Šupak Smolčić, V. (2017) Dealing with the positive publication bias: Why you should really publish your negative results. Biochemia medica. 27 (3), 447–452.

Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E. & den Bussche, J.V. (2011) The Open Provenance Model core specification (v1.1). Future Generation Computer Systems. 27 (6), 743–756. doi:10.1016/j.future.2010.07.005.

Moreau, L., Freire, J., Futrelle, J., McGrath, R.E., Myers, J. & Paulson, P. (2008) The Open Provenance Model: An Overview. In: J. Freire, D. Koop, & L. Moreau (eds.). Provenance and Annotation of Data and Processes. 2008 Berlin, Heidelberg, Springer. pp. 323–326. doi:10.1007/978-3-540-89965-5_31.

Moyne, J., Samantaray, J. & Armacost, M. (2016) Big Data Capabilities Applied to Semiconductor Manufacturing Advanced Process Control. IEEE Transactions on Semiconductor Manufacturing. 29 (4), 283–291. doi:10.1109/TSM.2016.2574130.

Mutalik, V.K., Guimaraes, J.C., Cambray, G., Lam, C., Christoffersen, M.J., Mai, Q.-A., Tran, A.B., Paull, M., Keasling, J.D., Arkin, A.P. & Endy, D. (2013a) Precise and reliable gene expression via standard transcription and translation initiation elements. Nature Methods. 10 (4), 354–360. doi:10.1038/nmeth.2404.

Mutalik, V.K., Guimaraes, J.C., Cambray, G., Mai, Q.-A., Christoffersen, M.J., Martin, L., Yu, A., Lam, C., Rodriguez, C., Bennett, G., Keasling, J.D., Endy, D. & Arkin, A.P. (2013b) Quantitative estimation of activity and quality for collections of functional genetic elements. Nature Methods. 10 (4), 347–353. doi:10.1038/nmeth.2403.

Najmabadi, P., Goldenberg, A.A. & Emili, A. (2006) Hardware Flexibility of Laboratory Automation Systems: Analysis and New Flexible Automation Architectures. JALA: Journal of the Association for Laboratory Automation. 11 (4), 203–216. doi:10.1016/j.jala.2006.05.014.

New England Biolabs (2020) PCR Protocol for Phusion High-Fidelity DNA Polymerase (M0530) v3. doi:10.17504/protocols.io.bdd3i28n.

Ng, N.S., Gately, R. & Ooi, L. (2021) Automated liquid handling for microplate assays : a simplified user interface for the Hamilton Microlab STAR. Journal of Applied Bioanalysis. 11–18. doi:10.17145/jab.21.003.

Nishii, R. (1993) Optimality of experimental designs. Discrete Mathematics. 116 (1), 209–225. doi:10.1016/0012-365X(93)90402-F.

Nishizaki, T., Tsuge, K., Itaya, M., Doi, N. & Yanagawa, H. (2007) Metabolic Engineering of Carotenoid Biosynthesis in Escherichia coli by Ordered Gene Assembly in Bacillus subtilis. Applied and Environmental Microbiology. 73 (4), 1355–1361. doi:10.1128/AEM.02268-06.

Norton, N.M. & Fischer, K.J. (2023) Effects of micropipette handle diameter and inclusion of finger rest on basilar thumb joint contact mechanics. Medical Engineering & Physics. 111, 103940. doi:10.1016/j.medengphy.2022.103940.

Notman, R., Noro, M., O'Malley, B. & Anwar, J. (2006) Molecular Basis for Dimethylsulfoxide (DMSO) Action on Lipid Membranes. Journal of the American Chemical Society. 128 (43), 13982–13983. doi:10.1021/ja063363t.

Ochs, J., Barry, F., Schmitt, R. & Murphy, J.M. (2017) Advances in automation for the production of clinical-grade mesenchymal stromal cells: the AUTOSTEM robotic platform. Cell and Gene Therapy Insights. 3 (8), 739–748. doi:10.18609/cgti.2017.073.

O'Connor, B.D., Yuen, D., Chung, V., Duncan, A.G., Liu, X.K., Patricia, J., Paten, B., Stein, L. & Ferretti, V. (2017) The Dockstore: enabling modular, community-focused sharing of Docker-based genomics tools and workflows. F1000Research. 6.

Oinn, T., Greenwood, M., Addis, M., Alpdemir, M.N., Ferris, J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M.R., Senger, M., Stevens, R., Wipat, A. & Wroe, C. (2006) Taverna: lessons in creating a workflow environment for the life sciences. Concurrency and Computation: Practice and Experience. 18 (10), 1067–1100. doi:10.1002/cpe.993.

de Oliveira, W., Braga, R., David, J.M.N., Stroele, V., Campos, F. & Castro, G. (2022) Visionary: a framework for analysis and visualization of provenance data. Knowledge and Information Systems. 64 (2), 381–413. doi:10.1007/s10115-021-01645-6.

Oliver, H., Shin, M., Matthews, D., Sanders, O., Bartholomew, S., Clark, A., Fitzpatrick, B., van Haren, R., Hut, R. & Drost, N. (2019) Workflow Automation for Cycling Systems. Computing in Science & Engineering. 21 (4), 7–21. doi:10.1109/MCSE.2019.2906593.

Oliver, H.J., Shin, M. & Sanders, O. (2018) Cylc: A Workflow Engine for Cycling Systems. Journal of Open Source Software. 3 (27), 737. doi:10.21105/joss.00737.

openCypher Implementers Group (2017) openCypher · openCypher. 2017. https://opencypher.org/ [Accessed: 21 August 2023].

Otero-Muras, I. & Carbonell, P. (2021) Automated engineering of synthetic metabolic pathways for efficient biomanufacturing. Metabolic Engineering. 63, 61–80.

doi:10.1016/j.ymben.2020.11.012.

Park, W.-J., You, S.-H., Choi, H.-A., Chu, Y.-J. & Kim, G.-J. (2015) Over-expression of recombinant proteins with N-terminal His-tag via subcellular uneven distribution in Escherichia coli. Acta Biochimica et Biophysica Sinica. 47 (7), 488–495. doi:10.1093/abbs/gmv036.

Parnas, D.L. (1972) On the criteria to be used in decomposing systems into modules. Communications of the ACM. 15 (12), 1053–1058.

parts.igem.org (n.d.) parts.igem.org. https://parts.igem.org/Main_Page [Accessed: 18 May 2023].

Peccoud, J. (2016) Synthetic Biology : fostering the cyber-biological revolution. Synthetic Biology. 1 (1), ysw001. doi:10.1093/synbio/ysw001.

Peccoud, J., Johnson, D., Peccoud, S., Setchell, J. & Zhou, W. (2022) Organizing laboratory information to analyze the reproducibility of experimental workflows. doi:10.1101/2022.04.05.487214.

Peltz, C. (2003) Web services orchestration and choreography. Computer. 36 (10), 46–52. doi:10.1109/MC.2003.1236471.

Pérez, B., Rubio, J. & Sáenz-Adán, C. (2018) A systematic review of provenance systems. Knowledge and Information Systems. 57 (3), 495–543. doi:10.1007/s10115-018-1164-3.

Plesser, H.E. (2018) Reproducibility vs. Replicability: A Brief History of a Confused Terminology. Frontiers in Neuroinformatics. 11. https://www.frontiersin.org/articles/10.3389/fninf.2017.00076.

Polizzi, K.M. & Kontoravdi, C. (2013) Synthetic Biology. Methods in Molecular Biology. Totowa, NJ, Humana Press. doi:10.1007/978-1-62703-625-2.

Pope, D.C. & Oliver, W.T. (1966) Dimethyl sulfoxide (DMSO). Canadian Journal of Comparative Medicine and Veterinary Science. 30 (1), 3–8.

Porr, M., Lange, F., Marquard, D., Niemeyer, L., Lindner, P., Scheper, T. & Beutel, S. (2021) Implementing a digital infrastructure for the lab using a central laboratory server and the SiLA2 communication standard. Engineering in Life Sciences. 21 (3–4), 208–219. doi:10.1002/elsc.202000053.

Porr, M., Schwarz, S., Lange, F., Niemeyer, L., Hentrop, T., Marquard, D., Lindner, P., Scheper, T. & Beutel, S. (2020) Bringing IoT to the Lab: SiLA2 and Open-Source-Powered Gateway Module for Integrating Legacy Devices into the Digital Laboratory. HardwareX. 8, e00118. doi:10.1016/j.ohx.2020.e00118.

Prasad, P.J. & Bodhe, G.L. (2012) Trends in laboratory information management system.

Chemometrics and Intelligent Laboratory Systems. 118, 187–192. doi:10.1016/j.chemolab.2012.07.001.

Qi, Q. & Tao, F. (2018) Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison. IEEE Access. 6, 3585–3593. doi:10.1109/ACCESS.2018.2793265.

Quaglia, D. (2017) Antha: a platform for engineering biology. The Official PLOS Blog. https://theplosblog.plos.org/2017/02/antha-a-platform-for-engineering-biology/.

Radivojević, T., Costello, Z., Workman, K. & Garcia Martin, H. (2020) A machine learning Automated Recommendation Tool for synthetic biology. Nature Communications. 11 (1), 4879. doi:10.1038/s41467-020-18008-4.

Rafiq, Q.A. & Thomas, R.J. (2016) The evolving role of automation in process development and manufacture of cell and gene-based therapies.

Raj, P. & Surianarayanan, C. (2020) Chapter Twelve - Digital twin: The industry use cases. In: P. Raj & P. Evangeline (eds.). Advances in Computers. The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases. Elsevier. pp. 285–320. doi:10.1016/bs.adcom.2019.09.006.

Rajakumar, P.D., Gowers, G.-O.F., Suckling, L., Foster, A., Ellis, T., Kitney, R.I., McClymont, D.W. & Freemont, P.S. (2019) Rapid Prototyping Platform for Saccharomyces cerevisiae Using Computer-Aided Genetic Design Enabled by Parallel Software and Workcell Platform Development. SLAS TECHNOLOGY: Translating Life Sciences Innovation. 24 (3), 291–297. doi:10.1177/2472630318798304.

Raptis, T.P., Passarella, A. & Conti, M. (2019) Data Management in Industry 4.0: State of the Art and Open Challenges. IEEE Access. 7, 97052–97093. doi:10.1109/ACCESS.2019.2929296.

Rauber, A., Asmi, A., van Uytvanck, D. & Proell, S. (2015) Data citation of evolving data: Recommendations of the Working Group on Data Citation (WGDC). Result of the RDA Data Citation WG. 20, 1–2.

RBS Calculator (2021) RBS Calculator. 2021. https://www.denovodna.com/software/predict_rbs_calculator.

Recupito, G., Pecorelli, F., Catolino, G., Moreschini, S., Nucci, D.D., Palomba, F. & Tamburri, D.A. (2022) A Multivocal Literature Review of MLOps Tools and Features. In: 2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). August 2022 pp. 84–91. doi:10.1109/SEAA56994.2022.00021.

Reis, A.C. & Salis, H.M. (2020) An Automated Model Test System for Systematic

Development and Improvement of Gene Expression Models. ACS Synthetic Biology. 9 (11), 3145–3156. doi:10.1021/acssynbio.0c00394.

Roehner, N., Mante, J., Myers, C.J. & Beal, J. (2021) Synthetic Biology Curation Tools (SYNBICT). ACS Synthetic Biology. 10 (11), 3200–3204. doi:10.1021/acssynbio.1c00220.

Røkke, G., Korvald, E., Pahr, J., Øyås, O. & Lale, R. (2014) BioBrick Assembly Standards and Techniques and Associated Software Tools. In: S. Valla & R. Lale (eds.). DNA Cloning and Assembly Methods. Methods in Molecular Biology. Totowa, NJ, Humana Press. pp. 1–24. doi:10.1007/978-1-62703-764-8_1.

Rugbjerg, P. & Sommer, M.O.A. (2019) Overcoming genetic heterogeneity in industrial fermentations. Nature Biotechnology. 37 (8), 869–876. doi:10.1038/s41587-019-0171-6.

Saeid, A., Eun, J.B., Sagor, Md.S.A., Rahman, A., Akter, Mst.S. & Ahmed, M. (2016) Effects of Extraction and Purification Methods on Degradation Kinetics and Stability of Lycopene from Watermelon under Storage Conditions: Effects of extraction and purification…. Journal of Food Science. 81 (11), C2630–C2638. doi:10.1111/1750-3841.13504.

Salis, H.M., Mirsky, E.A. & Voigt, C.A. (2009) Automated design of synthetic ribosome binding sites to control protein expression. Nature Biotechnology. 27 (10), 946–950. doi:10.1038/nbt.1568.

SAS (2019) JMP®, Version 15. SAS Institute Inc., Cary, NC, 1989-2019. https://www.jmp.com/.

Schwab, K. (2015) The Fourth Industrial Revolution. Foreign Affairs. https://www.foreignaffairs.com/world/fourth-industrial-revolution.

Schwartz, C., Frogue, K., Misa, J. & Wheeldon, I. (2017) Host and Pathway Engineering for Enhanced Lycopene Biosynthesis in Yarrowia lipolytica. Frontiers in Microbiology. 8, 2233. doi:10.3389/fmicb.2017.02233.

Scott, K.J. (2001) Detection and measurement of carotenoids by UV/VIS spectrophotometry. Current protocols in food analytical chemistry. (1), F2. 2.1-F2. 2.10.

Sevgili, A. & Erkmen, O. (2019) Improved Lycopene Production from Different Substrates by Mated Fermentation of Blakeslea Trispora. Foods. 8 (4), 120. doi:10.3390/foods8040120.

Shendure, J., Balasubramanian, S., Church, G.M., Gilbert, W., Rogers, J., Schloss, J.A. & Waterston, R.H. (2017) DNA sequencing at 40: past, present and future. Nature. 550 (7676), 345–353. doi:10.1038/nature24286.

Shi, J., Le Maguer, M., BRYAN, M. & KAKUDA, Y. (2003) Kinetics of lycopene degradation in tomato puree by heat and light irradiation. Journal of Food Process Engineering. 25 (6), 485–498.

Shields, M. (2007) Control- Versus Data-Driven Workflows. In: I.J. Taylor, E. Deelman, D.B. Gannon, & M. Shields (eds.). Workflows for e-Science. London, Springer London. pp. 167–173. doi:10.1007/978-1-84628-757-2_11.

Sikkema, J., de Bont, J.A. & Poolman, B. (1995) Mechanisms of membrane toxicity of hydrocarbons. Microbiological reviews. 59 (2), 201–222. doi:10.1128/MMBR.59.2.201-222.1995.

Singh, R., Graves, J.A., Anantharaj, V. & Sukumar, S.R. (2019) Evaluating Scientific Workflow Engines for Data and Compute Intensive Discoveries. In: 2019 IEEE International Conference on Big Data (Big Data). December 2019 pp. 4553–4560. doi:10.1109/BigData47090.2019.9006223.

Singleton, C., Gilman, J., Rollit, J., Zhang, K., Parker, D.A. & Love, J. (2019) A design of experiments approach for the rapid formulation of a chemically defined medium for metabolic profiling of industrially important microbes. PLOS ONE. 14 (6), e0218208. doi:10.1371/journal.pone.0218208.

Sinha, A. & Mann, M. (2020) A beginner’s guide to mass spectrometry–based proteomics. The Biochemist. 42 (5), 64–69. doi:10.1042/BIO20200057.

Sjöberg, G., Guevara-Martínez, M., van Maris, A.J.A. & Gustavsson, M. (2019) Metabolic engineering applications of the Escherichia coli bacterial artificial chromosome. Journal of Biotechnology. 305, 43–50. doi:10.1016/j.jbiotec.2019.09.002.

Smanski, M.J., Bhatia, S., Zhao, D., Park, Y., B A Woodruff, L., Giannoukos, G., Ciulla, D., Busby, M., Calderon, J., Nicol, R., Gordon, D.B., Densmore, D. & Voigt, C.A. (2014) Functional optimization of gene clusters by combinatorial design and assembly. Nature Biotechnology. 32 (12), 1241–1249. doi:10.1038/nbt.3063.

Smucker, B., Krzywinski, M. & Altman, N. (2018) Optimal experimental design. Nature Methods. 15 (8), 559–560. doi:10.1038/s41592-018-0083-2.

Spice, A.J., Aw, R., Bracewell, D.G. & Polizzi, K.M. (2020) Improving the reaction mix of a Pichia pastoris cell-free system using a design of experiments approach to minimise experimental effort. Synthetic and Systems Biotechnology. 5 (3), 137–144. doi:10.1016/j.synbio.2020.06.003.

Srivastava, S. & Srivastava, A.K. (2015) Lycopene; chemistry, biosynthesis, metabolism and degradation under various abiotic parameters. Journal of Food Science and Technology. 52 (1), 41–53. doi:10.1007/s13197-012-0918-2.

Stephanopoulos, G. & Sinskey, A.J. (1993) Metabolic engineering—methodologies and future prospects. Trends in biotechnology. 11 (9), 392–396.

Stephenson, A., Lastra, L., Nguyen, B., Chen, Y.-J., Nivala, J., Ceze, L. & Strauss, K. (2023) Physical Laboratory Automation in Synthetic Biology. ACS Synthetic Biology. 12 (11), 3156–3169. doi:10.1021/acssynbio.3c00345.

Stitz, H., Luger, S., Streit, M. & Gehlenborg, N. (2016) AVOCADO: Visualization of Workflow–Derived Data Provenance for Reproducible Biomedical Research. Computer Graphics Forum. 35 (3), 481–490. doi:10.1111/cgf.12924.

Storch, M., Casini, A., Mackrow, B., Fleming, T., Trewhitt, H., Ellis, T. & Baldwin, G.S. (2015) BASIC: A New Biopart Assembly Standard for Idempotent Cloning Provides Accurate, Single-Tier DNA Assembly for Synthetic Biology. ACS Synthetic Biology. 4 (7), 781–787. doi:10.1021/sb500356d.

Storch, M., Haines, M.C. & Baldwin, G.S. (2020) DNA-BOT: a low-cost, automated DNA assembly platform for synthetic biology. Synthetic Biology. 5 (1), ysaa010. doi:10.1093/synbio/ysaa010.

Subramanian, K. (2020) Digital Twin for Drug Discovery and Development—The Virtual Liver. Journal of the Indian Institute of Science. 100 (4), 653–662. doi:10.1007/s41745-020-00185-2.

Suckling, L., McClymont, D. & Hirst, C. (2017) Automated High-throughput Transformation of Bacteria Cells. Analytik Jena AG.

Sun, D., Wu, L. & Fan, G. (2021) Laboratory information management system for biosafety laboratory: Safety and efficiency. Journal of Biosafety and Biosecurity. 3 (1), 28–34. doi:10.1016/j.jobb.2021.03.001.

Sun, T., He, X. & Li, Z. (2023) Digital twin in healthcare: Recent updates and challenges. DIGITAL HEALTH. 9, 20552076221149651. doi:10.1177/20552076221149651.

Szabadváry, F. (1978) Joseph Louis Gay-Lussac (1778–1850) and analytical chemistry. Talanta. 25 (11), 611–617. doi:10.1016/0039-9140(78)80162-3.

Takehara, M., Nishimura, M., Kuwa, T., Inoue, Y., Kitamura, C., Kumagai, T. & Honda, M. (2014) Characterization and Thermal Isomerization of (all-E)-Lycopene. Journal of Agricultural and Food Chemistry. 62 (1), 264–269. doi:10.1021/jf404497k.

Taylor, G.M. & Heap, J.T. (2020) Combinatorial metabolic engineering platform enabling stable overproduction of lycopene from carbon dioxide by cyanobacteria. doi:10.1101/2020.03.11.983833.

Taylor, G.M., Hitchcock, A. & Heap, J.T. (2021) Combinatorial assembly platform enabling engineering of genetically stable metabolic pathways in cyanobacteria. Nucleic Acids Research. 49 (21), e123. doi:10.1093/nar/gkab791.

Taylor, G.M., Mordaka, P.M. & Heap, J.T. (2019) Start-Stop Assembly: a functionally scarless DNA assembly system optimized for metabolic engineering. Nucleic Acids Research. 47 (3), e17–e17. doi:10.1093/nar/gky1182.

Team Imperial College (2020) iGEM 2020 Team Imperial College. 2020. https://2020.igem.org/Team:Imperial_College [Accessed: 11 September 2023].

Tellechea-Luzardo, J., Otero-Muras, I., Goñi-Moreno, A. & Carbonell, P. (2022) Fast biofoundries: coping with the challenges of biomanufacturing. Trends in Biotechnology. S0167779921002997. doi:10.1016/j.tibtech.2021.12.006.

Tellechea-Luzardo, J., Winterhalter, C., Widera, P., Kozyra, J., de Lorenzo, V. & Krasnogor, N. (2020) Linking Engineered Cells to Their Digital Twins: A Version Control System for Strain Engineering. ACS Synthetic Biology. 9 (3), 536–545. doi:10.1021/acssynbio.9b00400.

TerMaat, J.R., Pienaar, E., Whitney, S.E., Mamedov, T.G. & Subramanian, A. (2009) Gene synthesis by integrated polymerase chain assembly and PCR amplification using a high-speed thermocycler. Journal of Microbiological Methods. 79 (3), 295–300. doi:10.1016/j.mimet.2009.09.015.

Terreros, D.A. & Grantham, J.J. (1982) Marshall Barber and the origins of micropipette methods. American Journal of Physiology-Renal Physiology. 242 (3), F293–F296. doi:10.1152/ajprenal.1982.242.3.F293.

Teytelman, L. & Stoliartchouk, A. (2015) Protocols.io: Reducing the knowledge that perishes because we do not publish it. Information Services & Use. 35 (1–2), 109–115. doi:10.3233/ISU-150769.

Teytelman, L., Stoliartchouk, A., Kindler, L. & Hurwitz, B.L. (2016) Protocols.io: Virtual Communities for Protocol Development and Discussion. PLOS Biology. 14 (8), e1002538. doi:10.1371/journal.pbio.1002538.

The Bioprotocols Working Group (2022a) The Bioprotocols Working Group. 2022. https://bioprotocols.github.io/labop/about.html [Accessed: 17 January 2023].

The Bioprotocols Working Group (2022b) The Laboratory Open Protocol Language (LabOP). 2022. https://bioprotocols.github.io/labop/ [Accessed: 17 January 2023].

The Economist (2006) Life 2.0. The Economist. https://www.economist.com/special-report/2006/08/31/life-20.

The Royal Academy of Engineering (2009) Synthetic Biology: scope, applications and implications. Synthetic Biology.

Thurow, K. (2022) Automation for Life Science Laboratories. In: S. Beutel & F. Lenk (eds.). Smart Biolabs of the Future. Advances in Biochemical Engineering/Biotechnology. Cham,

Springer International Publishing. pp. 3–22. doi:10.1007/10_2021_170.

Tian, T. & Salis, H.M. (2015) A predictive biophysical model of translational coupling to coordinate and control protein expression in bacterial operons. Nucleic acids research. 43 (14), 7137–7151.

Tietze, L. & Lale, R. (2021) Importance of the 5′ regulatory region to bacterial synthetic biology applications. Microbial Biotechnology. n/a (n/a). doi:10.1111/1751-7915.13868.

Venkatesh, K.P., Brito, G. & Kamel Boulos, M.N. (2024) Health Digital Twins in life Science and Health Care Innovation. Annual Review of Pharmacology and Toxicology. 64 (1), null. doi:10.1146/annurev-pharmtox-022123-022046.

Vicknair, C. (2010) Research issues in data provenance. In: Proceedings of the 48th Annual Southeast Regional Conference. ACM SE '10. 15 April 2010 New York, NY, USA, Association for Computing Machinery. pp. 1–4. doi:10.1145/1900008.1900037.

Vidal, L.S., Isalan, M., T. Heap, J. & Ledesma-Amaro, R. (2023) A primer to directed evolution: current methodologies and future directions. RSC Chemical Biology. 4 (4), 271–291. doi:10.1039/D2CB00231K.

Vivian, J., Rao, A.A., Nothaft, F.A., Ketchum, C., Armstrong, J., Novak, A., Pfeil, J., Narkizian, J., Deran, A.D. & Musselman-Brown, A. (2017) Toil enables reproducible, open source, big biomedical data analyses. Nature biotechnology. 35 (4), 314–316.

Voigt, C.A. (2020) Synthetic biology 2020–2030: six commercially-available products that are changing our world. Nature Communications. 11 (1), 6379. doi:10.1038/s41467-020-20122-2.

Volk, M.J., Tran, V.G., Tan, S.-I., Mishra, S., Fatma, Z., Boob, A., Li, H., Xue, P., Martin, T.A. & Zhao, H. (2023) Metabolic Engineering: Methodologies and Applications. Chemical Reviews. 123 (9), 5521–5570. doi:10.1021/acs.chemrev.2c00403.

Vrana, J., de Lange, O., Yang, Y., Newman, G., Saleem, A., Miller, A., Cordray, C., Halabiya, S., Parks, M., Lopez, E., Goldberg, S., Keller, B., Strickland, D. & Klavins, E. (2021) Aquarium: open-source laboratory software for design, execution and data management. Synthetic Biology. ysab006. doi:10.1093/synbio/ysab006.

W3C Working Group (2013a) PROV Model Primer. https://www.w3.org/TR/2013/NOTE-prov-primer-20130430/.

W3C Working Group (2013b) PROV-O: The PROV Ontology. https://www.w3.org/TR/prov-o/.

W3C Working Group (2013c) PROV-Overview. https://www.w3.org/TR/prov-overview/.

Wagner, L., Jules, M. & Borkowski, O. (2023) What remains from living cells in bacterial lysate-based cell-free systems. Computational and Structural Biotechnology Journal. 21,

3173–3182. doi:10.1016/j.csbj.2023.05.025.

Wang, C., Zhao, S., Shao, X., Park, J.-B., Jeong, S.-H., Park, H.-J., Kwak, W.-J., Wei, G. & Kim, S.-W. (2019) Challenges and tackles in metabolic engineering for microbial production of carotenoids. Microbial Cell Factories. 18 (1), 55. doi:10.1186/s12934-019-1105-1.

Wang, D. (2018) Building Value in a World of Technological Change: Data Analytics and Industry 4.0. IEEE Engineering Management Review. 46 (1), 32–33. doi:10.1109/EMR.2018.2809915.

Wang, M., Morón-Ortiz, Á., Zhou, J., Benítez-González, A., Mapelli-Brahm, P., Meléndez-Martínez, A.J. & Barba, F.J. (2023) Effects of pressurized liquid extraction with dimethyl sulfoxide on the recovery of carotenoids and other dietary valuable compounds from the microalgae Spirulina, Chlorella and Phaeodactylum tricornutum. Food Chemistry. 405, 134885.

Wang, S., Meng, Y., Liu, J., Cao, X. & Xue, S. (2018) Accurate quantification of astaxanthin from Haematococcus pluvialis using DMSO extraction and lipase-catalyzed hydrolysis pretreatment. Algal Research. 35, 427–431. doi:10.1016/j.algal.2018.08.029.

Wang, Y., Wang, H., Wei, L., Li, S., Liu, L. & Wang, X. (2020) Synthetic promoter design in Escherichia coli based on a deep generative network. Nucleic Acids Research. 48 (12), 6403–6412. doi:10.1093/nar/gkaa325.

Wang, Y., Xue, P., Cao, M., Yu, T., Lane, S.T. & Zhao, H. (2021) Directed Evolution: Methodologies and Applications. Chemical Reviews. 121 (20), 12384–12444. doi:10.1021/acs.chemrev.1c00260.

Watson, J.F. & García-Nafría, J. (2019) In vivo DNA assembly using common laboratory bacteria: A re-emerging tool to simplify molecular cloning. Journal of Biological Chemistry. 294 (42), 15271–15281. doi:10.1074/jbc.REV119.009109.

Weisstein, E.W. (2023a) Acyclic Graph. 2023. https://mathworld.wolfram.com/ [Accessed: 30 October 2023].

Weisstein, E.W. (2023b) Graph Cycle. 2023. https://mathworld.wolfram.com/ [Accessed: 30 October 2023].

Whitehead, E., Rudolf, F., Kaltenbach, H.-M. & Stelling, J. (2018) Automated Planning Enables Complex Protocols on Liquid-Handling Robots. ACS Synthetic Biology. 7 (3), 922–932. doi:10.1021/acssynbio.8b00021.

Wickham, H. (2011) The Split-Apply-Combine Strategy for Data Analysis. Journal of Statistical Software. 40 (1). doi:10.18637/jss.v040.i01.

Wikipedia (2021) Heterologous. Wikipedia. https://en.wikipedia.org/w/index.php?

title=Heterologous&oldid=1030630745.

Wikipedia (2023) Ribosome-binding site. Wikipedia. https://en.wikipedia.org/w/index.php?title=Ribosome-binding_site&oldid=1171093305.

Wilkinson, M.D., Dumontier, M., Aalbersberg, Ij.J., Appleton, G., Axton, M., et al. (2016) The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data. 3, 160018. doi:10.1038/sdata.2016.18.

Winston, C., Willsey, M. & Ceze, L. (2023) Virtualizing Existing Fluidic Programs. ACM Journal on Emerging Technologies in Computing Systems. 19 (3), 27:1-27:14. doi:10.1145/3558550.

Wong, M., Badri, A., Gasparis, C., Belfort, G. & Koffas, M. (2021) Modular optimization in metabolic engineering. Critical Reviews in Biochemistry and Molecular Biology. 56 (6), 587–602. doi:10.1080/10409238.2021.1937928.

Wratten, L., Wilm, A. & Göke, J. (2021) Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. Nature Methods. 18 (10), 1161–1168. doi:10.1038/s41592-021-01254-9.

Wu, J., Deng, A., Sun, Q., Bai, H., Sun, Z., Shang, X., Zhang, Y., Liu, Q., Liang, Y. & Liu, S. (2017) Bacterial genome editing via a designed toxin–antitoxin cassette. ACS Synthetic Biology. 7 (3), 822–831.

Wu, Z., Zhao, D., Li, S., Wang, J., Bi, C. & Zhang, X. (2019) Combinatorial modulation of initial codons for improved zeaxanthin synthetic pathway efficiency in Escherichia coli. MicrobiologyOpen. 8 (12), e930. doi:10.1002/mbo3.930.

Xiong, M. & Wang, H. (2022) Digital twin applications in aviation industry: A review. The International Journal of Advanced Manufacturing Technology. 121 (9), 5677–5692. doi:10.1007/s00170-022-09717-9.

Xu, M., David, J.M. & Kim, S.H. (2018) The Fourth Industrial Revolution: Opportunities and Challenges. International Journal of Financial Research. 9 (2), 90. doi:10.5430/ijfr.v9n2p90.

Xu, N., Wei, L. & Liu, J. (2019) Recent advances in the applications of promoter engineering for the optimization of metabolite biosynthesis. World Journal of Microbiology and Biotechnology. 35 (2), 33. doi:10.1007/s11274-019-2606-0.

Xu, P., Rizzoni, E.A., Sul, S.-Y. & Stephanopoulos, G. (2017) Improving Metabolic Pathway Efficiency by Statistical Model-Based Multivariate Regulatory Metabolic Engineering. ACS Synthetic Biology. 6 (1), 148–158. doi:10.1021/acssynbio.6b00187.

Yadav, V.G., De Mey, M., Giaw Lim, C., Kumaran Ajikumar, P. & Stephanopoulos, G. (2012) The future of metabolic engineering and synthetic biology: Towards a systematic practice.

Metabolic Engineering. 14 (3), 233–241. doi:10.1016/j.ymben.2012.02.001.

Yamano, S., Ishii, T., Nakagawa, M., Ikenaga, H. & Misawa, N. (1994) Metabolic engineering for production of beta-carotene and lycopene in Saccharomyces cerevisiae. Bioscience, Biotechnology, and Biochemistry. 58 (6), 1112–1114. doi:10.1271/bbb.58.1112.

Yen, H.-W., Palanisamy, G. & Su, G.-C. (2019) The Influences of Supplemental Vegetable Oils on the Growth and β-carotene Accumulation of Oleaginous Yeast-Rhodotorula glutinis. Biotechnology and Bioprocess Engineering. 24 (3), 522–528. doi:10.1007/s12257-019-0027-4.

Yeoh, J.W., Swainston, N., Vegh, P., Zulkower, V., Carbonell, P., Holowko, M.B., Peddinti, G. & Poh, C.L. (2021) SynBiopython: an open-source software library for Synthetic Biology. Synthetic Biology. (ysab001). doi:10.1093/synbio/ysab001.

Yeow (2014) Effects of Stress, Repetition, Fatigue and Work Environment on Human Error in Manufacturing Industries. 2014. doi:10.3923/jas.2014.3464.3471 [Accessed: 1 August 2023].

Yoon, S.-H., Kim, J.-E., Lee, S.-H., Park, H.-M., Choi, M.-S., Kim, J.-Y., Lee, S.-H., Shin, Y.-C., Keasling, J.D. & Kim, S.-W. (2007) Engineering the lycopene synthetic pathway in E. coli by comparison of the carotenoid genes of Pantoea agglomerans and Pantoea ananatis. Applied Microbiology and Biotechnology. 74 (1), 131–139. doi:10.1007/s00253-006-0623-z.

Yoon, S.-H., Lee, Y.-M., Kim, J.-E., Lee, S.-H., Lee, J.-H., Kim, J.-Y., Jung, K.-H., Shin, Y.-C., Keasling, J.D. & Kim, S.-W. (2006) Enhanced lycopene production inEscherichia coli engineered to synthesize isopentenyl diphosphate and dimethylallyl diphosphate from mevalonate. Biotechnology and Bioengineering. 94 (6), 1025–1032. doi:10.1002/bit.20912.

Young, R., Haines, M., Storch, M. & Freemont, P.S. (2021) Combinatorial metabolic pathway assembly approaches and toolkits for modular assembly. Metabolic Engineering. 63, 81–101. doi:10.1016/j.ymben.2020.12.001.

Yu, W., Patros, P., Young, B., Klinac, E. & Walmsley, T.G. (2022) Energy digital twin technology for industrial energy management: Classification, challenges and future. Renewable and Sustainable Energy Reviews. 161, 112407. doi:10.1016/j.rser.2022.112407.

Zang, L.-Y., Sommerburg, O. & van Kuijk, F.J.G.M. (1997) Absorbance Changes of Carotenoids in Different Solvents. Free Radical Biology and Medicine. 23 (7), 1086–1089. doi:10.1016/S0891-5849(97)00138-X.

Zhang, F., Carothers, J.M. & Keasling, J.D. (2012) Design of a dynamic sensor-regulator system for production of chemicals and fuels derived from fatty acids. Nature Biotechnology. 30 (4), 354–359. doi:10.1038/nbt.2149.

Zhang, M., Jia, C., Li, F., Li, C., Zhu, Y., Akutsu, T., Webb, G.I., Zou, Q., Coin, L.J.M. & Song,

J. (2022) Critical assessment of computational tools for prokaryotic and eukaryotic promoter prediction. Briefings in Bioinformatics. 23 (2), bbab551. doi:10.1093/bib/bbab551.

Zhang, X., Chen, S., Lin, Y., Li, W., Wang, D., Ruan, S., Yang, Y. & Liang, S. (2023) Metabolic Engineering of Pichia pastoris for High-Level Production of Lycopene. ACS Synthetic Biology. doi:10.1021/acssynbio.3c00294.

Zhang, Y., Chen, H., Zhang, Y., Yin, H., Zhou, C. & Wang, Y. (2021) Direct RBS Engineering of the biosynthetic gene cluster for efficient productivity of violaceins in E. coli. Microbial Cell Factories. 20, 38. doi:10.1186/s12934-021-01518-1.

Zhao, M., Yuan, Z., Wu, L., Zhou, S. & Deng, Y. (2022) Precise Prediction of Promoter Strength Based on a De Novo Synthetic Promoter Library Coupled with Machine Learning. ACS Synthetic Biology. 11 (1), 92–102. doi:10.1021/acssynbio.1c00117.

Zhao, W., Yan, T., Yin, W. & Xie, Y. (2020) Effect of pH on the Lycopene Stability and Its Degradation Kinetics. MODERN FOOD SCIENCE & TECHNOLOGY. 36 (1), 178–183.

Zhong, Z., Xu, M., Rodriguez, M.A., Xu, C. & Buyya, R. (2022) Machine Learning-based Orchestration of Containers: A Taxonomy and Future Directions. ACM Computing Surveys. 54 (10s), 217:1-217:35. doi:10.1145/3510415.

Zielinski, D., Gordon, A., Zaks, B.L. & Erlich, Y. (2014) iPipet: sample handling using a tablet. Nature Methods. 11 (8), 784–785. doi:10.1038/nmeth.3028.

Zimmerman, S.B., Little, J.W., Oshinsky, C.K. & Gellert, M. (1967) Enzymatic joining of DNA strands: a novel reaction of diphosphopyridine nucleotide. Proceedings of the National Academy of Sciences. 57 (6), 1841–1848. doi:10.1073/pnas.57.6.1841.

Zinnen, T. (2004) The Micropipette Story. 2004. Outreach. https://outreach.biotech.wisc.edu/the-micropipette-story/ [Accessed: 8 August 2023].

Appendices C3

C3 - Methods

BASIC DNA assembly

BASIC linkers and parts preparation were done using the standard BASIC parts preparation protocol (https://www.basic-assembly.org/protocols). BASIC assemblies were also done following the standard BASIC assembly protocol in a 96-well standard plate either manually or with the Opentrons OT-2.

Preparation of BASIC linkers and BASIC bioparts

BASIC linkers were obtained from Biolegio. Synthesised genes were ordered as gBlocks from NEB. New linkers are prepared as such:

1. Spin down tubes with lyophilized linkers to ensure oligos are at bottom of the tube.
2. Set heating block to 95°C.
3. Add 200 ul of linker annealing buffer to each linker tube and leave on bench for 1h.
4. Vortex tubes and collect liquid at tube bottom via a quick centrifuge spin.
5. After heating block reached 95°C, place tubes into block and slightly loosen tube caps to allow for heat expansion.
6. After 5 min switch of heating block and tighten tube caps again to avoid evaporation.
7. Allow tubes to cool down to room temperature over at least 1h in heat block.
8. Collect solution at tube bottom via a quick centrifuge spin.
9. Linkers are stored at -20°C. Bioparts for BASIC assembly are provided in storage plasmids (pSEVA18). For each BASIC linker ligation reaction 50 ng of plasmid per 1kb of total plasmid size (including BASIC part and storage backbone pSEVA18) are required. Usually, that amount of DNA is provided in 1 ul of a typical miniprep of biopart storage plasmids (200 ng/ul for a 4kb plasmid). If PCR products or gene fragments are used as reaction input, 50 ng per 1kb linear DNA are required.

BASIC reaction

For each BASIC linker ligation reaction, setup 1 PCR tube with 30 ul total volume: dH20 17 ul, Promega T4 buffer (10x), 3 ul Prefix Linker, 1 ul Suffix Linker, 1 ul BASIC biopart 0.5-6 ul

(50 ng per 1kb total plasmid size), add dH20 to reach 28.5 ul volume, NEB BsaI-HF v2 enzyme (R3733) 20 U/ul, 1 ul Promega T4 ligase (M1801) 1-3 U/ul 0.5 ul Mix by pipetting up and down. After mixing, tubes are placed in a PCR machine running the following programme: (37°C,2 min) X 20 cycles, (20°C, 1 min), (37°C 5 min), (80°C, 20 min)

Magbead purification

Using 0.5 ml of 70% EtOH per BASIC reaction and bringing magnetic beads stored at 4°C back into homogeneous mix by shaking thoroughly. We used 96 well Falcon plates (Falcon 351177) in combination with an Ambion magnetic plate (AM10050) for quick magbead immobilisation and easy pipetting access. Purification protocol was applied as followed:

1. Add 54 ul of magnetic beads into 96 well Falcon plate (one well per BASIC reaction) and add the 30 ul BASIC linker ligation from the PCR machine step, mix by pipetting 10 times.
2. Wait 5min to allow DNA binding to magbeads.
3. Place Falcon plate on the magnetic stand and wait for rings to form and the solution to clear.
4. Remove the solution with a 200 ul pipette tip from the centre of each well.
5. Add 190 ul 70% EtOH to each well and wait 30 s.
6. Remove solution from each well (pipette set to 200 ul volume)
7. Add 190 ul 70% EtOH to each well and wait 30 s.
8. Remove solution from each well (pipette set to 200 ul volume)
9. Leave the plate to dry for 1-2 min.
10. Remove Falcon plate from magnet and resuspend magbeads in 32 ul dH20.
11. Wait 1 min for DNA to elute.
12. Place Falcon plate back on the magnetic stand and allow ring to form and the solution to clear.
13. Pipette 30 ul of H20 with eluted DNA into fresh 1.5 ml Eppendorf tube for direct use in assembly or storage at -20°C for up to 1 month.

Assembly reaction

For each BASIC assembly parts were combined with buffer in a PCR tube: dH20 2 ul, NEB CutSmart buffer 10x 1 ul, Linker ligated BASIC part crtE 1 ul, Linker ligated BASIC part crtB 1 ul, Linker ligated BASIC part crtI 1 ul, dH20 Top up to 10 ul total volume. Assembly reaction is run in a PCR machine with following programme: 50°C for 45 min followed by 4°C on hold

Transformation

Use 50 uL of chemically competent cells DH5alpha with high transformation efficiency (109 CFU/ug pUC19, for instance NEB C2987I) to transform 5 uL of each BASIC assembly:

1. Chemically competent cells are stored at -80°C.
2. Thaw competent cells on ice (takes 5-10 min); 50 uL per BASIC assembly to be transformed.
3. Cool 5 uL of BASIC DNA assembly in 1.5 mL Eppendorf tube on ice.
4. Add 50 uL of competent cells to each precooled 5 uL BASIC reaction.
5. Incubate on ice for 20 min.
6. Apply heat shock in 42°C water bath for 45s and place back on ice for 2 min.
7. Add 200 uL SOC medium to each tube and incubate shaking at 37°C for 1h recovery.
8. Spot or plate cells on agar plates with appropriate antibiotics. Depending on the number of parts assembled and transformation efficiency 2-250 uL might be spotted or plated.
9. Incubate agar plates at 37°C overnight, next day pick colony for assay or miniprep. We used a PCR machine, Heat block (up to 95°C) fitting 2ml Eppendorf tubes, Water bath (42°C) for transformation, Magnetic plate Ambion AM10050 (Thermo), 96 well U-Bottom Falcon plate, Falcon 351177 (Thermo) Eppendorf tubes, Magnetic beads Ampliclean (Nimagen), dH20, 70% EtOH, Biolegio BASIC linkers BBT-18100 (Biolegio), BASIC parts in storage plasmids (200ng/ul), NEB BsaI-HF v2 enzyme (R3733), 20 U/ul; includes CutSmart buffer R3733 (NEB), Promega T4 ligase (M1801) 1-3U/ul; includes Promega T4 buffer M1801 (Promega), Chemically competent cells (DH5alpha, 1x109 CFU/ug pUC19); includes SOC media C2987I (NEB)

Pooled transformation

Instead of single RBS at a single location, we use 3 RBS while maintaining relative proportion of RBS (relative to volume). The promoters, RBS and genes were added to the transformation mix. The order of the linkers in the BASIC assembly allows the fixed ordering of the three genes (crtE, crtB, crtI) hence the six different pools containing six different gene order combinations.

Cloning and cultivation

DH5α (New England Biolabs) Escherichia coli were used for cloning with a standard approach. *E. coli* cells were grown in Luria Broth (LB) medium supplemented with 25 ug/mL kanamycin (Kan). DH5α *E. coli* starter cells were grown at 37 °C. Cell cultures for lycopene production were grown at 28 °C (optimum growth temperature for lycopene production).

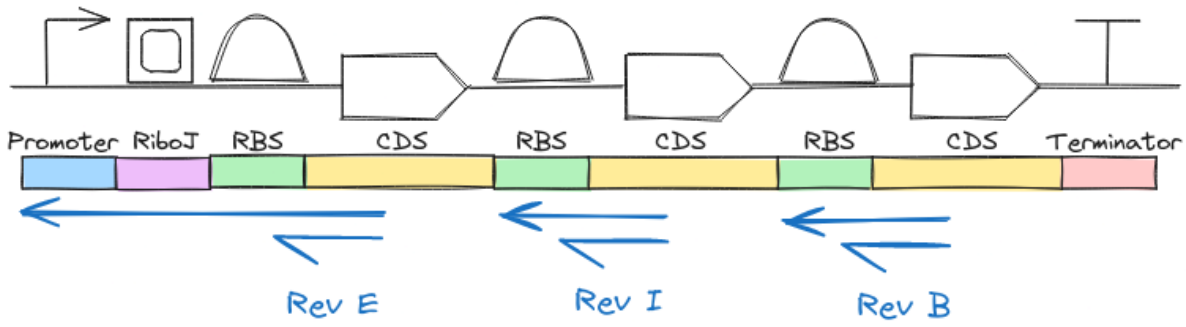
Extraction and measurement of lycopene

The 24-hour growth plate has its OD600 measured on a plate reader (Clariostar). (50 ul from the 1000 ul total volume is added to 150 ul of LB in a new, flat-bottom plate, it is mixed thoroughly, and then tested. The final OD value that the plate reader gives us is multiplied by 4 to compensate for this dilution) The remaining culture is spun at 4000rpm for 15min, at room temperature, until pellets appear. The supernatant is removed by turning the plate

upside-down onto paper towels). The pellets are resuspended in 500 ul of DMSO, using a robot (OT-2) or a multi-channel pipette The plates are then incubated at 37°C for 30 minutes at 900 rpm on a benchtop incubator, in order to extract the lycopene, while covered in aluminium foil in order to limit exposure to light. The plates are then spun down in order to generate pellets (15min at 4000rpm. 200 ul of the lycopene-containing DMSO was moved to a flat bottom plate, to measure the OD at 471 nm. (Clariostar)

Sequence analysis and part-matching

To analyse the Sanger sequences and identify the best candidate parts of the constructs, we have developed our own software based on biopython pairwise sequence alignment that uses uses a dynamic programming algorithm and implements respectively the Needleman-Wunsch algorithm and the Smith–Waterman algorithm to performs global and local sequence alignments. We use local sequence alignments as we are trying to match and find each of the parts of the lycopene operon. Whereas it is usually common to visually align matching sequences and recognise matching DNA parts and still identify them if mutations happened, however we are automating our workflow and need to procedurally match against 810 possible in-silico constructs and thus need a metric to rank the possible candidates. We normalise the matching score to the sequence length (and adjust the sequence matching length according to the trace, eliminating low quality reads at both ends of the sequencing) and know if we have a perfect match or a value that allows us to rank possible candidate parts.



C3-Figure S1 – Lycopene operon with the 3 reverse sequencing primers for the 3 genes crtE, crtI and crtB.

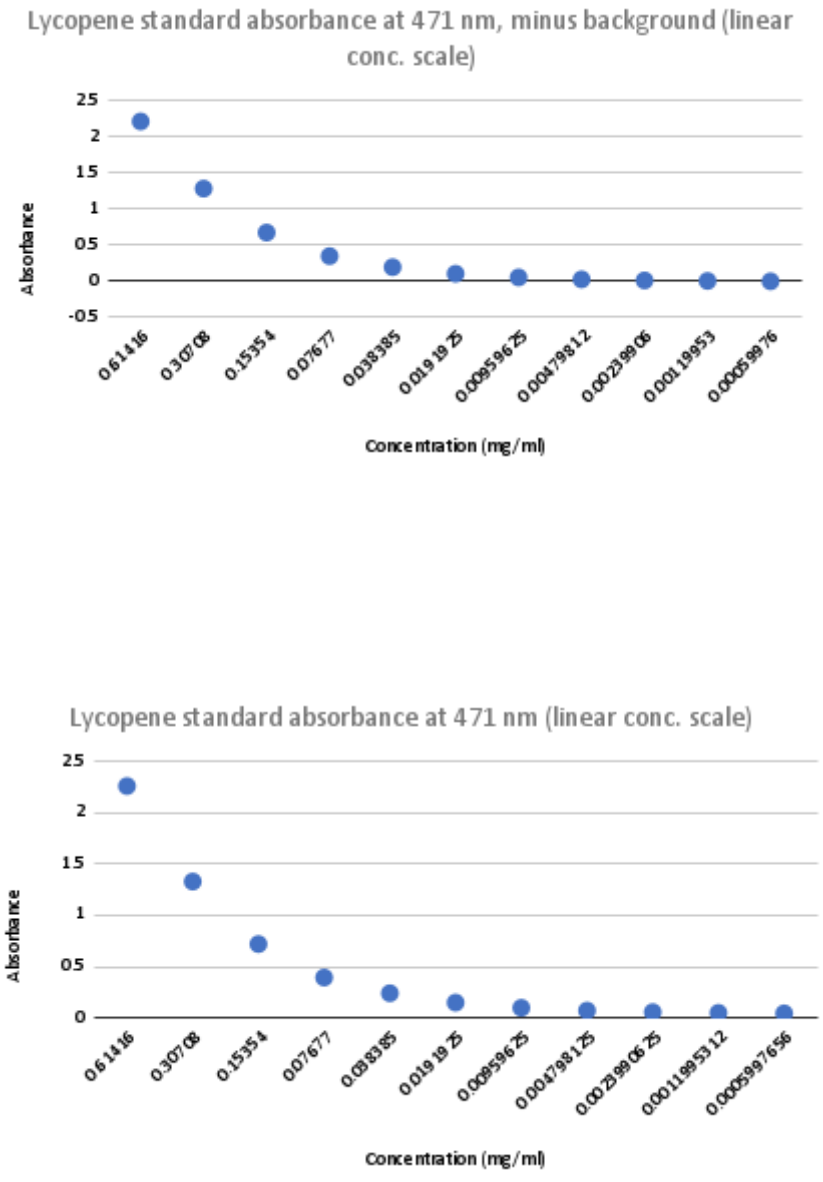
We sequenced with three distinct reverse primers starting at 1/3 in each of the three genes (crtB, crtI and crtE) (See Figure 10). The three sub-sequences obtained are approximately 800 base-pair long which is long enough to let us determine the upstream parts. Once we have all three sub-sequences we can easily infer the full sequence of the operon; as BASIC assembly requires the usage of three different UTR-RBS for the assembly, we have three different UTR1-RBSn UTR2-RBSn and UTR3-RBSn, $n \in [1, 3]$ at fixed positions in the operon, consequently the sub-sequence containing UTR1 and the promoter goes in the first

place, the sub-sequence containing the part of the CDS.

Yield calculation

Cell growth (*E. coli* DH5α) was followed by measuring the optical density at 600 nm and correlated to cell dry weight (cdw) with a ratio cdw/OD = 0.36.
Ratio of cell dry weight (g/L)/OD600

Lycopene Absorbance



C3-Figure S2 – Lycopene Absorbance.

C3 - Promoter Strength

Name	**Type**	**Strength** (in RPU)
K137085	Constitutive	0.08
J23106	Constitutive	0.35
J23108	Constitutive	0.61
J23101	Constitutive	1
apFab32	Constitutive	2.5

C3-Table S1 – The five promoters chosen for the lycopene study – ranging from weak (K137085; TPU = 0.08) to very strong (apFAB32, RPU=2.5).

C3 - Pool Number - Gene Order

The design does not fix the gene order and allows for permutation instead. Constructs are labelled according to their gene order (e.g BEI is labelled 1) as shown in Table S2.

Pool / Combination Number	**Gene Order**
1	BEI
2	BIE
3	EBI
4	EIB
5	IBE
6	IEB

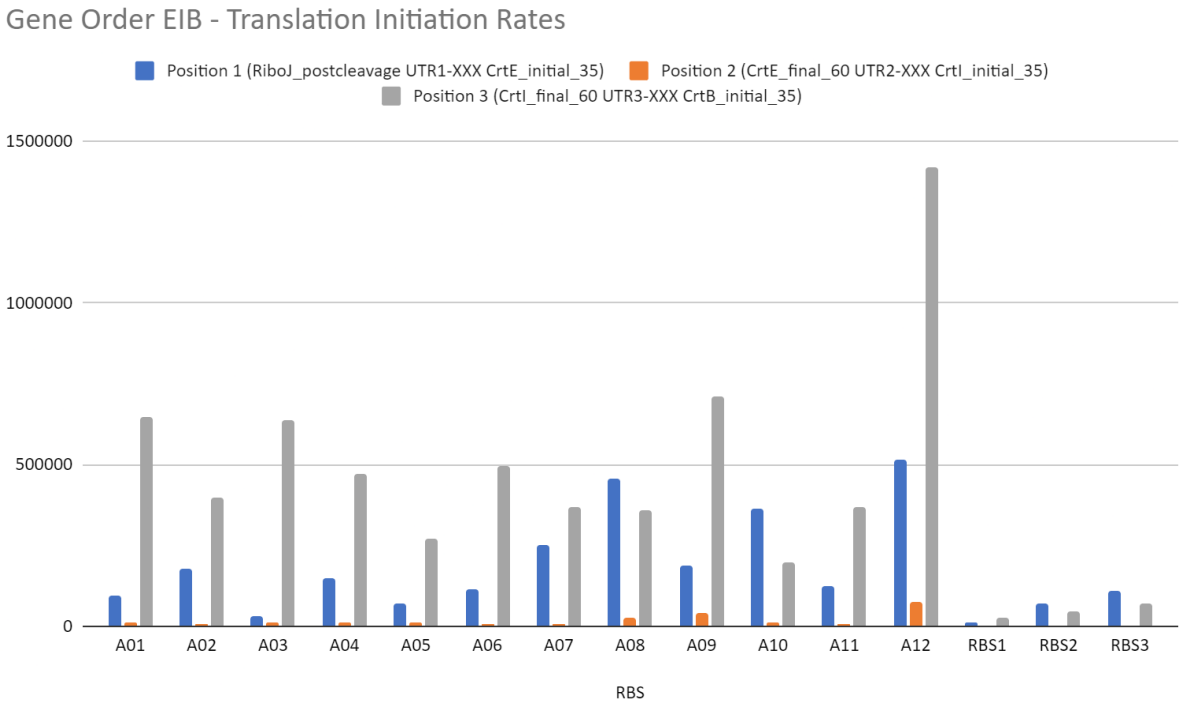
C3-Table S2 – The 6 possible gene orders in the lycopene operon.

C3 - Translation Rates

Variation of RBS Expression with its Context

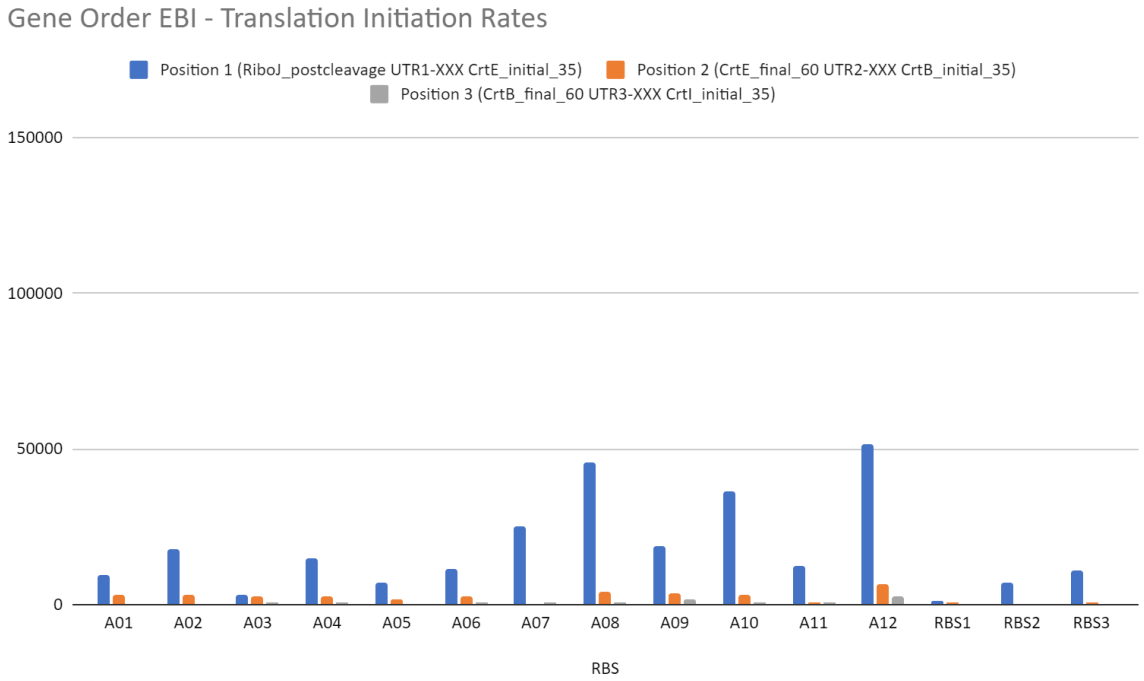
Effective translation rates were computed for all contexts with the RBS calculator (https://www.denovodna.com/) – see supplementary files Pool1- BEI, Pool2 - BIE, Pool 3 - EBI, Pool4 - EIB; Pool 5 - IBE and Pool 6 - IEB. Figures S2 and S3 show the variations of translation rates for the most commonly represented pools (EBI and EIB) for all RBS in the Biolegio library.

Translation rates vary by up to orders of magnitude and RBS ranking were not conserved.

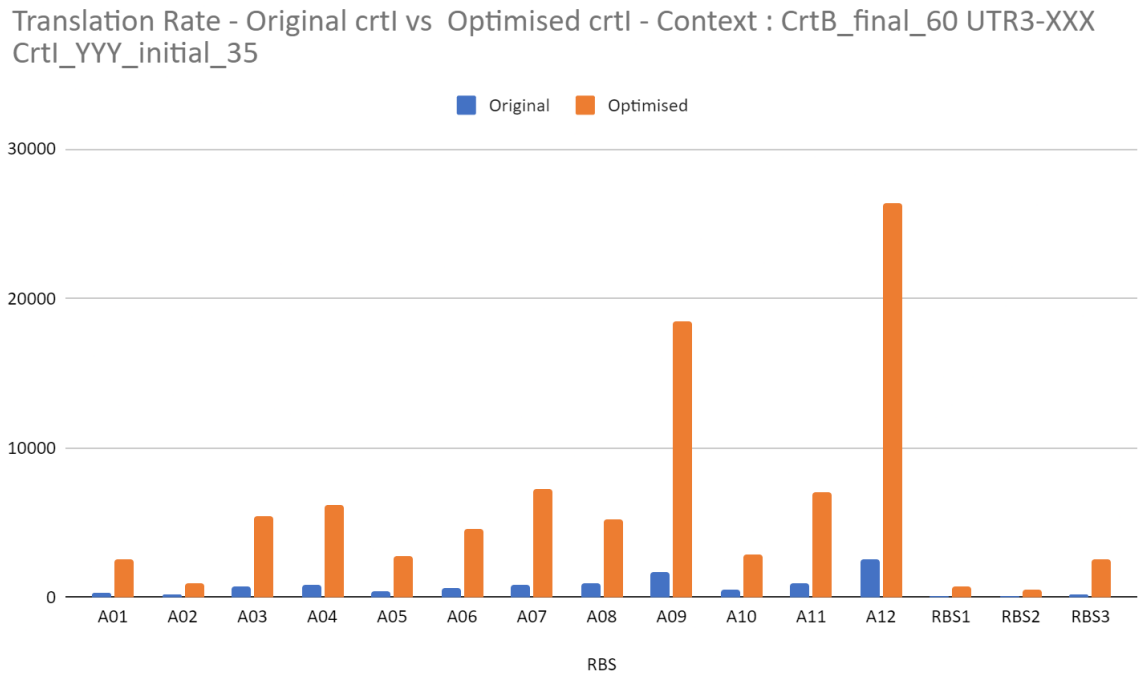


C3-Figure S4 – Effective translation rates of the BioLegio RBS in all three positions (denoted as Position 1, Position 2 and Position 3) for the gene order EIB. All values are in arbitrary units as returned by the RBS calculator.

Another practical effect of RBS context is that having CrtI downstream of the RBS drastically reduces the translation rate. Figure S4 compares translation rate with the CrtI sequence used in this study vs a codon optimised crtI sequence (https://eu.idtdna.com/pages/tools/codon-optimization-tool) – showing the translation rate is an order of magnitude large with the optimised sequence (although it was not optimised to increase translation).



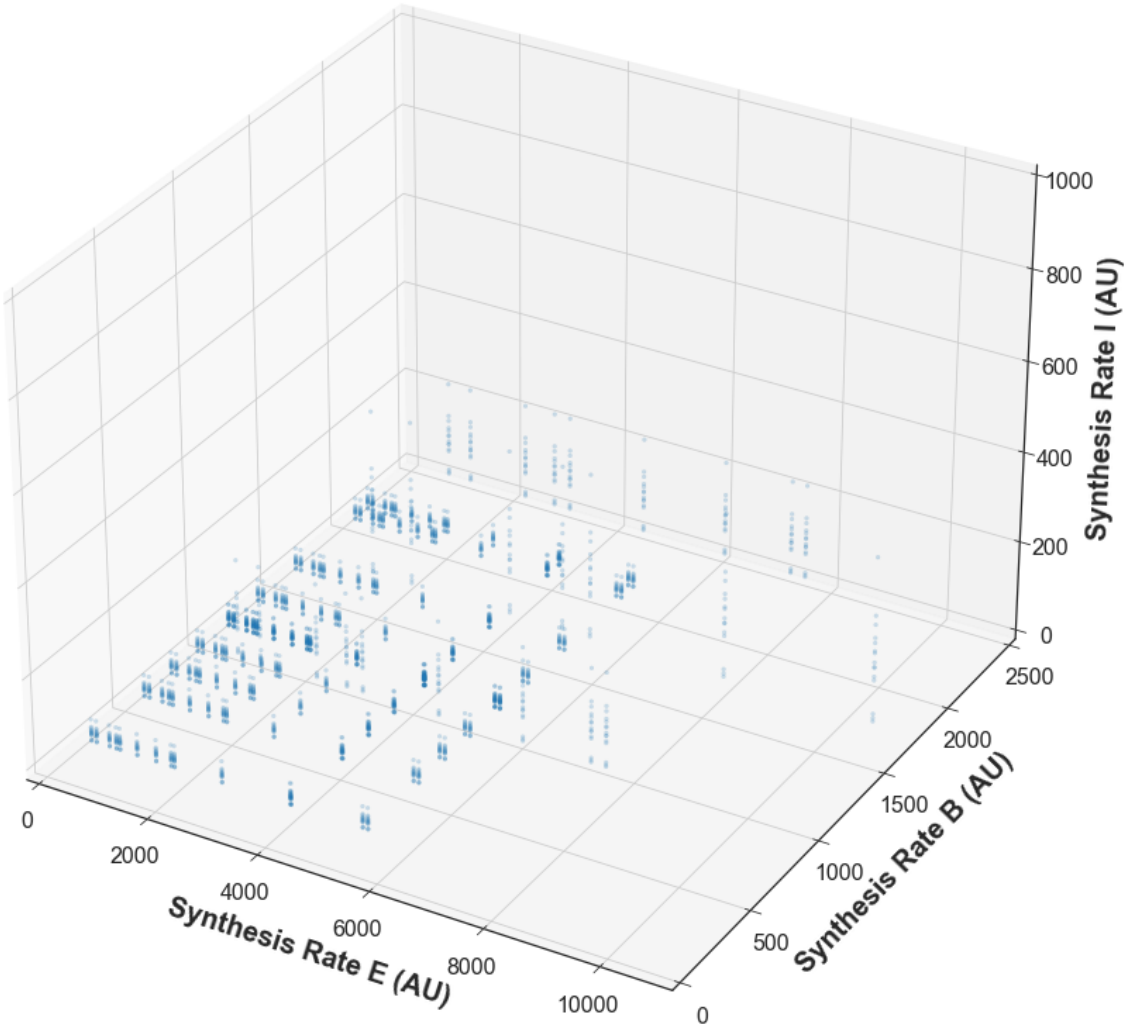
C3-Figure S3 – Effective translation rates of the Biolegio RBS in all three positions (denoted as Position 1, Position 2 and Position 3) for the gene order EBI.



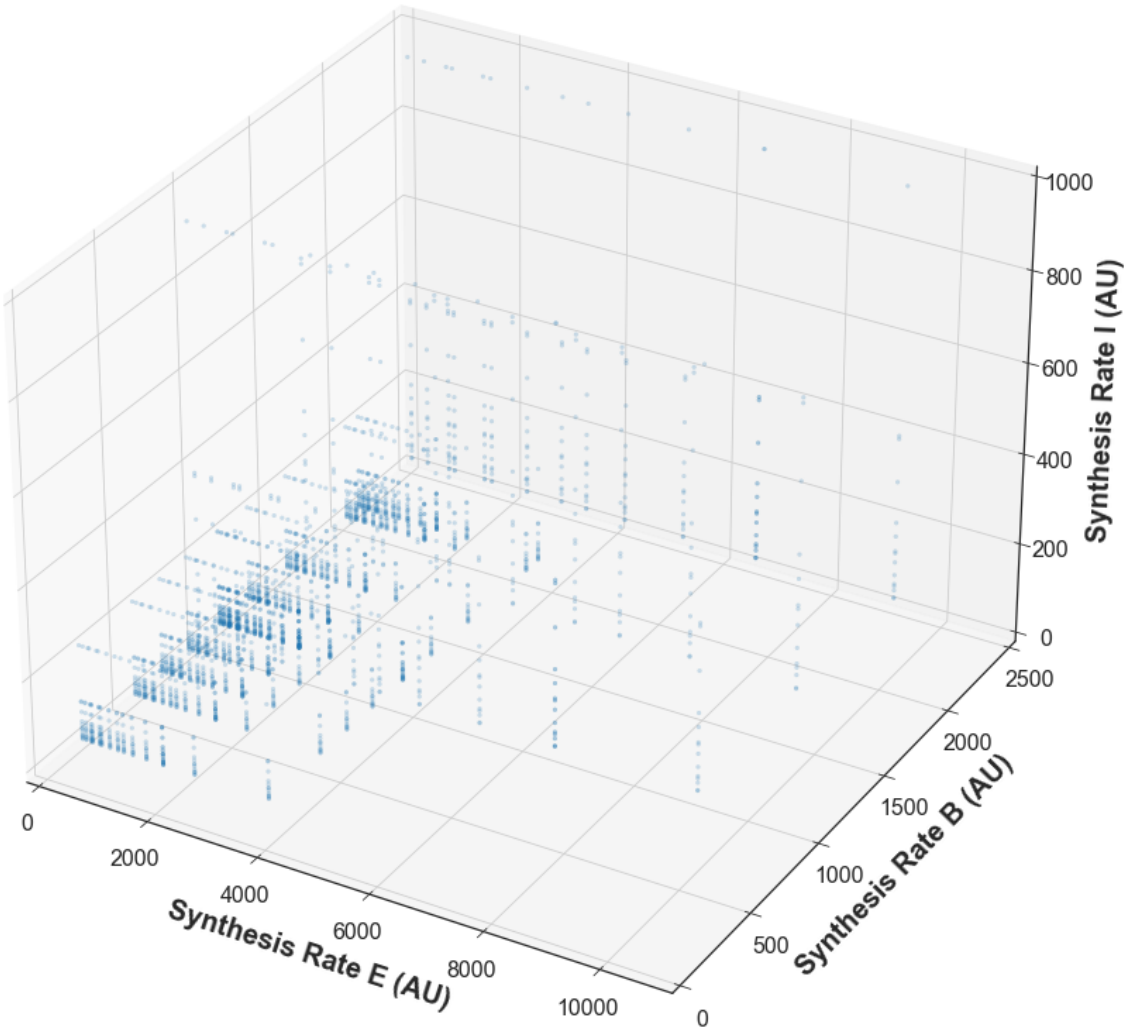
C3-Figure S4 – Translation rate with CrtI upstream of the RBS - comparison between the original sequence and a codon-optimised sequence.

C3 - Visualisation By Gene Order

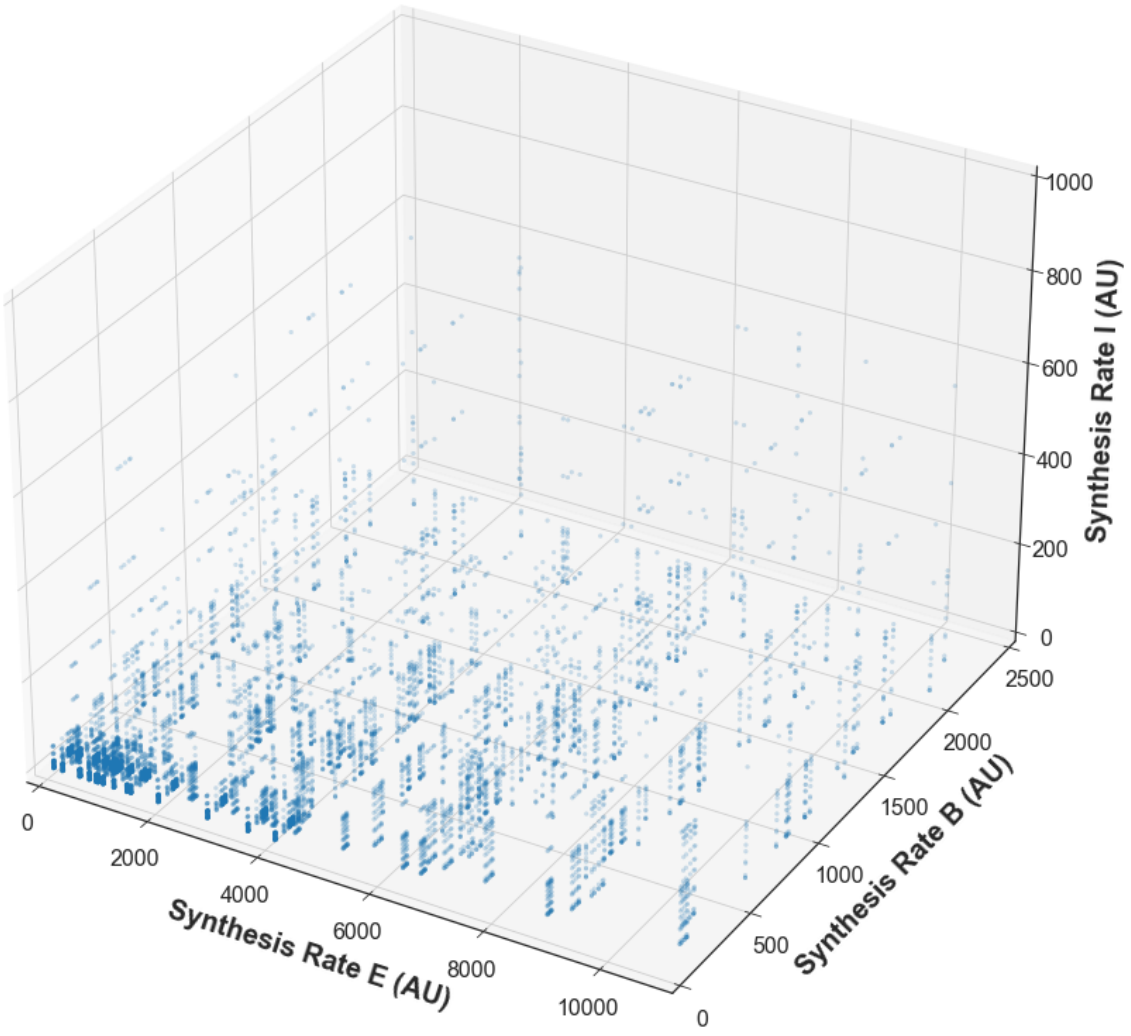
Influence on gene order on construct distribution can be observed on Figures S5 to S10.The 15-strong Biolegio RBS library (RBS-A01 to RBS-A12) and RBS 1,2 and 3 was used for this purpose. Some orders lead to clustering along some axes, whereas EBI and EIB sample the space more evenly - offering a simple explanation for their over-representation in constructs yielded by the pooled approach. Figure S11 displays the whole design space accessible with the lycopene operon and the whole BioLegiop library; Figure S12 the design space accessible with the RBS 1,2 and 3 subset.



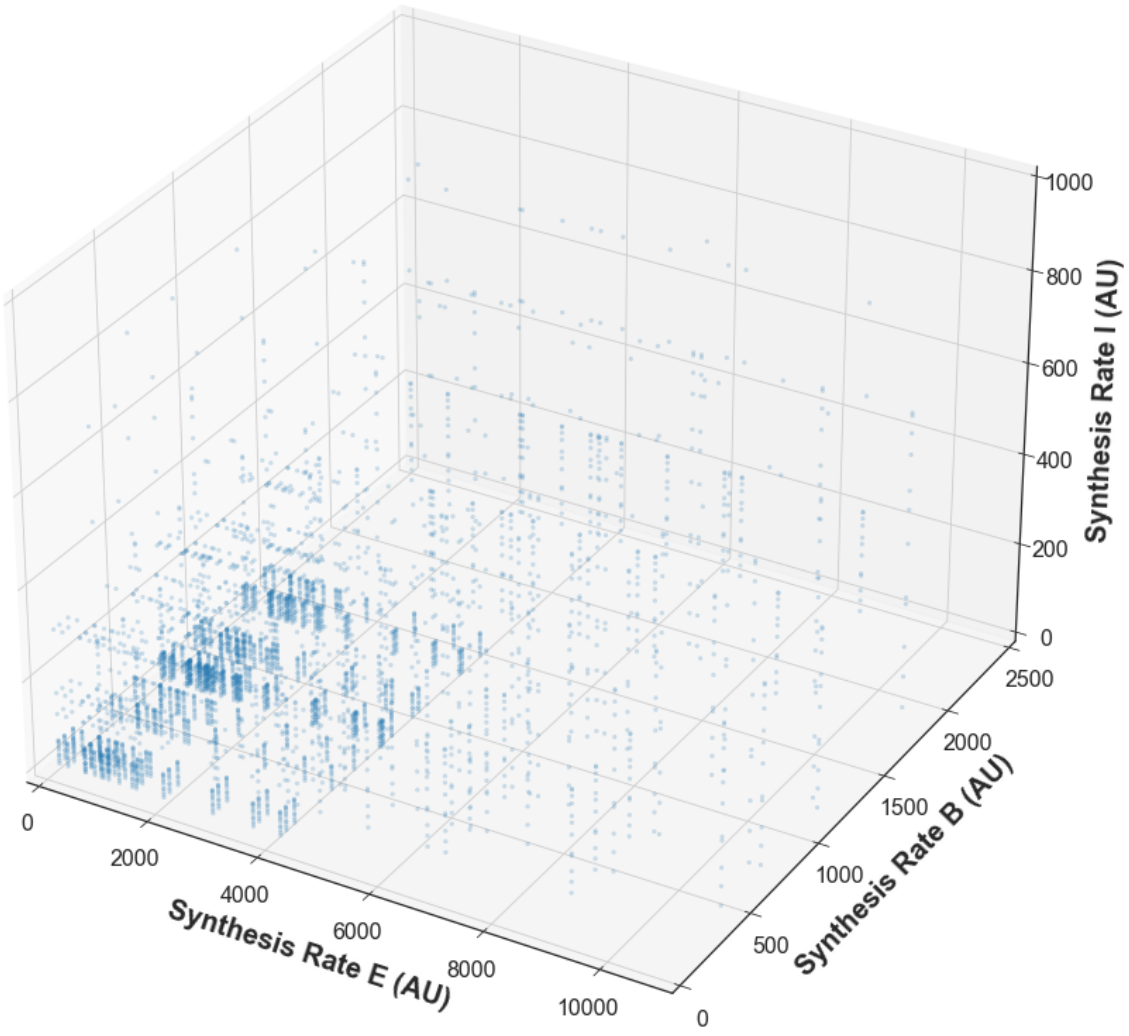
C3-Figure S5 – Design space for the BEI gene order.



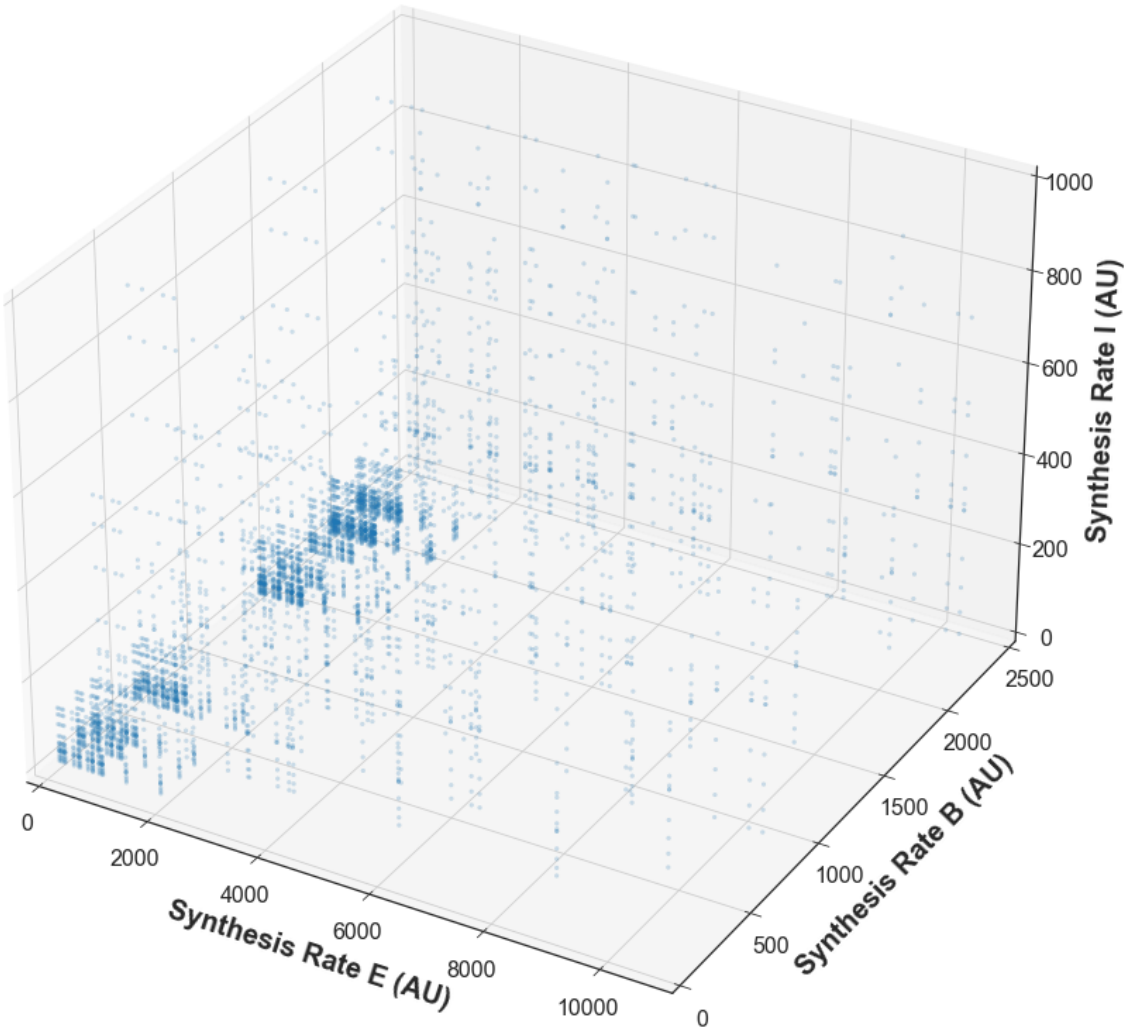
C3-Figure S6 – Design space for the BIE gene order.



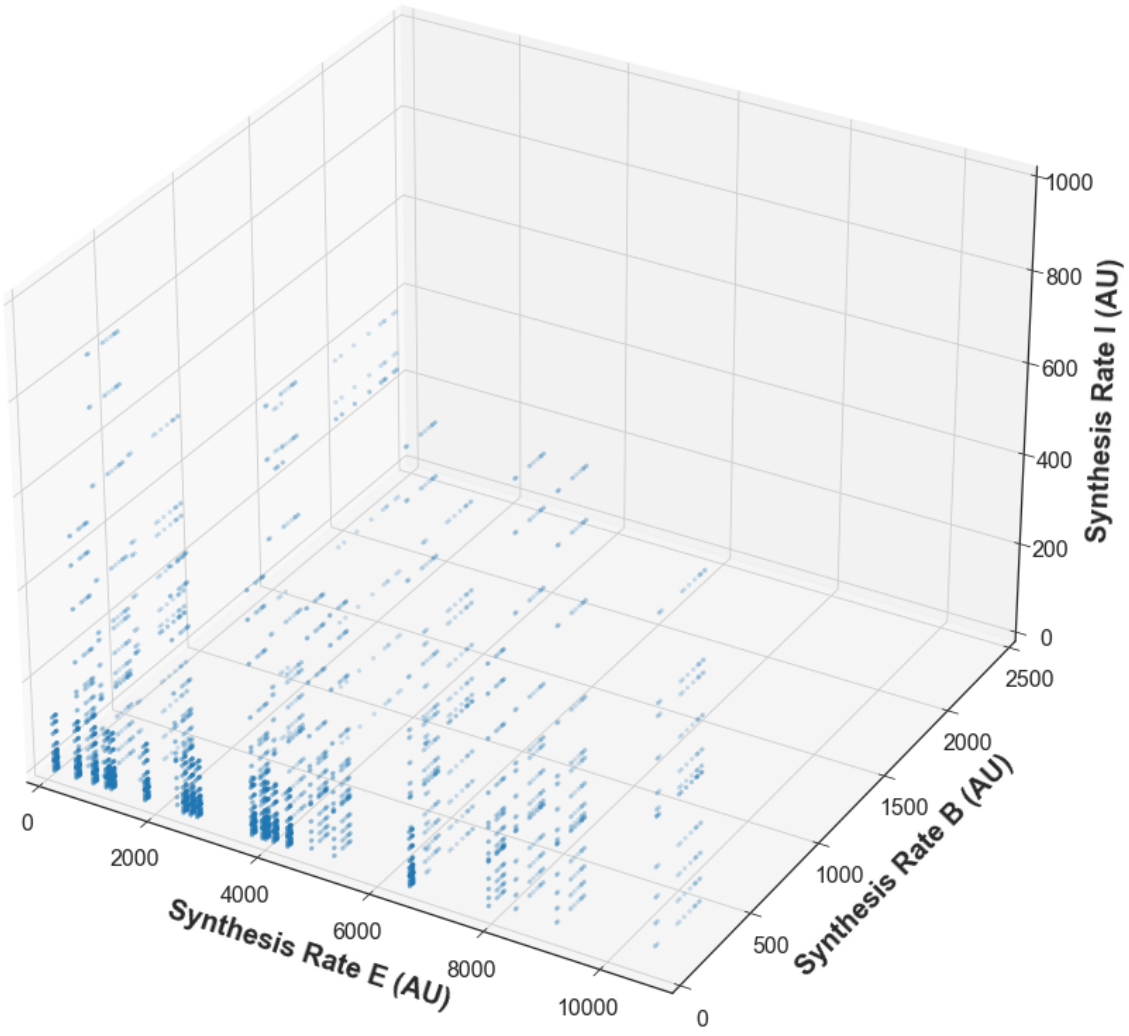
C3-Figure S7 – Design space for the EBI gene order.



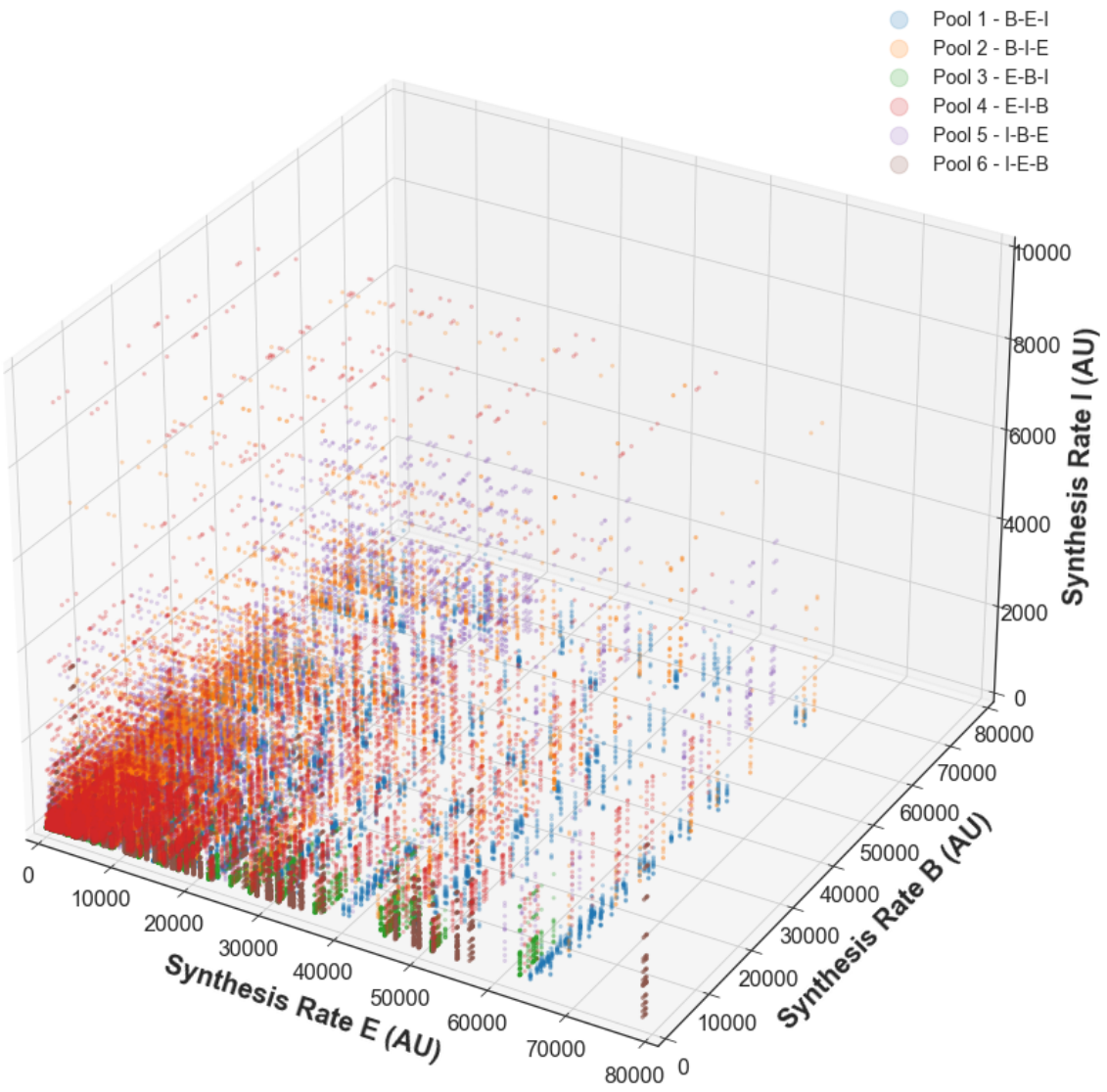
C3-Figure S8 – Design space for the EIB gene order.



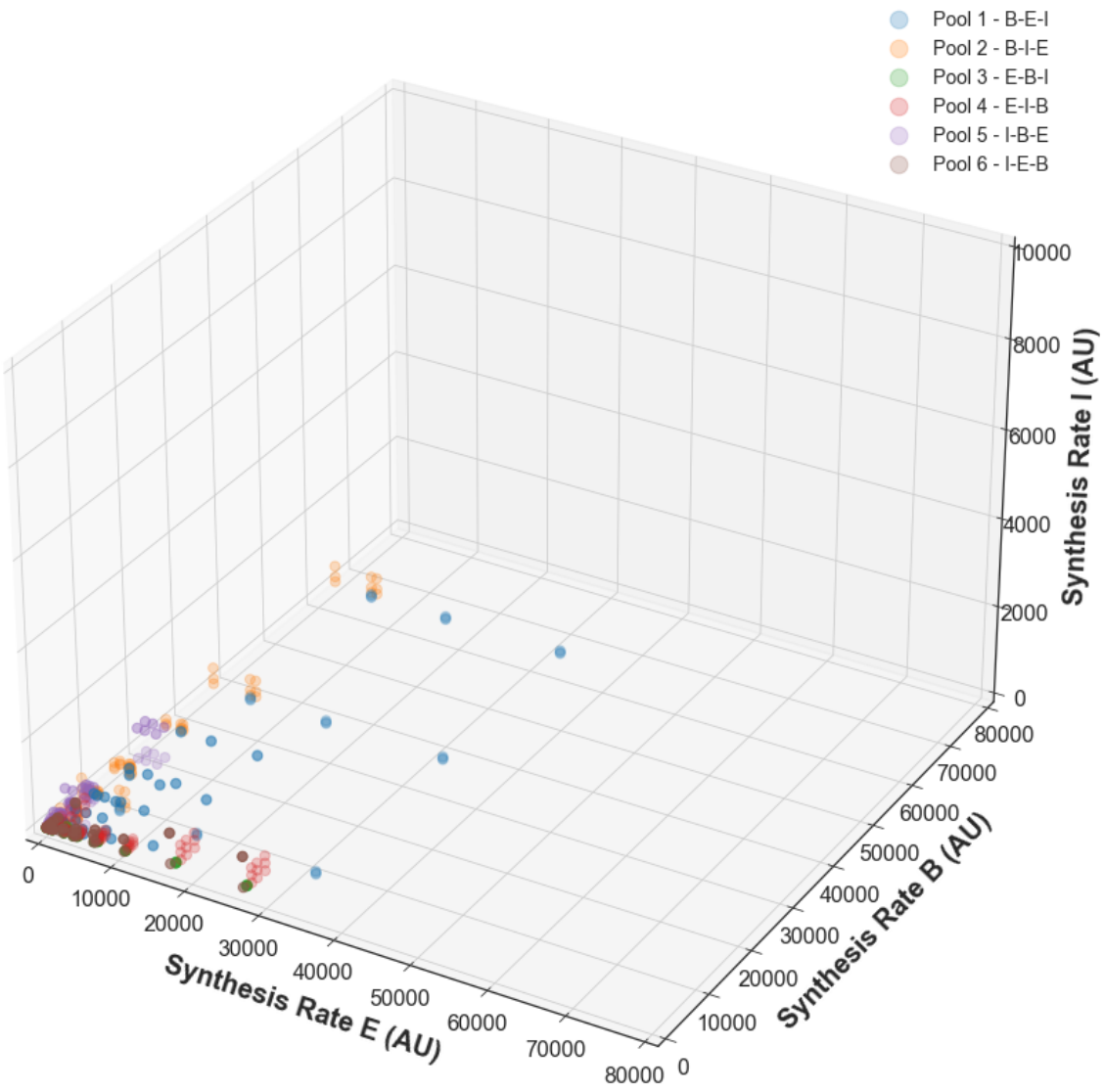
C3-Figure S9 – Design space for the IBE gene order.



C3-Figure S10 – Design space for the IEB gene order.



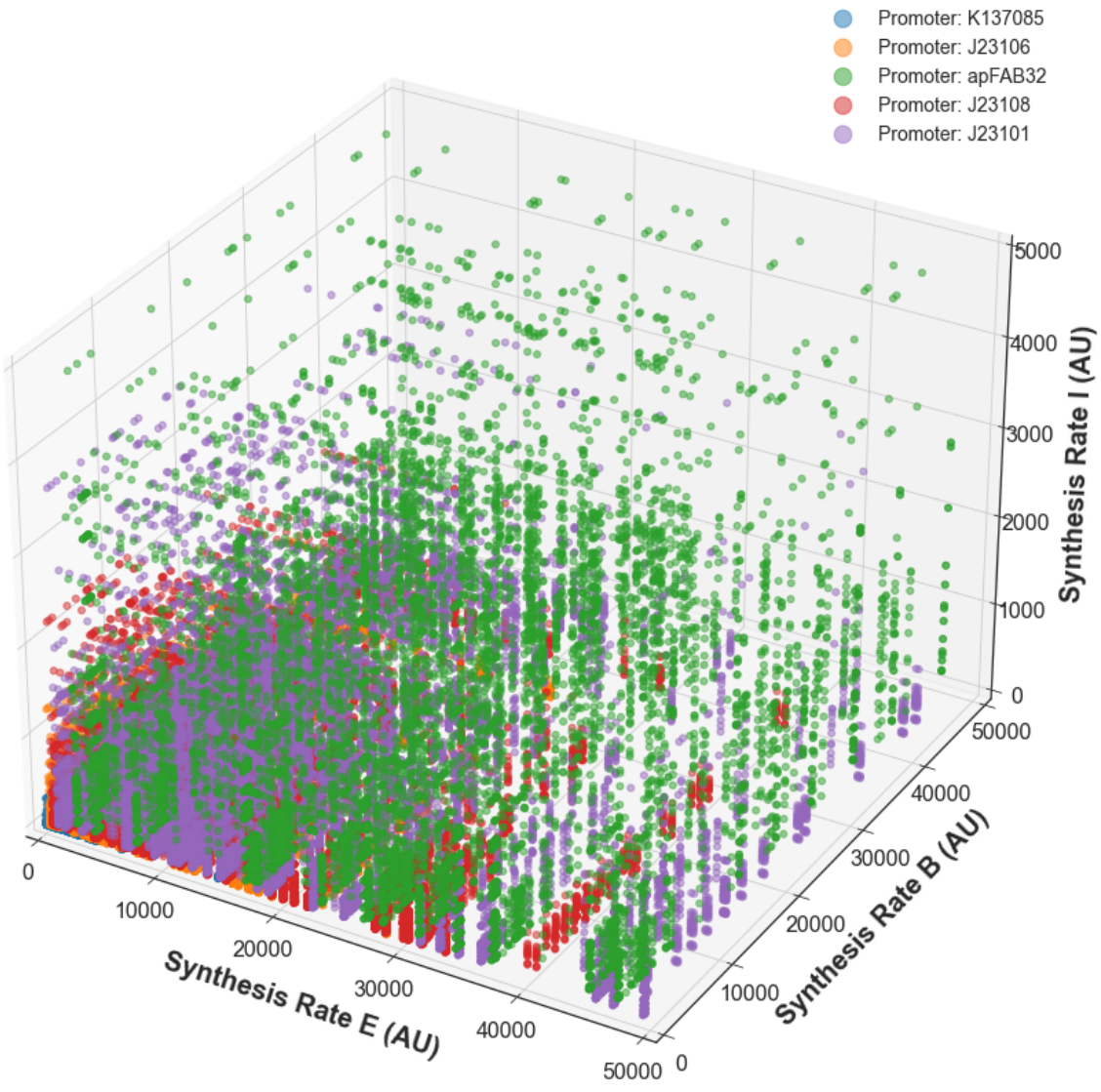
C3-Figure S11 – Complete design space with the entire Biolegio library - stratified by pool/gene order.



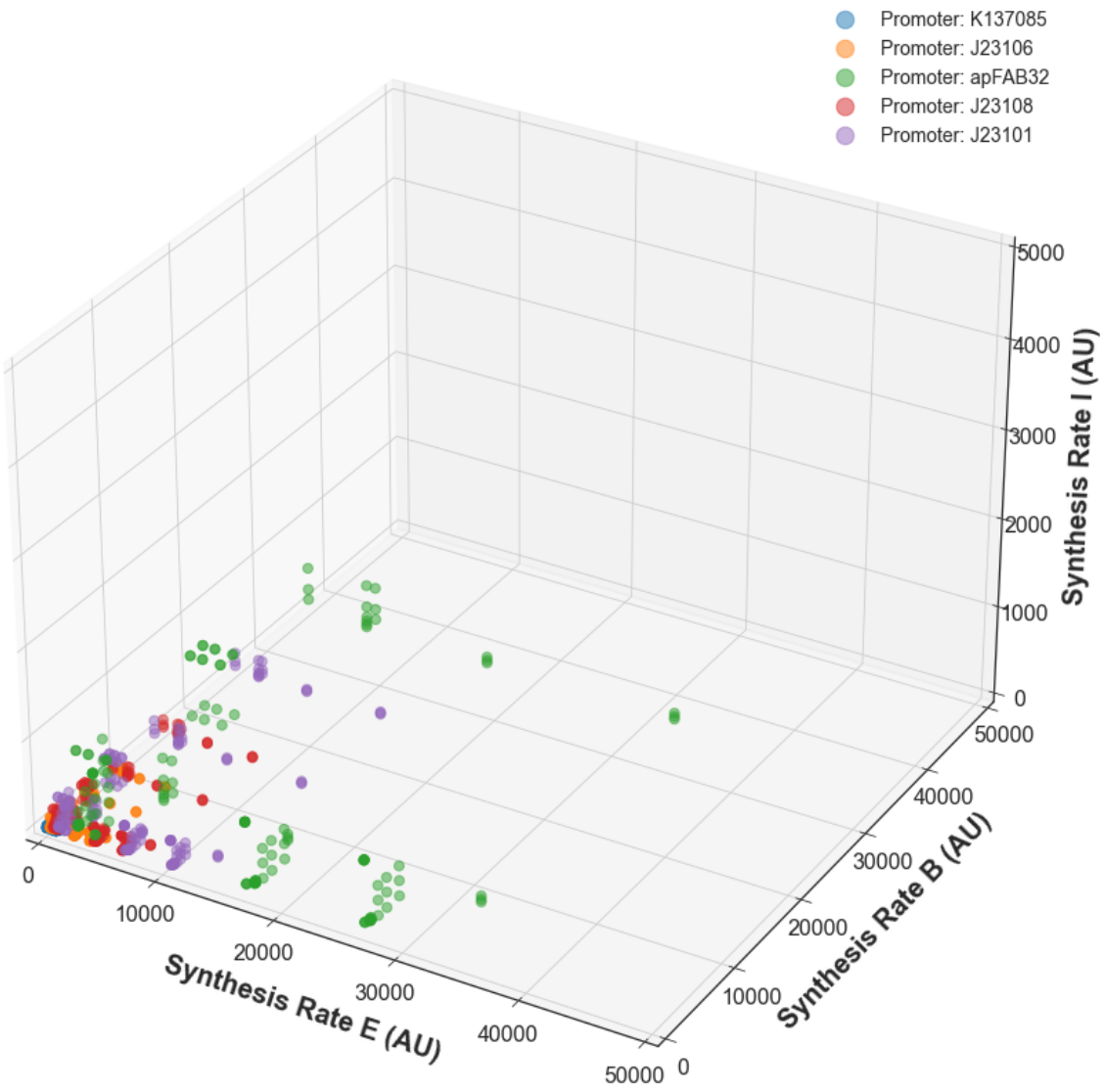
C3-Figure S12 – Design space for the {RBS1,RBS2,RBS3} subset - stratified by pool/gene order.

C3 - Visualisation By Promoter

Another way to visualise the design space is to stratify it according to the leading promoter of the operon. Again, the 15-strong Biolegio RBS library (Figure S13) and RBS 1,2 and 3 (Figure S14) are used for this purpose. Results show spatial distribution is clearly associated with promoter strength. Contrary to RBS, this time the results are more predictable since synthesis rates are proportional to promoter strength with the lycopene operon design. Practically the stronger the promoter, the more likely the construct is to be far from the origin.



C3-Figure S13 – Complete design space with the Entire Biolegio library - stratified by leading promoter.



C3-Figure S14 – Design space for the {RBS1,RBS2,RBS3} subset - stratified by leading promoter.

C3 - JSON Template

The "template" portion of the file encodes the construct template (structural constraints and corresponding combinatorics). A template lists its components, while a component lists the source for the library containing the sequence data, as well as positional information for the component. In the case of the two-gene operon used as an example in Figure 3(A), the "template" structure is as follows. To make the hierarchical structure more evident, the instances of components have been shaded in green.

```
"template": {
    # Template Description (type Template)
    # Contains sourcing information
    # as  well as header information
    "name": "Two Gene Operon",    # Template Name
    "UID": "two-gene-example",    # Internal UID
    "description": "Two-gene operon design",

    "structure":  [
    # Template Structure
    # All template components are listed here

    {     # Template component (type Component)
    "type": "utr-rbs",                  # Component type
    "library_source": "library_example_utr1_rbs", # Source
    "template_position": 2     # Template position (relative)
    },

    [.... ]    # Not all components are shown

    {
    "type": "backbone",
    "library_source": "library_example_backbone_seva26",
    "template_position": 7
     }
      ]
 }
```

Sourcing information is encoded into another portion of the file (in a structure called "component_sources"). All component libraries are listed - each one points to a repository address, and lists its elements. An element points towards the file containing the sequence of the element. To simplify data import, only text, JSON, GenBank and SBOL file formats are supported. In the case of the two-gene operon, the "component_sources" structure is as follows. To make the hierarchical structure more evident, the instances of libraries have been shaded in pink.

```
"component_sources": {

"library_example_utr1_rbs" : {
    # Library for UTR (type Library)
    # Contains sourcing information for the library
    # as  well as header information

    "score_threshold": 0.9, # Matching threshold
    "name":  "Example UTR-RBS", # Library Name
    "type":  "utr-rbs", # Type of all elements
    "Repository": # Location of data (local folder)
"supplementary/Real_Life_Example_Operon/repo/example",

    "parts": [
    # Lists all library elements (type Library_Element)

    # First element of library (type Library_Element)
    # Data in txt format
    {"name": "UTR1-RBS-A01", "filetype": "text", "filename": "UTR1-RBS-A01.txt"},

    [....]    # Not all elements are shown

    {"name": "UTR1-RBS-A12", "filetype": "text", "filename": "UTR1-RBS-A12.txt"} ]
    },

    [....]    # Not all libraries are shown

"library_example_backbone_seva26" : {
    # Library for Backbone (type Library)
    [...]    # Not all attributes are shown

    "repository": "supplementary/Real_Life_Example_Operon/repo/example",

     "parts": [
    # Only one element - Data in Genbank format
    {"name": "pSEVA26", "filetype": "genbank", "filename": "SEVA_26.gb"}]
    }
}
```

C3 - cMatch Logs

Logs can be found in a git repository on Github at: https://github.com/kitneylab/cmatch/logs/

As an example, the log from CM_2 with the Violacein example (1 vs All) with a Threshold of 0.75. File: run_cm2-vio-1-easy-1vsAll-B0030-B0031-B0032-B0033-B0064-th75-20211111-202448.log

```
2021-11-11 20:24:48,413:INFO: CM_2: Violacein - 1 vs All - Easy B0030-B0064 -
Threshhold: 0.75 - Targets:['rbs_tu_easy/'] -
Template:template_violacein_02_tu_1vsAll.json - Nb runs: 10
2021-11-11 20:24:48,413:INFO: Test run: (1/10)
2021-11-11 20:24:48,444:INFO: Target sequence: vio_A-0030_tu1
2021-11-11 20:24:48,465:INFO: J23101
2021-11-11 20:24:48,619:INFO: Time spent on match_part(): 0:00:00.153934
2021-11-11 20:24:48,621:INFO: B0030
2021-11-11 20:24:48,745:INFO: Time spent on match_part(): 0:00:00.124298
2021-11-11 20:24:48,745:INFO: B0031
2021-11-11 20:24:48,757:INFO: Time spent on match_part(): 0:00:00.011666
2021-11-11 20:24:48,757:INFO: B0032
2021-11-11 20:24:48,769:INFO: Time spent on match_part(): 0:00:00.011629
2021-11-11 20:24:48,769:INFO: B0033
2021-11-11 20:24:48,779:INFO: Time spent on match_part(): 0:00:00.010518
2021-11-11 20:24:48,780:INFO: B0064
2021-11-11 20:24:48,898:INFO: Time spent on match_part(): 0:00:00.118232
2021-11-11 20:24:48,899:INFO: VioA
2021-11-11 20:24:50,088:INFO: Time spent on match_part(): 0:00:01.188616
2021-11-11 20:24:50,088:INFO: B0015
2021-11-11 20:24:50,209:INFO: Time spent on match_part(): 0:00:00.120052
2021-11-11 20:24:50,210:INFO: Time spent on match_libs(): 0:00:01.765092
2021-11-11 20:24:50,213:INFO: Target sequence: vio_B-0031_tu2
2021-11-11 20:24:50,214:INFO: J23101
2021-11-11 20:24:50,813:INFO: Time spent on match_part(): 0:00:00.599429
2021-11-11 20:24:50,815:INFO: B0030
2021-11-11 20:24:50,841:INFO: Time spent on match_part(): 0:00:00.025693
2021-11-11 20:24:50,841:INFO: B0031
2021-11-11 20:24:51,397:INFO: Time spent on match_part(): 0:00:00.556183
2021-11-11 20:24:51,397:INFO: B0032
2021-11-11 20:24:51,422:INFO: Time spent on match_part(): 0:00:00.024883
2021-11-11 20:24:51,422:INFO: B0033
2021-11-11 20:24:51,441:INFO: Time spent on match_part(): 0:00:00.018576
2021-11-11 20:24:51,441:INFO: B0064
2021-11-11 20:24:51,996:INFO: Time spent on match_part(): 0:00:00.555243
2021-11-11 20:24:51,998:INFO: VioB
2021-11-11 20:24:58,062:INFO: Time spent on match_part(): 0:00:06.063767
2021-11-11 20:24:58,064:INFO: B0015
2021-11-11 20:24:58,350:INFO: Time spent on match_part(): 0:00:00.285827
2021-11-11 20:24:58,350:INFO: Time spent on match_libs(): 0:00:08.137033
2021-11-11 20:24:58,353:INFO: Target sequence: vio_C-0032_tu3
2021-11-11 20:24:58,354:INFO: J23101
2021-11-11 20:24:58,526:INFO: Time spent on match_part(): 0:00:00.171877
2021-11-11 20:24:58,533:INFO: B0030
2021-11-11 20:24:58,551:INFO: Time spent on match_part(): 0:00:00.018512
2021-11-11 20:24:58,551:INFO: B0031
```

```
2021-11-11 20:24:58,571:INFO: Time spent on match_part(): 0:00:00.019409
2021-11-11 20:24:58,571:INFO: B0032
2021-11-11 20:24:58,723:INFO: Time spent on match_part(): 0:00:00.151524
2021-11-11 20:24:58,723:INFO: B0033
2021-11-11 20:24:58,734:INFO: Time spent on match_part(): 0:00:00.011282
2021-11-11 20:24:58,734:INFO: B0064
2021-11-11 20:24:58,811:INFO: Time spent on match_part(): 0:00:00.076580
2021-11-11 20:24:58,813:INFO: VioC
2021-11-11 20:25:00,248:INFO: Time spent on match_part(): 0:00:01.435305
2021-11-11 20:25:00,250:INFO: B0015
2021-11-11 20:25:00,389:INFO: Time spent on match_part(): 0:00:00.138770
2021-11-11 20:25:00,390:INFO: Time spent on match_libs(): 0:00:02.035930
2021-11-11 20:25:00,395:INFO: Target sequence: vio_D-0033_tu4
2021-11-11 20:25:00,398:INFO: J23101
2021-11-11 20:25:00,547:INFO: Time spent on match_part(): 0:00:00.149733
2021-11-11 20:25:00,550:INFO: B0030
2021-11-11 20:25:00,571:INFO: Time spent on match_part(): 0:00:00.020280
2021-11-11 20:25:00,571:INFO: B0031
2021-11-11 20:25:00,585:INFO: Time spent on match_part(): 0:00:00.014482
2021-11-11 20:25:00,586:INFO: B0032
2021-11-11 20:25:00,600:INFO: Time spent on match_part(): 0:00:00.014145
2021-11-11 20:25:00,600:INFO: B0033
2021-11-11 20:25:00,717:INFO: Time spent on match_part(): 0:00:00.117081
2021-11-11 20:25:00,717:INFO: B0064
2021-11-11 20:25:00,728:INFO: Time spent on match_part(): 0:00:00.011049
2021-11-11 20:25:00,729:INFO: VioD
2021-11-11 20:25:02,013:INFO: Time spent on match_part(): 0:00:01.283453
2021-11-11 20:25:02,015:INFO: B0015
2021-11-11 20:25:02,141:INFO: Time spent on match_part(): 0:00:00.125988
2021-11-11 20:25:02,142:INFO: Time spent on match_libs(): 0:00:01.746392
2021-11-11 20:25:02,160:INFO: Target sequence: vio_E-0064_tu5
2021-11-11 20:25:02,161:INFO: J23101
2021-11-11 20:25:02,217:INFO: Time spent on match_part(): 0:00:00.055611
2021-11-11 20:25:02,218:INFO: B0030
2021-11-11 20:25:02,258:INFO: Time spent on match_part(): 0:00:00.039475
2021-11-11 20:25:02,258:INFO: B0031
2021-11-11 20:25:02,267:INFO: Time spent on match_part(): 0:00:00.008758
2021-11-11 20:25:02,267:INFO: B0032
2021-11-11 20:25:02,276:INFO: Time spent on match_part(): 0:00:00.008485
2021-11-11 20:25:02,276:INFO: B0033
2021-11-11 20:25:02,282:INFO: Time spent on match_part(): 0:00:00.005471
2021-11-11 20:25:02,282:INFO: B0064
2021-11-11 20:25:02,324:INFO: Time spent on match_part(): 0:00:00.042113
2021-11-11 20:25:02,325:INFO: VioE
2021-11-11 20:25:02,656:INFO: Time spent on match_part(): 0:00:00.330932
2021-11-11 20:25:02,657:INFO: B0015
2021-11-11 20:25:02,720:INFO: Time spent on match_part(): 0:00:00.062718
2021-11-11 20:25:02,721:INFO: Time spent on match_libs(): 0:00:00.560369
2021-11-11 20:25:02,725:INFO: Time spent on reconstruct(): 0:00:00.002372
```

Appendices C4

C4 - Methods

Preparation of the lycopene standard

10 mg Standard pharma-grade lycopene powder (Sigma-Aldrich Lycopene >=98% (HPLC) from Tomato SMB00706, REF: PHR1770-5X100MG 5 X 100 MG) was weighted on a precision scale (REF: OHAUS Pioneer PA214 Analytical Balance) in 20mL pure DMSO (Fisher Scientific UK Ltd, DMSO, Dimethyl Sulfoxide, GC Headspace Grade, Fisher Chemical, Quantity: 1L, Packaging: Glass Bottle, CAS: 67-68-5, Colourless, Grade: GC Headspace, Molecular Formula: C2H6OS, Molecular Weight: 78.129, Percent Purity: >-99.9%, REF: 15572393) in a 50 mL Falcon tube, wrapped in aluminium foil to avoid exposure to light and vortexed (REF: Vortex IR StarLab Smart Instruments) for 5 minutes. This gives us an initial lycopene standard concentration C0.

Automated dilution liquid-handler setup

Automation was carried out on an automated liquid handler (Analytik Jena CyBio Felix) with an 8-channel pipette head (CyBi-FeliX Head 8/250µL, CyBi-FeliX SELECT Head 8/250µL, REF: OL318-14-100).
DMSO was poured in a reservoir (Thermo Scientific Nalgene Robotic Reservoirs Flat Bottom, Sterile, Size 300mL, REF: 1200-1301) and placed on deck position 11. (See Supplementary Information for the CyBio-FeliX deck layout).
A volume of 10 mL Lycopene standard at initial concentration C0 was manually pipetted in a 12-channel reservoir plate (4titude 12 Channel Reservoir Plate, 21 ml square channels, pyramid bottom, REF: 4ti-0131) and placed on deck position 8.
Two boxes of 250µL tips (AJ CyBio RoboTipTray 96/250µL, REF: OL3810-25-669) were placed on deck position 1 and 2.
For the serial dilution protocol, a mixing deep-well plate (Greiner Masterblock 96 well, 2mL, PP, V-Bottom, Natural, REF: 780285) was placed on deck position 9.
A measurement plate (Greiner Flat-bottom plate REF: 655161) was placed on deck position 12.

Food colourant

Sunset red (E110), Royal Blue (E110, E123, E133), Fruit Green (E102, E133) (Ingredients: water, sodium carboxymethylcellulose, sorbitol, propylene, glycol, glycerin) (REF: Ktdorns Food colouring)

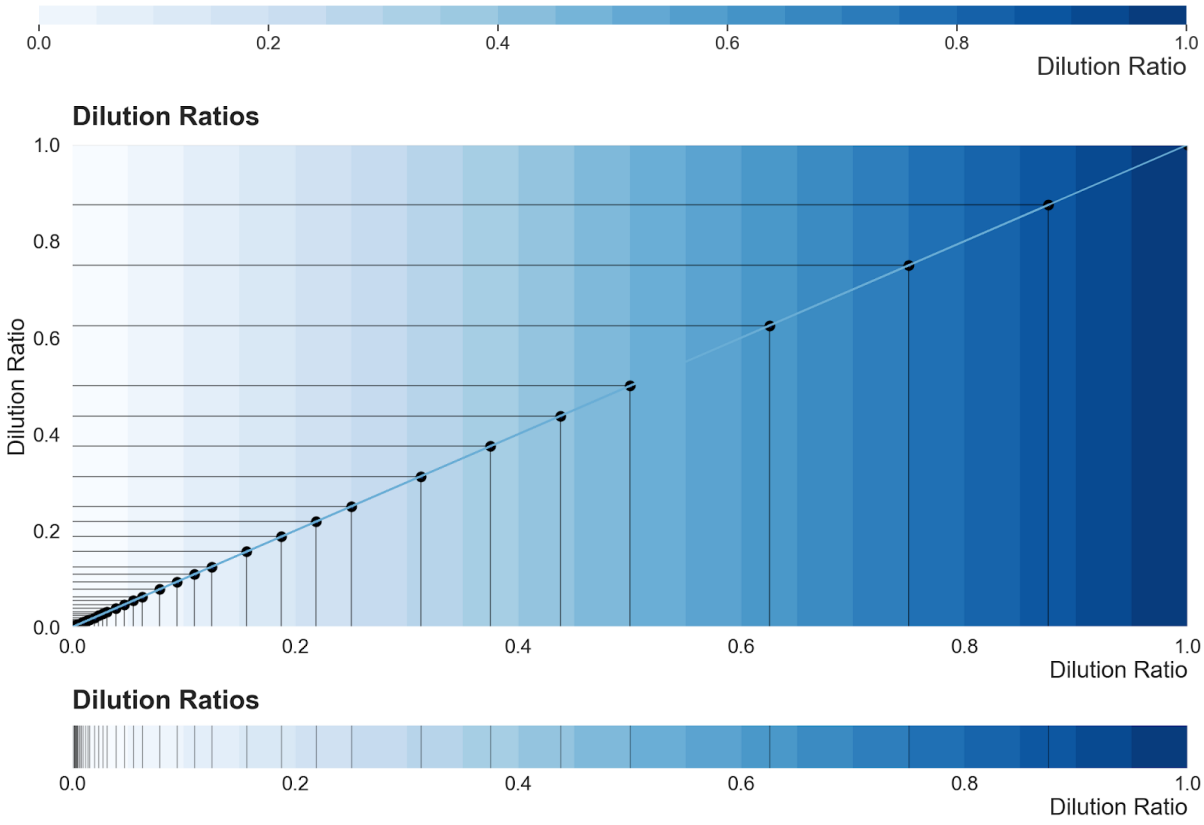
Spectral measurements

UV/vis absorbance (full spectrum: wavelength: 350-650 nm, wavelength step width: 2 nm) measurements were done in two different microplate readers of the same model (REF: BMG LABTECH CLARIOstar Plus).

C4 - Section 1 - Implemented Dilution Schemes

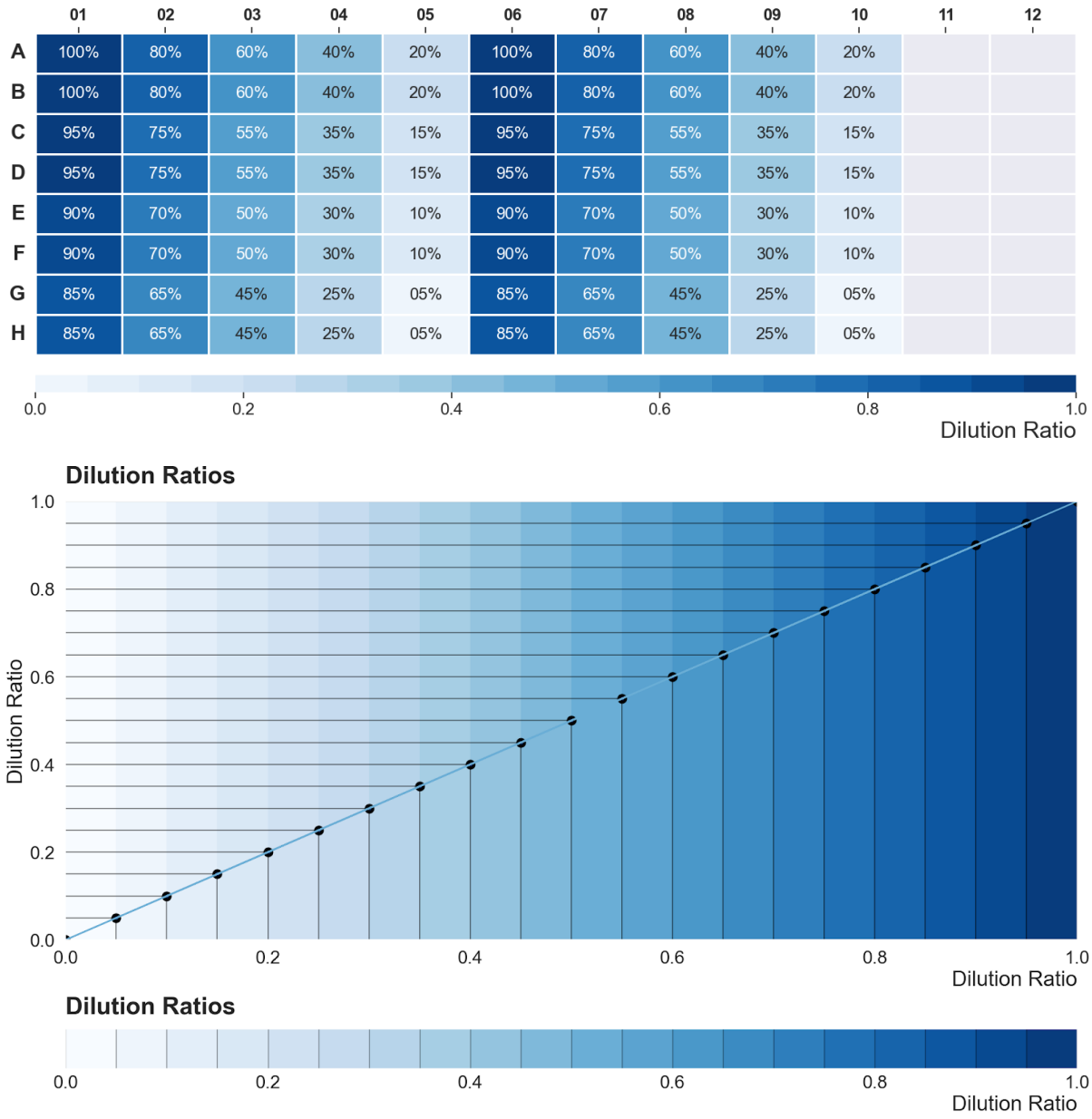
1.1 - Geometric dilution scheme

	01	02	03	04	05	06	07	08	09	10	11	12
A	1	1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/256	1/512		
B	1	1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/256	1/512		
C	7/8	7/16	7/32	7/64	7/128	7/256	7/512	7/1024	7/2048	7/4096		
D	7/8	7/16	7/32	7/64	7/128	7/256	7/512	7/1024	7/2048	7/4096		
E	3/4	3/8	3/16	3/32	3/64	3/128	3/256	3/512	3/1024	3/2048		
F	3/4	3/8	3/16	3/32	3/64	3/128	3/256	3/512	3/1024	3/2048		
G	5/8	5/16	5/32	5/64	5/128	5/256	5/512	5/1024	5/2048	5/4096		
H	5/8	5/16	5/32	5/64	5/128	5/256	5/512	5/1024	5/2048	5/4096		



C4-Figure S1 – Plate map for the geometric scheme and corresponding dilution density.

1.2 - Linear dilution scheme



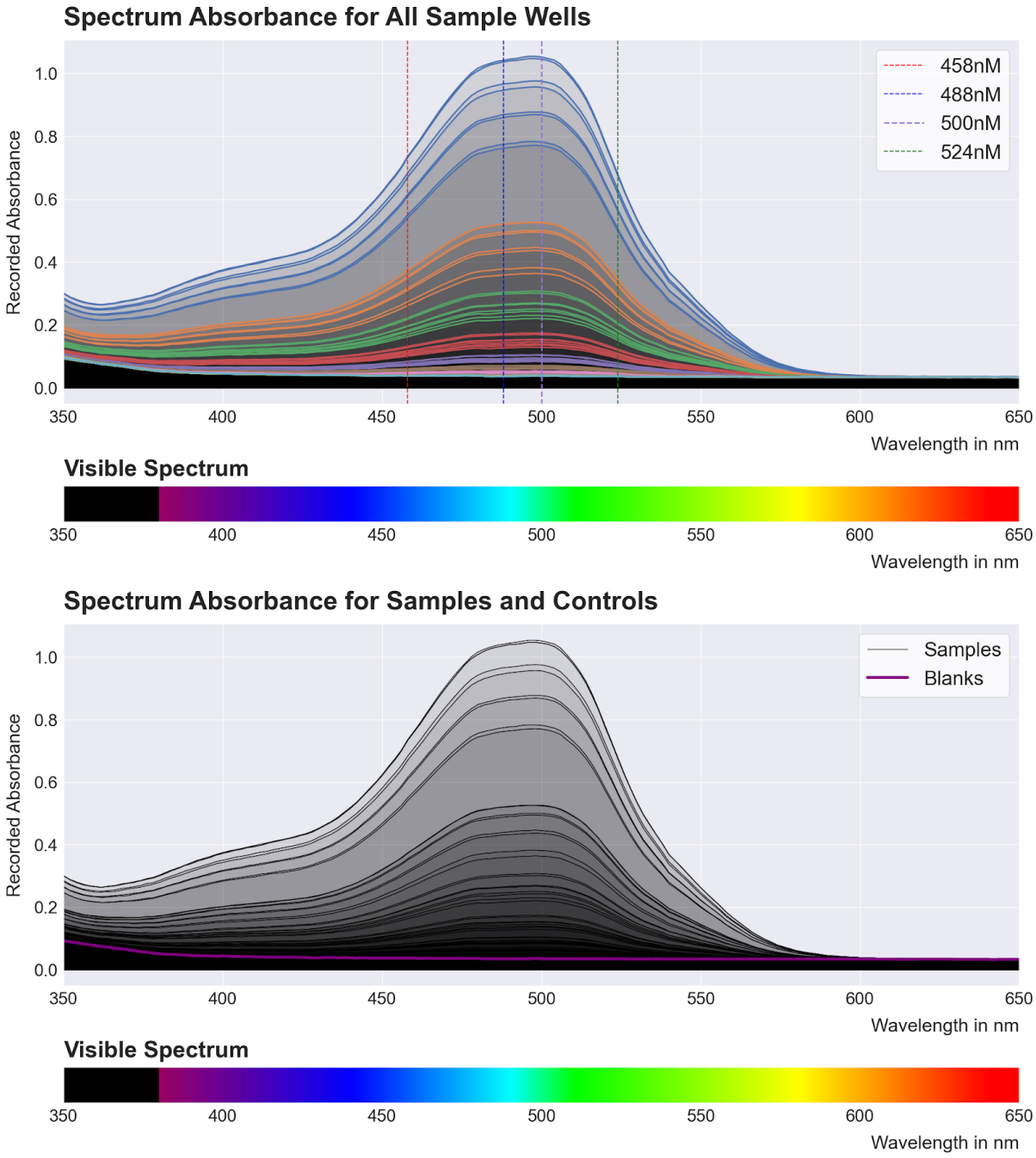
C4-Figure S2 – Plate map for the linear scheme and corresponding dilution density.

C4 - Section 2 - Geometric Dilution Scheme - Food Colorant and Water

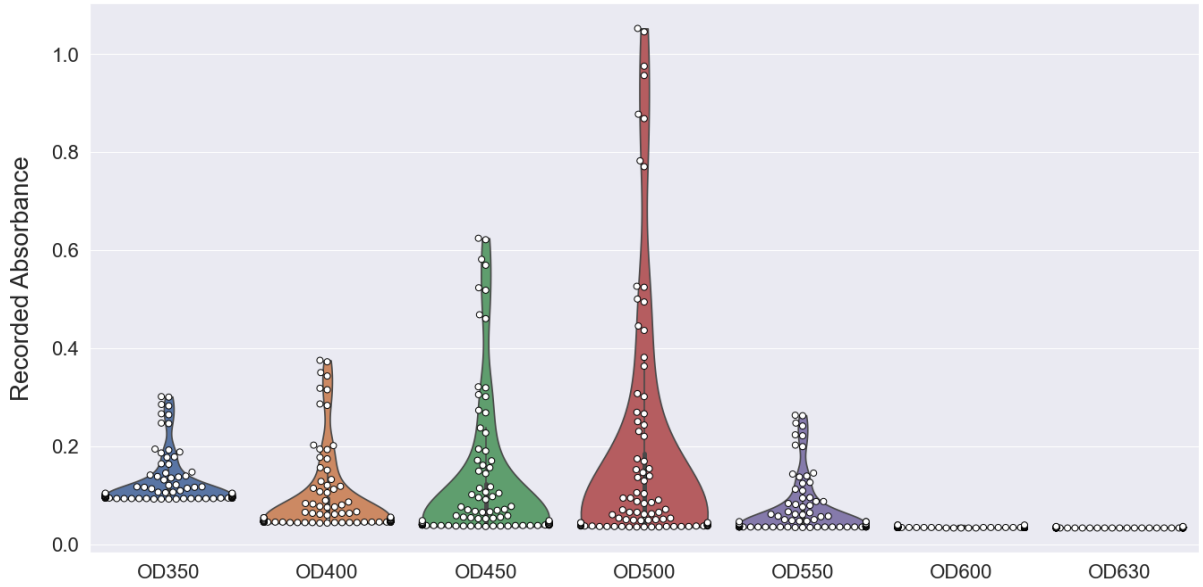
In this section we present the main experimental results for the preliminary study of the geometric dilution scheme.

2.1 - Run 55 - Plate 01 - automated

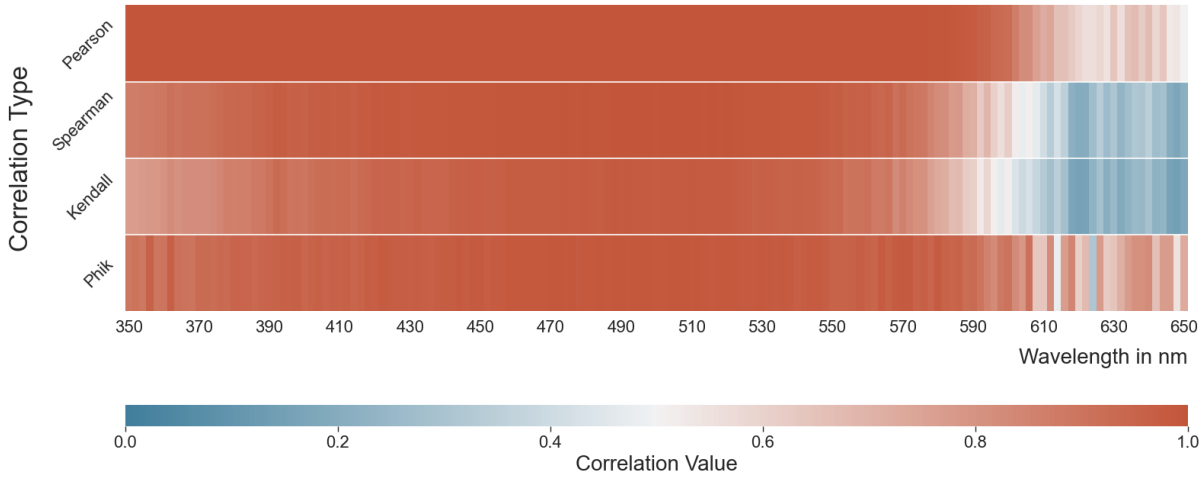
Food colorant in Water - Protocol V1.0.0 (Automated)



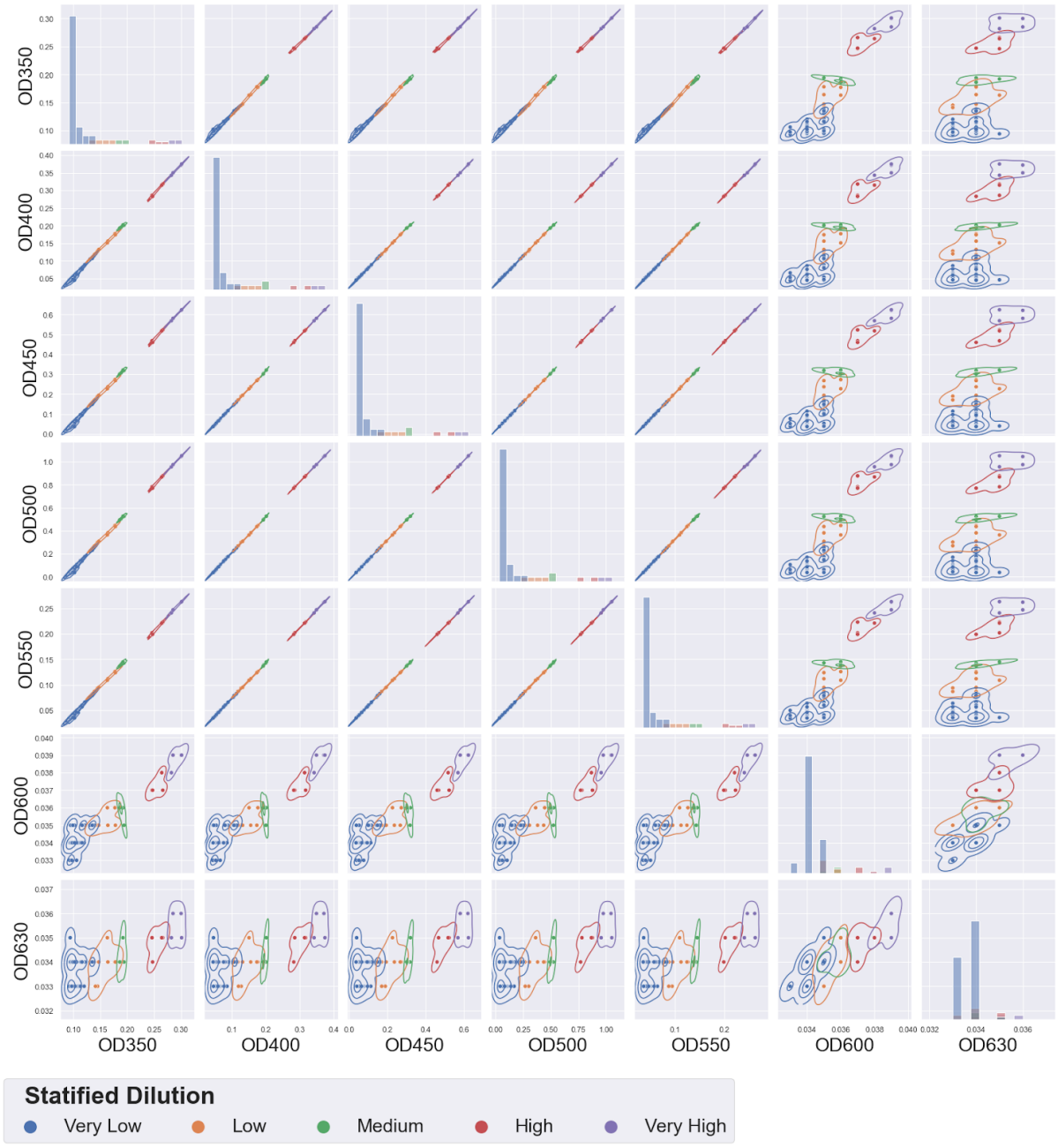
C4-Figure S3 – Run 55Plate 01Absorbance spectrum for samples and blanks.



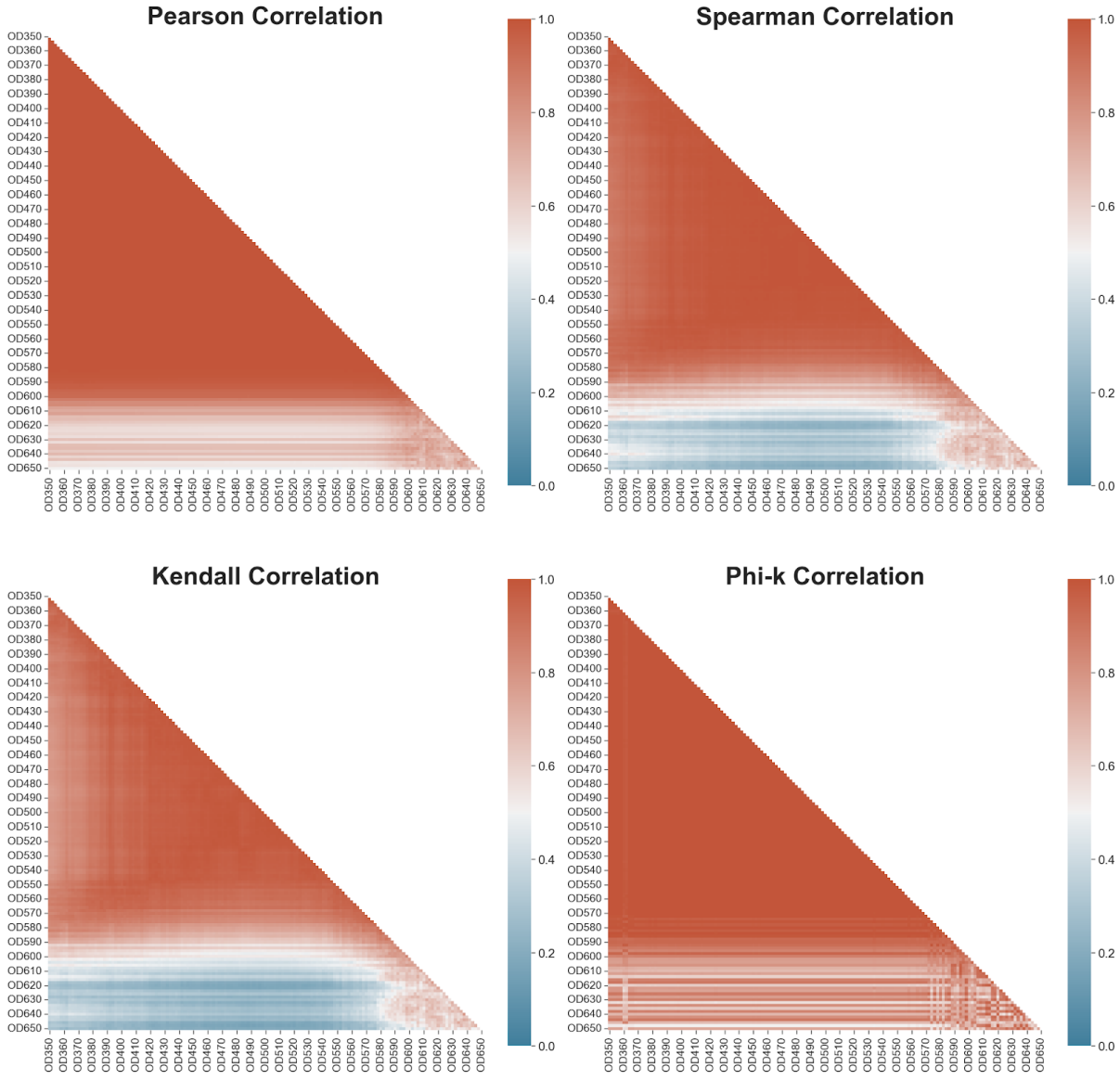
C4-Figure S4 – Run 55 - Plate 01 - Corresponding violin plot at selected wavelengths. Very asymmetrical due to lopsided dilution scheme - Accumulation at small concentrations. Similar shapes, albeit stretched out, for wavelengths from 350 to 550 - No information to be found above 600.



C4-Figure S5 – Run 55 - Plate 01 - Correlations between dilution ratio and measurements.



C4-Figure S6 – Run 55 - Plate 01 - Pairplot between measurements at selected wavelengths.
Very high correlation between measurements for wavelengths between 400 and 550 nm.



C4-Figure S7 – Run 55 - Plate 01 - Correlation matrices between measurements at different wavelengths.

2.2 - Run 55 - Plate 02 - manual

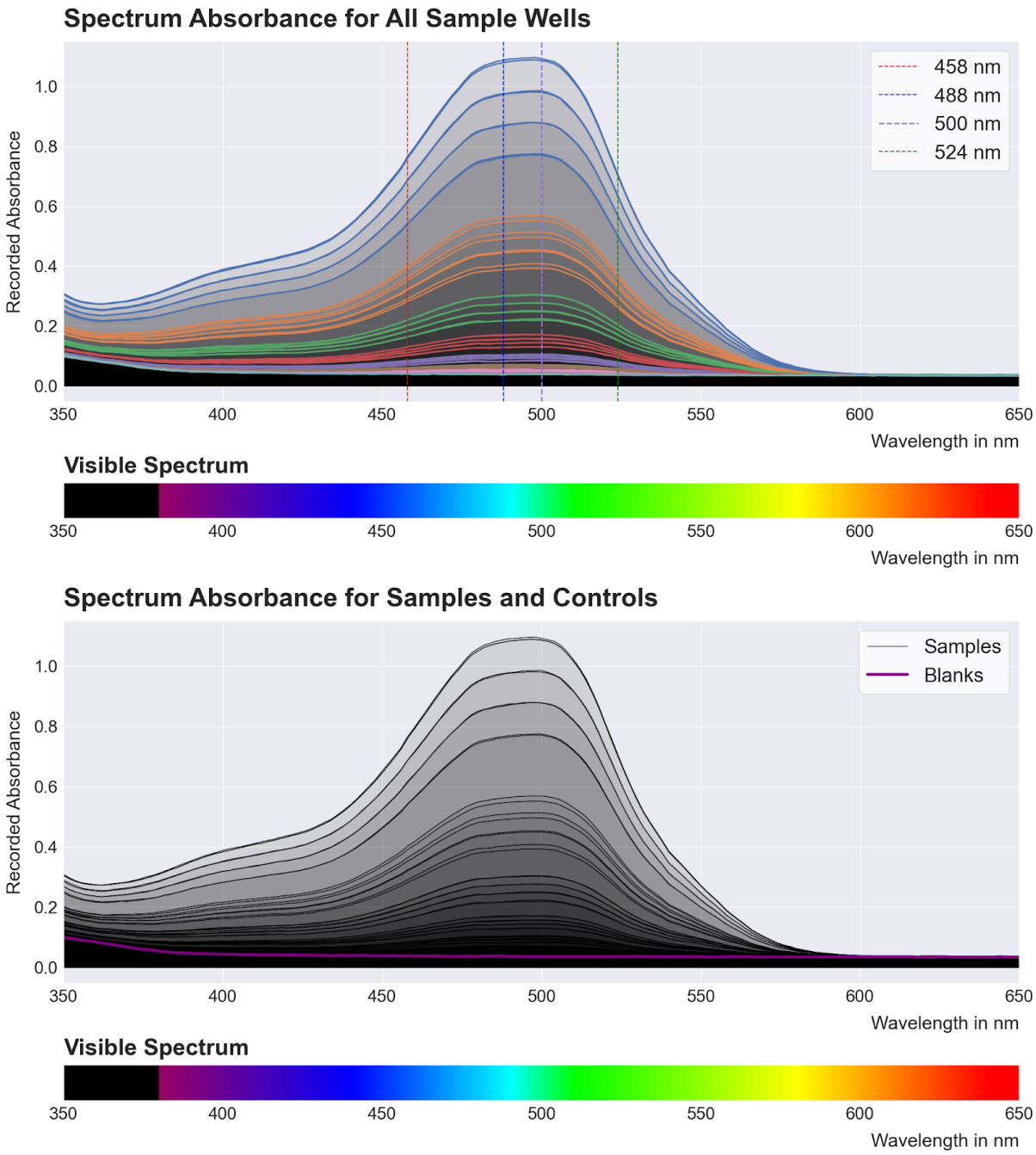
Food colourant and Water

Manual Protocol - Used same preparation mix (Tube 01) as Run 055 - Plate 01

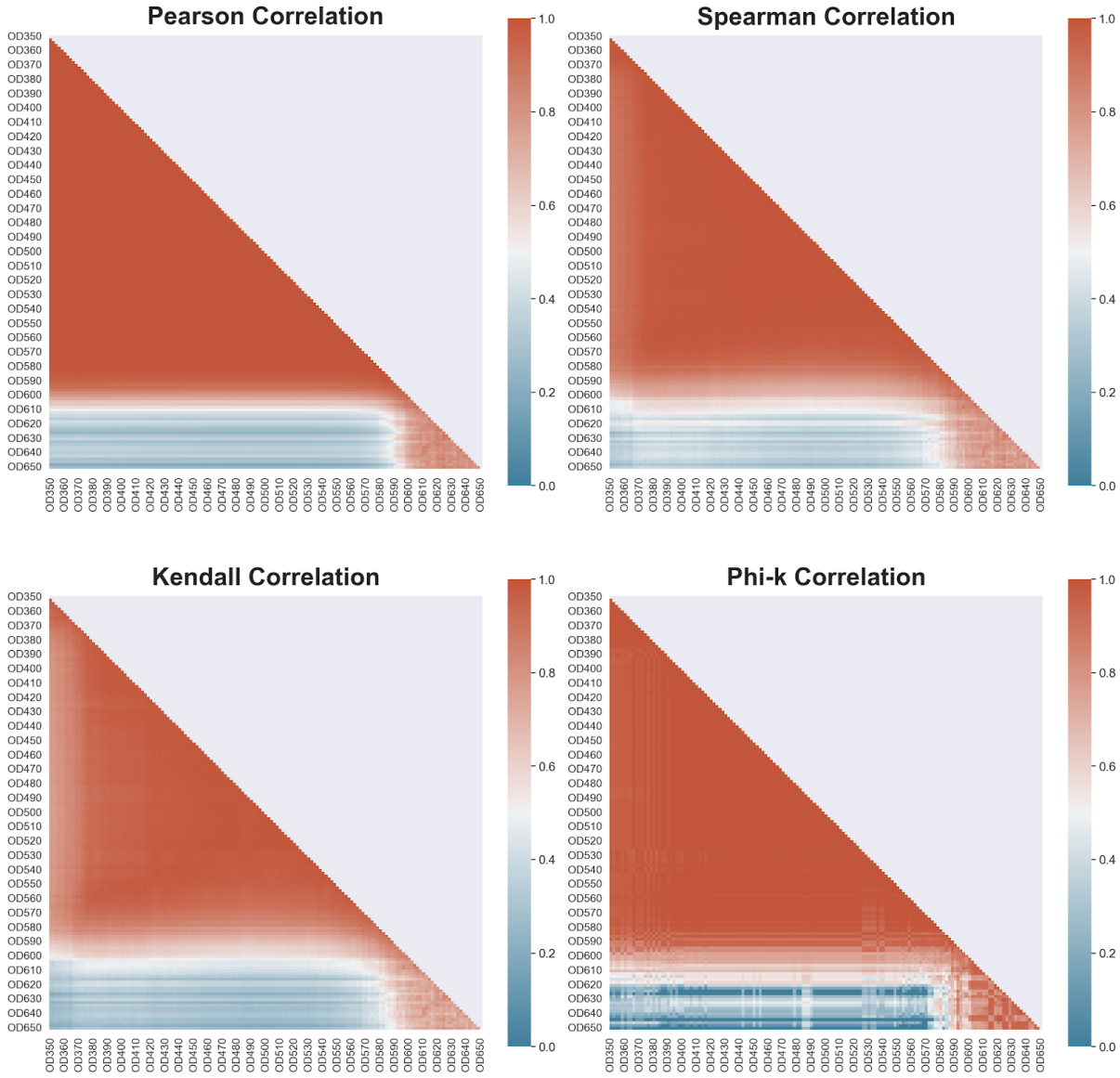
Tighter repeats (noticeable at larger concentrations).

Sharper correlations matrices.

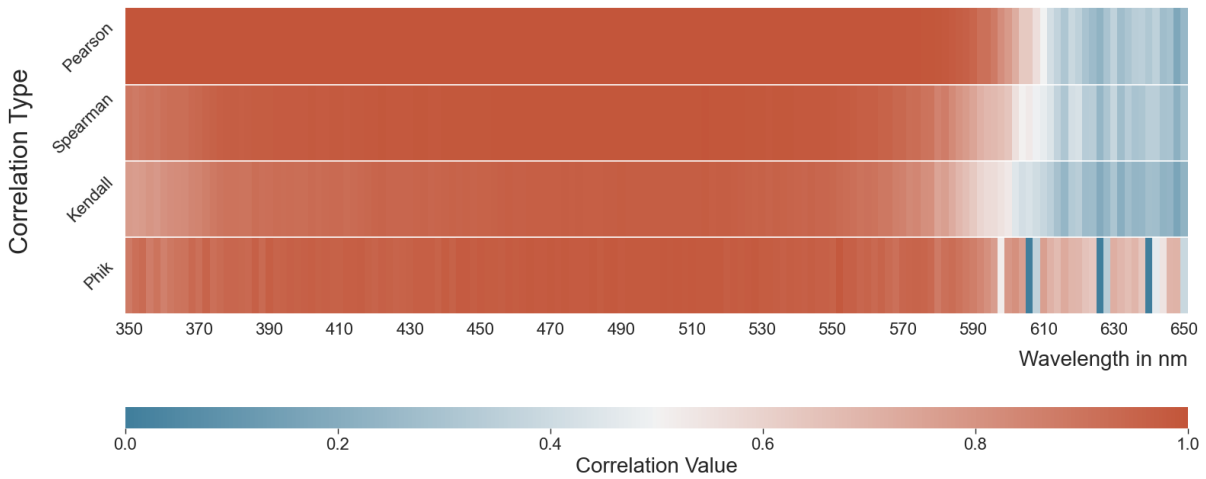
The lack of correlation at wavelengths larger than 600 nm is more pronounced.



C4-Figure S8 – Run 55 - Plate 02 - Absorbance spectrum for samples and blanks.



C4-Figure S9 – Run 55 - Plate 02 - correlation matrices between measurements.

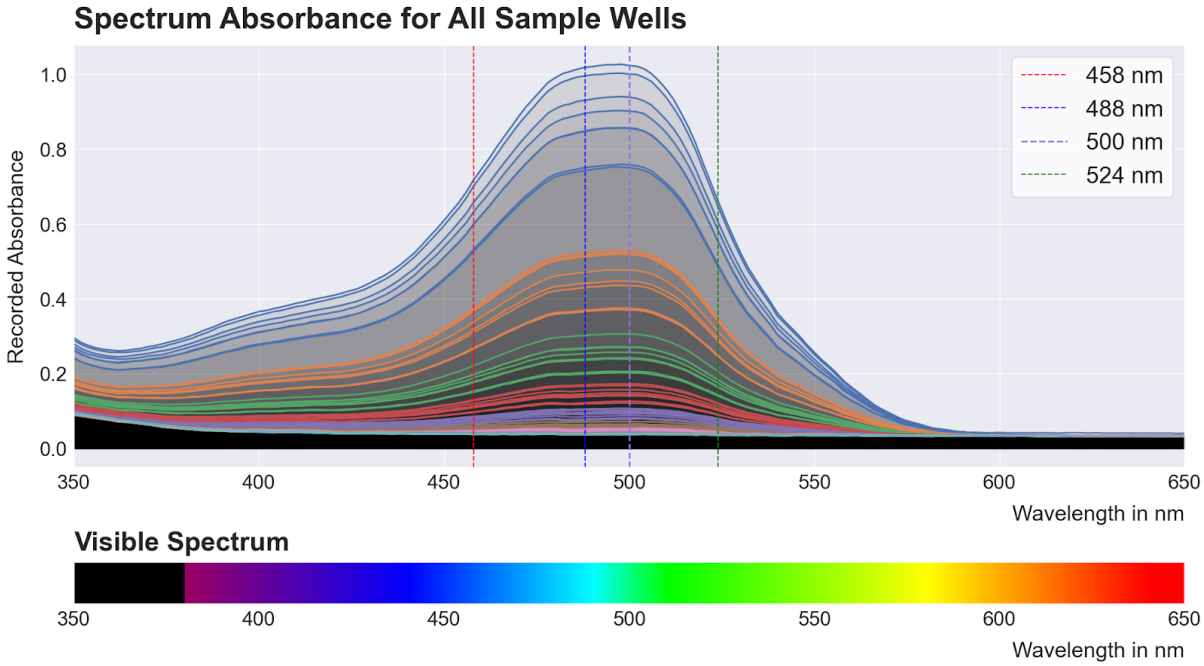


C4-Figure S10 – Run 55 - Plate 02 - Correlations between dilution ratio and measurements.

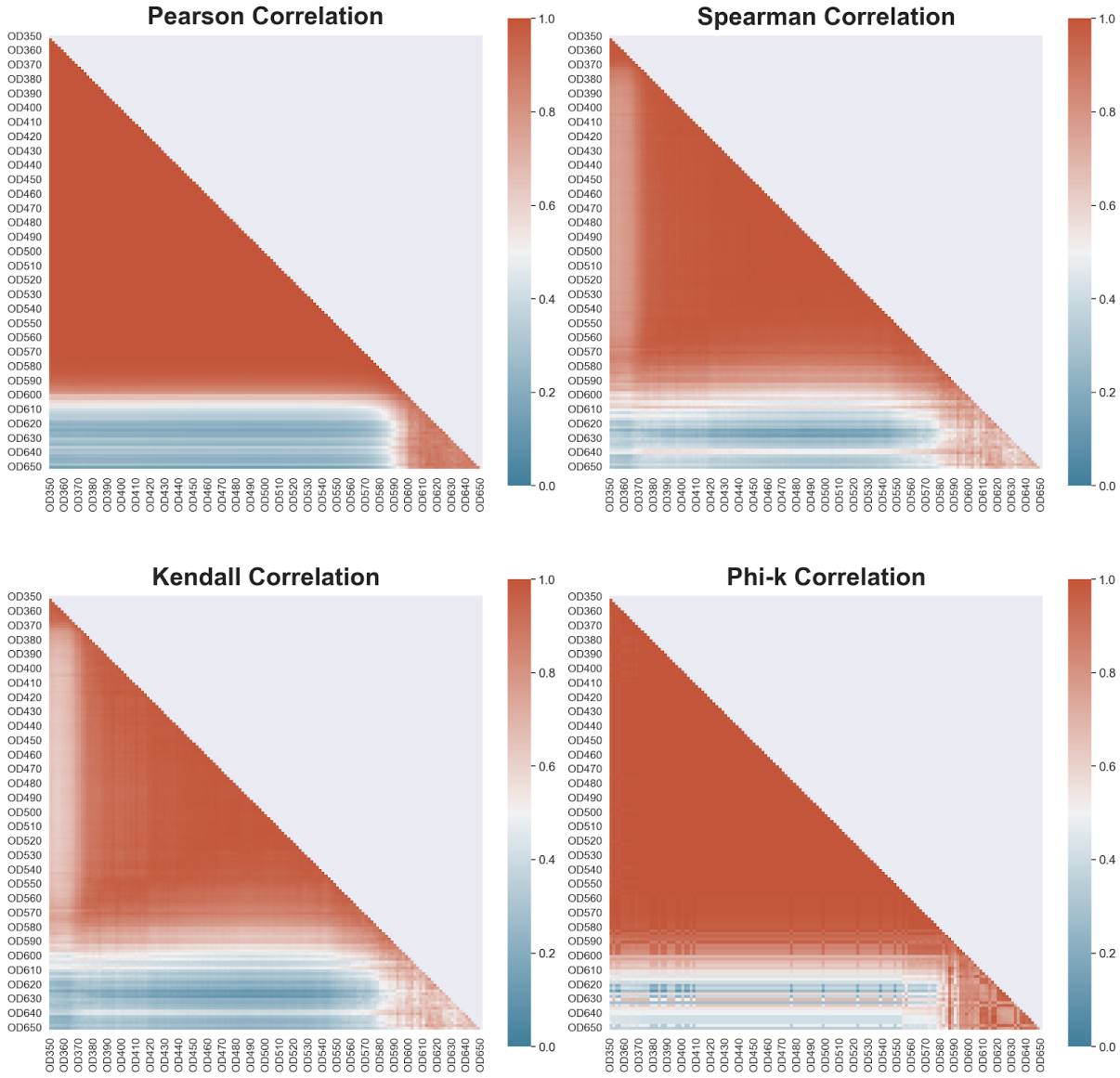
2.3 - Run 57 - Plate 01

Food colourant in Water (but Different Mix from Mix used in Run 055)

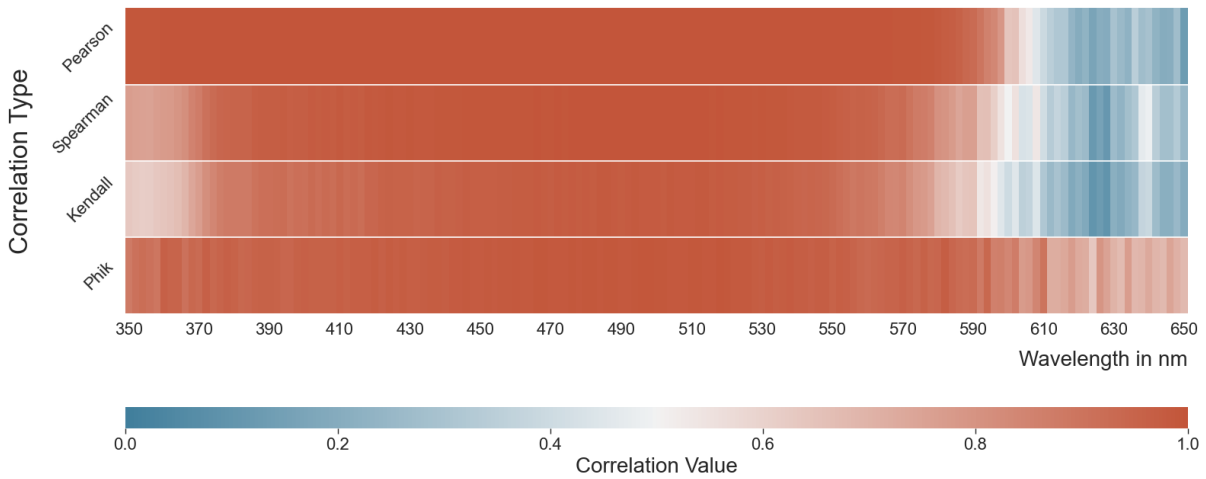
Protocol V1.0.0 (Automated)



C4-Figure S11 – Run 57 - Plate 01 - Absorbance spectrum for all samples.



C4-Figure S12 – Run 57 - Plate 01 - Correlation matrices between measurements.

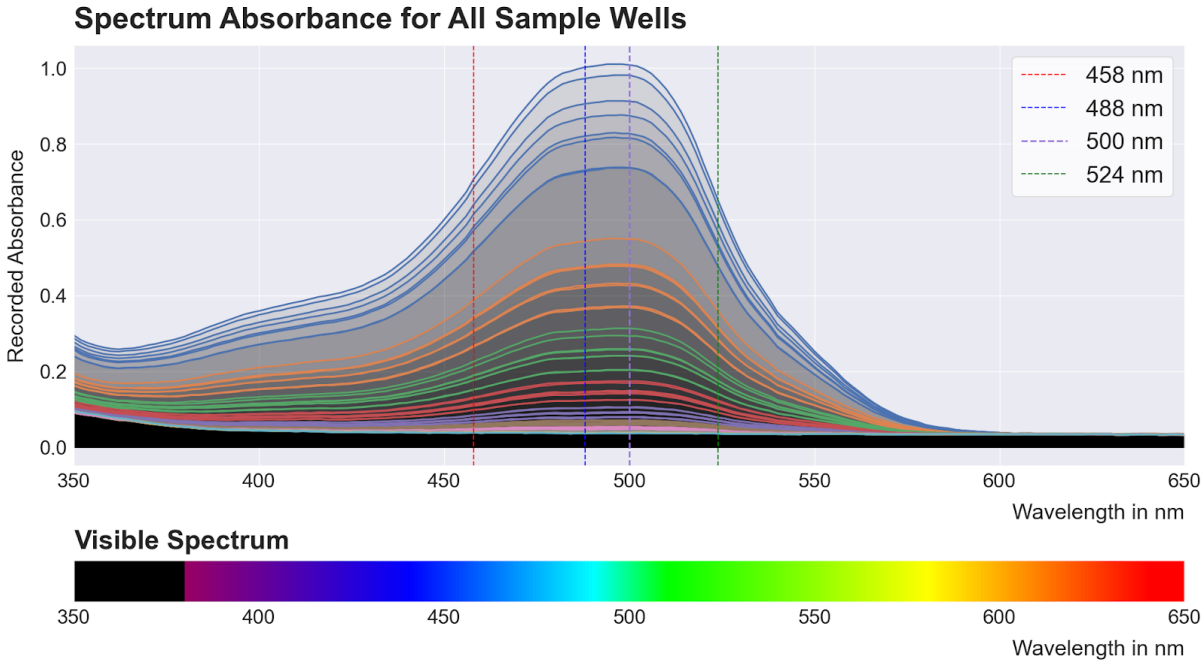


C4-Figure S13 – Run 57 - Plate 01 - Correlations between dilution ratio and measurements.

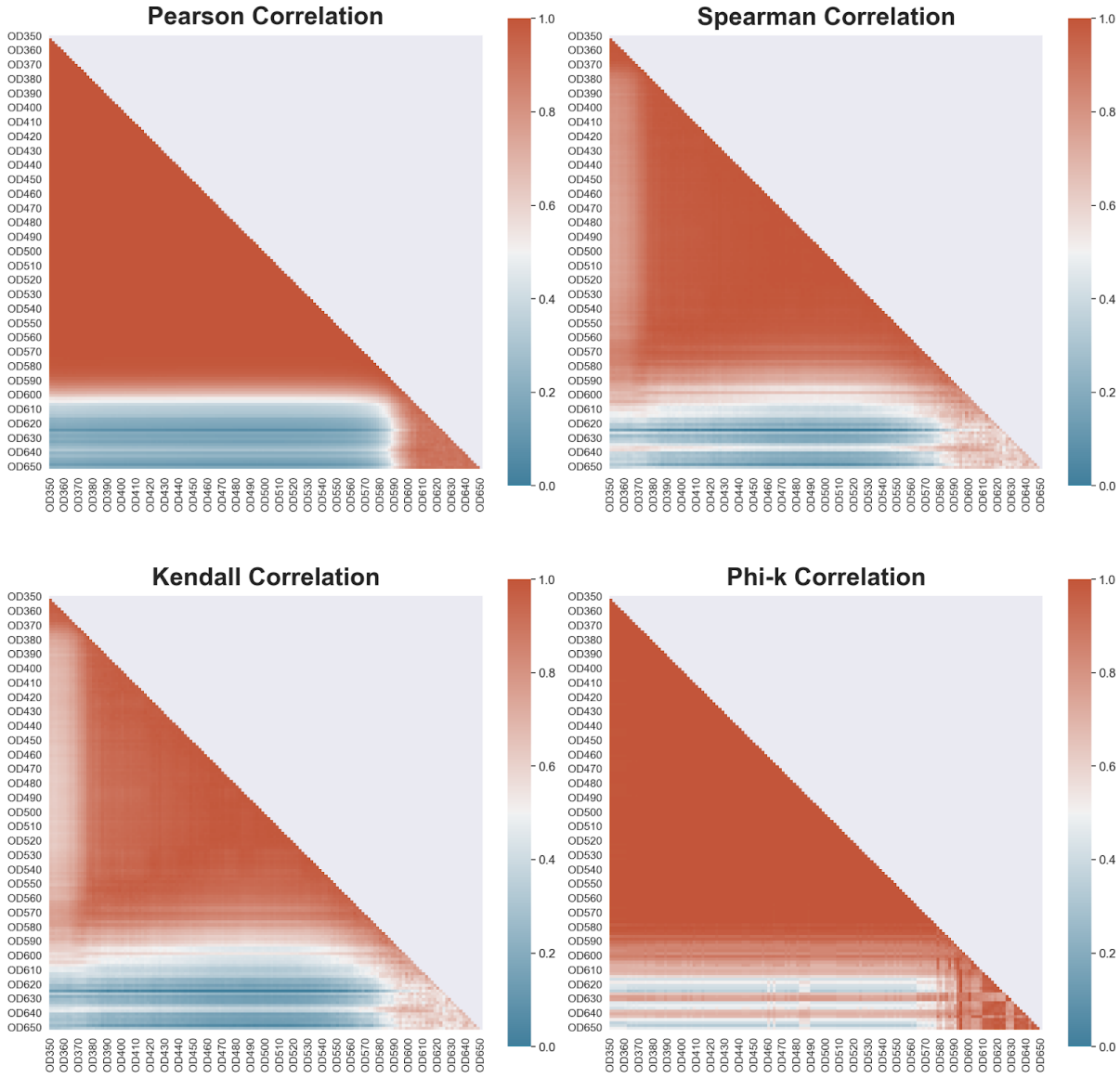
2.4 - Run 57 - Plate 02

Food colourant in Water (Same mix as used in Run 057 - Plate 01)

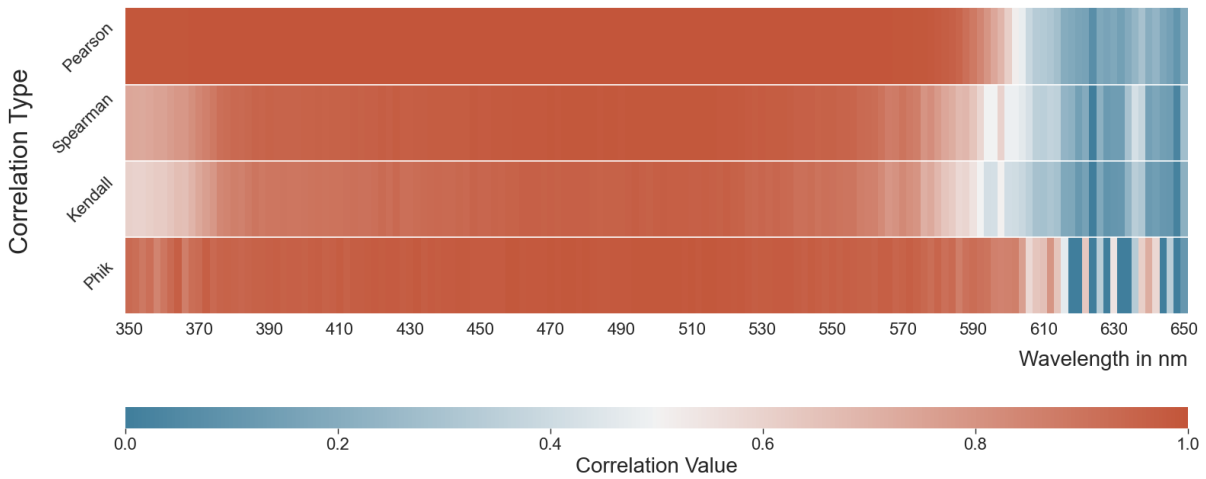
Protocol: Manual



C4-Figure S14 – Run 57 - Plate 02 - Absorbance spectrum for all samples.

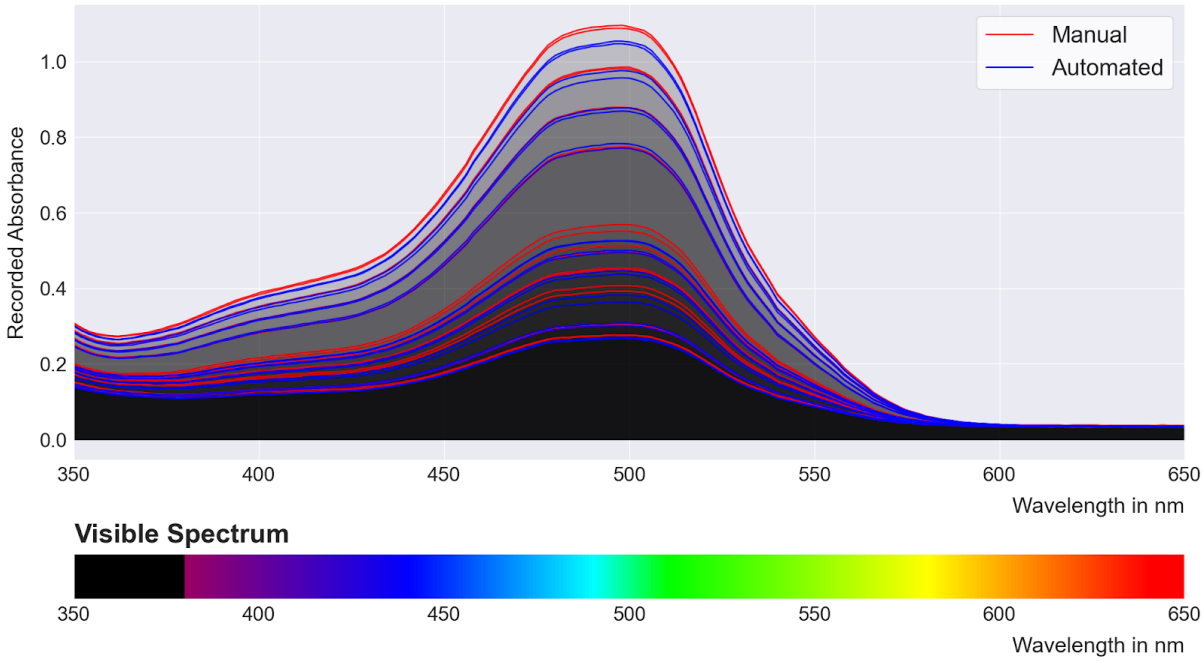


C4-Figure S15 – Run 57 - Plate 02 - Correlation matrices between measurements.

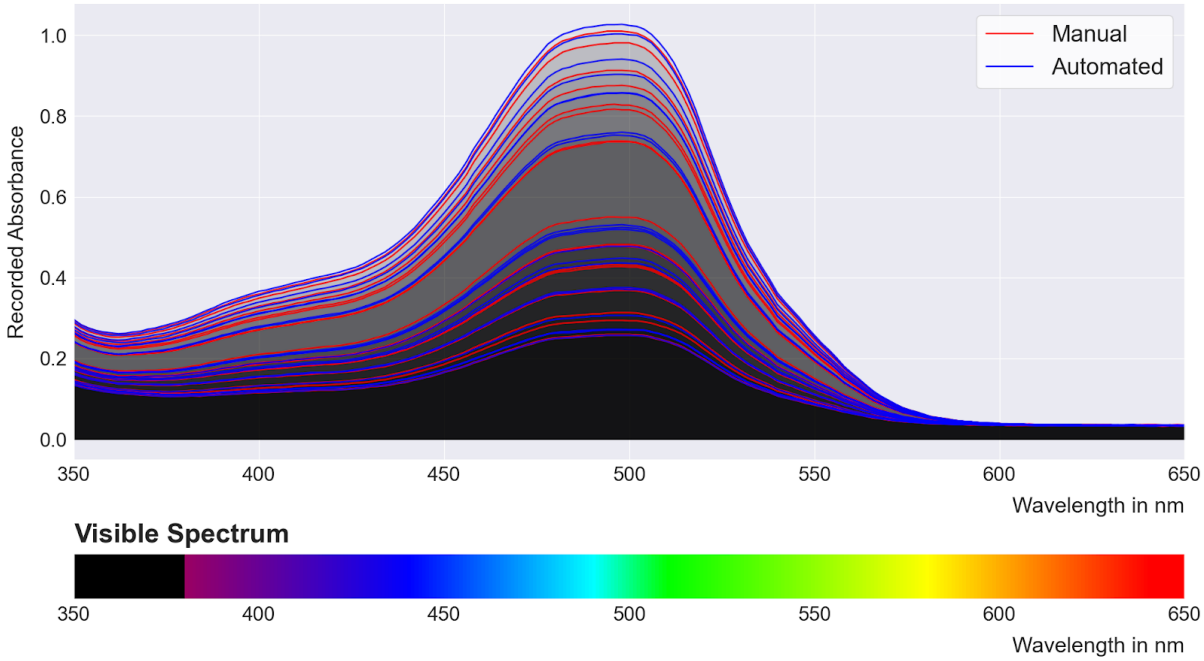


C4-Figure S16 – Run 57 - Plate 02 - Correlations between dilution ratio and measurements.

2.5 - Run 55 & 57 - comparison automated vs manual



C4-Figure S17 – Run 55 - Comparison between automation-generated data (Plate 01, in blue) and manually generated data (Plate 02 in red).



C4-Figure S18 – Run 57 - Comparison between automation-generated data (Plate 01, in blue) and manually generated data (Plate 02 in red).

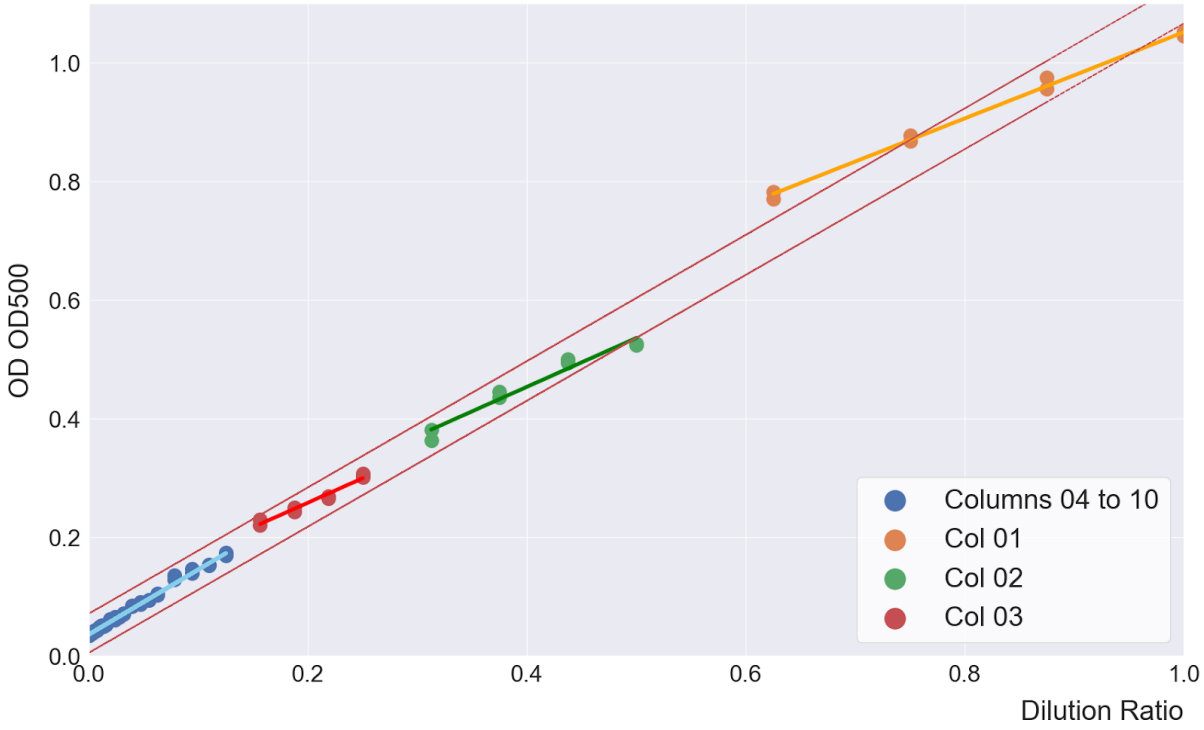
2.6 - Assessing the effects of the iterative transfers

Since the protocol uses iterative transfers from one column to the next we stratified the data by column and checked for they exhibited similar patterns as the first column (since they inherited them by iterative transfer). In practice we could only observe the first 3 columns - the dilution ratios were too small afterwards). And due to the limited size of the data from a column, linear regression was used on the 8 sample values in a column.

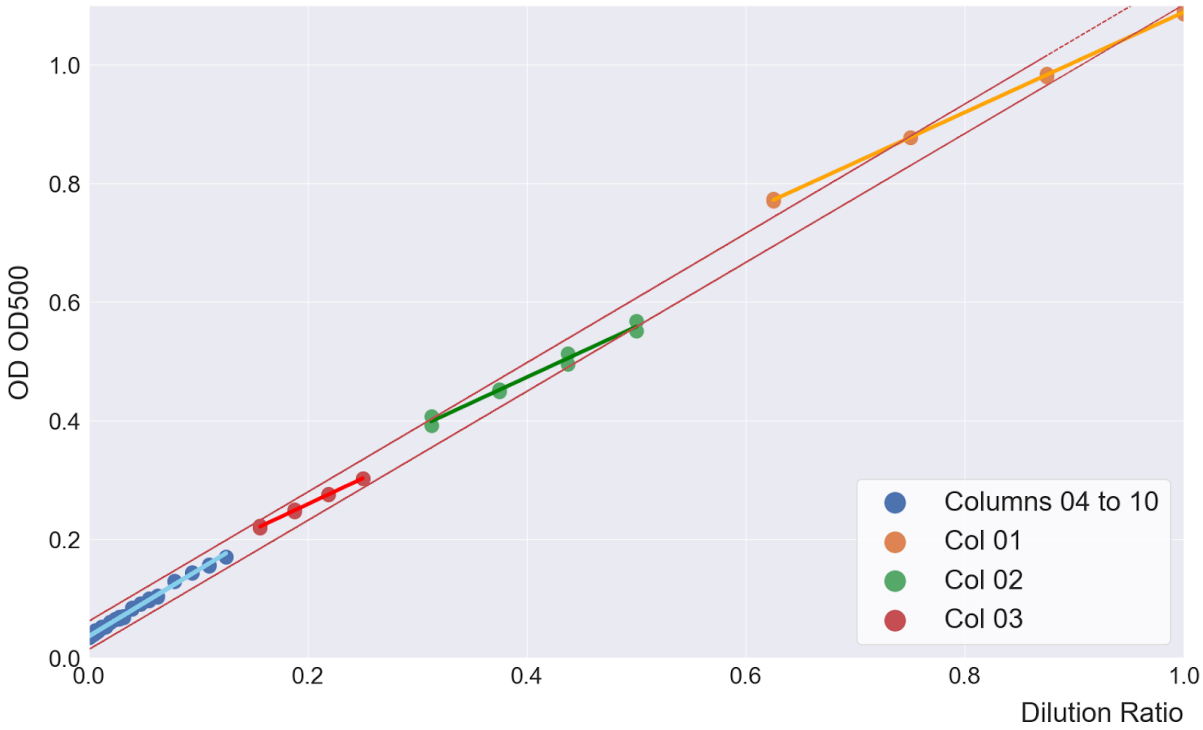
The following points are used to explain the data:

1. If the transfer between columns are perfectly implemented, then the trendline in the first also appears in the subsequent columns – that is the trendlines are parallel
2. Also, if the transfers in the first columns (dilution ratios of 100%,87.5%,75% and 62.5 %) are perfectly implemented, the trendlines are not only parallel, but also line up
3. Conversely, if the volume transferred from one column to the next is off by a factor α (instead of transferring a volume V as programmed, (1- α)V is transferred) then the trendlines do not stay parallel and instead their slope increases (the ratio of the column 2 and column 1 is 2+ α for instance).

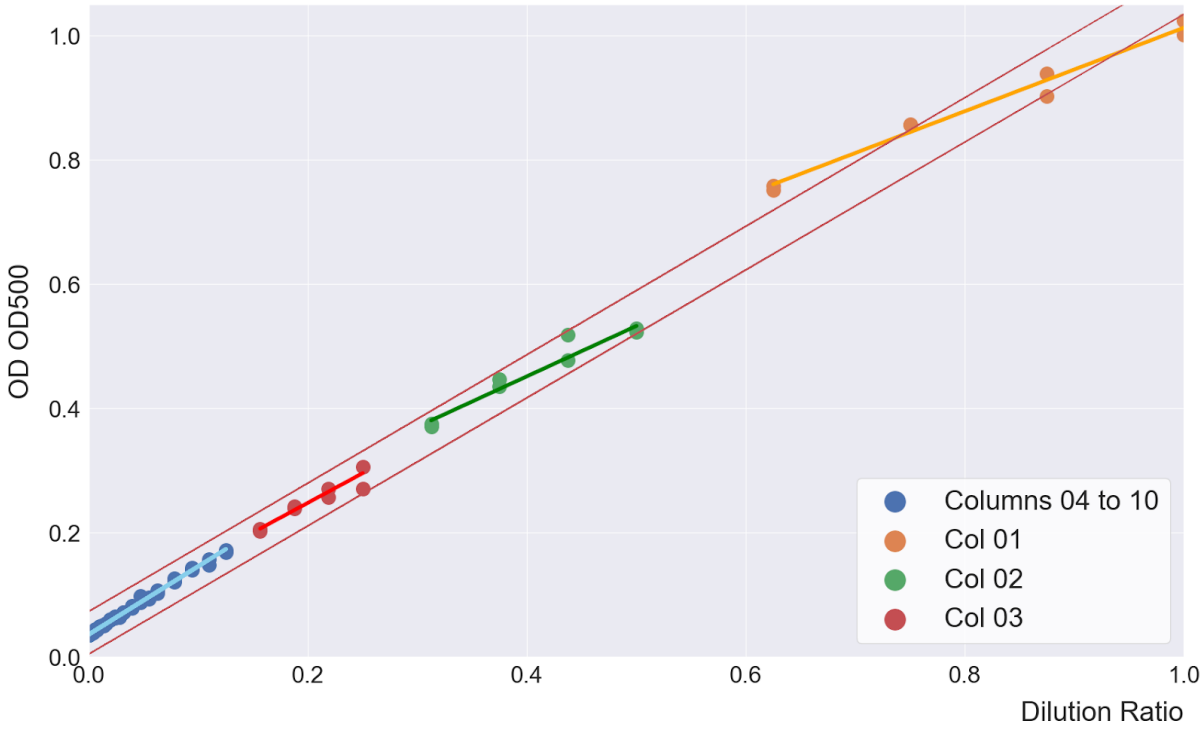
Figures C4 S19-22 show results that are consistent with the third scenario: imperfect transfer from one column to the next. Automated protocol performs worse than the manual protocol – although even data for the manual protocol show jumps between columns, and trendlines that do not stay parallel.



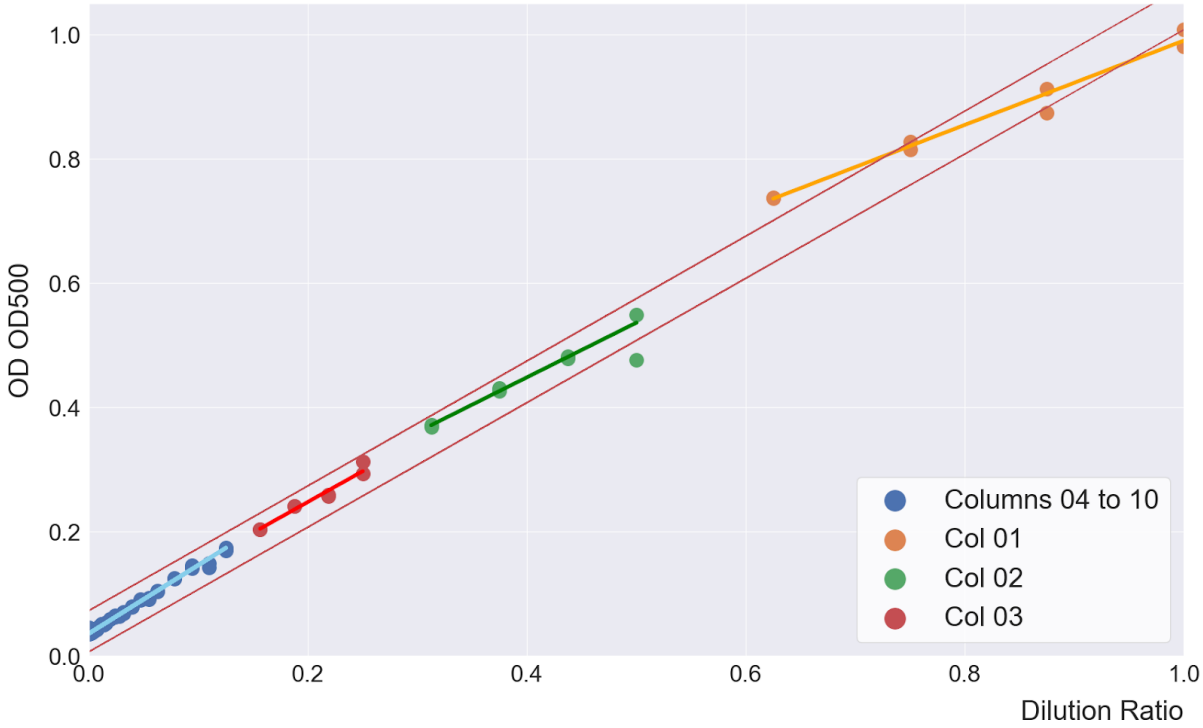
C4-Figure S19 – Run 55 - Plate 01 (Automated) - Effects of the Iterative dilutions at 500 nm.



C4-Figure S20 – Run 55 - Plate 02 (Manual) - Effects of the Iterative dilutions at 500 nm.



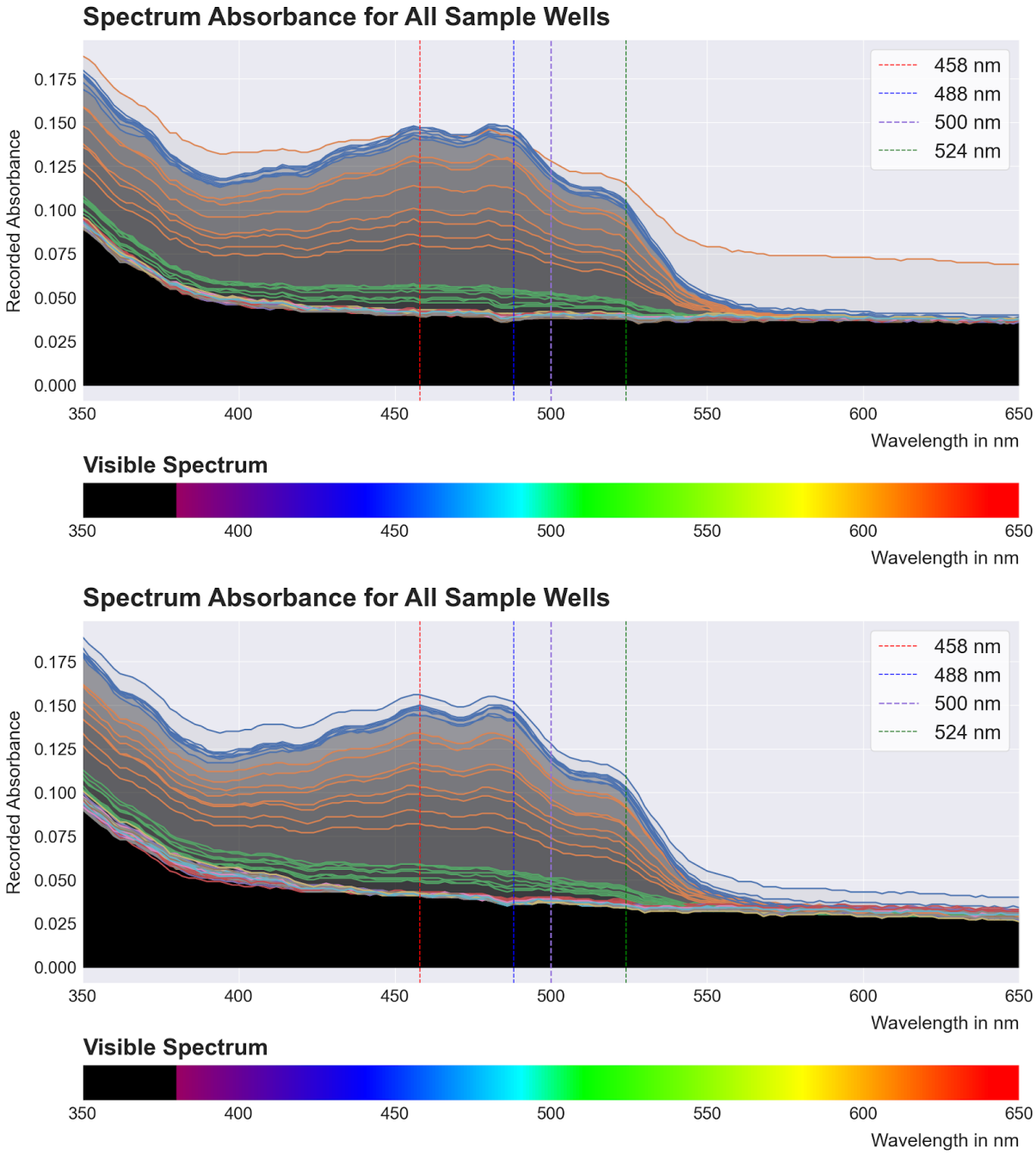
C4-Figure S21 – Run 57 - Plate 01 (Automated) - Effects of the iterative dilutions at 500 nm.



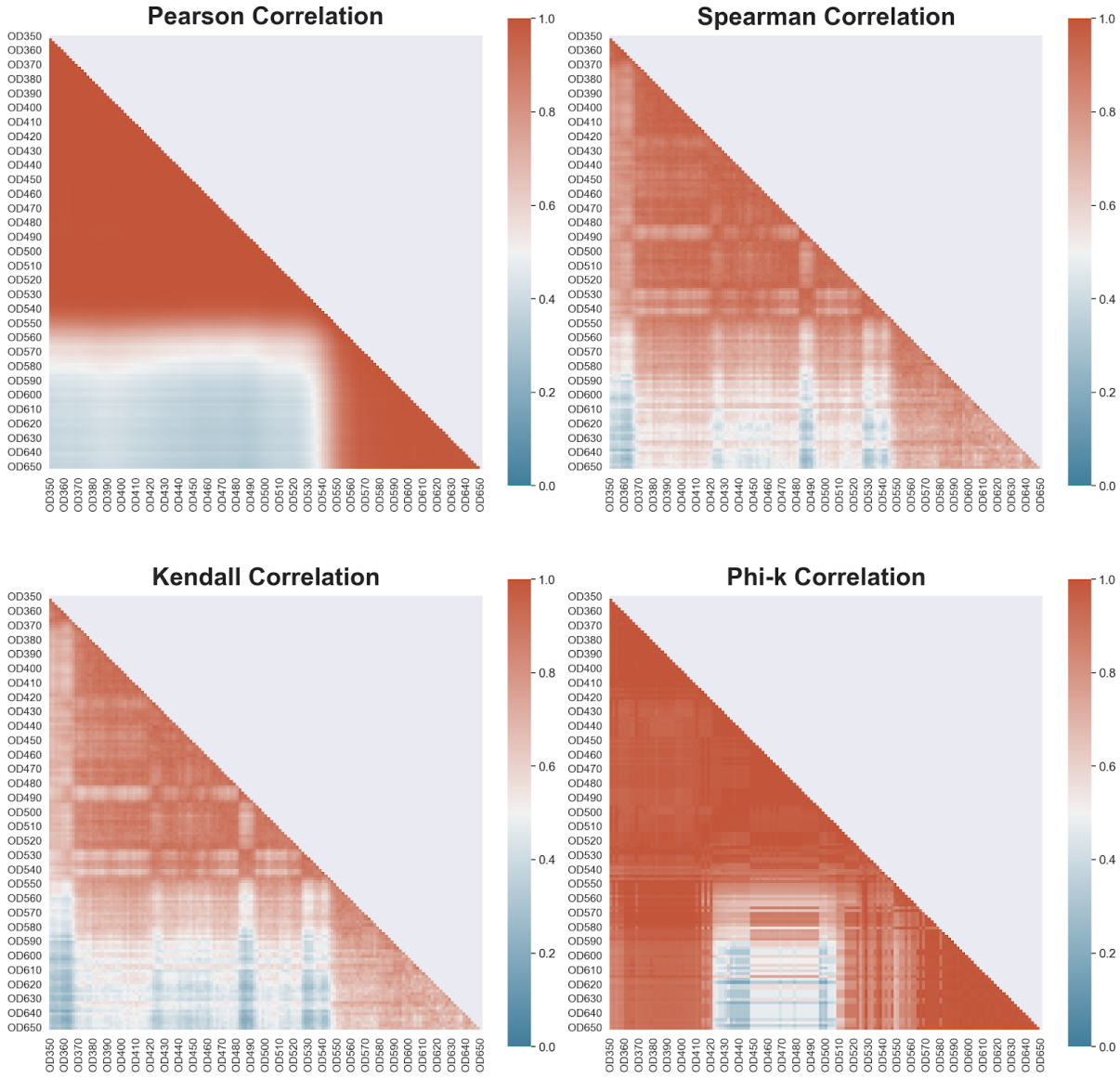
C4-Figure S22 – Run 57 - Plate 02 (Manual) – Effects of the iterative dilutions at 500 nm.

C4 - Section 3 - Geometric Dilution Scheme - Lycopene in DMSO

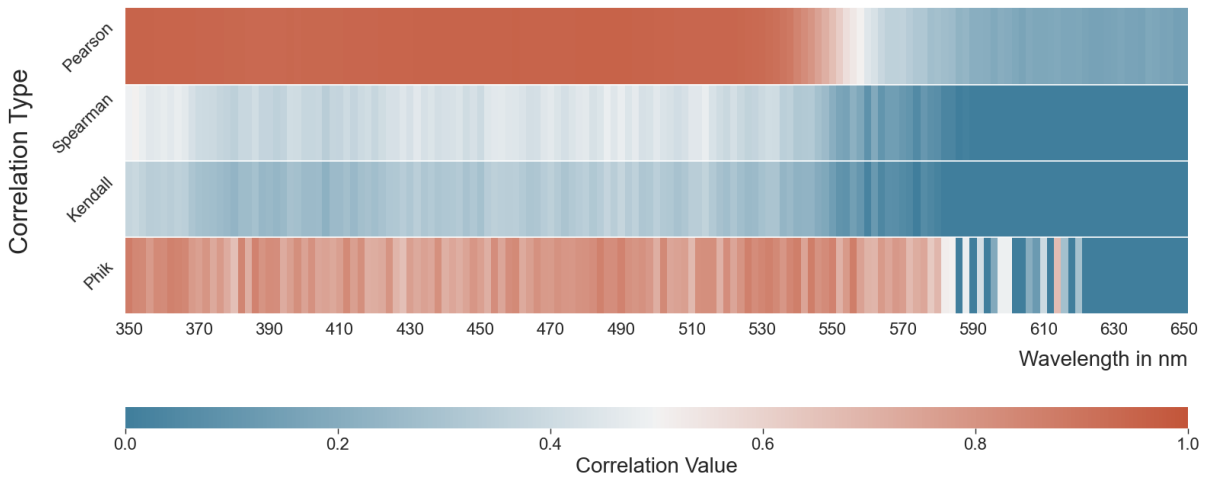
3.1 - Run 62 - Plate 01 - automated



C4-Figure S23 – Run 62 - Plate 01 - Absorbance Spectrum for All Samples. Top: Measurements in Clariostar A - **Bottom:** Measurements in Clariostar B.

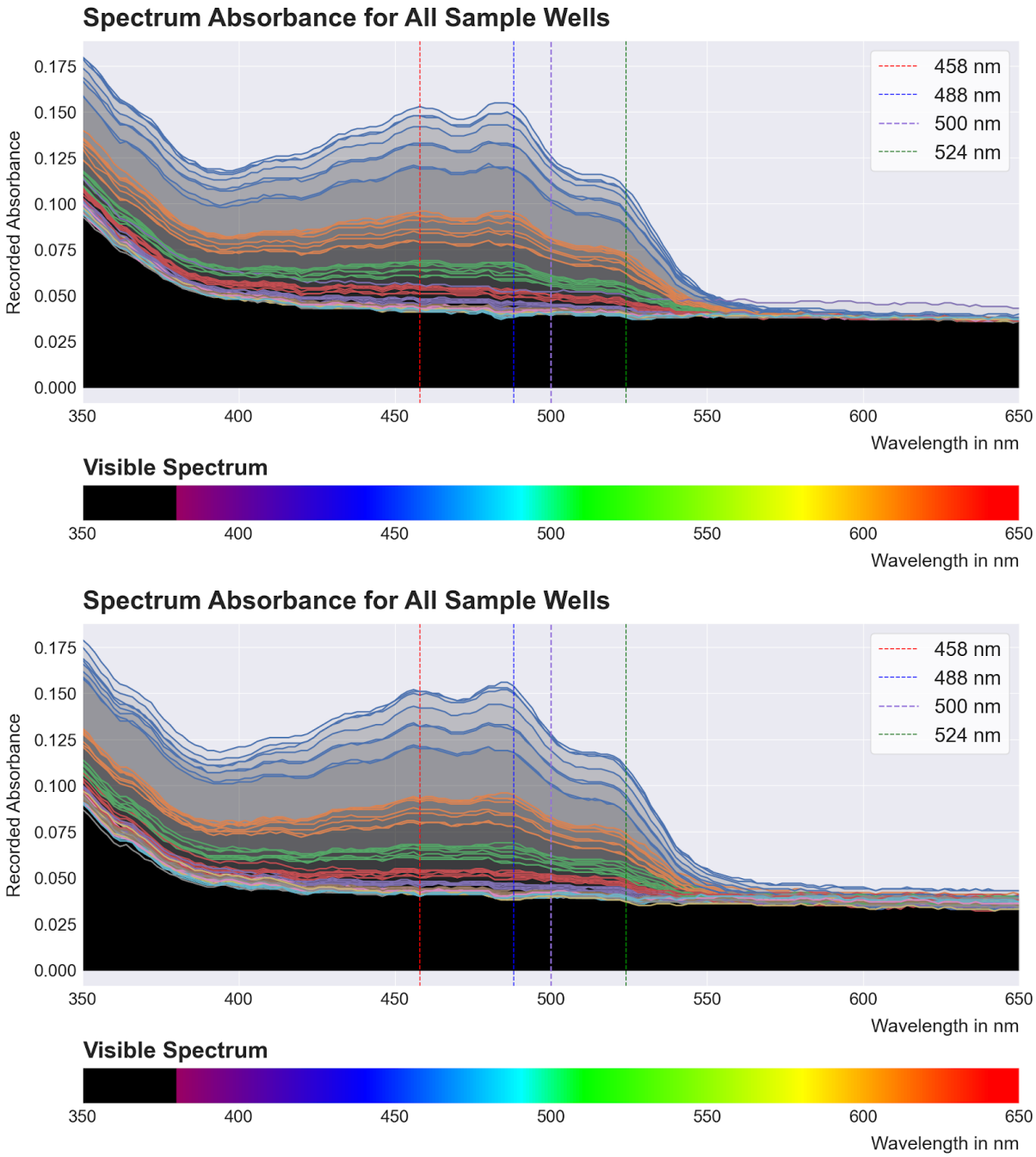


C4-Figure S24 – Run 62 - Plate 01 - Correlation matrices between measurements.

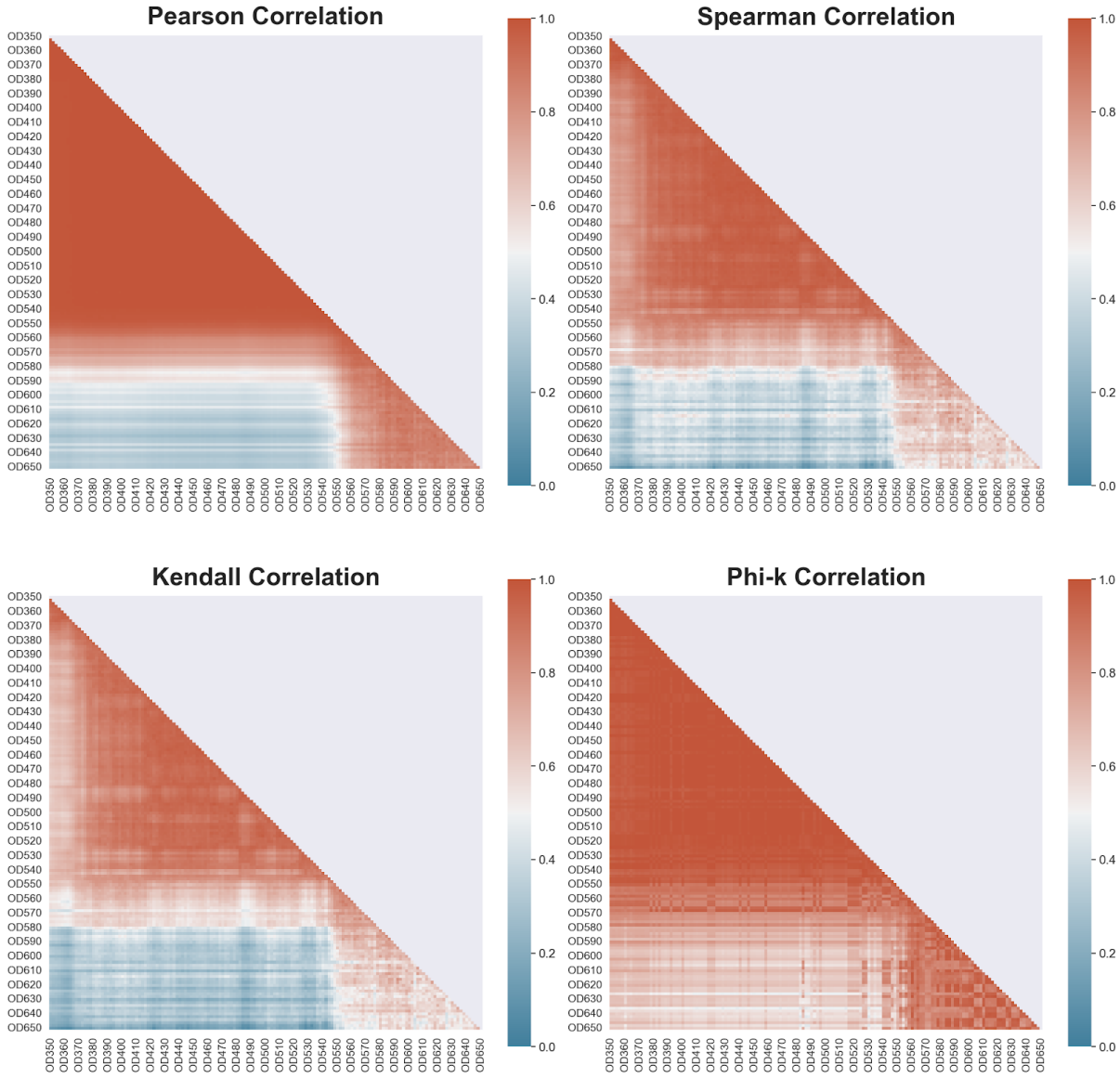


C4-Figure S25 – Run 62 - Plate 01 - Correlations between dilution ratio and measurements.

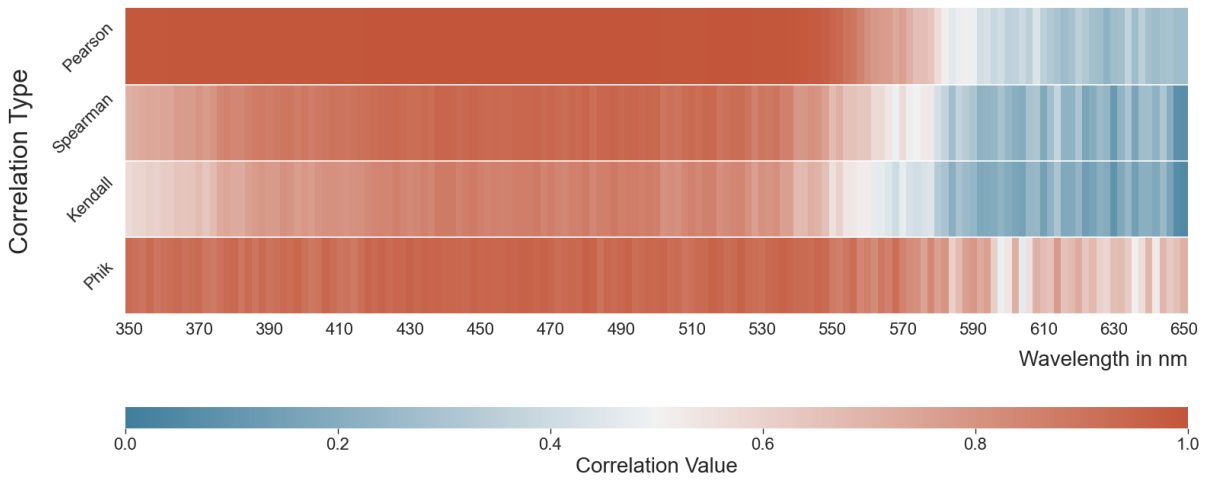
3.2 - Run 63 - Plate 01 - manual



C4-Figure S26 – Run 63 - Plate 01 - Absorbance spectrum for all samples.



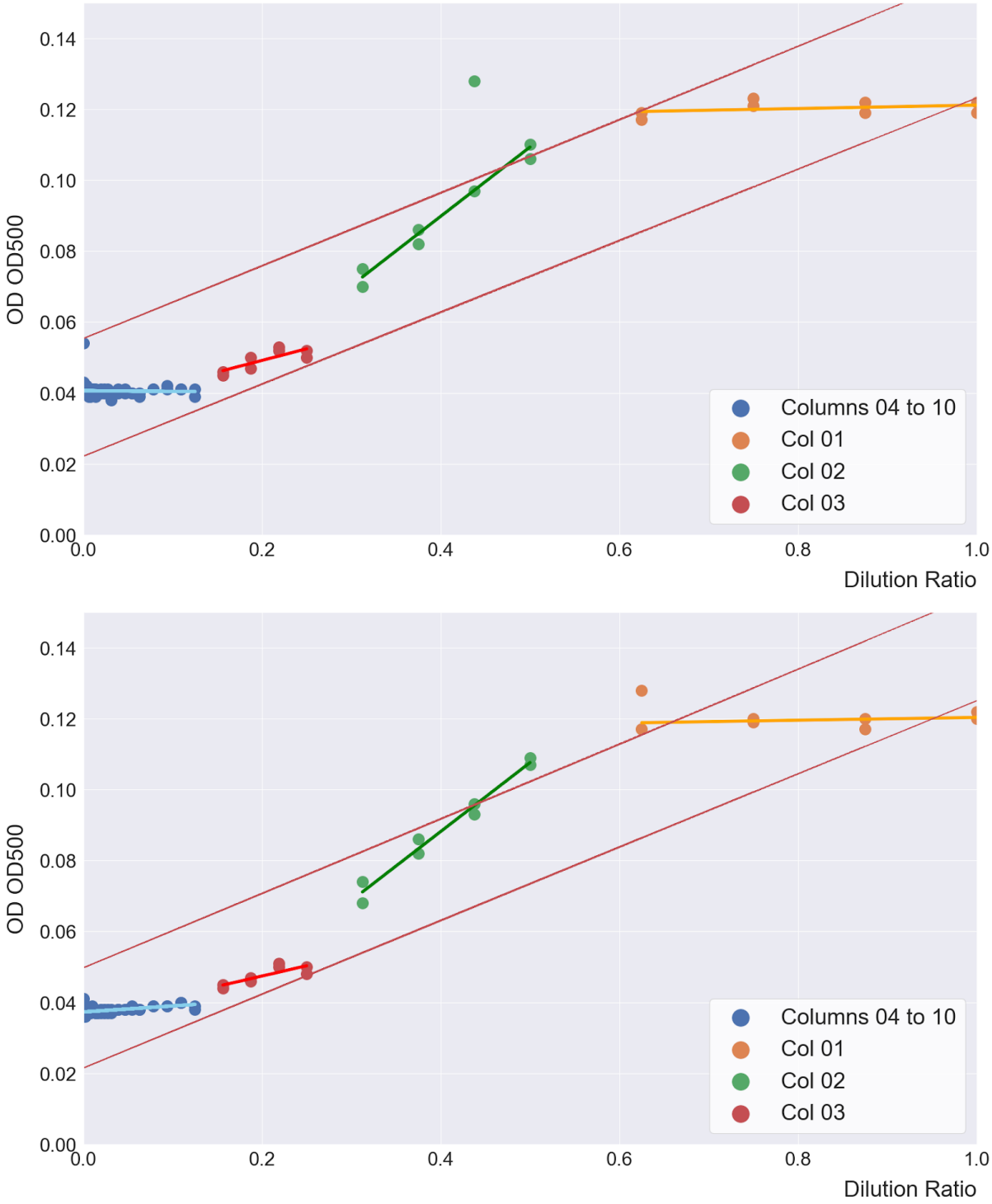
C4-Figure S27 – Run 63 - Plate 01 - Correlation matrices between measurements.



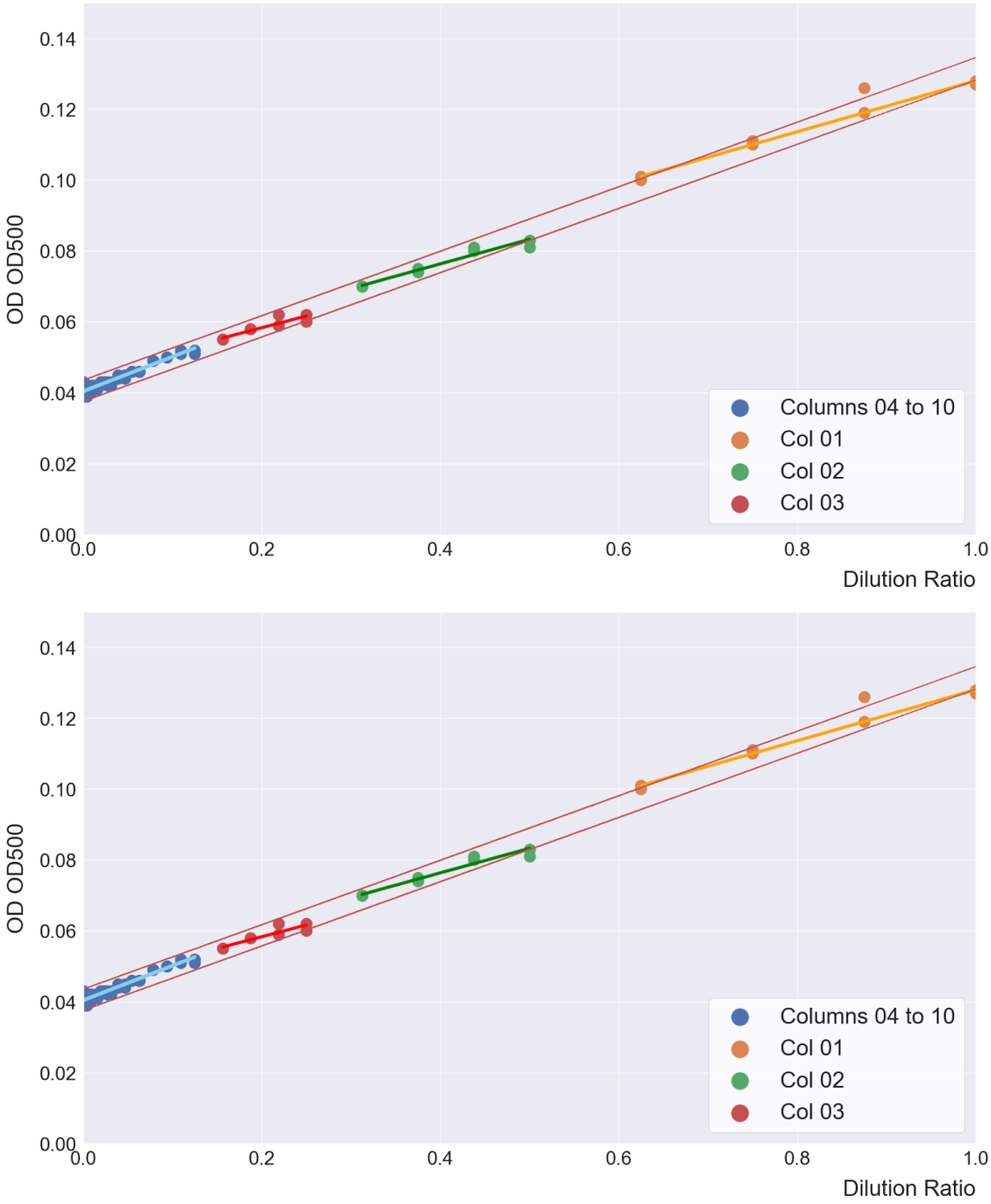
C4-Figure S28 – Run 63 - Plate 01 - Correlations between dilution ratio and measurements.

3.3 - Assessing the effects of the iterative transfer

The stratified analysis was conducted on the data for DMSO and Lycopene. Results with the automated protocol varied greatly depending on the column – measurements were conducted in two Clariostar plate readers to be certain the effects were not caused by the measuring instrument. Results for the plate generated by manually were much smoother and closer to those with water and food colourant.



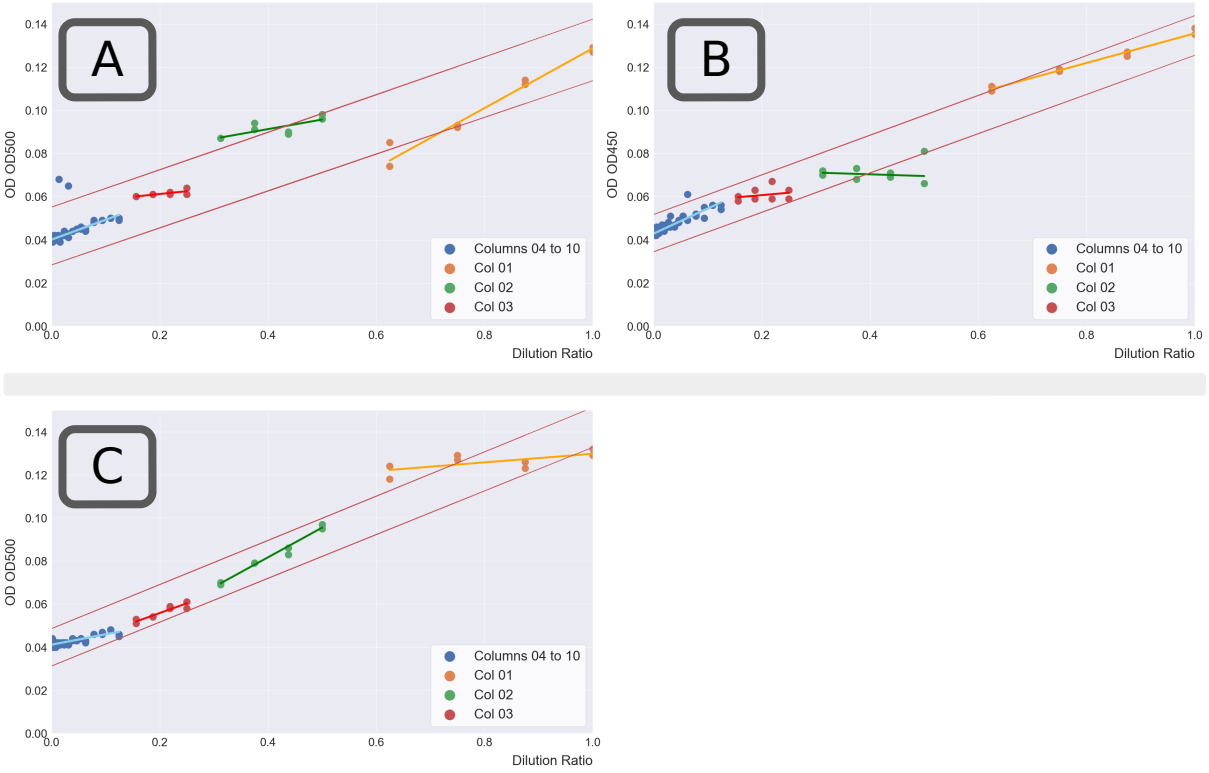
C4-Figure S29 – Run 62 - Plate 01 (Automated) - Effects of the iterative dilutions at 500 nm
Top : Measurements in Clariostar A - **Bottom** in Clariostar B.



C4-Figure S30 – Run 63 - Plate 01 (Manual) - Effects of the iterative dilutions at 500 nm. Top: Measurements in Clariostar A **- Bottom:** in Clariostar B.

Other low concentration mixes of Lycopene in DMSO

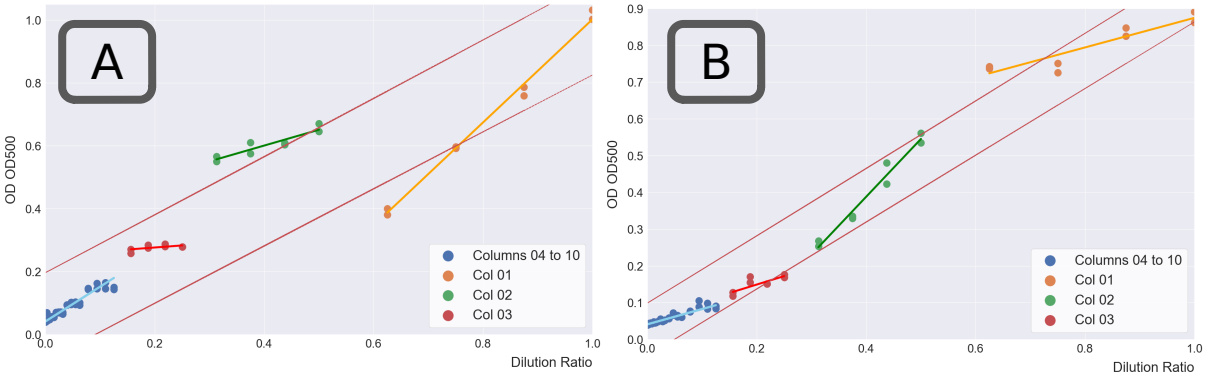
The same analysis was run on a set of runs using lycopene in DMSO at low concentration – the plates were only generally automatically. Again a great deal of variation could be observed from one column to the next, but also from one plate to another



C4-Figure S31 – Effects of the Iterative Dilutions at 500 nm (low concentration mixes). (A) Run 58 - Plate 01 - **(B)** Run 64 - Plate 01 - **(C)** Run 61 - Plate 01.

High concentration mixes of lycopene in DMSO

Results for high concentration mixes (plates generated with automation) showed even more discrepancies from one column to the next (and wider confidence intervals)



C4-Figure S32 - Effects of the Iterative Dilutions at 500 nm (High Concentration Mixes). (A) Run 59 - Plate 01 - **(B)** Run 60 - Plate 01.

Stratification by column - variation of the trend lines

Dataset	Trend Whole	Trend Col 01	Trend Col 02	Trend Col 03
55/01/A	Y=1.06309X + 0.03823	Y=0.72717X + 0.32441	Y=0.8248X + 0.12355	Y=0.8256X + 0.09305
55/02/A	Y=1.08856X + 0.03803	Y=0.8448X + 0.24385	Y=0.8528X + 0.13205	Y=0.86716X + 0.08578
57/01/A	Y=1.0316X + 0.0389	Y=0.67182X + 0.34059	Y=0.81095X + 0.12701	Y=0.952X + 0.0575
57/02/A	Y=1.00307X + 0.03972	Y=0.67836X + 0.31177	Y=0.88X + 0.09617	Y=0.98699X + 0.0503

C4-Table S1 – Food colourant in water mixes - Column trends vs dataset trends.

Dataset	Trend Whole	Trend Col 01	Trend Col 02	Trend Col 03
59/01/A	Y=0.91728X + 0.05648	Y=1.63901X + -0.63707	Y=0.5024X + 0.39965	Y=0.13035X + 0.25038
59/01/B	Y=0.95853X + 0.0516	Y=1.4484X + -0.4397	Y=0.56168X + 0.37807	Y=0.15045X + 0.23919
60/01/A	Y=0.91326X + 0.02688	Y=0.4026X + 0.47208	Y=1.5712X + -0.2403	Y=0.45918X + 0.05736
60/01/B	Y=0.91606X + 0.02607	Y=0.3204X + 0.5463	Y=1.42838X + -0.19535	Y=0.50862X + 0.04845

C4-Table S2 – High concentration lycopene mixes - Column trends vs dataset trends.

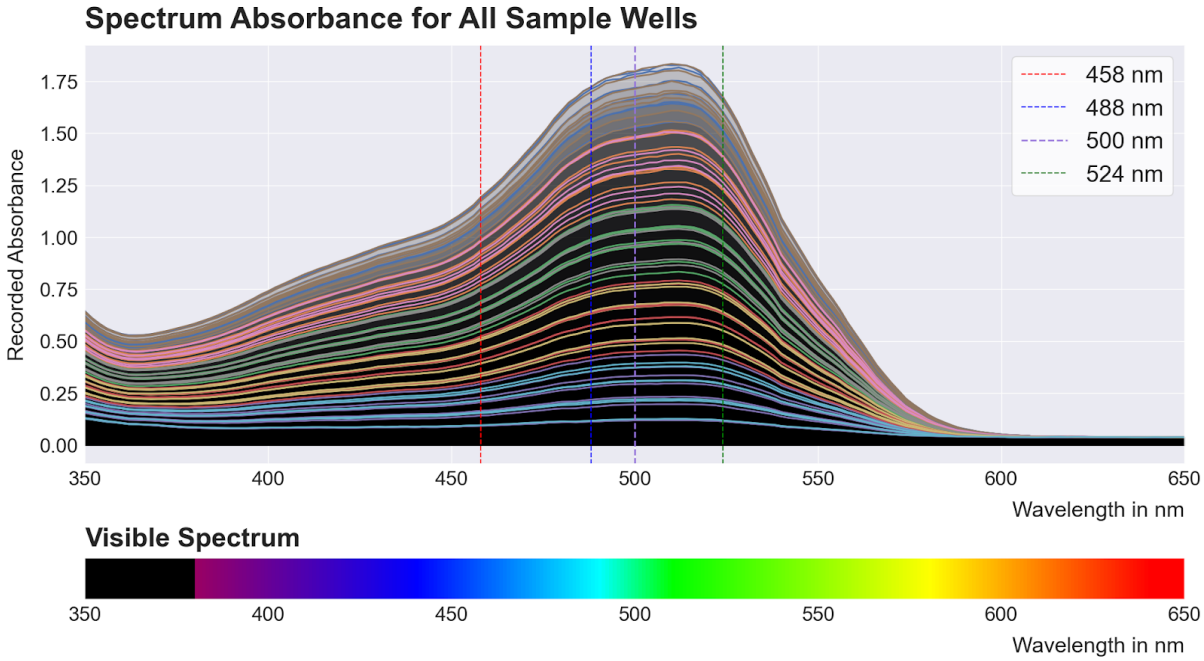
Dataset	Trend Whole	Trend Col 01	Trend Col 02	Trend Col 03
62/01/A	Y=0.10206X + 0.03876	Y=0.00483X + 0.11632	Y=0.19528X + 0.01168	Y=0.0656X + 0.03605
62/01/B	Y=0.10443X + 0.03562	Y=0.00402X + 0.11632	Y=0.1944X + 0.0104	Y=0.05821X + 0.03578
63/01/A	Y=0.08986X + 0.04028	Y=0.06332X + 0.06302	Y=0.06053X + 0.05162	Y=0.0544X + 0.0472
63/01/B	Y=0.08986X + 0.04028	Y=0.08986X + 0.04028	Y=0.08986X + 0.04028	Y=0.08986X + 0.04028
58/01/A	Y=0.08615X + 0.04172	Y=0.13839X + -0.00977	Y=0.04425X + 0.07352	Y=0.02564X + 0.05606
58/01/B	Y=0.08666X + 0.04195	Y=0.12597X + 0.00253	Y=0.04901X + 0.07075	Y=0.02094X + 0.05746
64/01/A	Y=0.07725X + 0.04083	Y=0.0568X + 0.0621	Y=-0.016X + 0.07033	Y=0.01131X + 0.05334
64/01/B	Y=0.07725X + 0.04083	Y=0.0568X + 0.0621	Y=-0.016X + 0.07033	Y=0.01131X + 0.05334
61/01/A	Y=0.1021X + 0.03989	Y=0.02X + 0.10975	Y=0.13783X + 0.02654	Y=0.09004X + 0.03781
61/01/B	Y=0.10318X + 0.03851	Y=0.02776X + 0.10262	Y=0.14933X + 0.0212	Y=0.0992X + 0.03535

C4-Table S3 – Low concentration lycopene mixes - Column trends vs dataset trends.

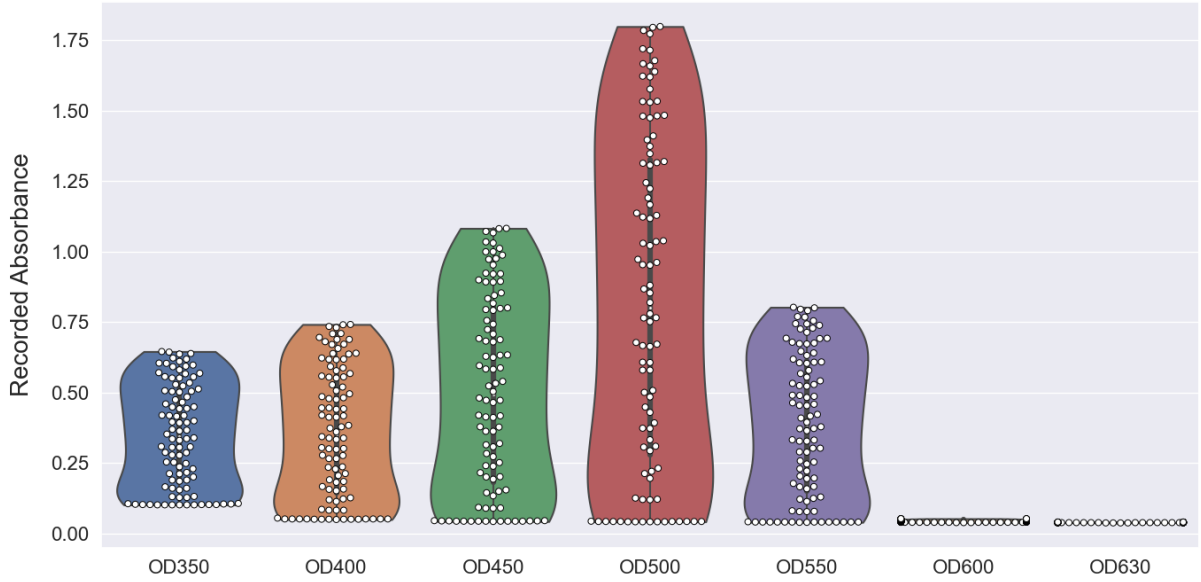
C4 - Section 4 - Linear Dilution Scheme - Preliminary Studies

4.1 - Development runs - food colorant in DMSO

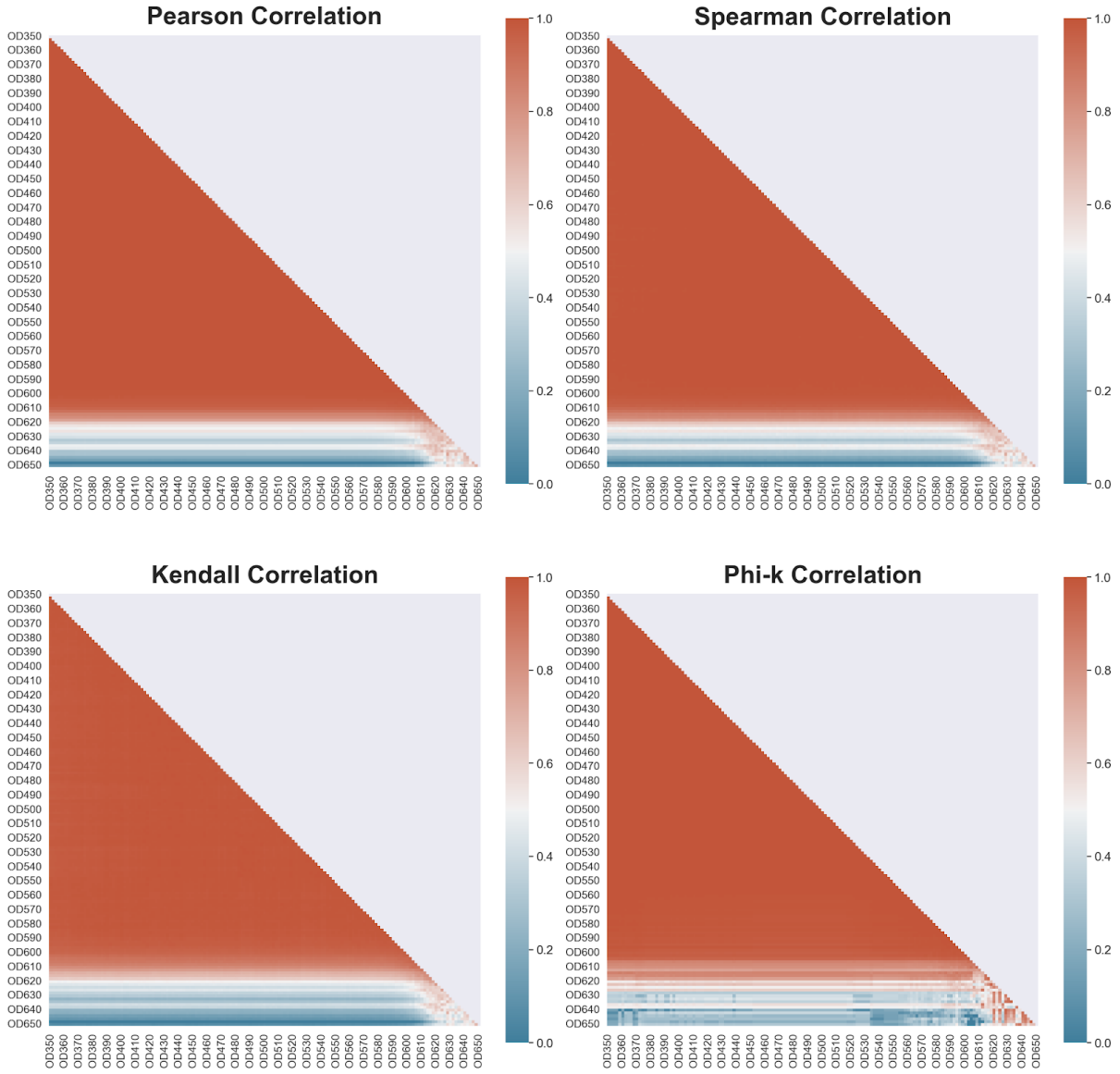
We present the results for Run 74 - Protocol 2.0.3 (minor alteration of the speed of aspiration from DMSO reservoir; all automated implementations of the linear scheme are denoted protocol 2.X.X). Runs 70,74 and 75 were used for the development phase.



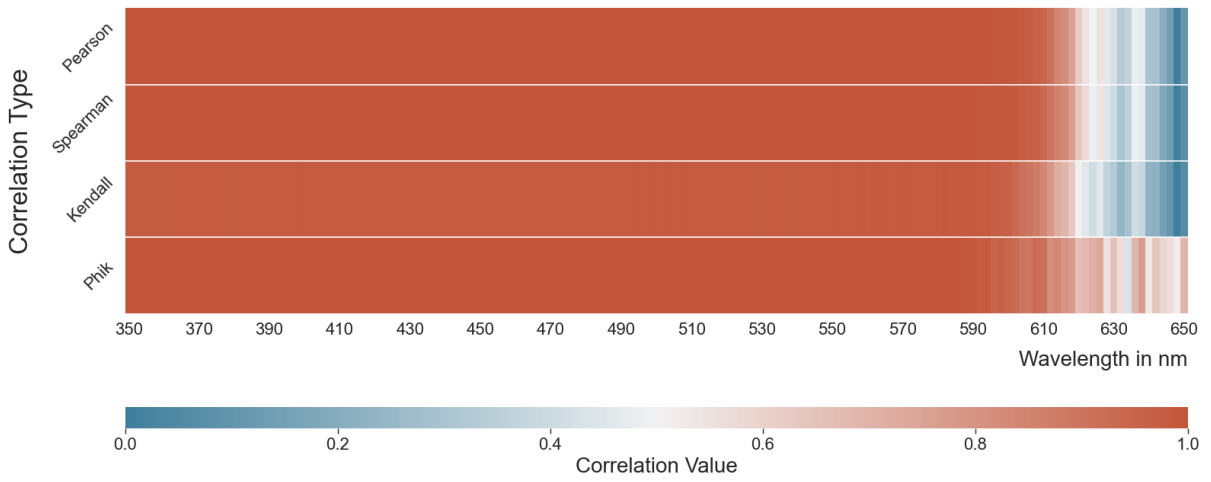
C4-Figure S33 – Run 74 - Plate 01 - Absorbance spectrum for all samples.



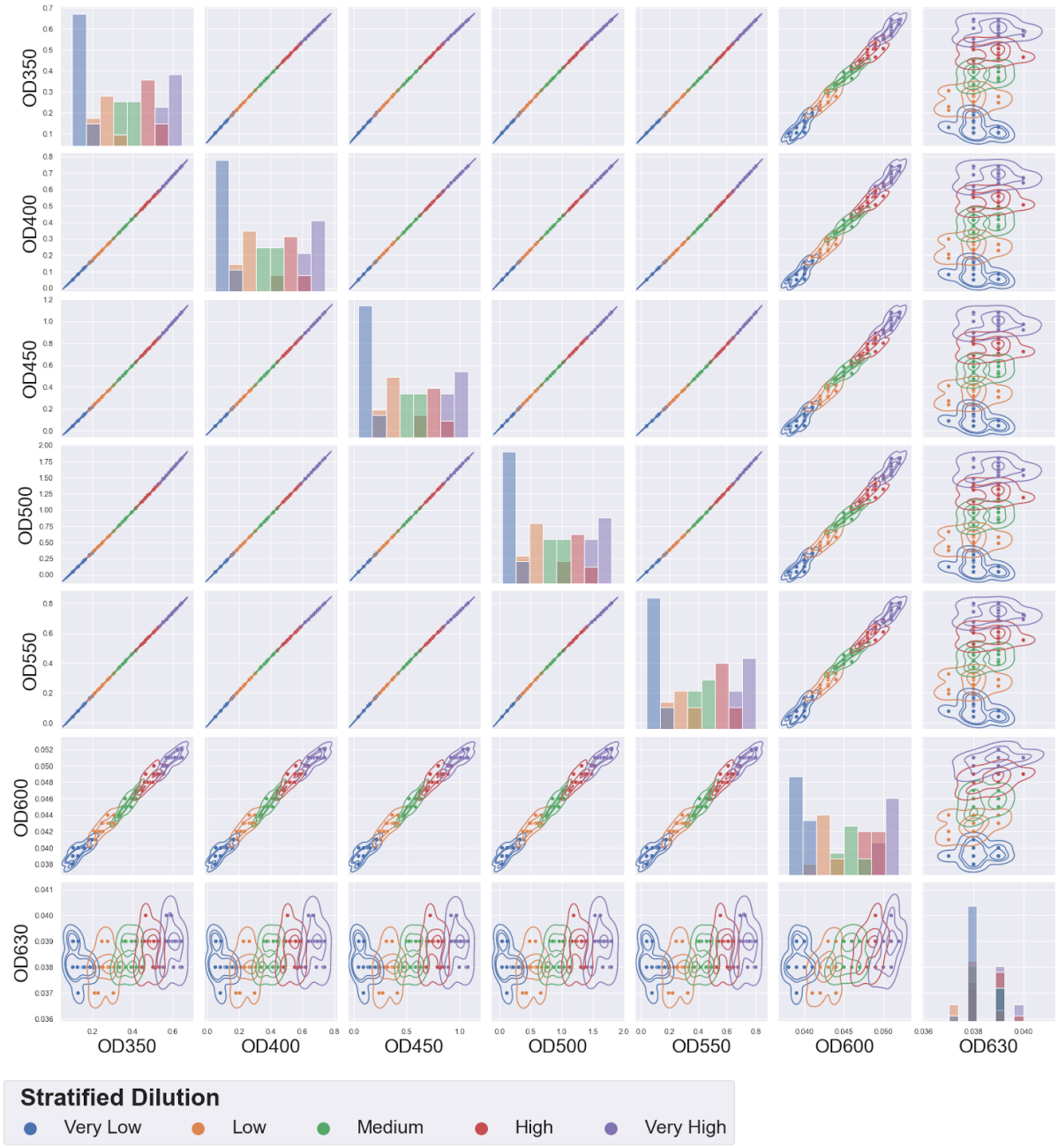
C4-Figure S34 – Run 74 - Plate 01 - Corresponding violin plot at selected wavelengths. The violin plot for the measurements is not as lopsided as for the geometric scheme - as expected from the linear distribution of the concentrations.



C4-Figure S35 - Run 74 - Plate 01 - Correlation matrices between measurements.



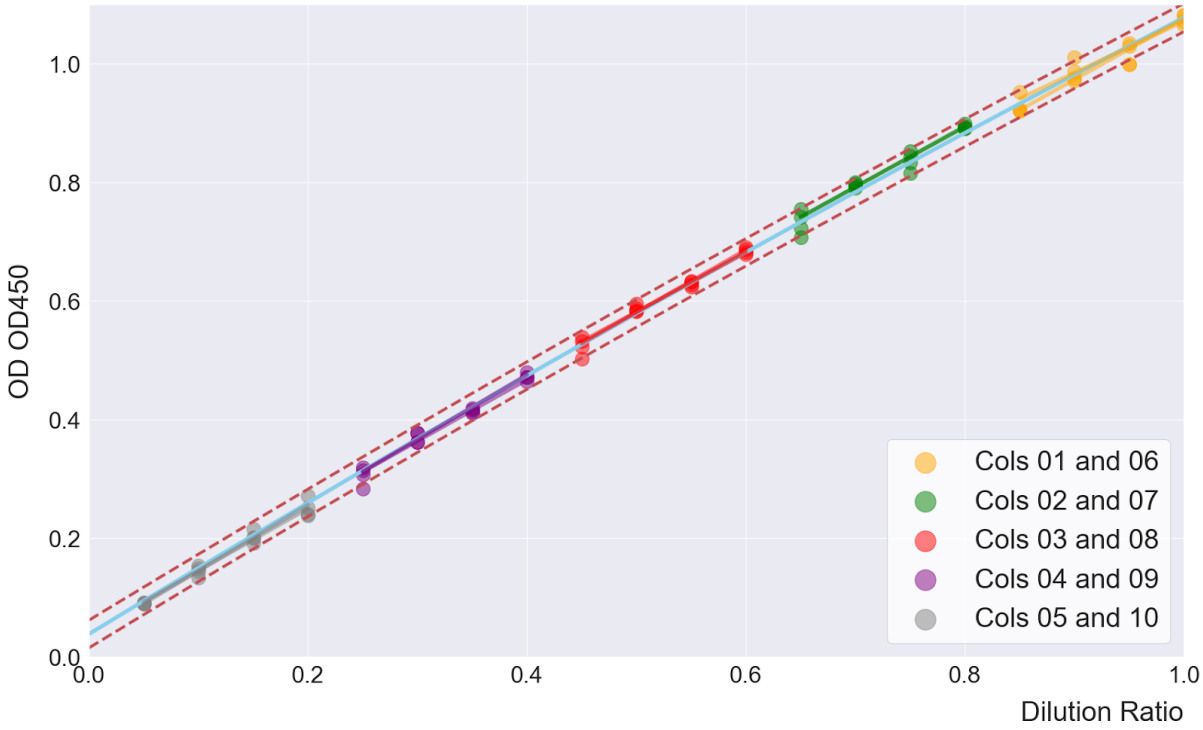
C4-Figure S36 – Run 74 - Plate 01 - Correlations between dilution ratio and measurements.



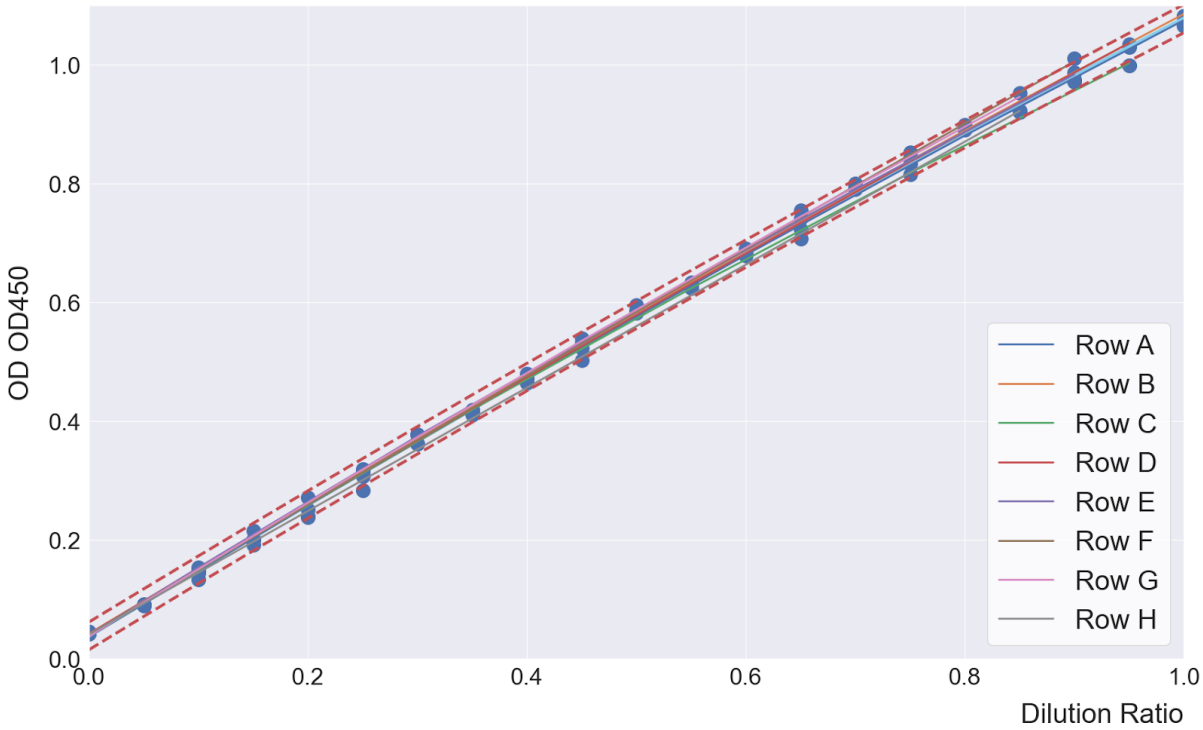
C4-Figure S37 – Run 74 - Plate 01 - Pairplot between measurements at selected wavelengths.

4.2 - Stratified analysis

The measurement data were stratified by columns and by rows.



C4-Figure S38 – Run 74 - Plate 01 – Stratification by column of the measurements at 450 nm.



C4-Figure S39 – Run 74 - Plate 01 – Stratification by row of the measurements at 450 nm.

4.3 - Main analytical approach

Again the approach is illustrated on Run 74.

Take data at a given wavelength (here 450 nm) and apply the following three steps

Step 1. Two models were used for the regression: a liner model and a quadratic model to capture what curvature is in the data. A linear regression was conducted on the dataset (in blue, confidence interval is displayed in red), and a robust linear model approach (in green) was also run – weighing the data points so the outliers are less prominent.

Step 2. After inversion of the regression models, the real dilution ratios are estimated for all the samples from their measurement. When plotted against the targeted dilution, they should fall on the Y=X line. Global (quality) metrics are ten computed that quantify how well they fall on said line – in practice we use the Pearson correlation (to quantify the linear relationship) and the mean square error (or equivalent in our case the square root of it) to quantify the magnitude of the residuals (how far the points fall off the line).

Step 3. Visualisation and reporting.

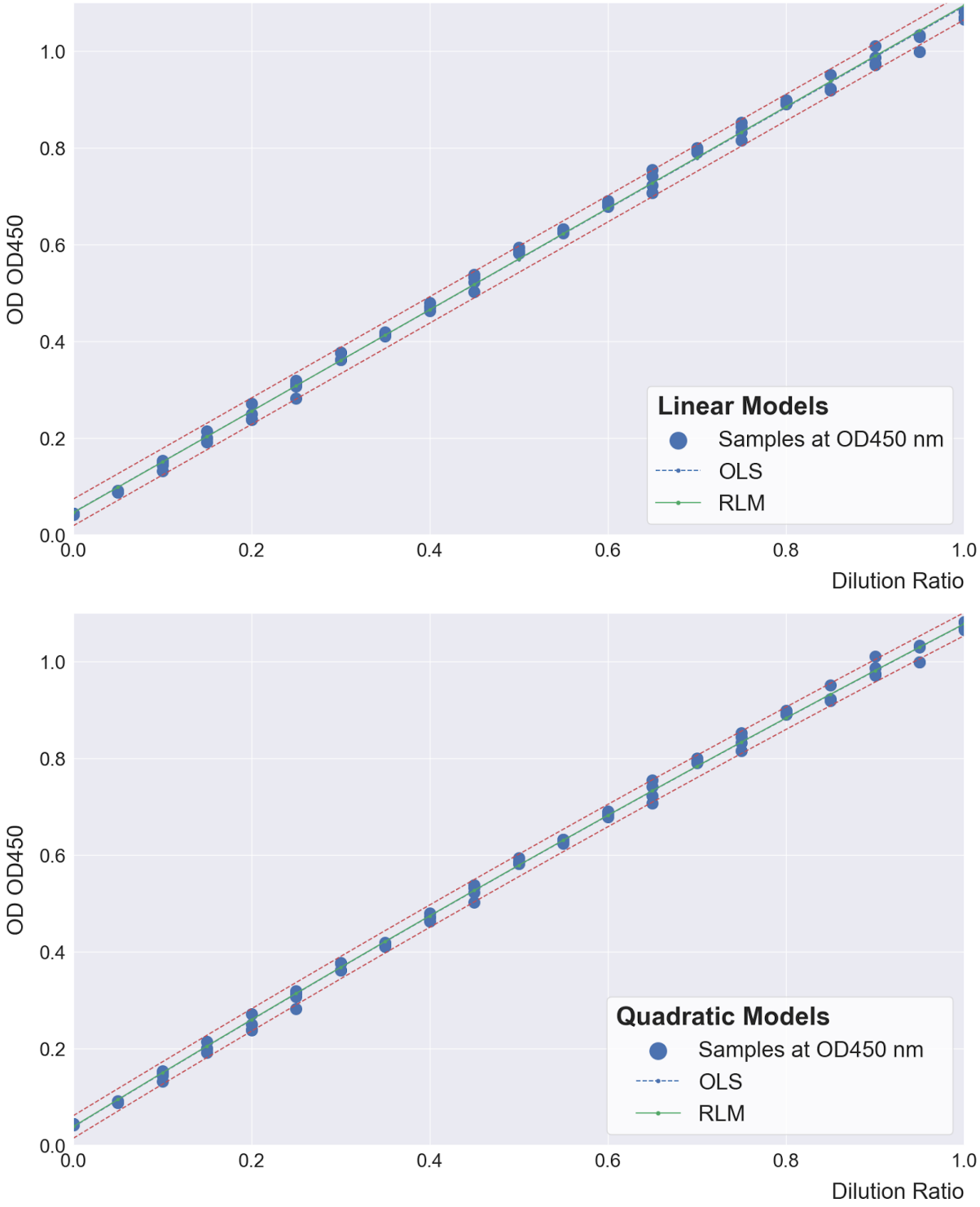
	Model of Order 1	**Model of Order 2**
Pearson Corr	0.9992221245	0.999427306
Spearman Corr	0.997359134	0.9973591339
√MSE	0.012952181	0.011121466

C4-Table S4 – Global quality metrics for Run 74.

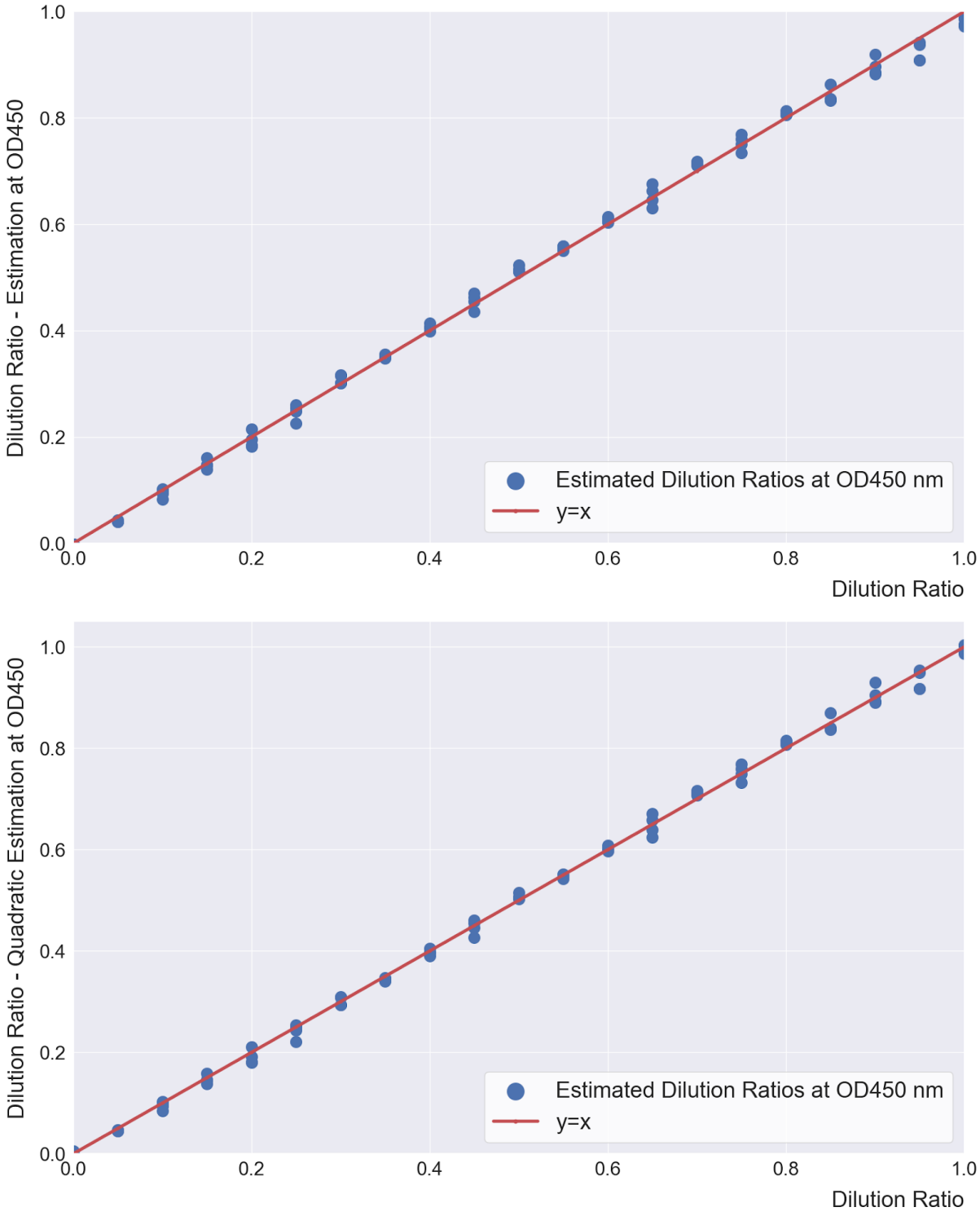
These metrics can be used as global quality metrics - the higher the correlation, the better the model is at explaining the data. Deviations from the trend

For all samples, The residual errors from the predictions can be turned into a relative error (residual/expected dilution value), and displayed on a plate map.

Results for the training runs with Food Colorant in DMSO yielded very tight regression and relative errors of a few percent except for the lowest dilution ratios/concentrations. The quadratic model yielded minor improvements – mostly visible at these low dilutions



C4-Figure S40 – Regression on the measurements at 450 nm (Top: Linear model, Bottom: Quadratic model). Note the slight bend for the highest dilution ratios indicative of some saturation.



C4-Figure S41 – Inversion of the regression models and estimation of the real dilution ratio (Top: Linear model, Bottom: Quadratic model).



C4-Figure S42 – Relative errors mapped onto the plate. Top: dilution ratios targeted by the linear dilution scheme; **Middle:** Estimated relative errors (linear model); **Bottom:** Estimated relative errors (quadratic model).

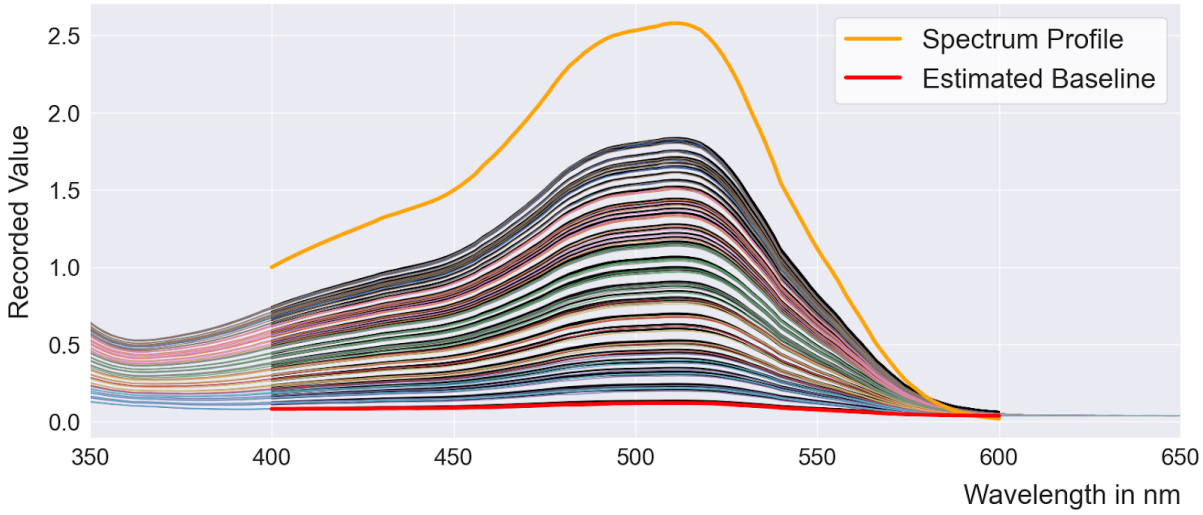
4.3 - Spectrum-based approach

The high degree of linear correlation between measurements at different wavelengths led to the construction of a model (called SM, for spectrum model) which binds measurements for any well over a range of wavelengths to a vector S - the spectrum of the solute - and a baseline B - the background measurement of the plate over the same range. The coefficient αi is well dependent and indicates the quantity of signal producing-solute in the well.

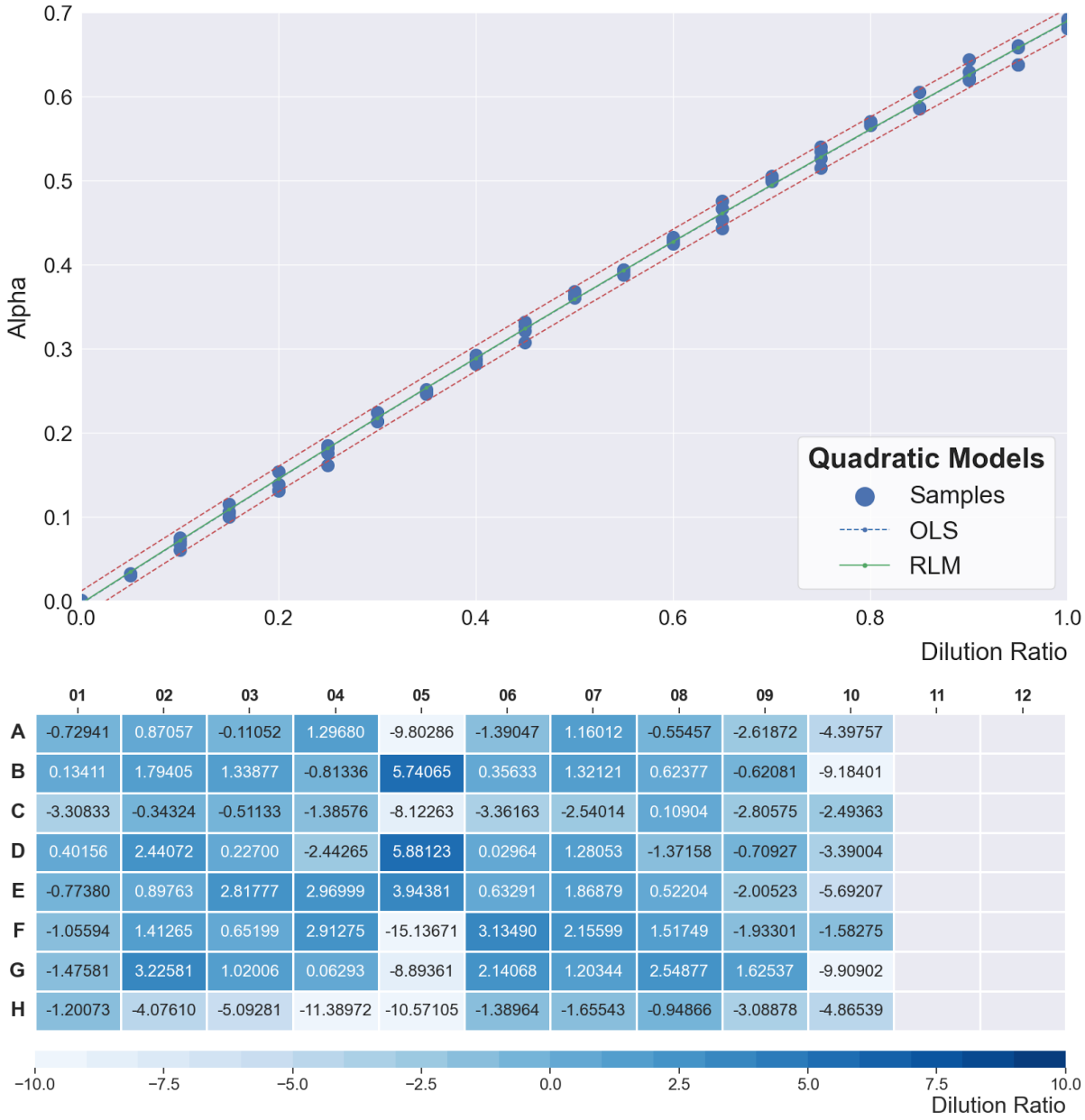
$$(SM)\ Yi = \alpha i \times S + B$$

The parameters B,S (αi not shown) are estimated at once from the plate data (under constraints that all components of S and B and coefficients αi.be positive and for penalty function that the baseline B stay close to blank values). The baseline B was plotted in red - and matches the absorbance spectra of the blanks on the plate. The spectrum S , was plotted in amber; and for legibility purposes its amplitude was normalised so its value is 1 at wl= 400 nm. For all wells, the spectrum is reconstructed from the estimated parameters and plotted in black. As can be seen on Figure S43, the match between data and reconstruction is very tight.

The previous regression-inversion-error estimation workflow can then be applied to the coefficients αi



C4-Figure S43 - Run 74 - Reconstructed spectrum, baseline and absorbance spectra.



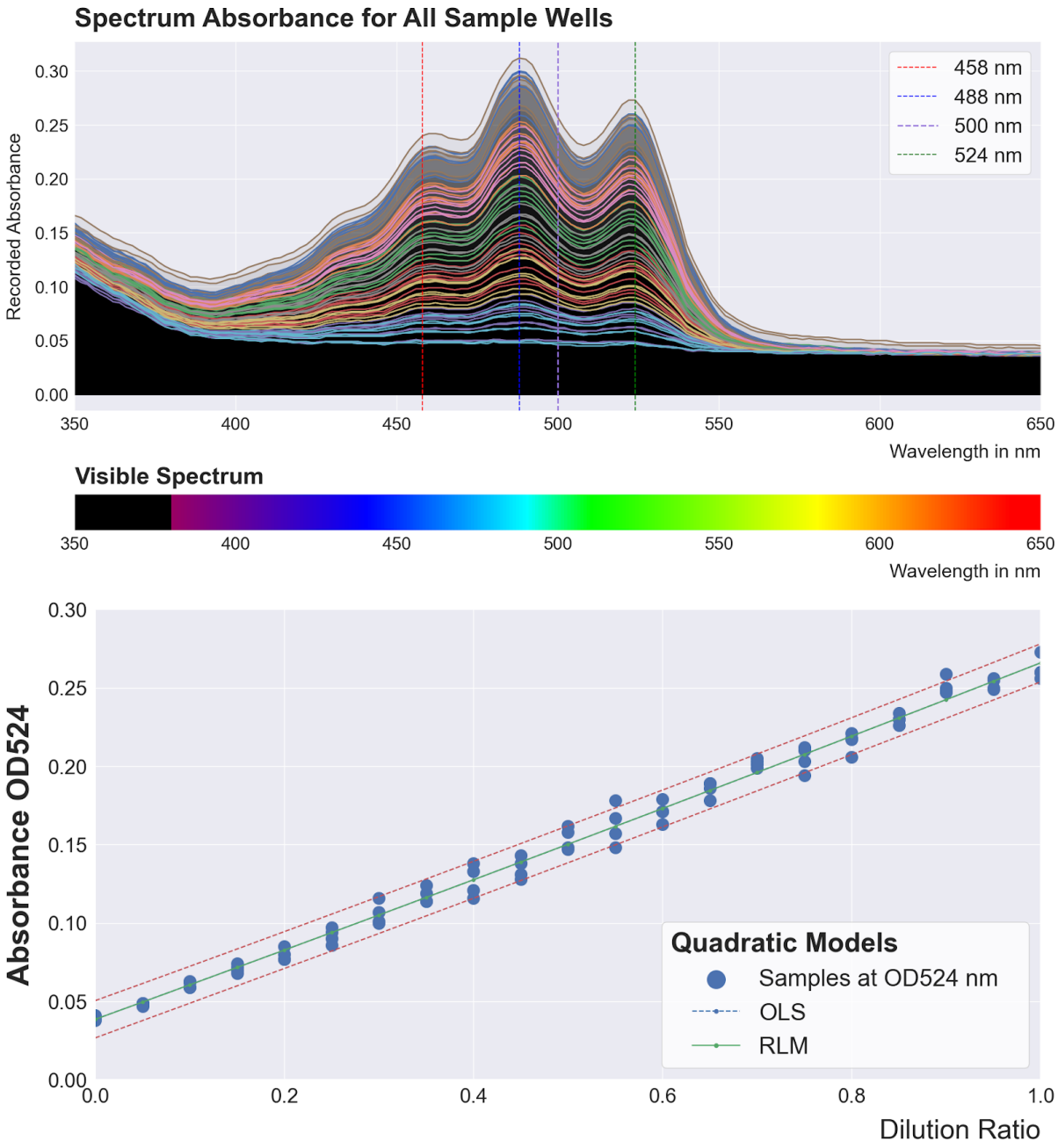
C4-Figure S44 - Run 74 - Error estimation workflow for the spectrum method. Top: Regression of the coefficients αi (quadratic model) – **Bottom:** Relative errors (quadratic model).

C4 - Section 5 - Linear Dilution Scheme - Lycopene in DMSO

5.1 - Development runs lycopene in DMSO

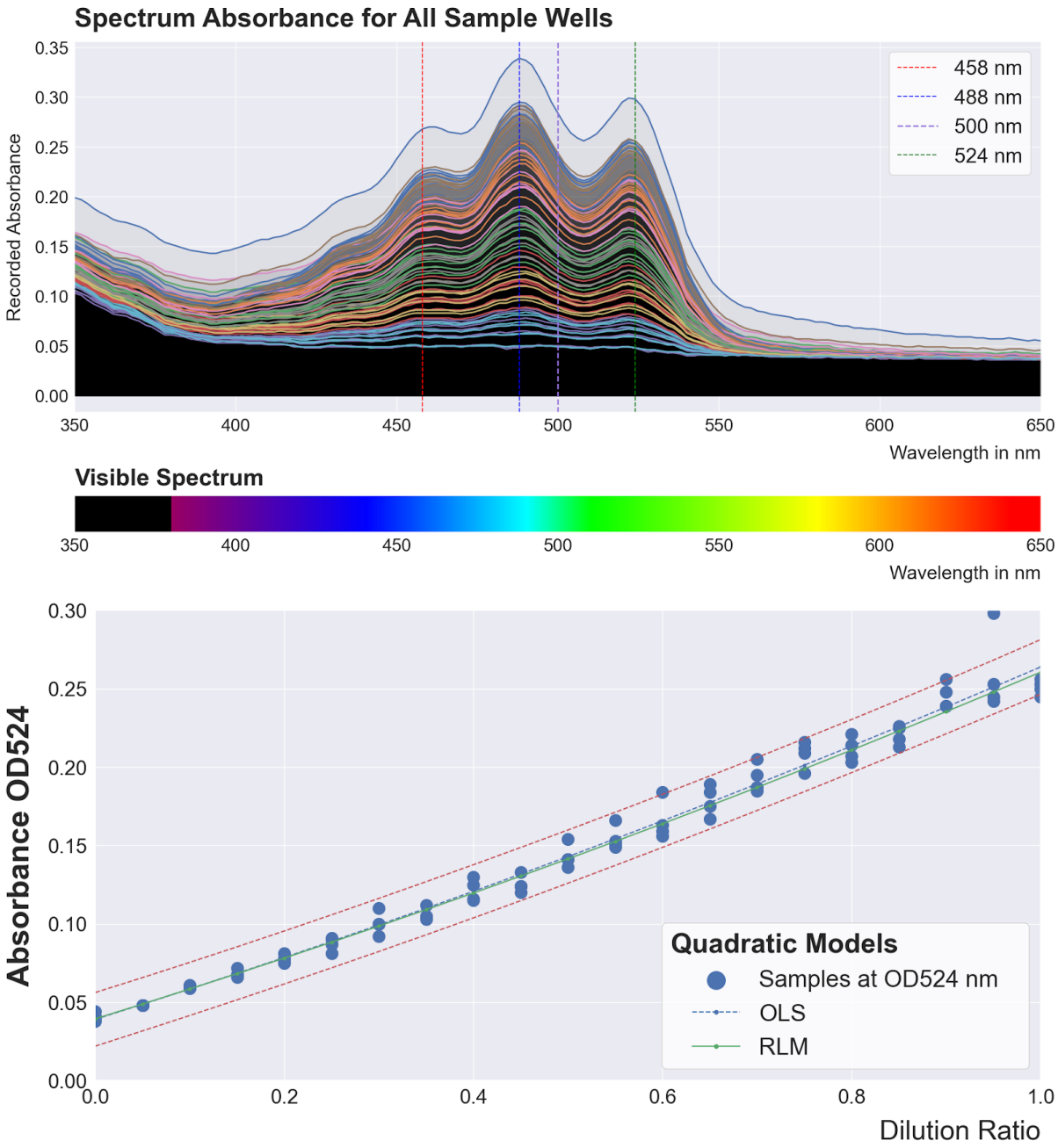
Several development runs for the linear dilution scheme wee conducted - to tune up the protocol and test the error assessment framework.

Run 81 - Low concentration



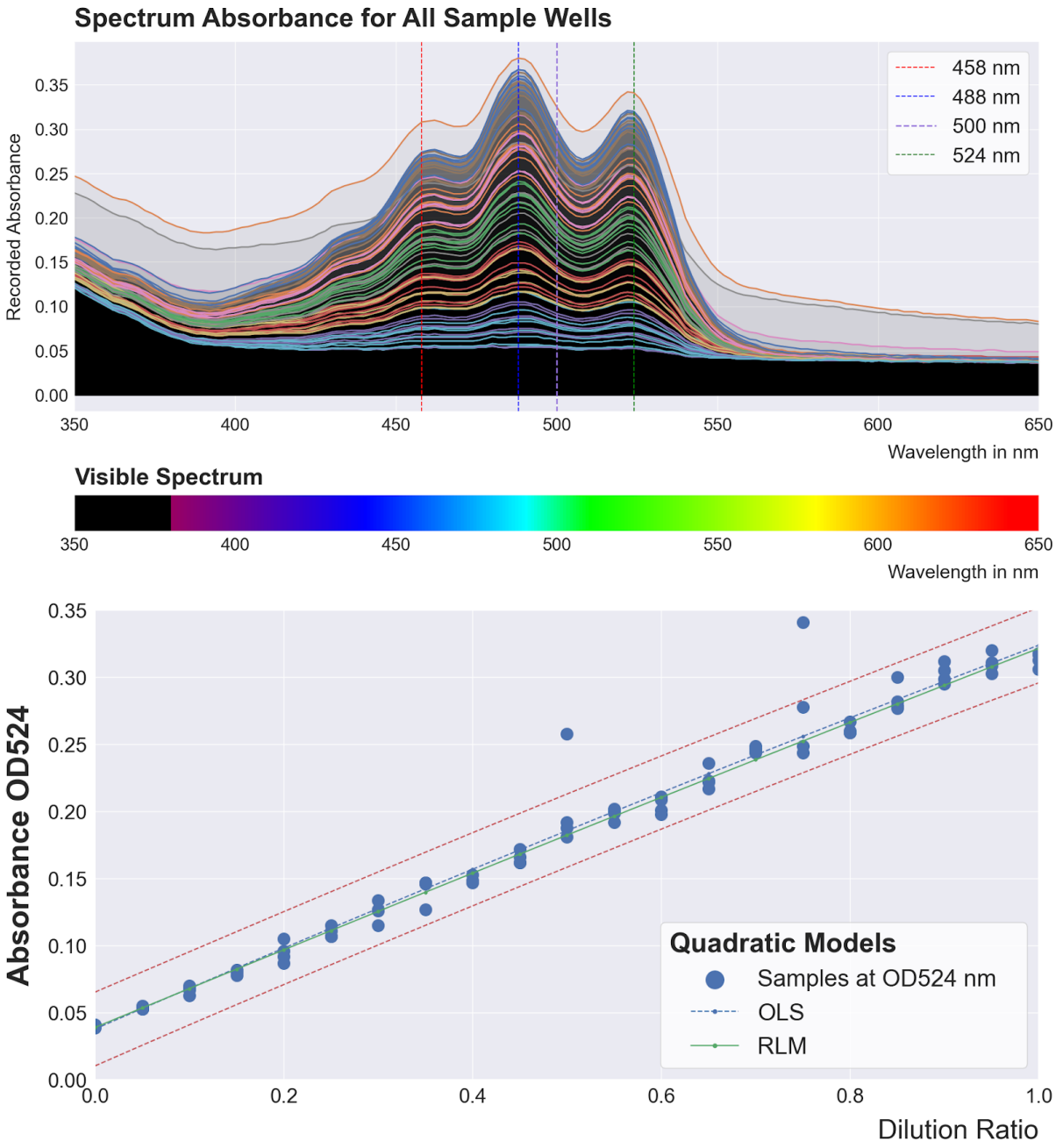
C4-Figure S48 - Run 81 – Absorbance spectrum and regression at 524 nm.

Run 80 - Low concentration



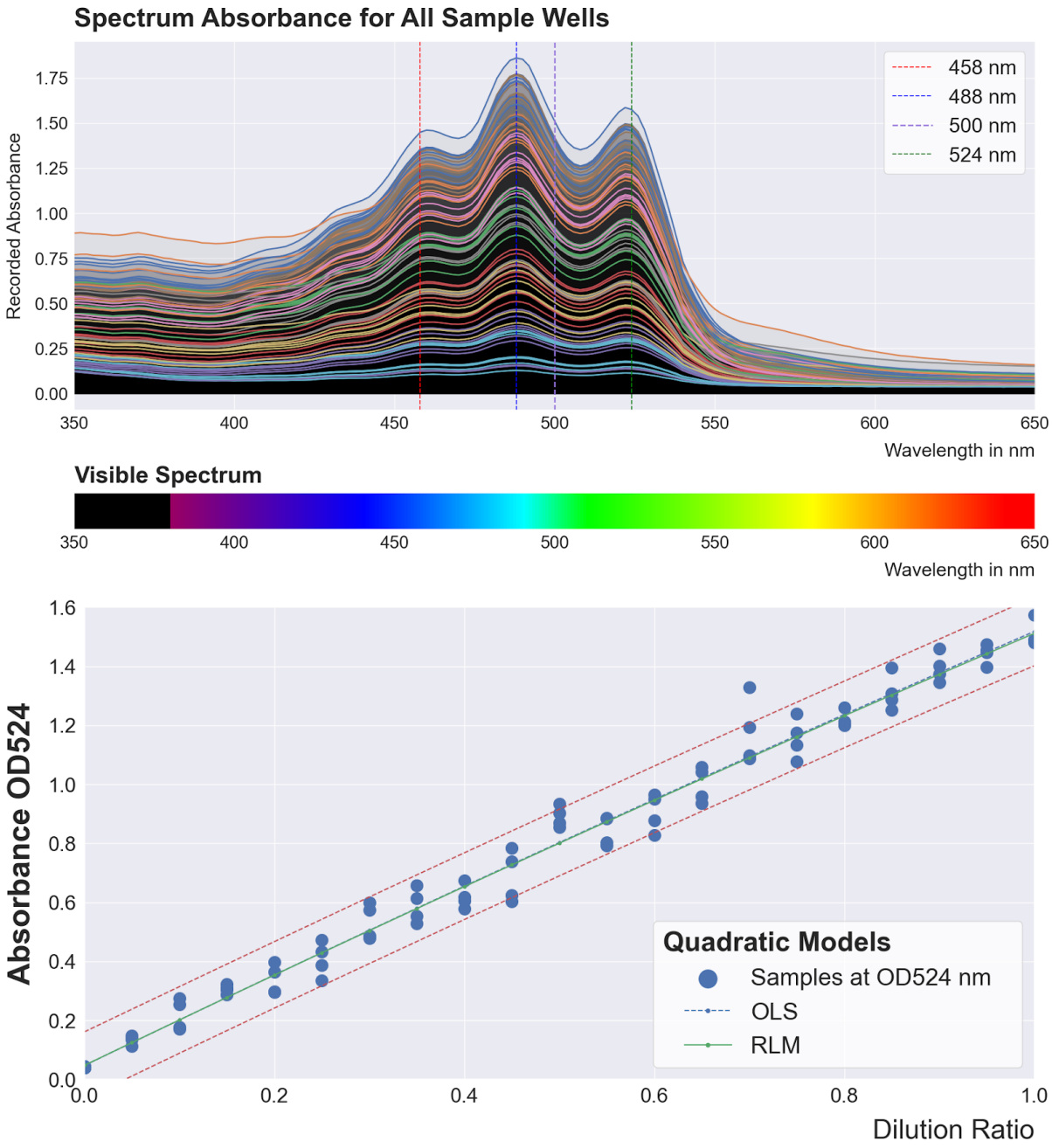
C4-Figure S49 - Run 80 – Absorbance spectrum and regression at 524 nm.

Run 82 - Low concentration



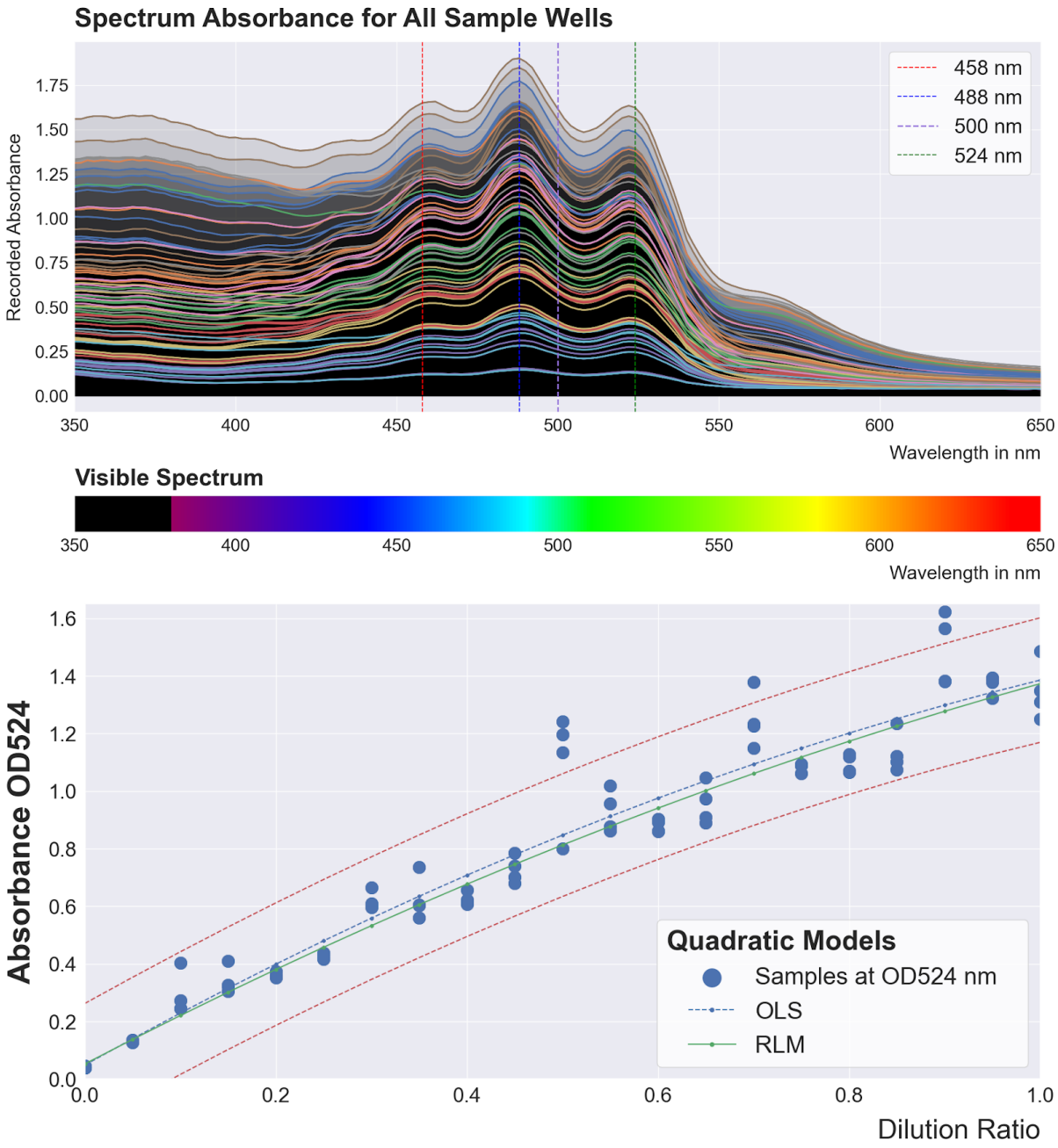
C4-Figure S50 – Run 82 – Absorbance spectrum and regression at 524 nm. Note the outliers

Run 76 – High Concentration Mix



C4-Figure S51 – Run 76 – Absorbance spectrum and regression at 524 nm.

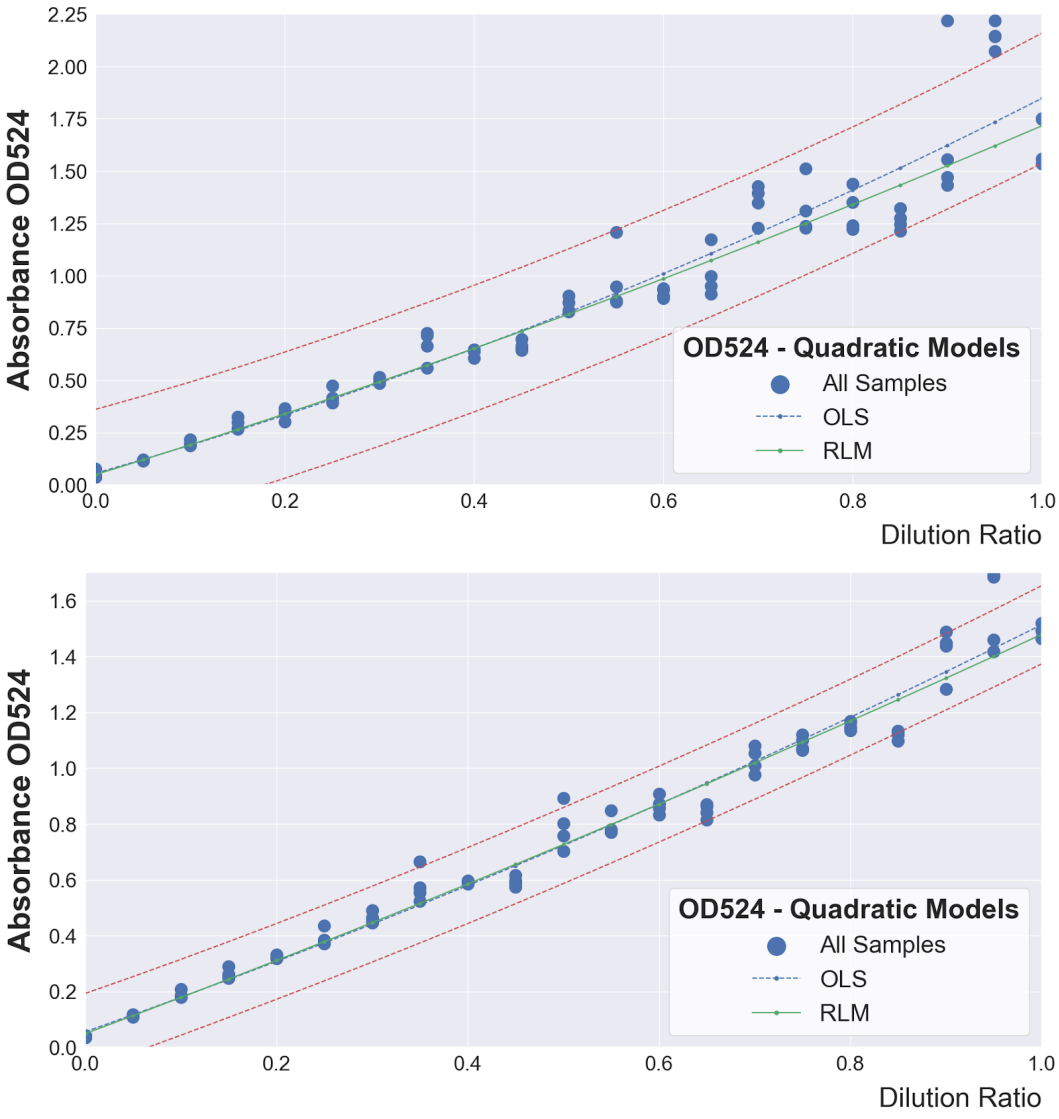
Run 79 – High Concentration Mix



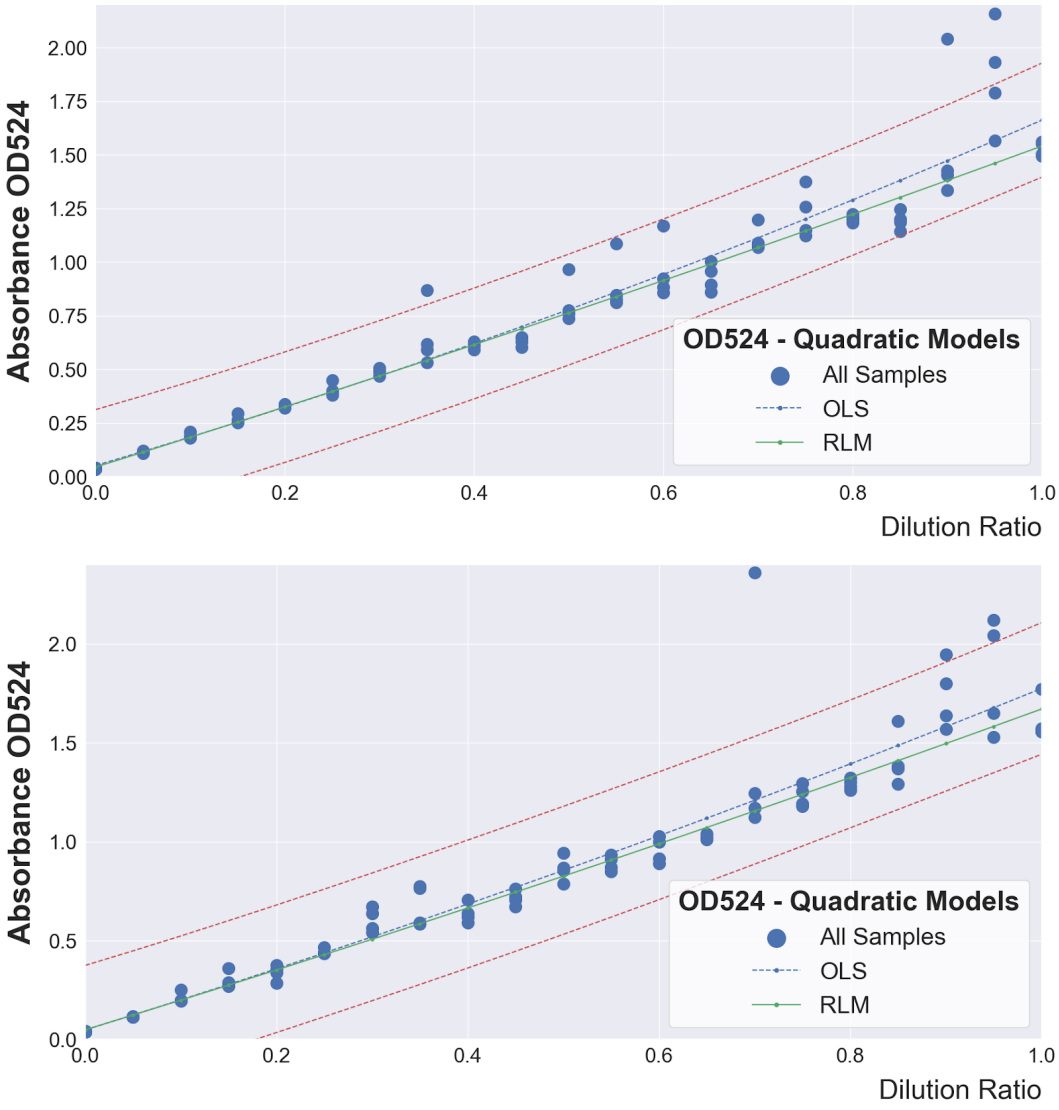
C4-Figure S52 – Run 79 – Absorbance spectrum and regression at 524 nm.

5.2 - Measurement study

Results below correspond to Run 085 – a high concentration lycopene mix. All technical repeats (measurements 1,3,5,7 and 9) were taken in a time scale of less than 30 minutes. Some samples were going on a “journey” - many regressing towards the trend of the dataset from the outset, some deviating (over a few measurements before returning to the trend.



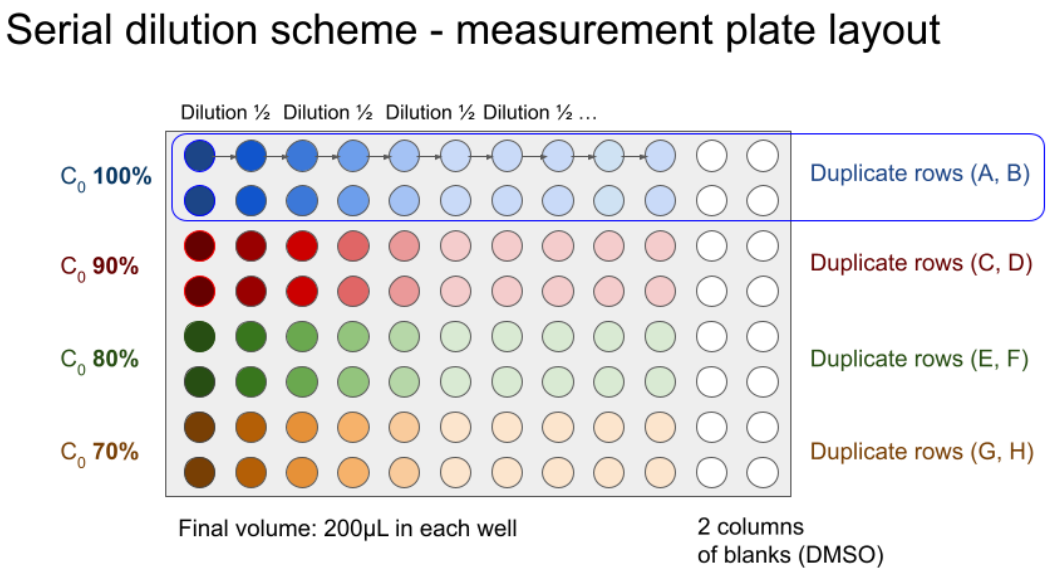
C4-Figure S53a - Evolution of the repeats of Run 085 - Repeats 1 and 3 at 524 nm.



C4-Figure S53b - Evolution of the repeats of Run 085 - Repeats 5 and 9 at 524 nm.

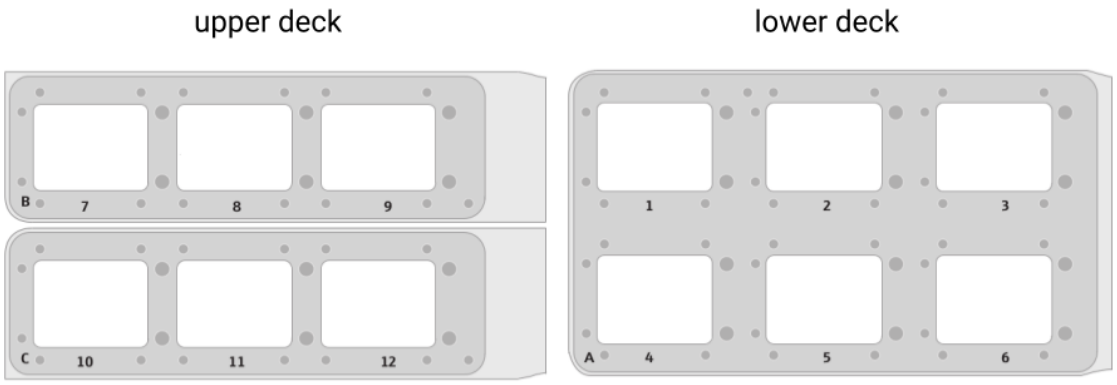
C6 - Automated Protocols

6.1 - Automated serial dilution protocol

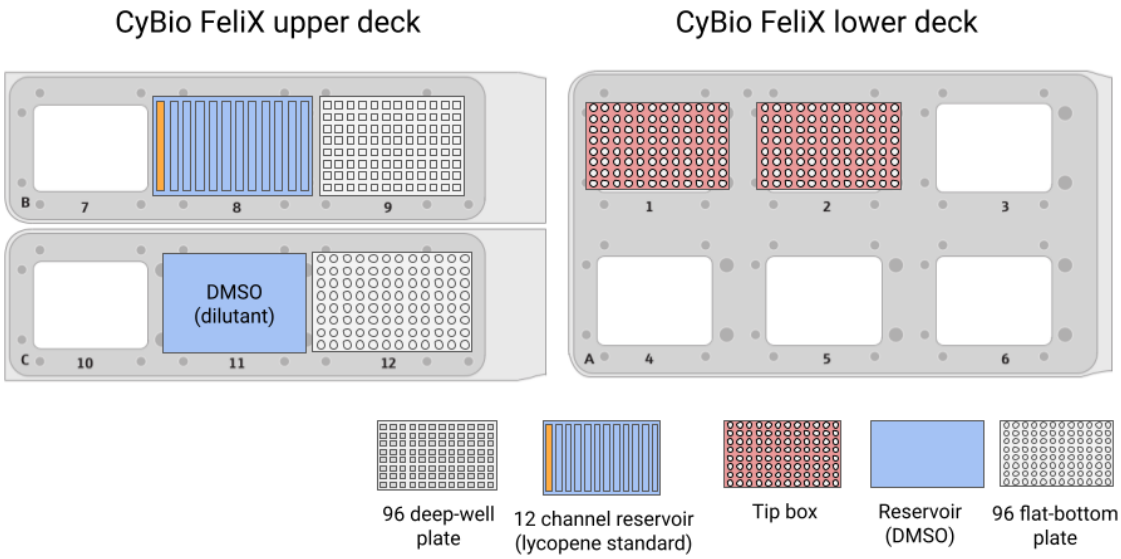


C4-Figure S54 – Automated serial dilution protocol - Plate map.

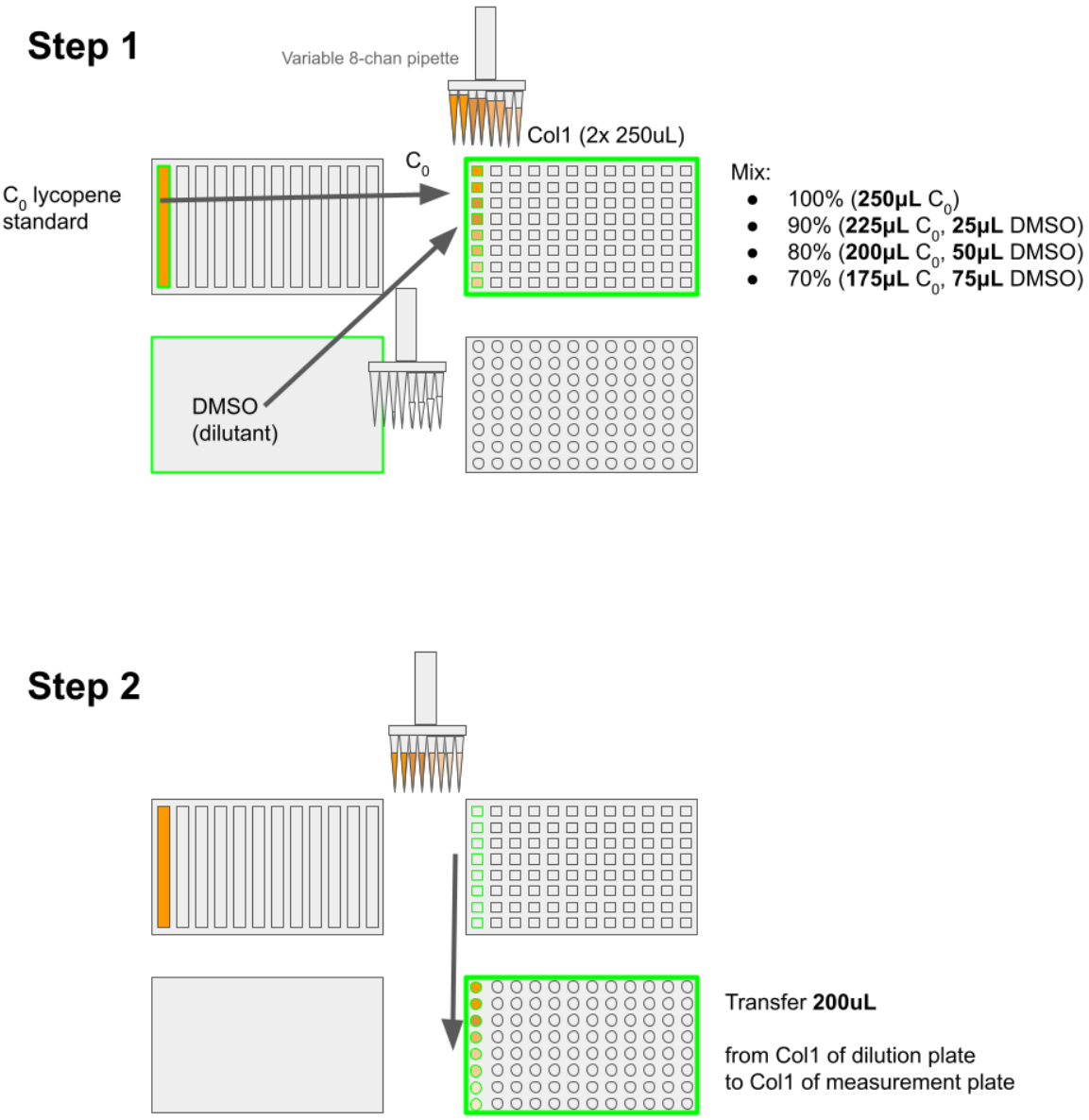
CyBio FeliX deck layout



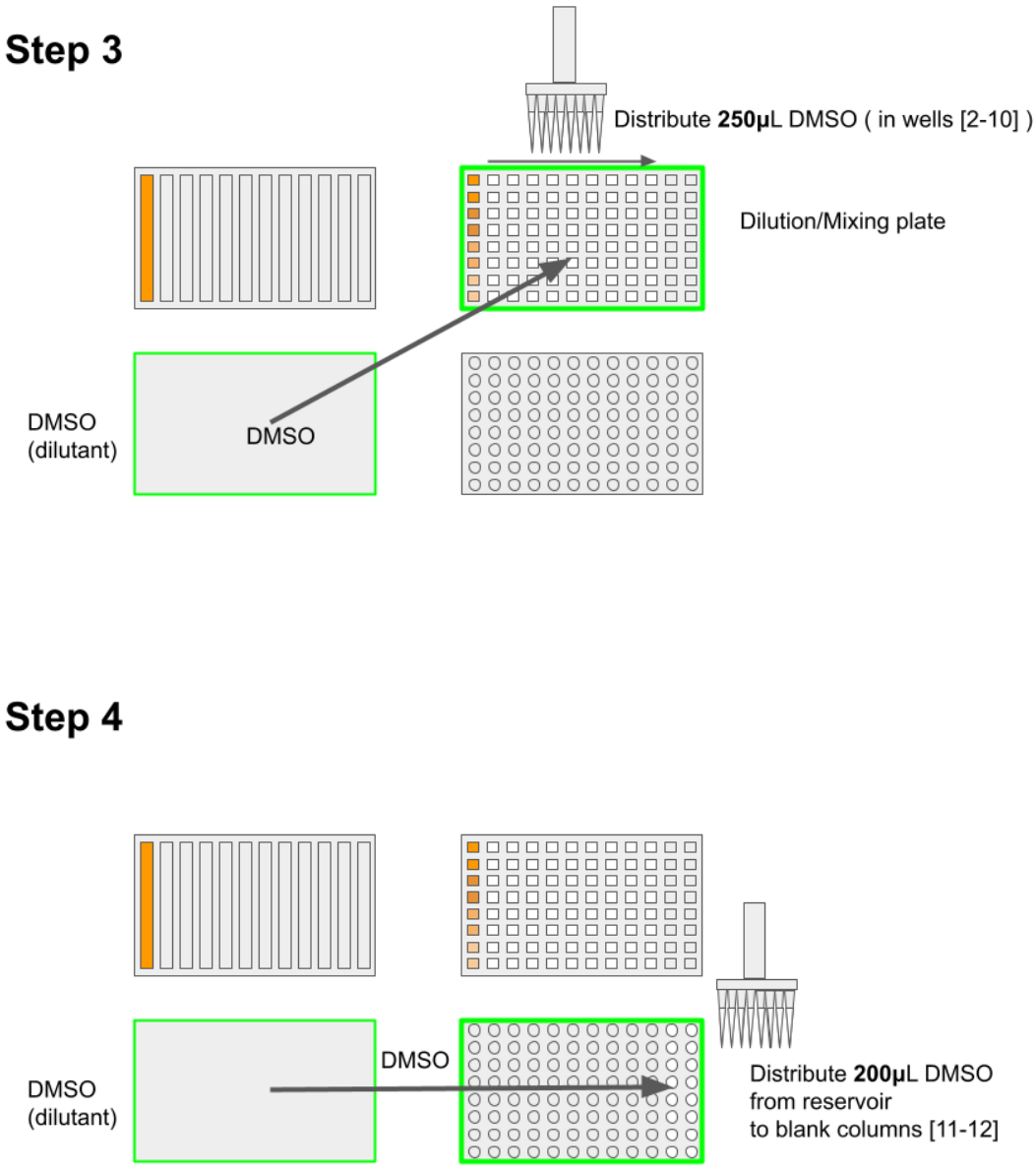
Serial dilution



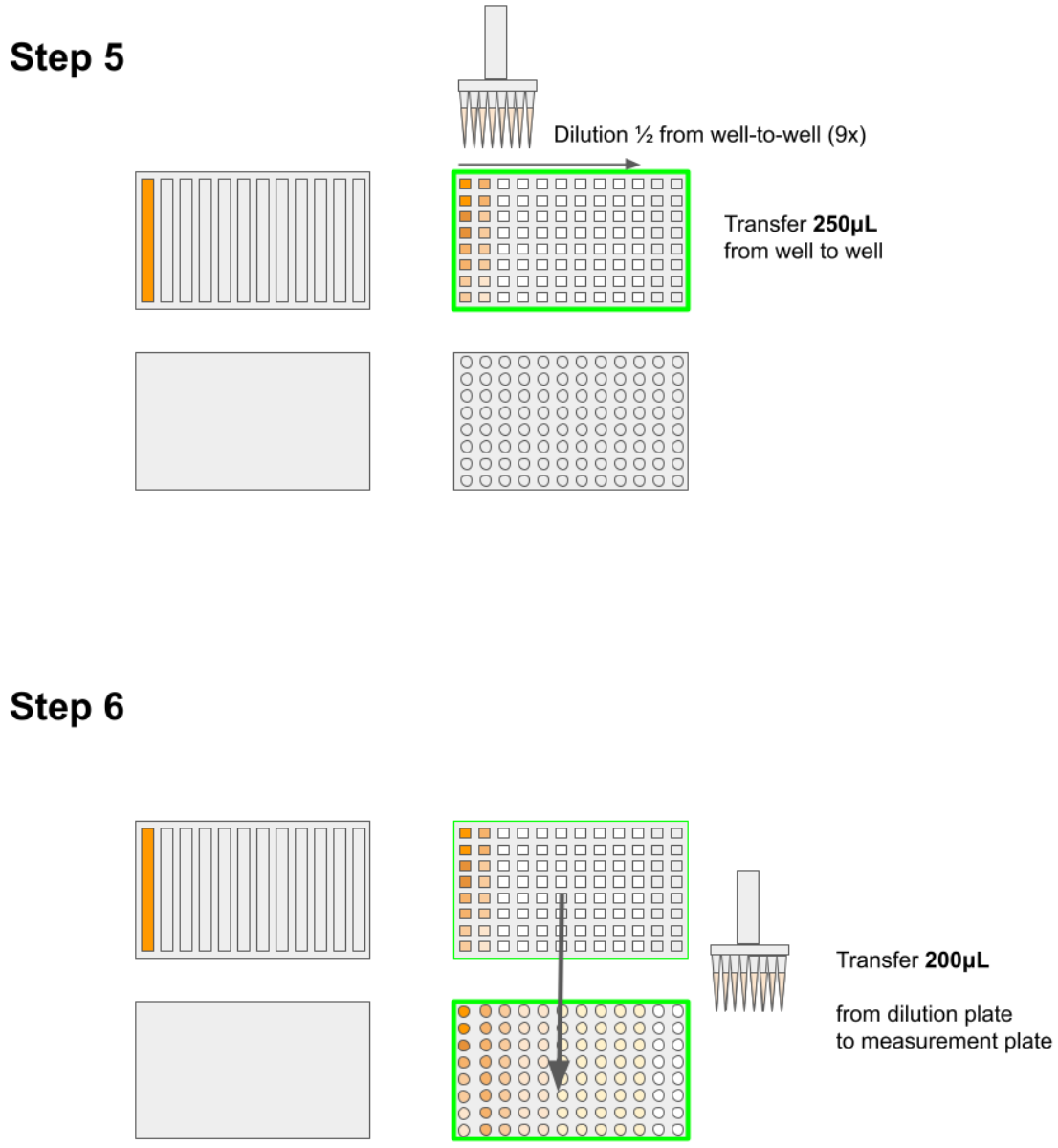
C4-Figure S55 – Automated serial dilution protocol - Robotic liquid handler (Felix) deck setup



C4-Figure S56a – Automated serial dilution protocol steps (Step 1 and 2).

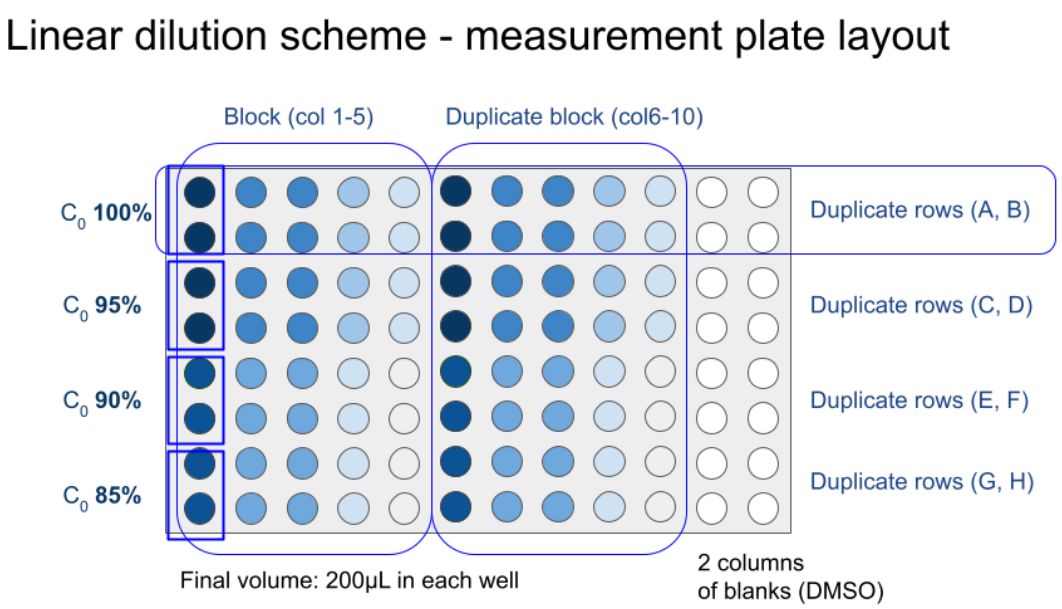


C4-Figure S56b – Automated serial dilution protocol steps (Step 3 and 4).



C4-Figure S56c – Automated serial dilution protocol steps (Step 5 and 6).

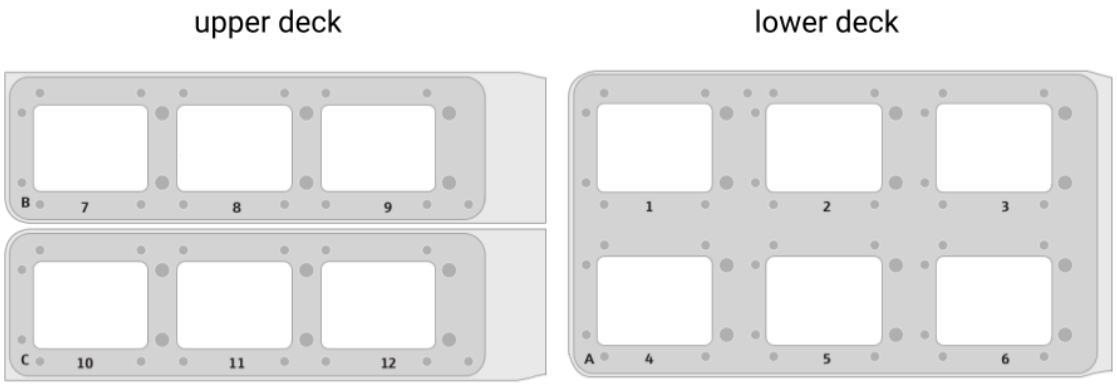
6.2 - Automated linear dilution protocol



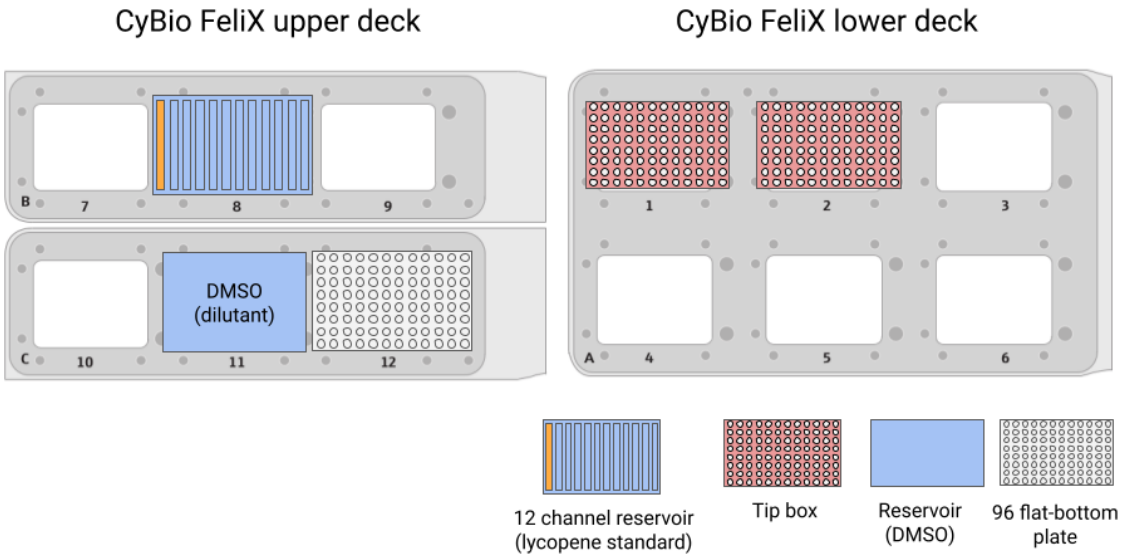
Dilution from 100% to 5%					Dilution (repeat)					Blanks	
A1											
100 %	80 %	60 %	40 %	20 %	100%	80 %	60 %	40 %	20 %	B	B
100 %	80 %	60 %	40 %	20 %	100%	80 %	60 %	40 %	20 %	B	B
95 %	75 %	55 %	35 %	15 %	95 %	75 %	55 %	35 %	15 %	B	B
95 %	75 %	55 %	35 %	15 %	95 %	75 %	55 %	35 %	15 %	B	B
90 %	70 %	50 %	30 %	10 %	90 %	70 %	50 %	30 %	10 %	B	B
90 %	70 %	50 %	30 %	10 %	90 %	70 %	50 %	30 %	10 %	B	B
85 %	65 %	45 %	25 %	5 %	85 %	65 %	45 %	25 %	5 %	B	B
85 %	65 %	45 %	25 %	5 %	85 %	65 %	45 %	25 %	5 %	B	B

C4-Figure S57 – Automated LinearDilution Protocol – Plate Map.

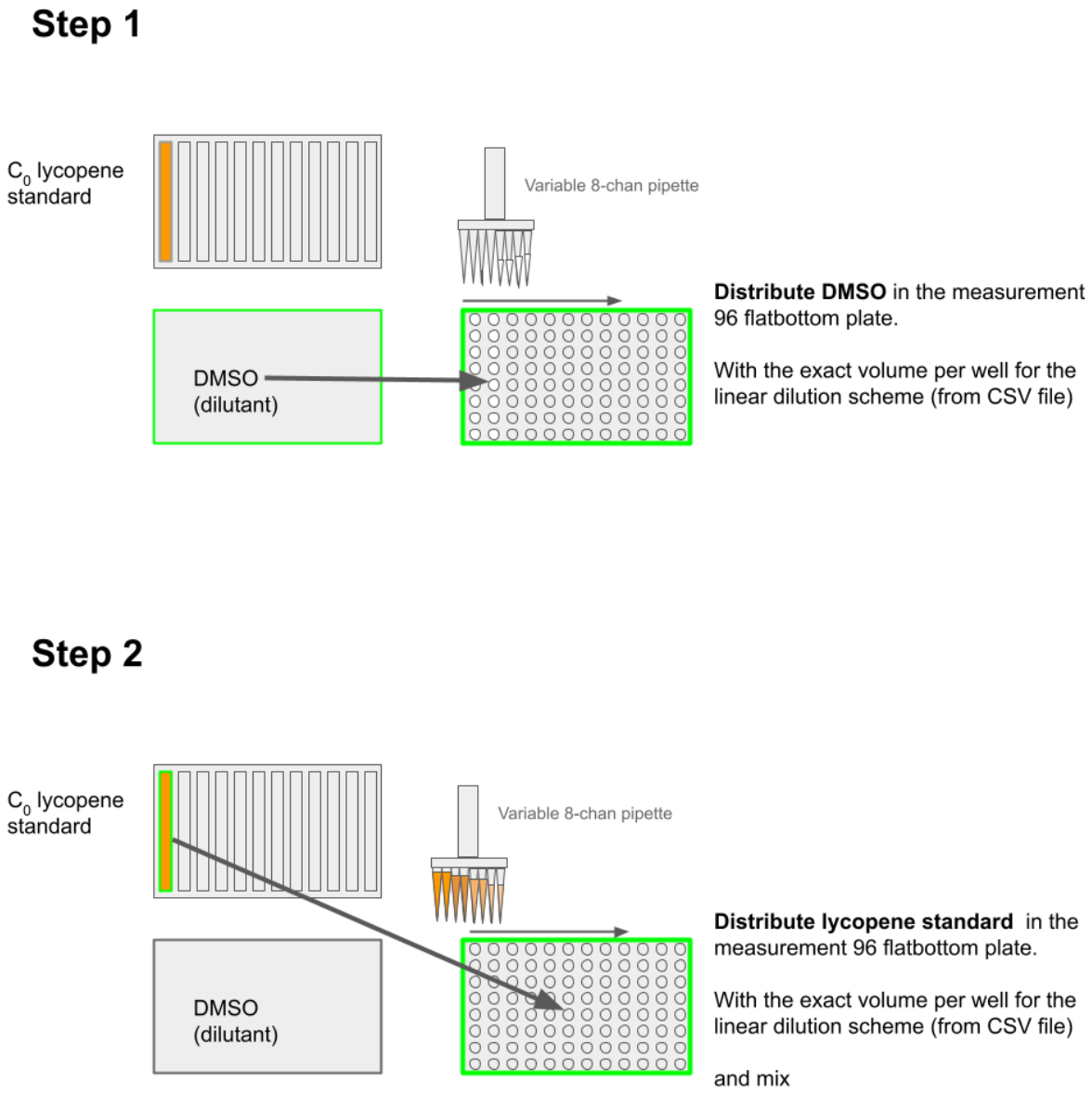
CyBio FeliX deck layout



Linear dilution



C4-Figure S58 – Automated linear dilution protocolRobotic liquid handler (Felix) deck setup.

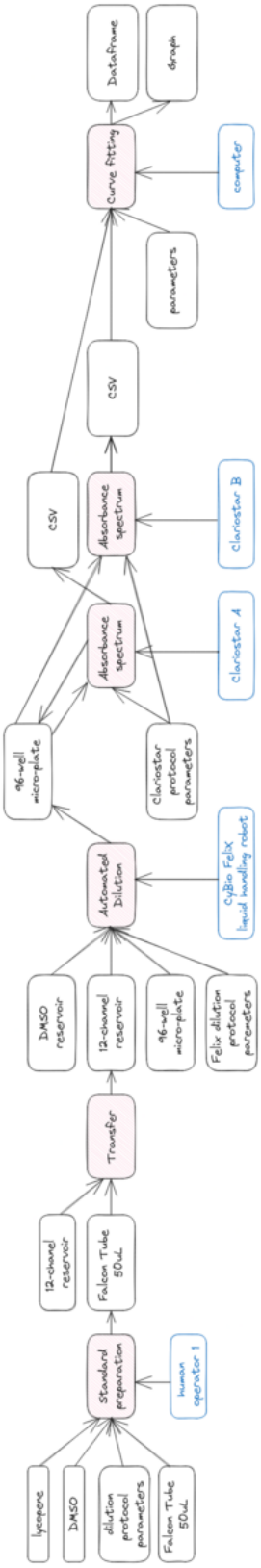


C4-Figure S59 – Automated serial dilution protocol steps.

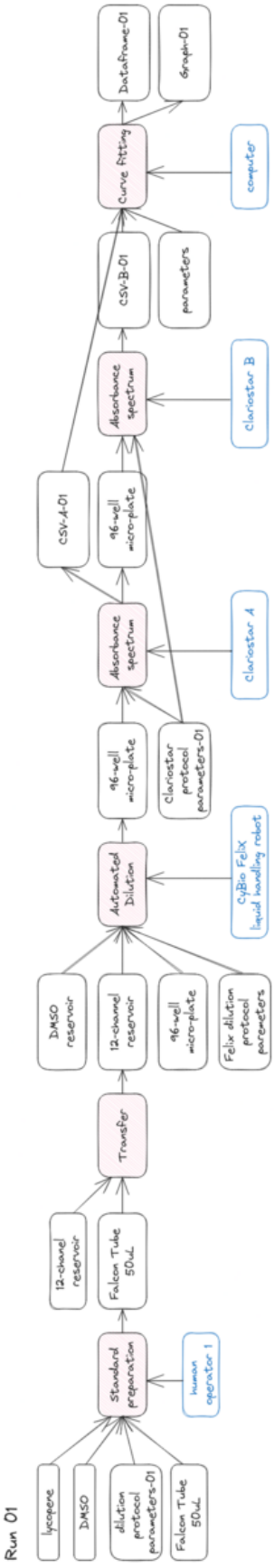
Appendices C6

C6 - Model Graph

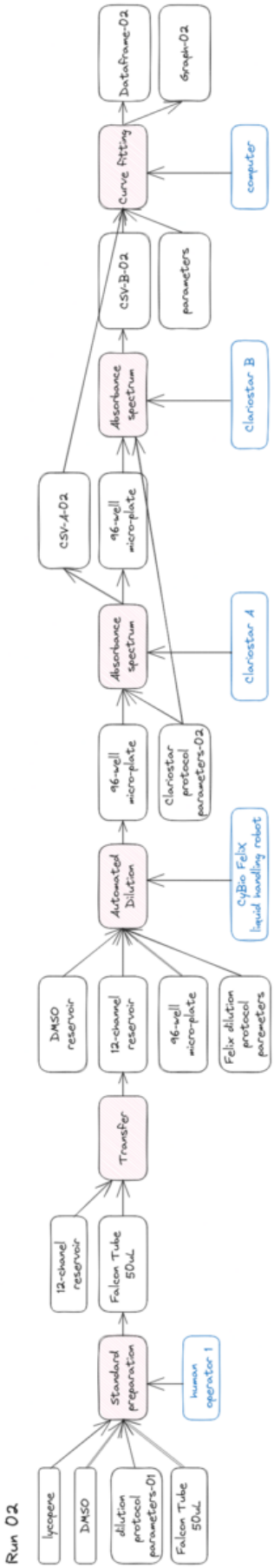
The next three figures C6 Figure S1, C6 Figure S3 and C6 Figure S3 are full page (landscape orientation) versions of Figure 6-14, Figure 6-15 and Figure 6-16 found in Chapter 6.



C6 Figure S1 – Example of the Automated Dilution Standard Model Graph. Resources are shown in blue.



C6-Figure S2 – Example of the Automated Dilution Standard Execution-Graph: Run 01.



C6-Figure S3 – Example of the Automated Dilution Standard Execution-Graph: Run 02

C6 - Examplar 2 – Serial Dilution for Standard Curve

Example of a workflow: serial dilution for standard curve

In the linear workflow below we automate a serial dilution for a lycopene standard curve. The goal of this wetlab process is to take a lycopene standard (pharma grade) and dissolve it in the diluant used for lycopene extraction (we used DMSO), and then make serial dilutions with a dilution step of 5% in a liquid handling robot (Cybio Felix) and then measure the absorption spectrum in a plate-reader (BMG Clariostar) to derive a standard curve.

Next page (Figure C6-S4) is the complete workflow of the serial dilution workflow.

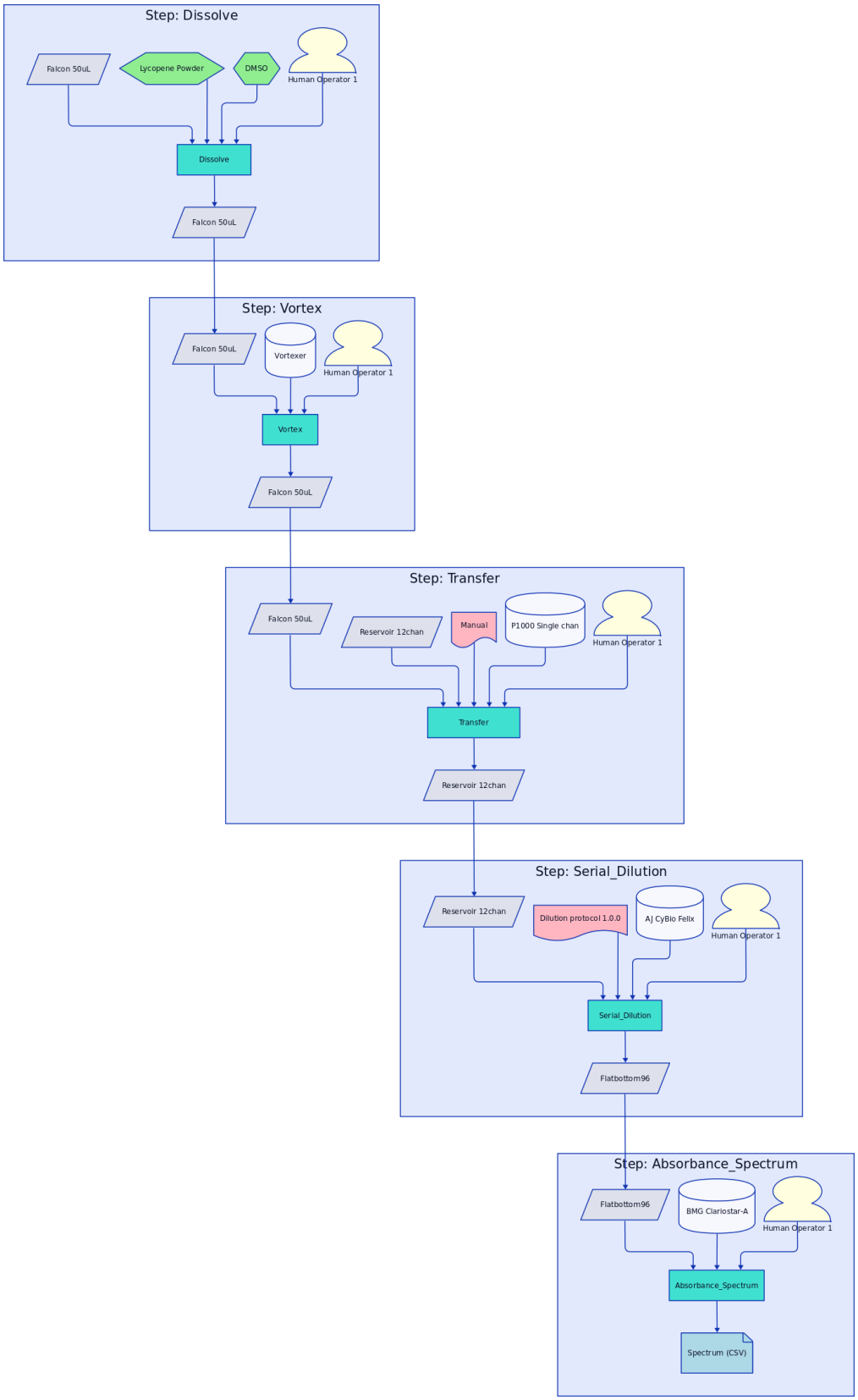


Figure C6-S4: Complete serial dilution workflow (same graphic as Figure 6-27, duplicated here to help the reader).

The pseudo-code below shows the tasks functions definitions with their parameters as well as the dependencies represented by the bit-shift operator symbol ”>>”.

‘Task A >> Task B’ means that Task B is dependent on Task A.

Workflow pseudo-code:

```
# Tasks definition
dissolve = dissolve( in_container=falcon, powder=lyc_powder, dilutant=dmso,
       user=HOP1 )
vortex = vortex( falcon, vortexer, user=user )
transfer = transfer( in_containers=[falcon, reservoir12],
       out_container=reservoir12, protocol=manual, equipment=p1000, user=HOP1 )
serial_dilution = serial_dilution( in_containers=[reservoir12],
       out_container=flatbottom96, protocol=dissolve, equipment=robot, user=HOP1)
absorbance_spectrum = absorbance_spectrum( container=flatbottom96,
       equipment=platereader, user=HOP1 )
```

Step 1: Dilution step

The first step in the pipeline is the dilution of the lycopene crystals in a 50mL Flacon tube of DMSO.

Why? Having a liquid form of pharma graded lycopene allows us to measure it's absorbtion spectrum to derive a lycopene standard.

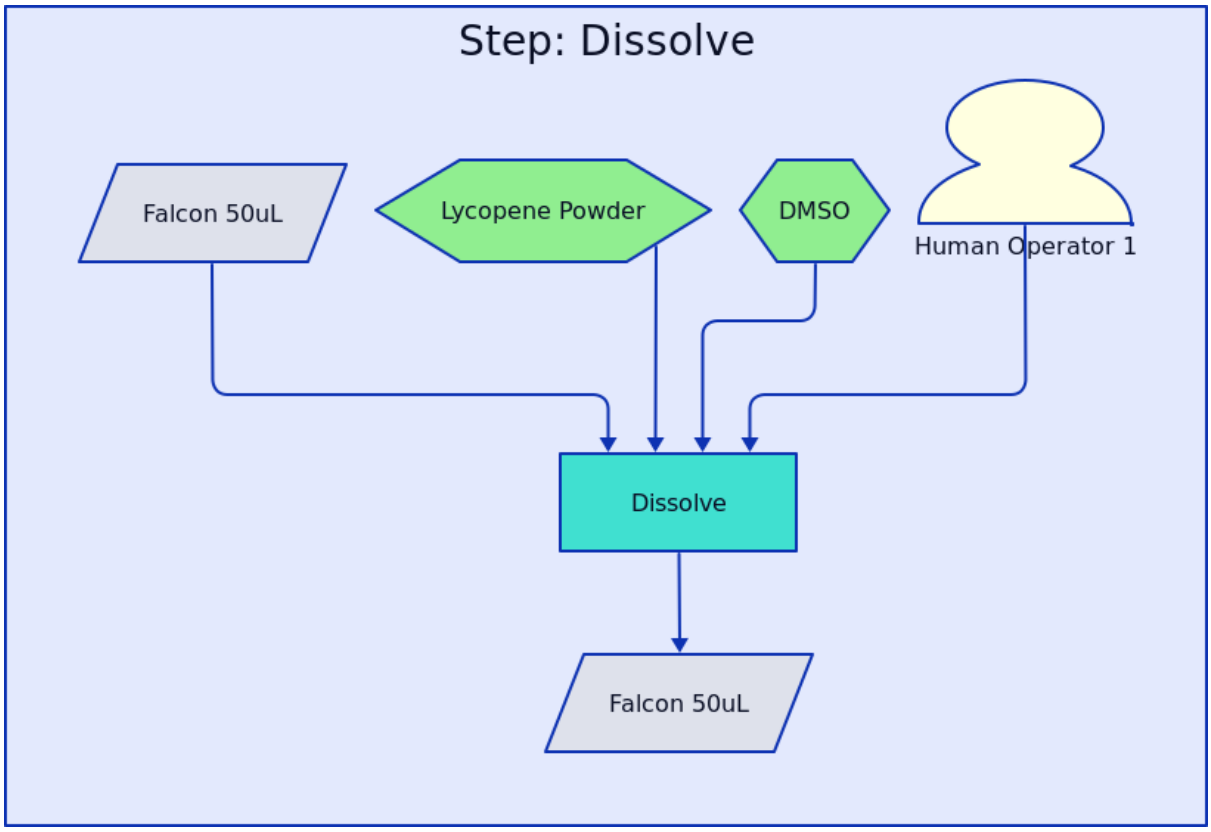


Figure C6-S5 – Workflow step: Dissolve.

FUNCTION DEFINITION:
Dissolve(in_container, powder, dilutant, user) → container

INPUTS:
- in_container (Container)
- powder (Reagent)
- dilutant (Reagent)
- user (Person)

OUTPUTS:
- container (Container)

FUNCTION DESCRIPTION:
Dissolve Lycopene powder/crystals
Example: Dissolve(Tube, Lycopene Powder, DMSO, Human Operator 1) → Falcon 50uL

INPUTS:
- Tube Falcon 50mL(Thermo Fisher) (Container)
- Lycopene Powder (Sigma-Aldrich) (Reagent)
- DMSO (Fisher Scientific) (Reagent)
- Human Operator 1(Person)

OUTPUTS:
- Falcon 50uL (Container)

Step 2: Vortex

This step consits of vortexing a Flacon tube with a vortexer.

• Why? In order to dissolve the lycopene crystals, the tube containing DMSO and lycopene crystals is vortexed for 5 minutes.

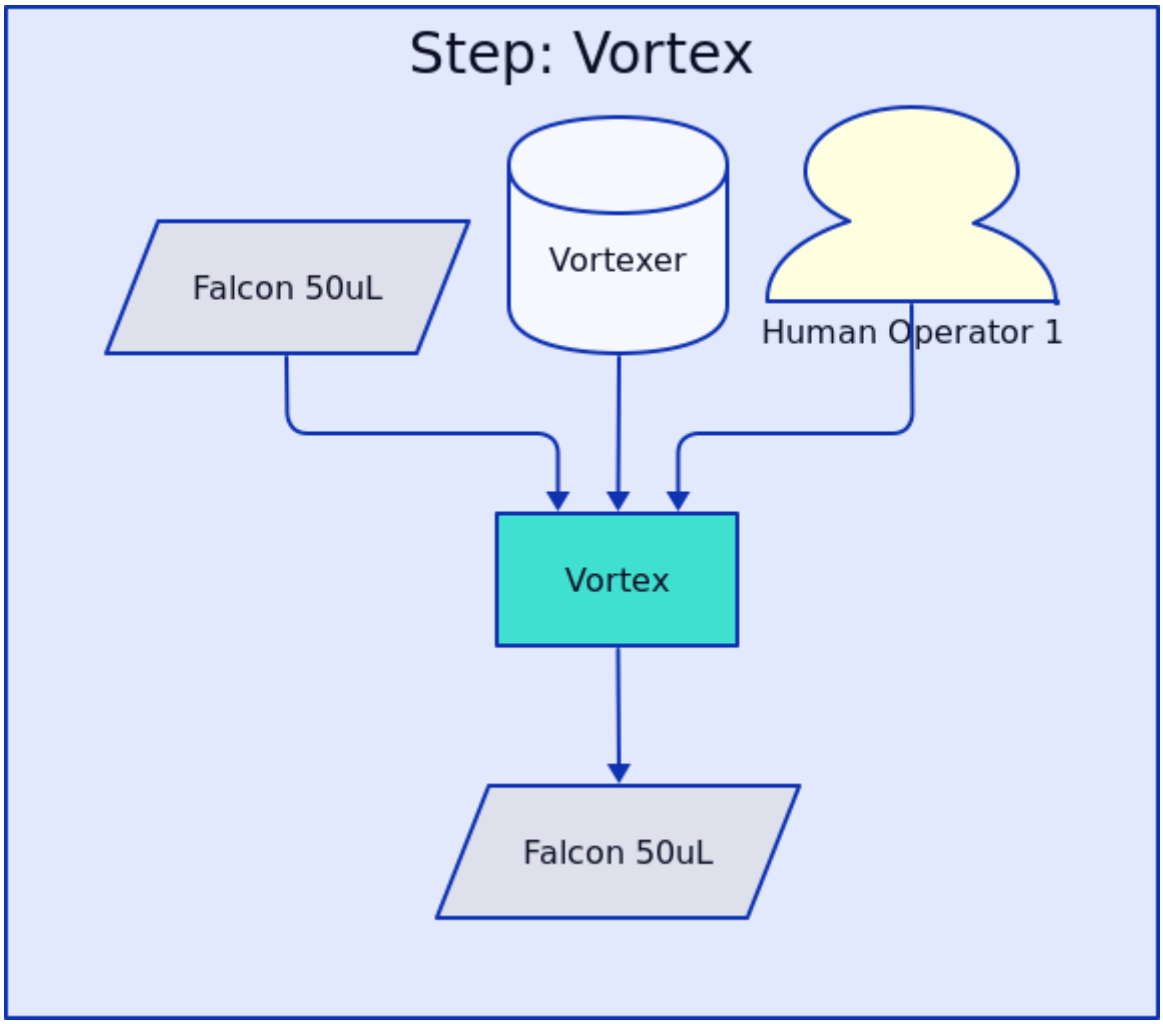


Figure C6-S6 – Workflow step: Vortex.

FUNCTION DEFINITION:
Vortex(in_container, equipment, user) → container

INPUTS:
- in_container (Container)
- equipment (Equipment)
- user (Person)

OUTPUTS:
- container (Container)

FUNCTION DESCRIPTION:
Vortex
Example: Vortex(Tube, Vortexer, Human Operator 1) → Falcon 50uL

INPUTS:
- Tube Falcon 50mL(Thermo Fisher) (Container)
- Vortexer (Vendor Model) (Equipment)
- Human Operator 1 (Person)

OUTPUTS:
- Falcon 50uL (Container)

Step 3: Transfer

In this step, 10uL of the dissolved DMSO/lycopene tube is transfered in a 12-channel reservoir.

• Why? Transfering the lycopene + DMSO solution in a 12-channel reservoir is a container more appropriate for pipetting with a 8-channel pipette in the liquid handling robot.

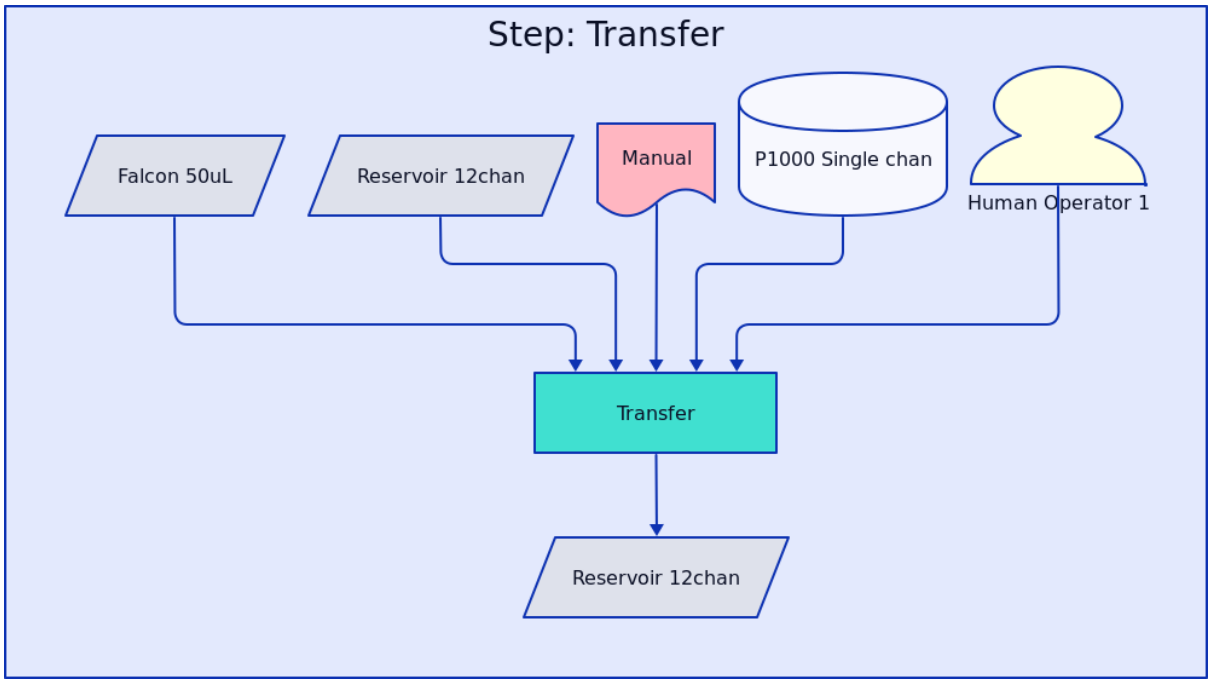


Figure C6-S7 – Workflow step: Transfer.

FUNCTION DEFINITION:
Transfer(in_containers, out_container, protocol, equipment, user) → container

INPUTS:
- in_containers (Container)
- out_container (Container)
- protocol (Protocol)
- equipment (Equipment)
- user (Person)

OUTPUTS:
- container (Container)

FUNCTION DESCRIPTION:
Transfer
Example:
Transfer(Tube, Reservoir 12chan, Manual, Single channel pipette, Human Operator 1) → Reservoir 12chan

INPUTS:
- Tube Falcon 50mL(Thermo Fisher) (Container)
- (Container)
- Manual-protocol v0.0.0 (Protocol)
- Single channel pipette (Generic P1000) (Equipment)
- Human Operator 1 (Person)

OUTPUTS:
- Reservoir 12chan (Container)

Step 4: Serial Dilution

A liquid handling robot (Felix) is then used to perform successive transfer and dilutions of the initial standard diluted lycopene in DMSO with DMSO. Steps of 5% dilutions are performed giving a final range of 5% to 100% diluted lycopene in DMSO.

• Why? This way of generating a serial dilution gives us many more points to interpolate the curve for the standard. The dilution is done in a 96-flatbottom plate at a total volume of 200uL per well.

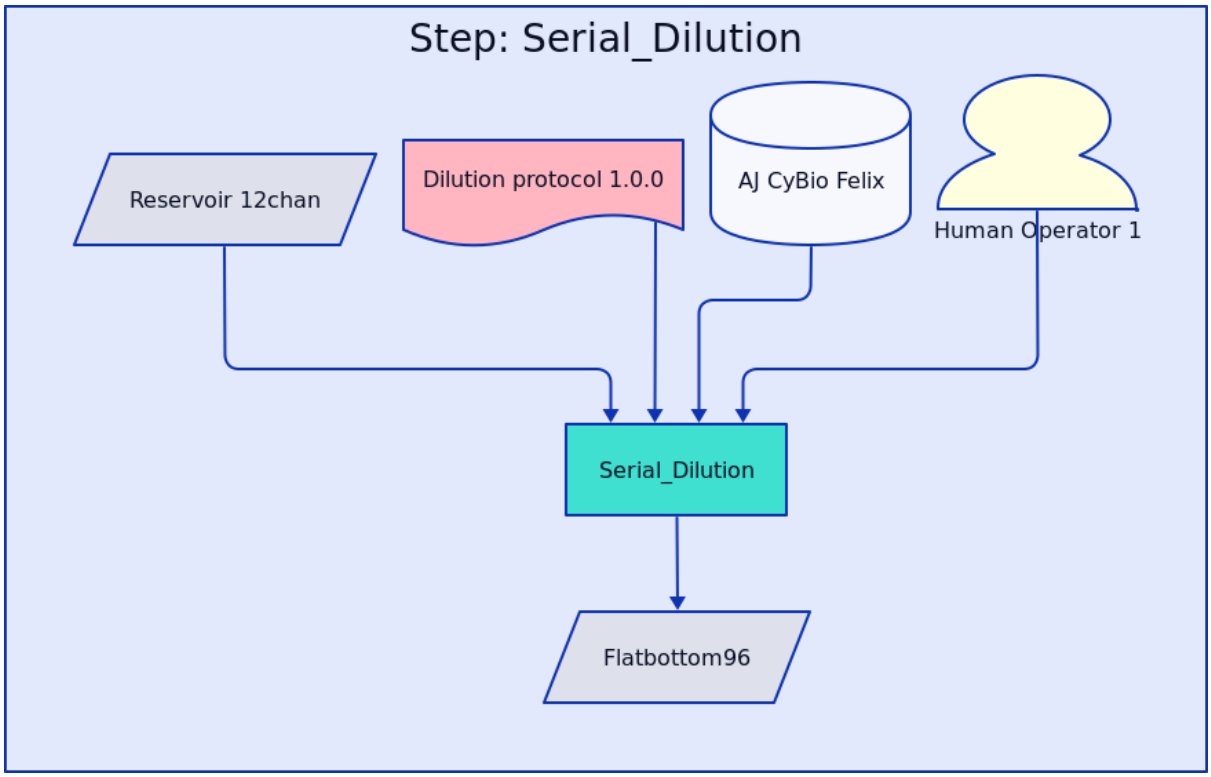


Figure C6-S7 – Workflow step: Serial_Dilution.

FUNCTION DEFINITION:
Serial_Dilution(in_containers, out_container, protocol, equipment, user) → container

INPUTS:
- in_containers (Container)
- out_container (Protocol)
- protocol (Equipment)
- equipment (Person)

OUTPUTS:
- container (Container)

FUNCTION DESCRIPTION:
Serial Dilution
Example:Serial_Dilution(Reservoir 12chan, Dilution, Robotic Liquid Handler, Human Operator 1) → Flatbottom96

INPUTS:
- (Container)
- Dilution-protocol v1.0.0 (Protocol)
- Robotic Liquid Handler (Analytik Jena CyBio-Felix) (Equipment)
- Human Operator 1 (Person)

OUTPUTS:
- Flatbottom96 (Container)

Step 5: Absorbance measurement

The 96-flatbottom plate is tranfered to a microplate-reader (BMG Clariostar) for a spectral absorbance measurement. The resulting measurement data is exported to a CSV file for further analysis and creation of a lycopene standard curve.

• Why? The micro-plate reading of the absorbance spectrum will give us the final values for the interpolation of the lycopene standard curve

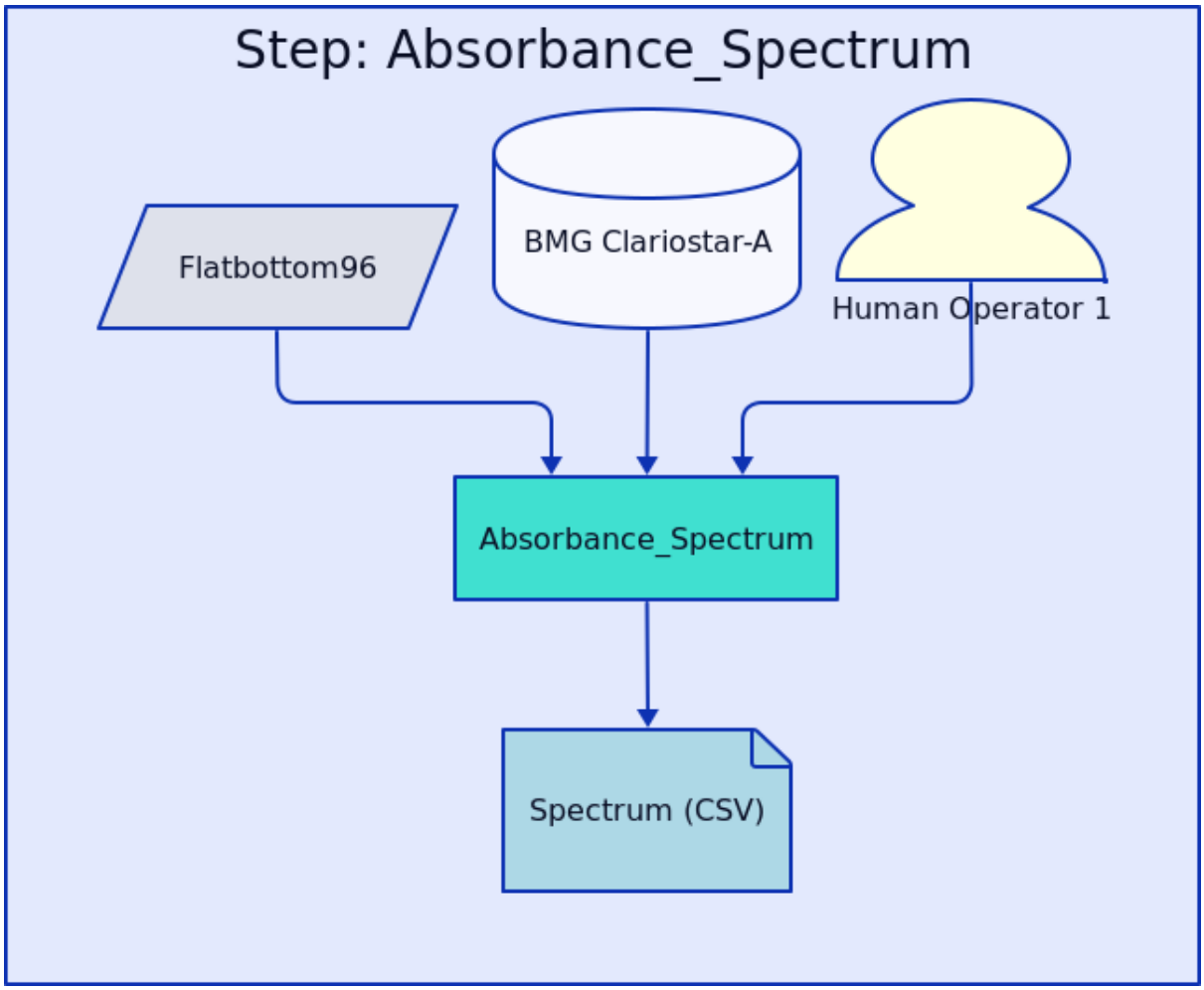


Figure C6-S8 – Workflow step: Absorbance_Spectrum.

FUNCTION DEFINITION:
Absorbance_Spectrum(container, equipment, user) → spectrum_csv

INPUTS:
- container (Container)
- equipment (Equipment)
- user (Person)

OUTPUTS:
- spectrum_csv (File)

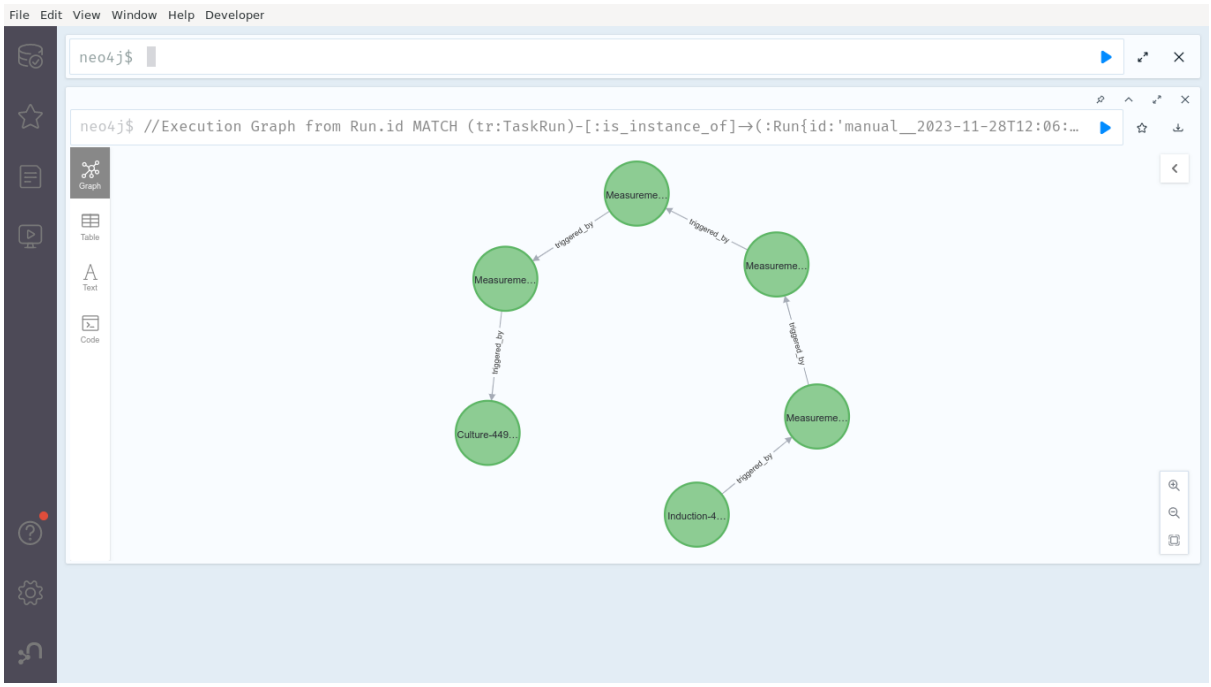
FUNCTION DESCRIPTION:
Absorbance spectrum
Example: Absorbance_Spectrum(Flatbottom96, Clariostar-A, Human Operator 1) → Spectrum

INPUTS:
- (Container)
- Clariostar-A (BMG Clariostar Plus) (Equipment)
- Human Operator 1 (Person)

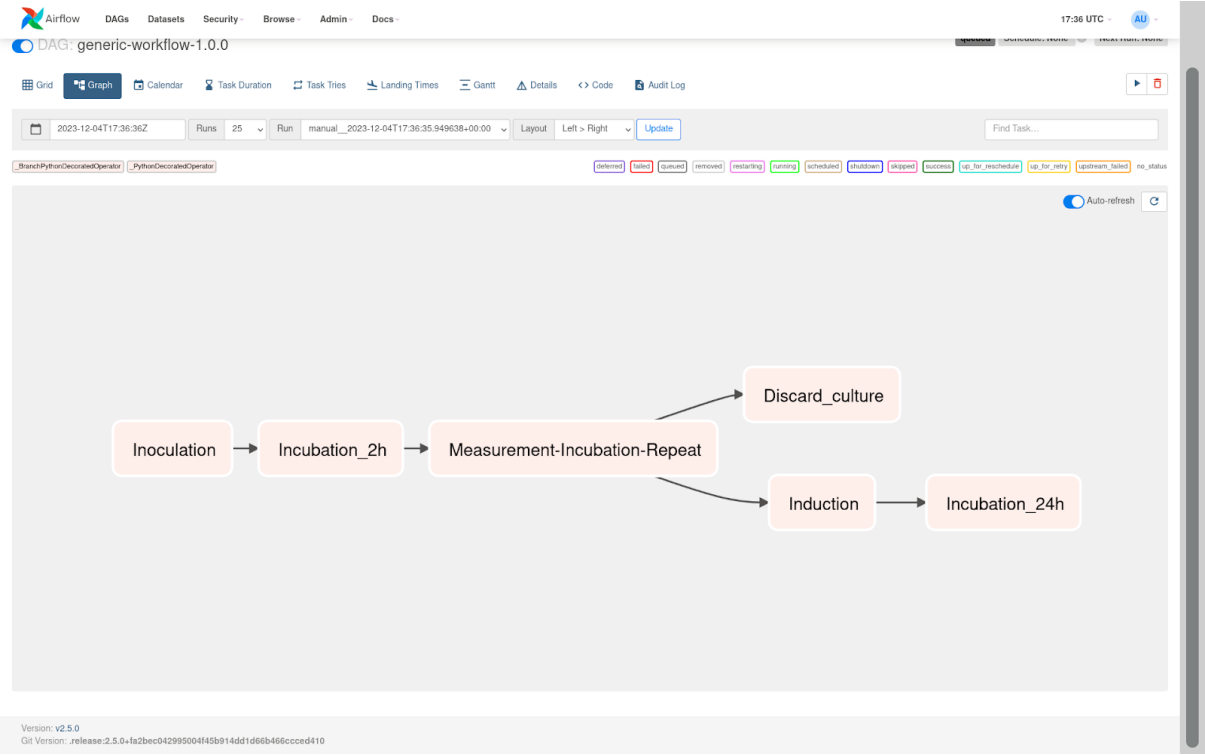
OUTPUTS:
- Spectrum (File)

Appendices C7

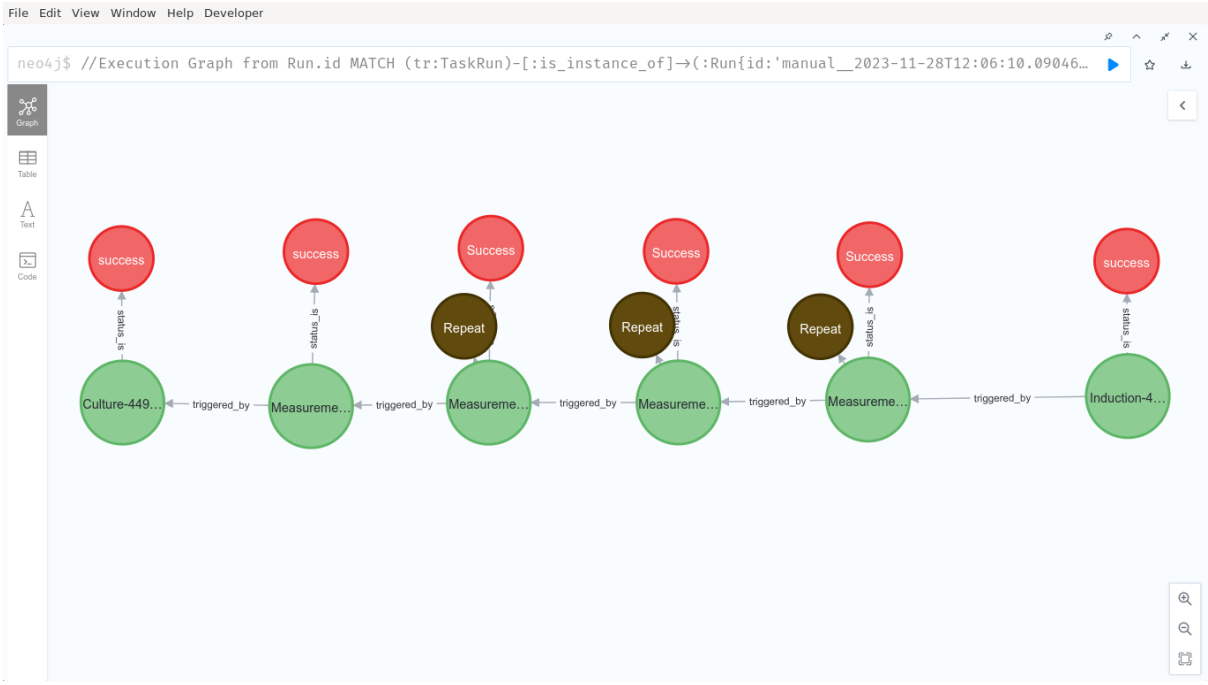
C7- Examples



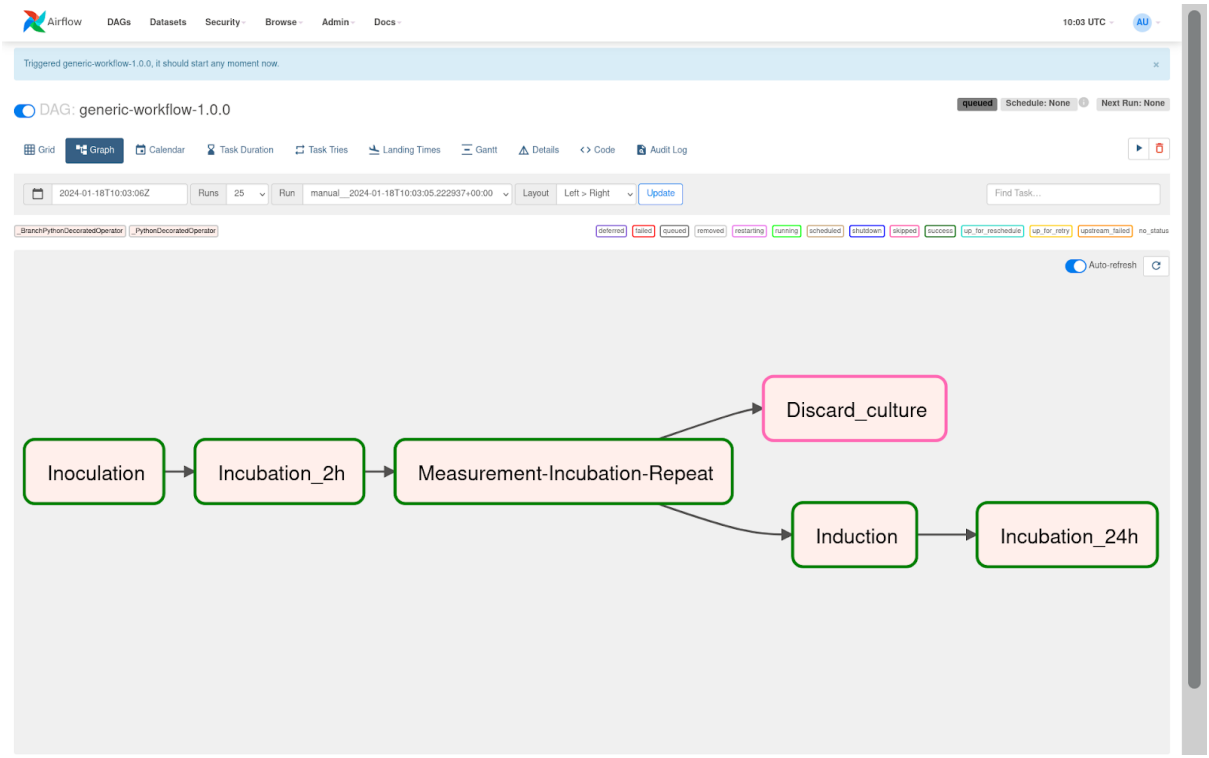
C7-Figure S1 – Execution Graph of the run of the workflow (Screenshot of Neo4j Browser user interface). Screenshot of the user interface of Neo4j Browser showing the workflow with a repetition on the task called ‘Measurement’ (implemented with Strategy 2).

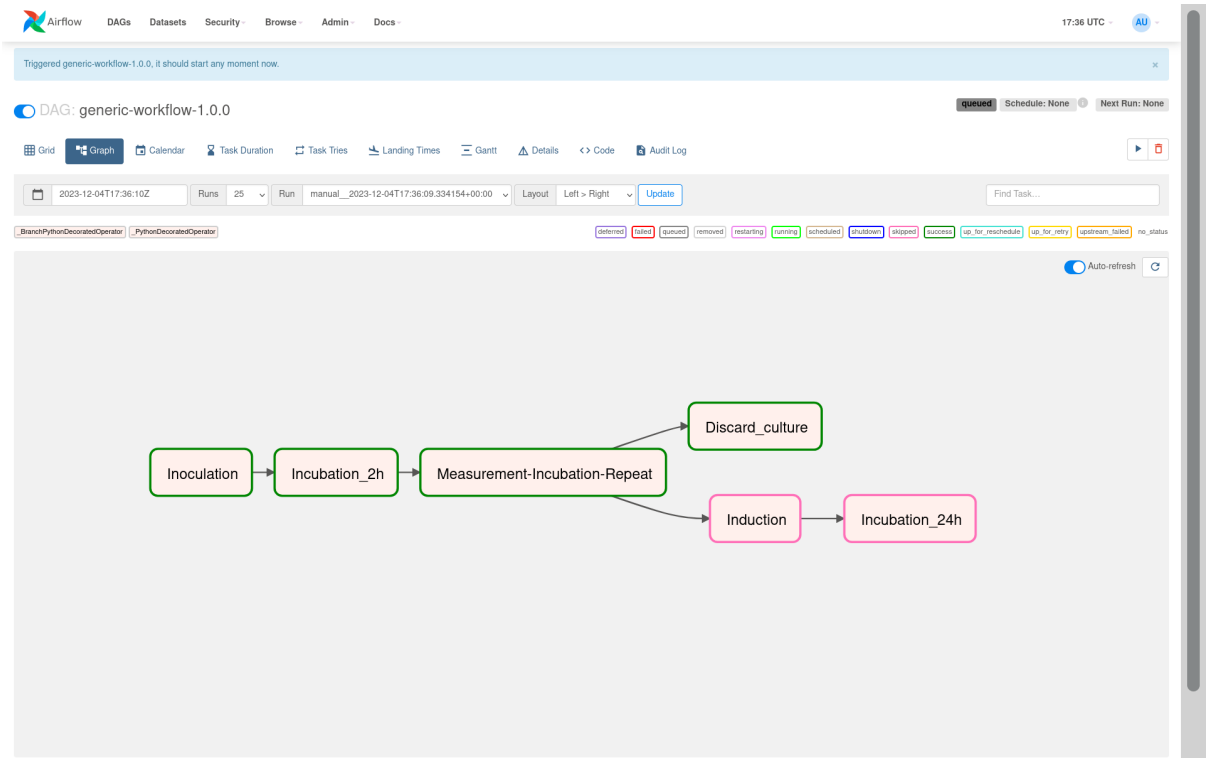


C7-Figure S2 – 'Generic workflow' as represented in Airflow's graphical interface. Screenshot of the user interface of Airflow.

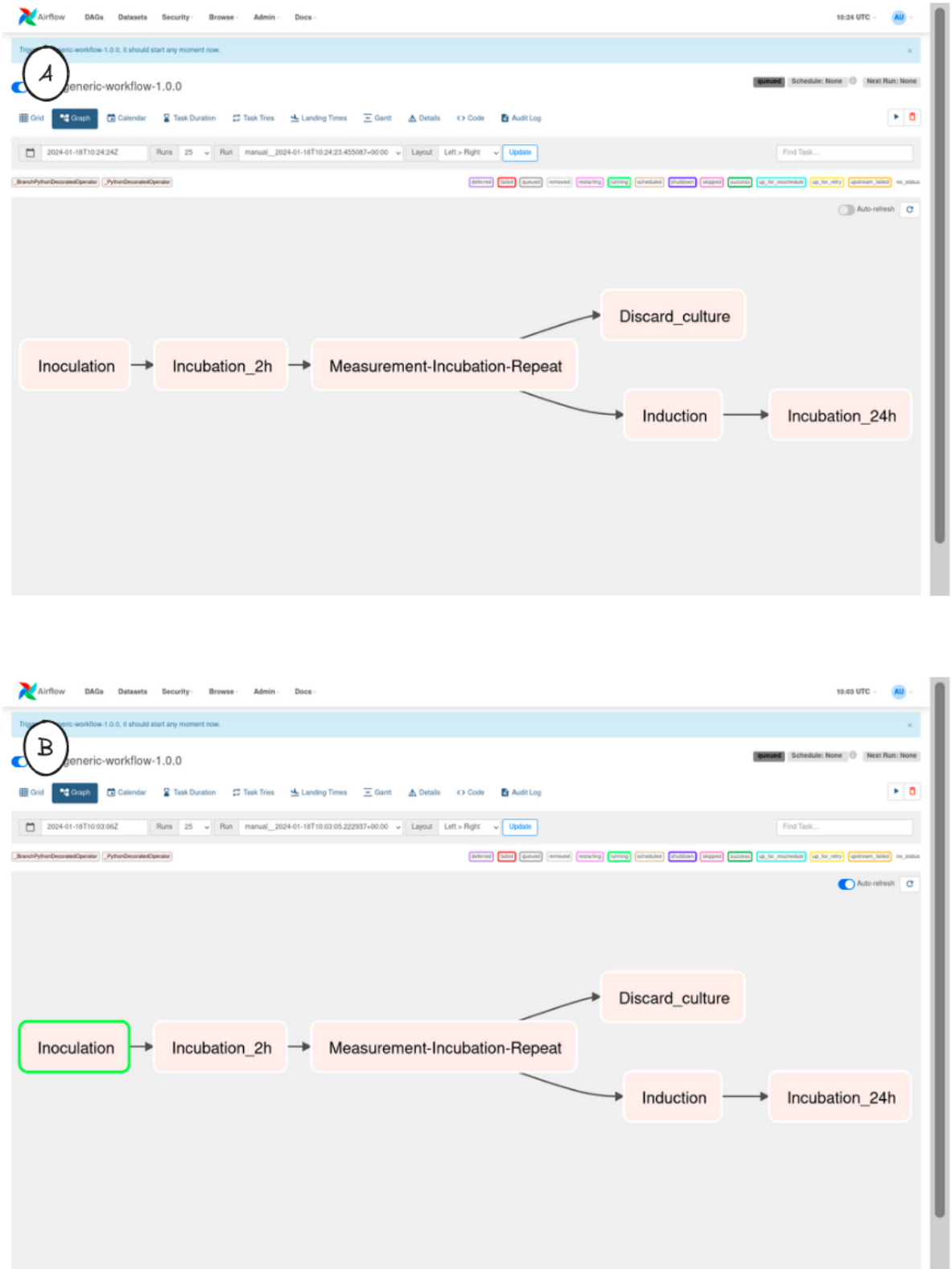


C7 Figure S3 – Execution Graph of repeats (Screenshot of Neo4j Browser user interface). Screenshot of the user interface of Airflow showing the workflow with a repetition on the task called 'Measurement' (implemented with Strategy 2).

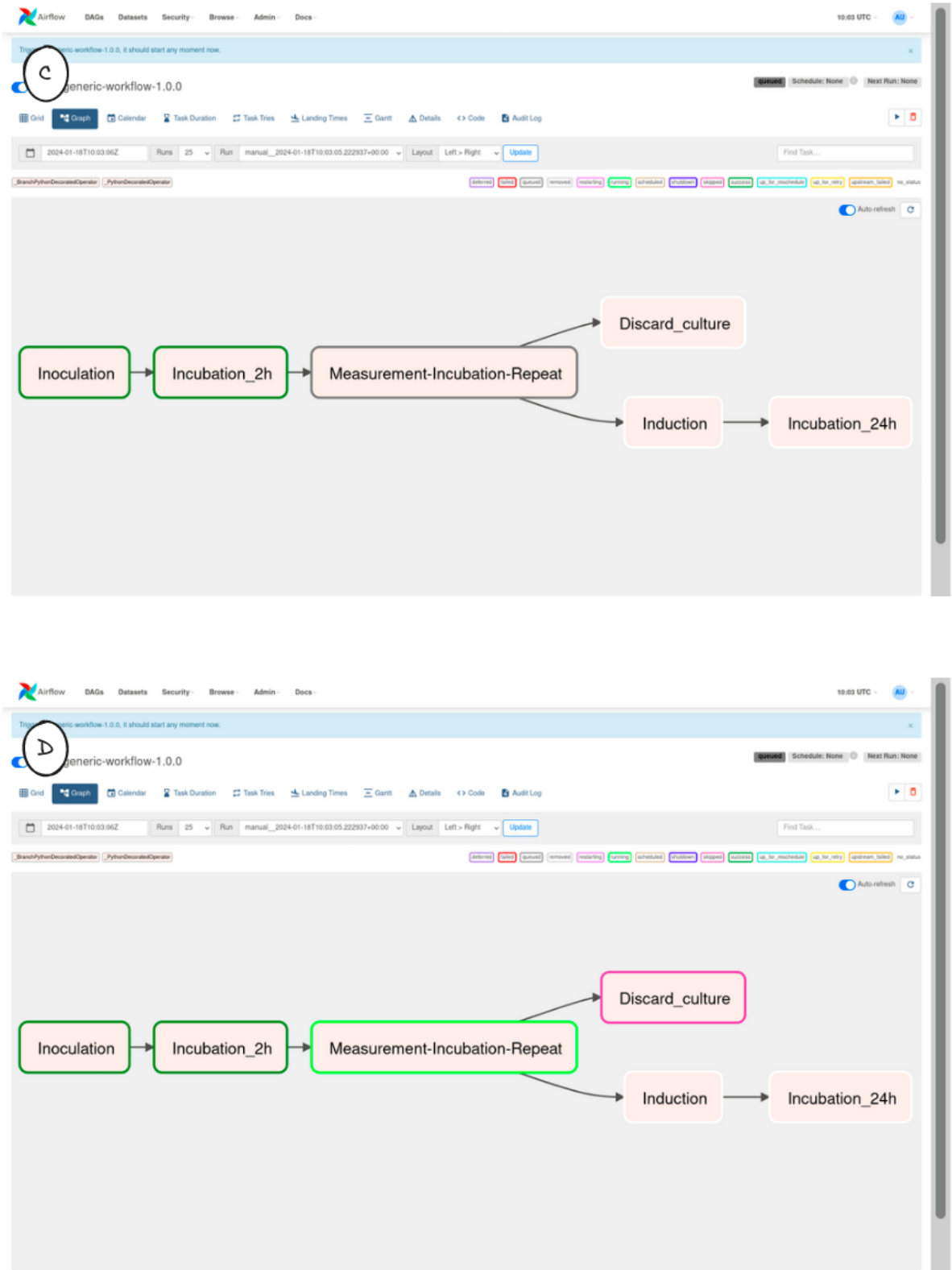




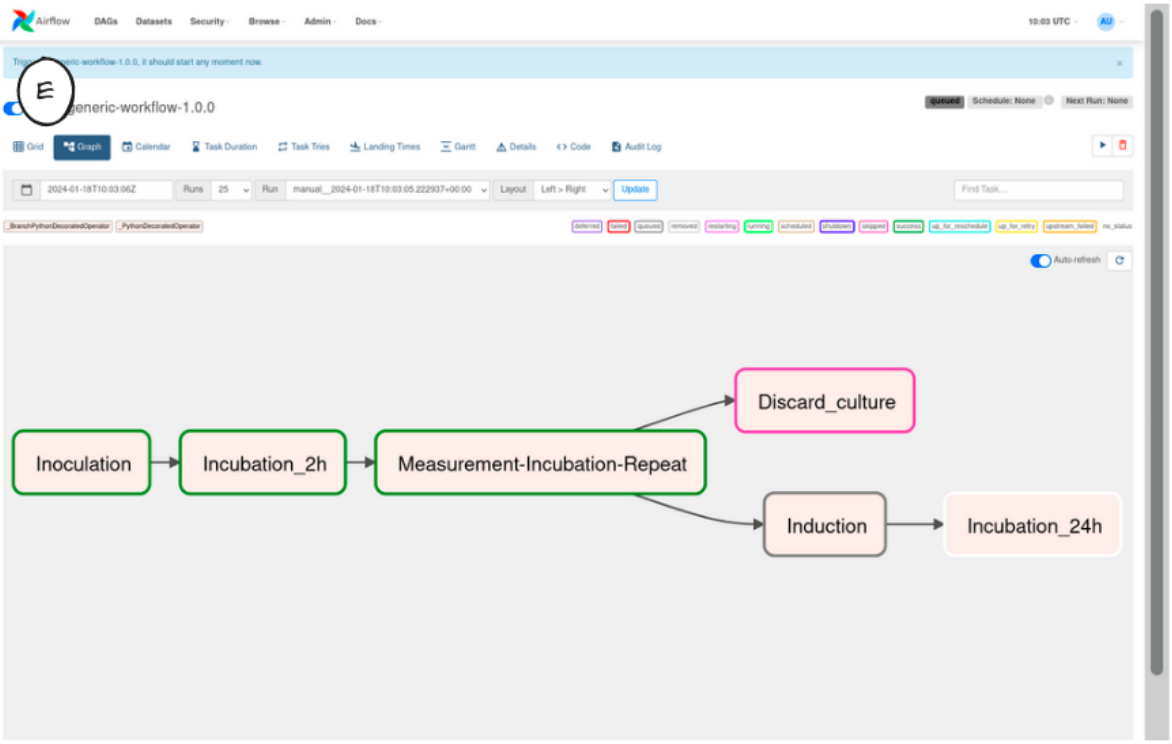
C7 Figure S4 – Figure 7.26 – 'Generic workflow' as represented in Airflow's graphical interface (Screenshot of Neo4j Browser user interface). **Top**: The execution of the workflow repeats a measurement and proceeds to the induction of the culture. **Bottom:** The execution of the workflow repeats a measurement and discards the culture.

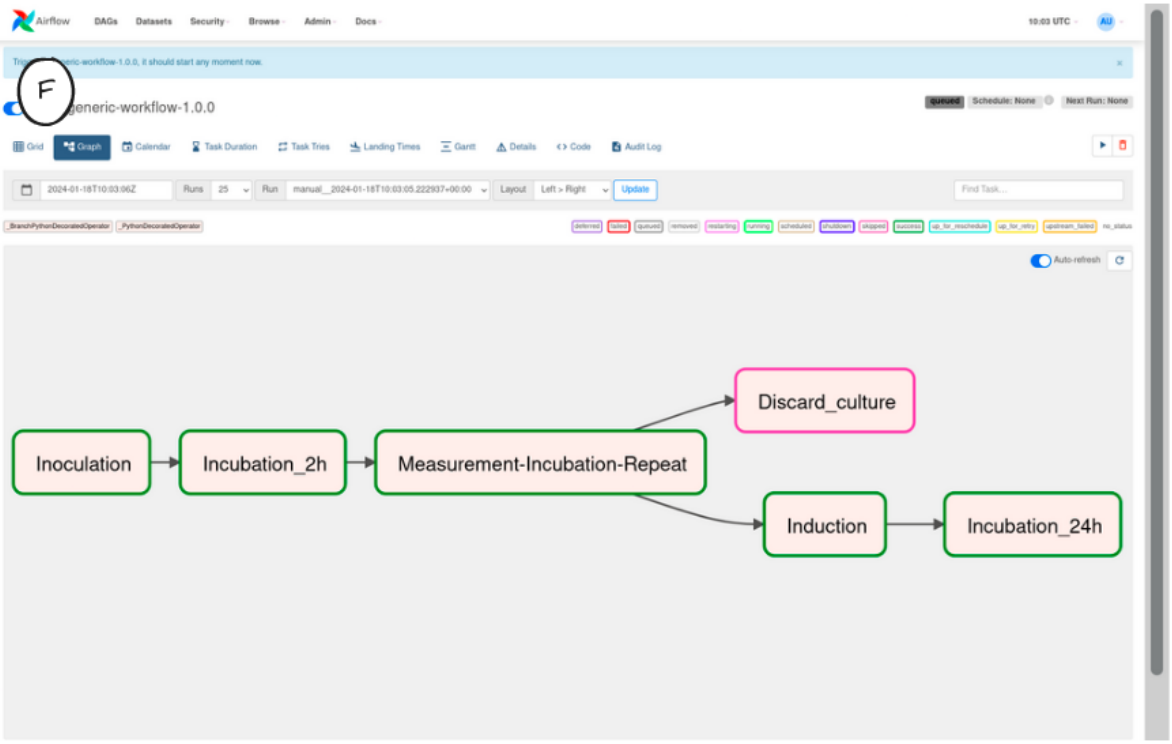


C7-Figure S5a – Execution Run of the 'Generic workflow' (screenshot of the user interface of Airflow). The screenshots shows the progress of the workflow at various states of the execution: **(A)** 'Generic workflow' before execution. (**B)** Current execution of the task 'Incubation' (in light green).

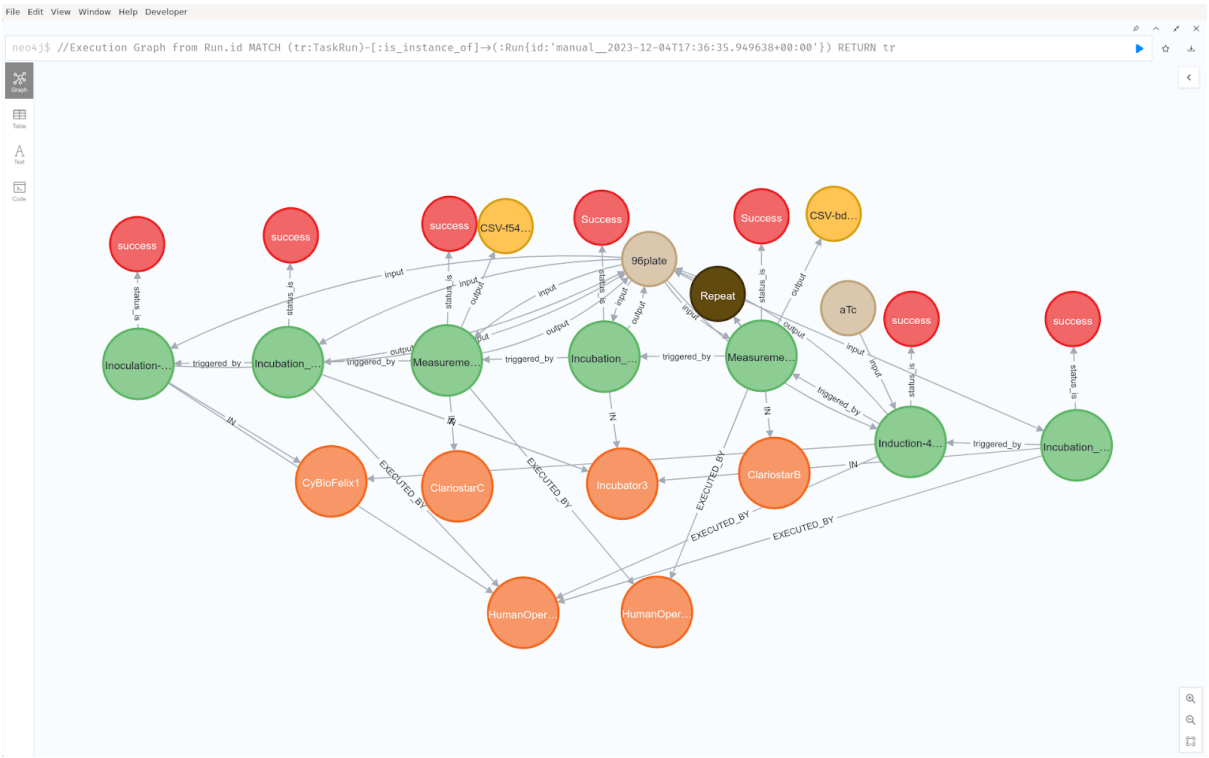


C7-Figure S5b – Execution Run of the 'Generic workflow' (continuation of C7 Figure S5a). (**C)** Tasks 'Induction' and 'Incubation_2h' have been successfully executed. (**D)** Current execution of the task 'Measurement-Incubation-Repeat' (in light green), the task 'Discard_culture' has been skipped (in pink).

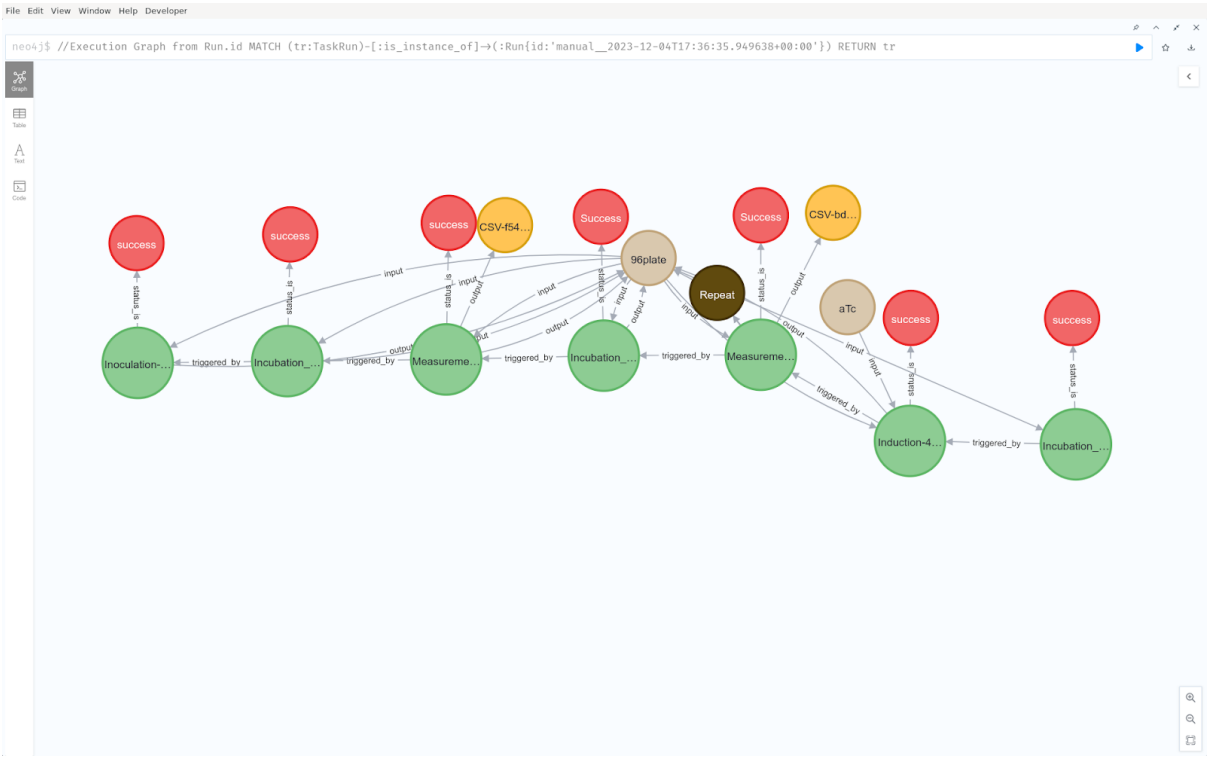




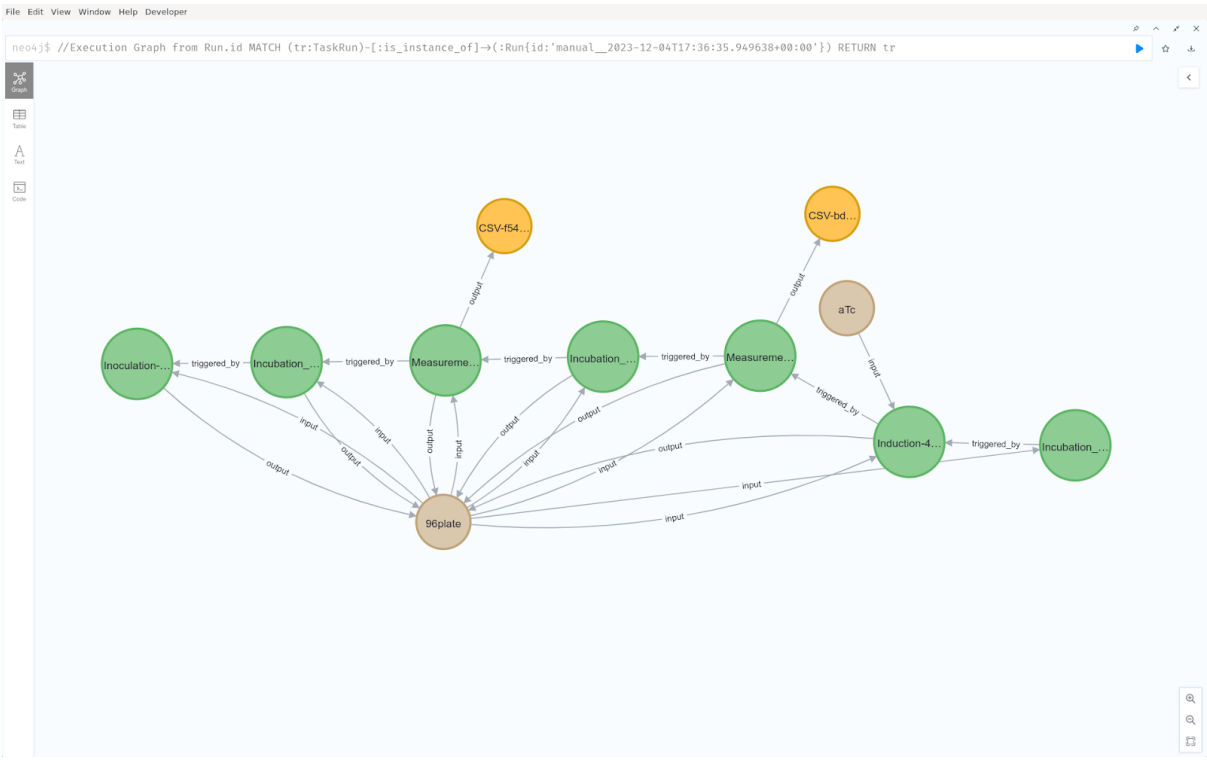
C7-Figure S5c – Execution Run of the 'Generic workflow' (continuation of C7 Figure S5b). (**E)** Task 'Induction' is being queued (in gray). **F)** Tasks 'Induction' and 'Incubation_24h' have been successfully executed (in dark green).



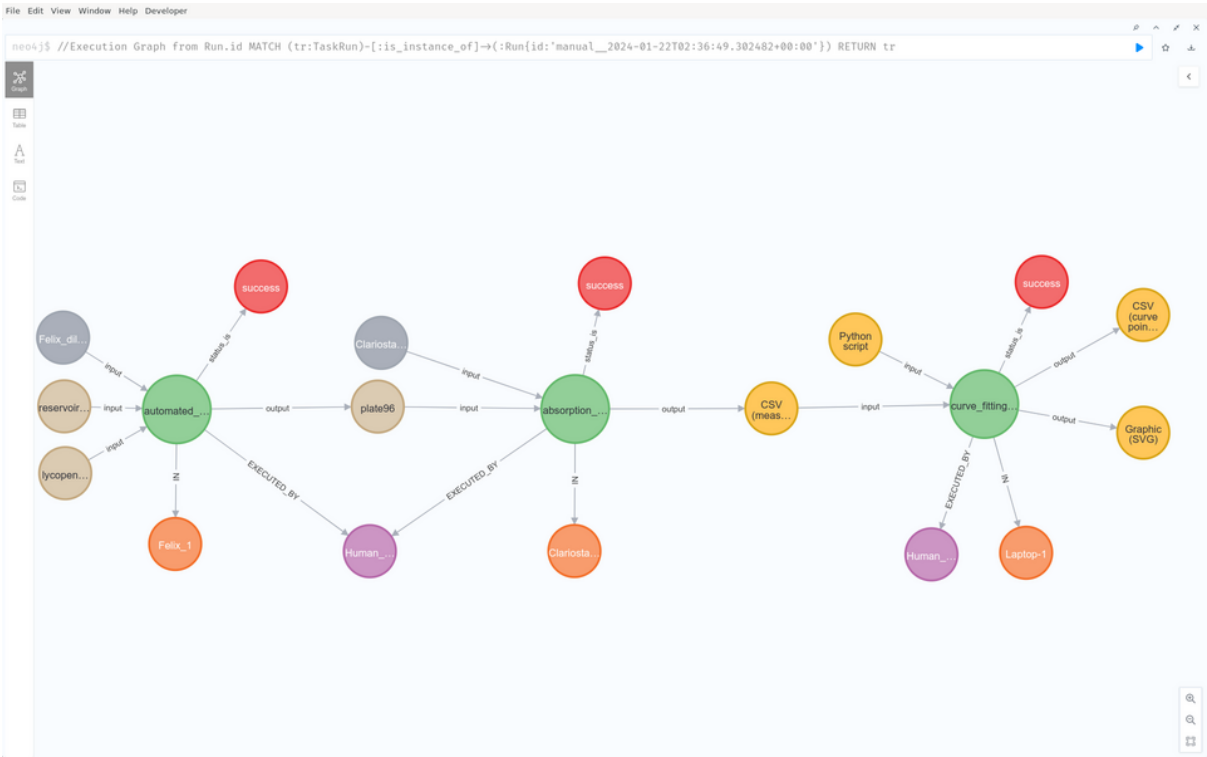
C7-Figure S6 – Execution Graph of the 'Generic workflow'. Screenshot of the Neo4j browser graph view. The graph view displays the executed Tasks (in green), the status of the task (in red), the resources (in orange), if a task is a repeat (in brown), the output files (in yellow) and the containers (labware) (in beige) of the Execution Graph resulting from the Run.



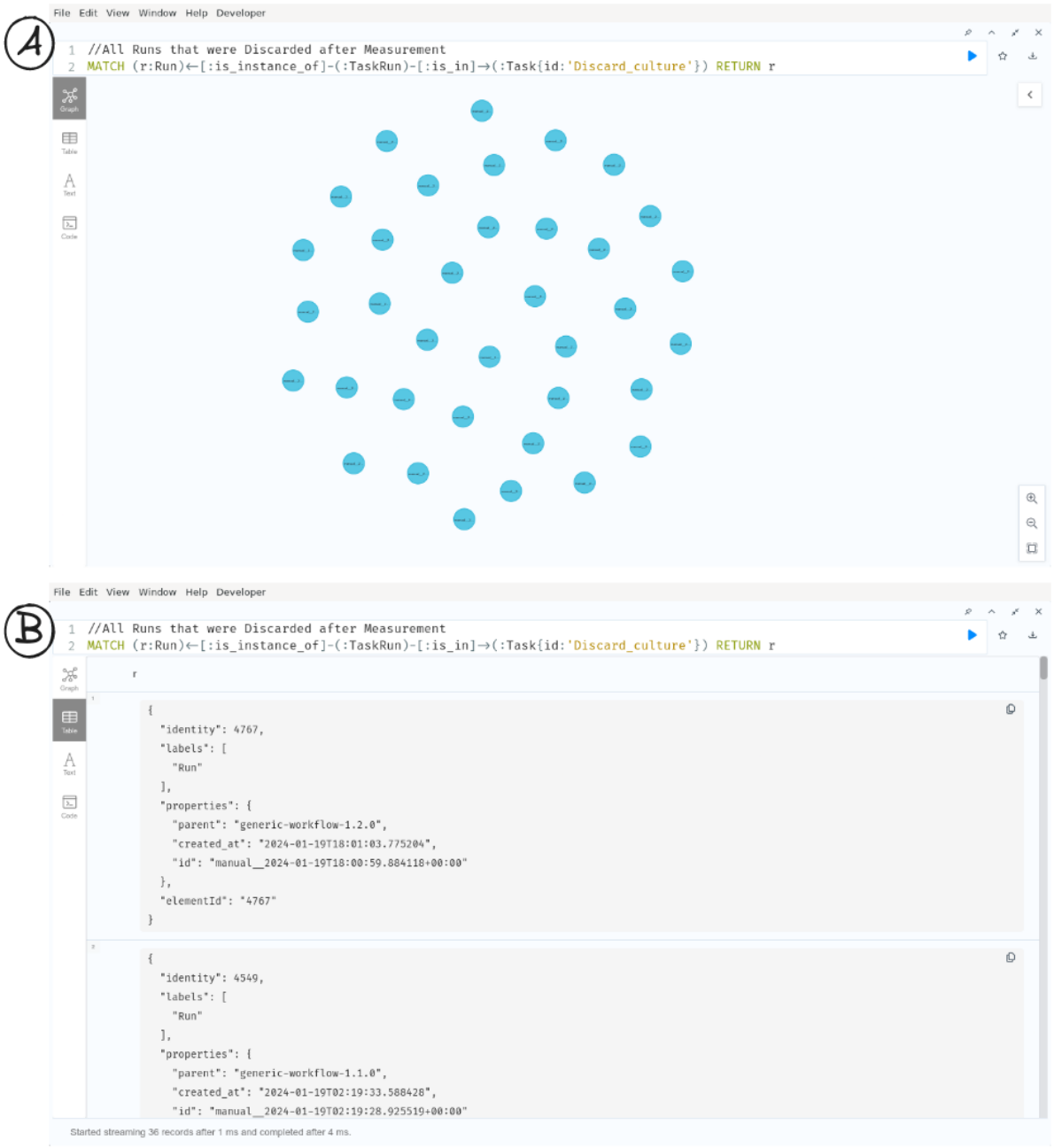
C7-Figure S8 – Execution Graph of the 'Generic workflow'. Screenshot of the Neo4j browser. The view displays the Task (in green) and their status (in red and brown) as well as the I/O objects (in yellow and beige) of the Execution Graph of the Run.



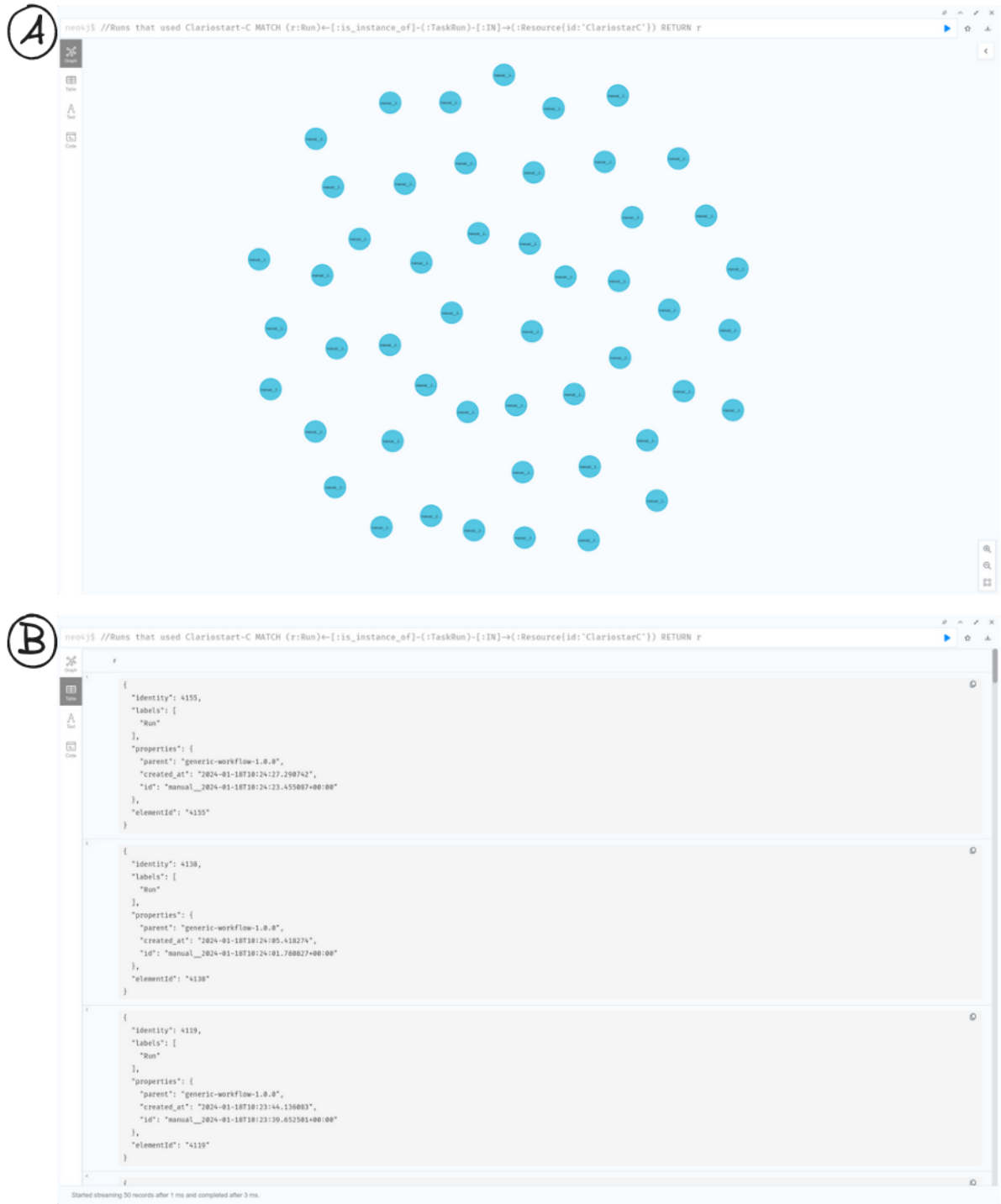
C7-Figure S9 – Execution Graph of the 'Generic workflow'. Screenshot of the Neo4j browser. The view displays only the Task (in green) the I/O objects (in yellow and beige) of the Execution Graph of the Run.



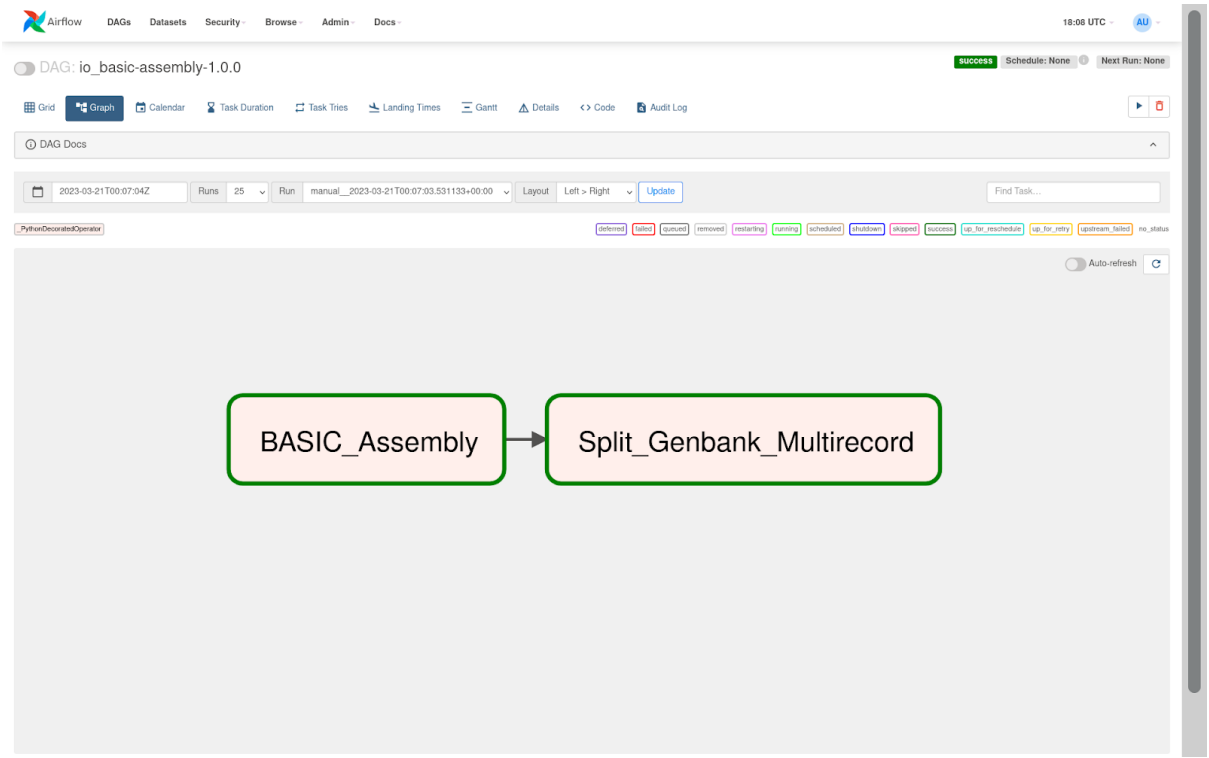
C7 Figure S10 – Visualisation of the Execution Graph of the 'Automated Standard Curve'. Screenshot of the Neo4j browser graph view. The graph view displays the executed Tasks (in green), the status of the task (in red), the input and output objects (files in yellow and labware in beige, protocols in gray, equipment in purple and human resources in orange).



C7-Figure S11 – Result of the Cypher query. Screenshot of the Neo4j browser. (**A)** Run IDs visualised as graph nodes (in blue). (**B)** Run IDs visualised in JSON objects.



C7-Figure S12 – Result of the Cypher query. Screenshot of the Neo4j browser. (**A)** Run IDs visualised as graph nodes (in blue). (**B)** Run IDs visualised in JSON objects.



C7-Figure S13 – 'BASIC DNA Assembly' workflow as represented in Airflow's graphical interface. Screenshot of the user interface of Airflow.