PLOS ONE

RESEARCH ARTICLE

A new method for computing the projection median, its influence curve and techniques for the production of projected quantile plots

Fan Chen[1☯], **Guy Nason**[2☯*]

1 School of Mathematics, University of Bristol, Fry Building, Woodland Road, Bristol, England, United Kingdom, **2** Dept. Mathematics, Imperial College, London, England, United Kingdom

☯ These authors contributed equally to this work.
* g.nason@imperial.ac.uk

Citation: Chen F, Nason G (2020) A new method for computing the projection median, its influence curve and techniques for the production of projected quantile plots. PLoS ONE 15(5): e0229845. https://doi.org/10.1371/journal.pone.0229845

Editor: Chengming Huang, Huazhong University of Science and Technology, CHINA

Received: August 6, 2019

Accepted: February 15, 2020

Published: May 7, 2020

Data Availability Statement: All relevant beetle data is already published and available in the article "Lubischew, A. A. (1962). On the use of discriminant functions in taxonomy. Biometrics, 18:455–477." All other data is available in an R package via CRAN at https://cran.r-project.org/package=Yamm and also available as a Supporting Information file.

Funding: GN:supported by UK Engineering and Physical Sciences Research Council EP/K020951/1 http://www.epsrc.ac.uk FC: supported by a

Abstract

This article introduces a new formulation of, and method of computation for, the projection median. Additionally, we explore its behaviour on a specific bivariate set up, providing the first theoretical result on form of the influence curve for the projection median, accompanied by numerical simulations. Via new simulations we comprehensively compare our performance with an established method for computing the projection median, as well as other existing multivariate medians. We focus on answering questions about accuracy and computational speed, whilst taking into account the underlying dimensionality. Such considerations are vitally important in situations where the data set is large, or where the operations have to be repeated many times and some well-known techniques are extremely computationally expensive. We briefly describe our associated R package that includes our new methods and novel functionality to produce animated multidimensional projection quantile plots, and also exhibit its use on some high-dimensional data examples.

1 Introduction: Overview of multivariate medians

The median is an estimator of location that is robust, i.e. not heavily influenced by outlying values, which are, loosely speaking, points that are far from the main body of the data. Let $\mathbf{x} = (x_1, \ldots, x_k)^T$ be a mutually independent and identically distributed (i.i.d.) sample of length $k \in \mathbb{N}$ from a univariate distribution with distribution function F. The univariate population *median* functional $M(F)$ is

$$M(F) = \inf\{x : F(x) \geq 1/2\} = \sup\{x : F(x) \leq 1/2\}. \quad (1)$$

There are several equivalent definitions of the univariate median that all yield same unique value of true median μ for a distribution F with a bounded and continuous density $f(\mu)$ at μ.

For multivariate data there is no natural ordering of the data to enable the choice of the middle observation in the same way as for one-dimensional data. However, several different multivariate median concepts have been developed that retain some characteristics of the univariate median. For example, an early extension of the multivariate median was suggested by Hayford [1], which is simply the component-wise median, also known as the vector of

University of Bristol/China Scholarship Council batch award. There is no specific grant number https://www.chinesescholarshipcouncil.com http://www.bristol.ac.uk The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

marginal medians. The spatial median, also known as the L_1 median [2, 3], and Tukey's median [4] are two other popular variants. Oja's median [5] provides an alternative to the spatial median, but it is known to be more computationally expensive than other choices. These, and others, are reviewed in [6–8]. We briefly review some of them here next, not least as we use them later in our simulation study.

1.1 Component-wise median

Let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)^T$ be an n-dimensional i.i.d. sample with distribution function $F : \mathbb{R}^n \to \mathbb{R}$. We assume that the n marginal distributions have bounded densities $f_1(\mu_1), \ldots, f_n(\mu_n)$ at the uniquely defined marginal medians $\mu = (\mu_1, \ldots, \mu_n)$. The component-wise median, also known as the marginal sample median, $M_C(\boldsymbol{X}) \in \mathbb{R}^n$ minimises

$$k^{-1}\sum_{i=1}^{k}\{(|x_{i1} - m_1| + \cdots + |x_{in} - m_n|) - (|x_{i1}| + \cdots + |x_{in}|)\}, \tag{2}$$

the sum of component-wise distances over $\boldsymbol{m} \in \mathbb{R}^n$, where $\mathbf{m} = (m_1, \ldots, m_n)$. The corresponding population functional, $M_C(F)$, for the vector of population medians minimises

$$E\{(|x_1 - m_1| + \cdots + |x_n - m_n|) - (|x_1| + \cdots + |x_n|)\}. \tag{3}$$

1.2 Spatial median

The spatial median $M_S(\mathbf{X})$, also known as the L_1 median, minimises

$$k^{-1}\sum_{i=1}^{k}\{||\mathbf{x}_i - \mathbf{m}|| - ||\mathbf{x}_i||\}, \tag{4}$$

over $\mathbf{m} \in \mathbb{R}^n$, where $||\mathbf{m}||^2 = \sum_{i=1}^{n} m_i^2$ is the (squared) Euclidean norm. The corresponding functional spatial median, $M_S(F)$, minimises

$$E_F\{||\mathbf{x} - \mathbf{m}|| - ||\mathbf{x}||\}. \tag{5}$$

1.3 Oja's median

Let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)^T$ be an i.i.d. sample in $\mathbb{R}^n$ with distribution function $F : \mathbb{R}^n \to \mathbb{R}$. The volume of the n-variate simplex determined by the $n + 1$ vertices $(\mathbf{m}_1, \ldots, \mathbf{m}_{n+1})$ is

$$V(\mathbf{m}_1, \ldots, \mathbf{m}_{n+1}) = \frac{1}{p!}\left|det\begin{pmatrix} 1 & \cdots & 1 \\ \mathbf{m}_1 & \cdots & \mathbf{m}_{n+1} \end{pmatrix}\right|. \tag{6}$$

The Oja median, $M_O(\mathbf{X})$, minimises

$$\binom{k}{n}^{-1}\sum_{i_1<\cdots<i_n} V(\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_n}, \mathbf{m}), \tag{7}$$

over $\mathbf{m} \in \mathbb{R}^n$. The corresponding functional $M_O(F)$ minimises

$$E_F\{V(\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_n}, \mathbf{m})\}. \tag{8}$$

1.4 Tukey's median

Let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)^T$ be an i.i.d. sample of size k in $\mathbb{R}^n$ with distribution function $F : \mathbb{R}^n \to \mathbb{R}$. Let $\mathcal{H}$ be the class of all closed half spaces in $\mathbb{R}^n$. For each $H \in \mathcal{H}$, define the empirical distribution

$$\hat{F}(H) = n^{-1} \sum_{i=1}^{k} \mathbb{I}(\mathbf{x}_i \in H), \tag{9}$$

where $\mathbb{I}$ is the usual indicator function. Then, define the *depth*, $D(\boldsymbol{\mu})$, of a point $\boldsymbol{\mu} \in \mathbb{R}^n$ within the dataset, to be the infinum of $\hat{F}(H)$, that is taken over all closed half spaces H for which $\boldsymbol{\mu} \in H$. Tukey's median is defined as the set of points $\boldsymbol{\mu}$ of maximal depth.

2 The projection median

This section introduces our new method for computing the projection median, yamm. We prove that yamm is equivalent to the projection median, as defined by Durocher and Kirkpatrick [9] in $\mathbb{R}^2$ and then generalised to higher dimensions by Basu *et al.* [10]. We also explore, theoretically and numerically, the statistical behaviour of yamm using a mixture of two bivariate normal distributions.

2.1 Review of the projection median

2.1.1 Projection median in $\mathbb{R}^2$. Let $\mathbf{X}$ be a multiset of points in $\mathbb{R}^2$ and $\theta \in [0, 2\pi)$ be an angle. Let $\mathbf{X}_\theta$ denote the multiset defined by the projection of $\mathbf{X}$ onto the unit vector $u_\theta = (\cos\theta, \sin\theta)$, so

$$\mathbf{X}_\theta = \{u_\theta \langle \mathbf{x}, u_\theta \rangle \mid \mathbf{x} \in \mathbf{X}\}, \tag{10}$$

where $\langle \cdot \rangle$ denotes the usual inner product.

The projection median of a non-empty finite set $\mathbf{X}$ with points in $\mathbb{R}^2$ is

$$M_P(\mathbf{X}) = \pi^{-1} \int_0^{2\pi} \text{med}(\mathbf{X}_\theta)\, d\theta, \tag{11}$$

where $\text{med}(\mathbf{X}_\theta) \in \mathbb{R}^2$ is the median of the projection of $\mathbf{X}$ onto the line through the origin, parallel to u_θ.

2.1.2 Generalisation of the projection median. Given a fixed positive integer, $n \geq 2$, and a finite set of points $\mathbf{X}$ in $\mathbb{R}^n$, the n-dimensional projection median of $\mathbf{X}$ is

$$M_P(\mathbf{X}) = n \frac{\int_{\mathbf{X}^{n-1}} \text{med}(\mathbf{X}_{\boldsymbol{a}})\, d\boldsymbol{a}}{\int_{\mathbf{X}^{n-1}} d\mathbf{a}} = n \int_{\mathbf{X}^{n-1}} \text{med}(\mathbf{X}_{\boldsymbol{a}})\, df(\mathbf{a}), \tag{12}$$

where $\mathbf{X}^{n-1} = \{\mathbf{x} \in \mathbb{R}^n : ||\mathbf{x}|| = 1\}$ is the unit n-dimensional hypersphere, $\text{med}(\mathbf{X}_\mathbf{a})$ is the median of the projection of $\mathbf{X}$ onto the line through the origin parallel to $\mathbf{a}$, and f is the normalised uniform measure over $\mathbf{X}^{n-1}$. Hence, for a point $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbf{X}^{n-1}$, the n-

dimensional spherical coordinates are given by

$$\begin{aligned} x_1 &= \cos\theta_1 \\ x_2 &= \sin\theta_1 \cos\theta_2 \\ x_3 &= \sin\theta_1 \sin\theta_2 \cos\phi_3 \\ &\cdots \\ x_{n-1} &= \sin\theta_1 \cdots \sin\theta_{n-2} \cos\theta_{n-1} \\ x_n &= \sin\theta_1 \cdots \sin\theta_{n-2} \sin\theta_{n-1}, \end{aligned} \tag{13}$$

where each angle $\theta_1, \theta_2, \ldots, \theta_{n-2}$ has a range of π and θ_{n-1} has range of 2π. Also, the normalised uniform measure f over $\mathbf{X}^{n-1}$ is given by

$$df = \frac{d_{\mathbf{X}^{n-1}} V}{\int_0^{\pi} \int_0^{\pi} \cdots \int_0^{2\pi} d_{\mathbf{X}^{n-1}} V}, \tag{14}$$

where $\mathrm{d}_{\mathbf{X}^{n-1}} V = \sin^{n-2}\theta_1 \sin^{n-3}\theta_2 \ldots \sin\theta_{n-2}\, d\theta_1 d\theta_2 \ldots d\theta_{n-1}$ is the volume element of the $(n-1)$-sphere.

Basu *et al.* [10] proved that the projection median has a breakdown point of 1/2 for all $n \geq 2$.

2.2 Yet another multivariate median (Yamm)

Let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)^T \in \mathbb{R}^{k \times n}$ be a random sample of size $k \in \mathbb{N}$, $\mathbf{x}_i \in \mathbb{R}^n$. Let $\mathbf{a}$ be a $n \times 1$ projection vector of unit length, 1_k be the $k \times 1$ vector of ones and $\boldsymbol{\mu}$ a shift vector of length n. Let $\mathbf{y}$ be the projection of $\mathbf{X}$ onto $\mathbf{a}$ after $\mathbf{X}$ has been shifted by $\boldsymbol{\mu}$:

$$\mathbf{y} = (\mathbf{X} - 1_k \boldsymbol{\mu}^T)\, \mathbf{a}, \tag{15}$$

where $\mathbf{y} \in \mathbb{R}^k$. The univariate median m of the projected points $\mathbf{y}$ is

$$m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a}) = m(\mathbf{y}). \tag{16}$$

Now define the integral

$$M_{\mathbf{X},m}(\boldsymbol{\mu}) = \int_{\{\mathbf{a}:\mathbf{a}^T\mathbf{a}=1\}} m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})^2 d\mathbf{a}. \tag{17}$$

The yamm estimator of location for $\mathbf{X}$ is

$$\hat{\boldsymbol{\mu}} = \mathrm{yamm}(\mathbf{X}) = \mathrm{argmin}_{\boldsymbol{\mu}} M_{\mathbf{X},m}(\boldsymbol{\mu}). \tag{18}$$

Eqs (17) and (18) illustrate the rationale behind yamm. Intuitively, if the shift vector $\boldsymbol{\mu}$ is far away from the true 'middle' of the dataset, then the magnitude of $m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})$, as well as the integral $M_{\mathbf{X},\, m}(\boldsymbol{\mu})$, will be large. By contrast, a smaller $m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})$ can be obtained when the $\boldsymbol{\mu}$ is moving closer to the true 'middle' of the data set.

Instead of computing the squared value of $m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})$ for the integral, we also considered the absolute value as an alternative. However, this leads to similar numerical results.

Example. We now generate two polar plots of the absolute value of $m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})$, when $\boldsymbol{\mu}$ is both close to, and far away, from the true median, respectively. A random two-dimensional dataset with $k = 100$ points was generated, whose Tukey's median computed as (2.78, 8.16). Here, the Tukey median is to be interpreted as a 'sensible' middle of the data set. The shift vector $\boldsymbol{\mu}$ is set to be (2.2, 8) and (2, 7.5) respectively, and for each plot, two thousand random projections were used to calculate the univariate median $m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})$, using methods to be explained in

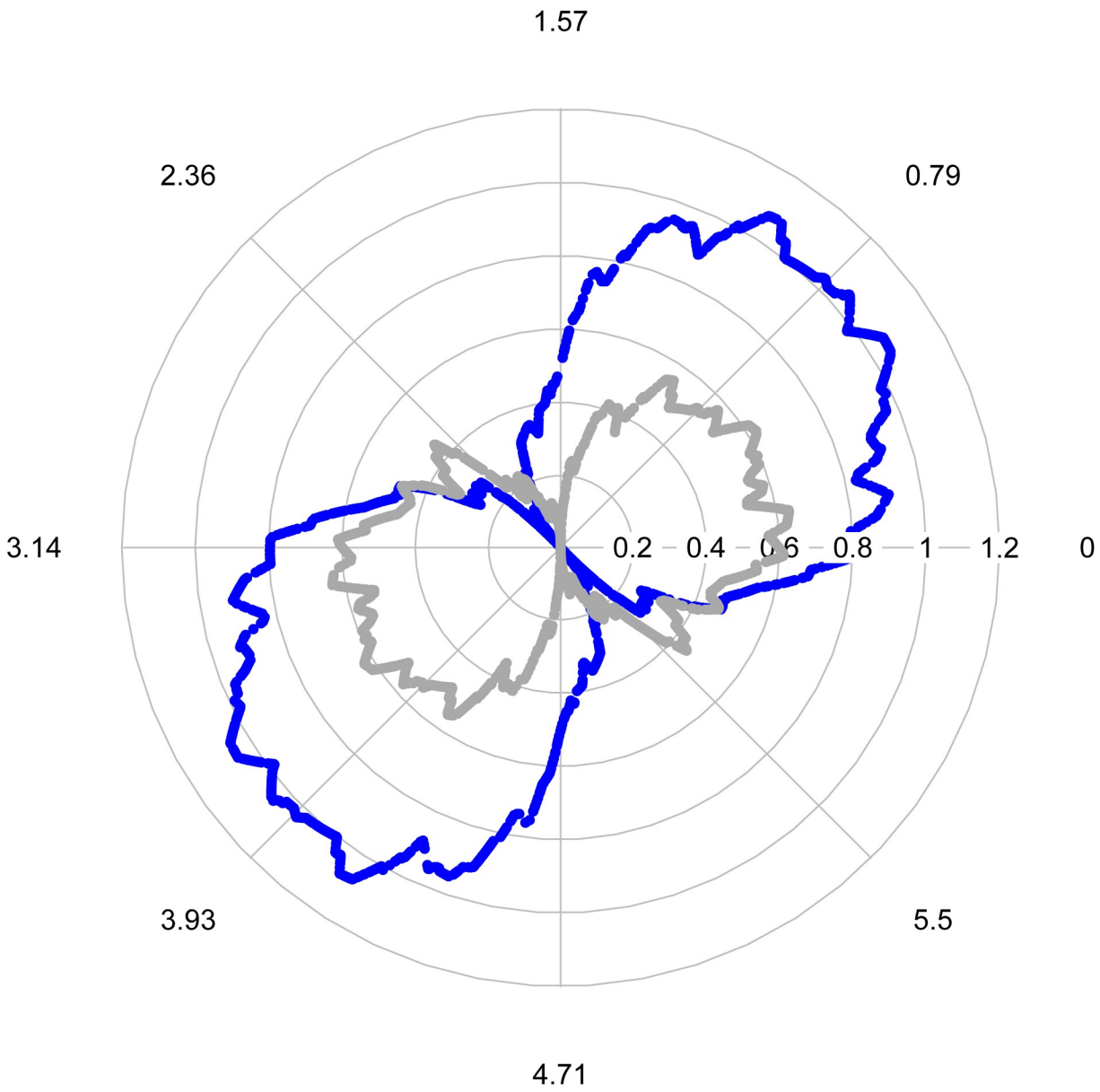

Fig 1. Polar plot (in radians) of the magnitude of $m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})$. *Grey line*: $\boldsymbol{\mu}$ = (2.2, 8) and *Blue line*: $\boldsymbol{\mu}$ = (2, 7.5).

https://doi.org/10.1371/journal.pone.0229845.g001

Section 2.4. Fig 1 shows that when $\boldsymbol{\mu}$ is near the Tukey's median, the magnitude of each $m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})$ is less than 0.65, while a larger value, ranging from 0 to 1.2, is shown in the figure when $\boldsymbol{\mu}$ is far away from the median. Overall, when integrated the quantity involving the $\boldsymbol{\mu}$ is closer to the Tukey median it gives a smaller result.

The projection median and yamm definitions seem similar, as both project the multiset onto the line passing through the origin, and then take the median. However, the projection median integrates $\text{med}(\mathbf{X}_{\mathbf{a}})$ directly over the unit hypersphere in $\mathbb{R}^n$, whereas yamm minimises the objective function $M_{\mathbf{X},m}(\boldsymbol{\mu}) \in \mathbb{R}$ over the shift vector $\boldsymbol{\mu}$. Despite these differences, the following theorem shows that the projection median and yamm are identical.

Theorem. *For any finite multiset* $\mathbf{X} \subseteq \mathbb{R}^n$ *with* $n \geq 2$, *yamm is equivalent to the projection median.*

For the proof of the theorem, see S1 Appendix.

2.3 Yamm behaviour on a bivariate normal mixture

To gain insight about the theoretical behaviour of yamm we study the case of yamm applied to a mixture of two bivariate normals, where one is thought of as the bulk and the other as the outlier of the distribution. Such a setup enables us to evaluate the robustness of yamm. We numerically and theoretically assess the influence curve when moving the outlier far from the bulk.

2.3.1 Bivariate mixture setup. Let $\mathbf{X}_1 \sim \mathcal{N}_2(v_1, \Sigma_1)$ and $\mathbf{X}_2 \sim \mathcal{N}_2(v_2, \Sigma_2)$ be independent bivariate normal random variables, where $\mathbf{X}_1 = (X_{11}, X_{12})^T$, $\mathbf{X}_2 = (X_{21}, X_{22})^T$ with mean vector $\boldsymbol{v}_1 = (v_{11}, v_{12})^T$ and $\boldsymbol{v}_2 = (v_{21}, v_{22})^T$. Let $\mathbf{R}(\theta)$ be a rotation matrix with angle θ given by

$$\boldsymbol{R}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}. \tag{19}$$

We are interested in the first row of this matrix, which describes the projection onto direction θ. Let $\mathbf{Y}_i = (Y_{i1}, Y_{i2})^T = \mathbf{R}(\mathbf{X}_i - \boldsymbol{\mu})$ for $i = 1, 2$ respectively, where $\boldsymbol{\mu} = (\mu_1, \mu_2)^T$ is a shift vector mentioned in (15). Basic multivariate theory shows that

$$\mathbf{Y}_i \sim \mathcal{N}_2\{\mathbf{R}(\boldsymbol{v}_i - \boldsymbol{\mu}),\ \mathbf{R}\Sigma_i\mathbf{R}^T\}, \qquad \text{for } i = 1, 2. \tag{20}$$

Denote $\mathbf{Y}_i = (Y_{i1}, Y_{i2})^T$, Y_{i1} is the first entry of $\mathbf{Y}_i$ for $i = 1, 2$. Then, it is immediate that $Y_{i1} \sim \mathcal{N}(s_i,\ \sigma_i^2)$, where

$$s_1 = (v_{11} - \mu_1)\cos\theta - (v_{12} - \mu_2)\sin\theta \ \text{ and } \ \sigma_1^2 = (\mathbf{R}\Sigma_1\mathbf{R}^T)_{1,1}, \tag{21}$$

$$s_2 = (v_{21} - \mu_1)\cos\theta - (v_{22} - \mu_2)\sin\theta \ \text{ and } \ \sigma_2^2 = (\mathbf{R}\Sigma_2\mathbf{R}^T)_{1,1}. \tag{22}$$

The mixture distribution that we study is

$$f_W(w_1, w_2) = (1 - \epsilon)f_{\mathbf{X}_1}(w_1, w_2) + \epsilon f_{\mathbf{X}_2}(w_1, w_2), \tag{23}$$

where $f_{\mathbf{X}_i}$ is the density of X_i, and $\epsilon \in [0, 1]$, is typically small. Here, $f_{\mathbf{X}_1}$ is considered to be the bulk of the distribution and $f_{\mathbf{X}_2}$ the outlier.

2.3.2 Projected distribution. Based on the bivariate setup above, the projected distribution is

$$f_Y(y) = (1 - \epsilon)\phi_{s_1,\sigma_1^2}(y) + \epsilon\phi_{s_2,\sigma_2^2}(y), \tag{24}$$

where $s_1, s_2, \sigma_1^2, \sigma_2^2$ are as above and ϕ is the standard normal density.

The distribution function of the projected $Y(\theta)$ is

$$F_Y(y) = (1 - \epsilon)\Phi_{s_1,\sigma_1^2}(y) + \epsilon\Phi_{s_2,\sigma_2^2}(y), \tag{25}$$

where Φ is the standard normal distribution function. We require the median of the projected distribution, i.e. find

$$y_m(\epsilon, \theta, s_1, s_2, \Sigma_1, \Sigma_2) \ \text{ such that } \ F_Y(y_m) = 1/2. \tag{26}$$

Finding an analytic exact solution for y_m is difficult. Hence, we will simplify the problem and assume that $\Sigma_1 = \Sigma_2 = I_2$, the identity matrix. Since $\mathbf{R}(\theta)$ is an orthogonal matrix, this means that $\sigma_1^2 = \sigma_2^2 = 1$ and Eq (25) becomes

$$F_Y(y) = (1-\epsilon)\Phi(y - s_1) + \epsilon\Phi(y - s_2). \tag{27}$$

For small ϵ, we know that the median should be close to the median of the bulk, so the median of F_Y should be close to s_1, the median of the first component of the mixture in Eq (27).

2.3.3 Theoretical approximation of yamm on the mixture. We derive a theoretically based approximation to the empirical influence function. We proceed by using a Taylor series expansion of $F_Y(y)$ around s_1, the quantity we know is close to our median:

$$\begin{aligned} F_Y(y) \quad &\approx [1 + \epsilon - \epsilon\ \mathrm{Erfc}\,\{(s_1 - s_2)/\sqrt{2}\}]/2 \\ &+ (2\pi)^{-1/2}[1 - \epsilon + \epsilon \exp\{-(s_1 - s_2)^2/2\}](y - s_1) \\ &+ O\{(y - s_1)^2\}, \end{aligned} \tag{28}$$

where $\mathrm{Erfc}\,(y) = 2\pi^{-1/2}\int_y^\infty e^{-t^2}\,dt$. When y is close to s_1, Eq (28) is approximately equal to 1/2 when ϵ is small, which is the behaviour we expect.

To find an approximation to the median we solve $F_Y\{y_m(\theta)\} = 1/2$. Ignoring remainders, subtracting 1/2 off both sides of Eq (28) gives

$$\frac{\epsilon}{2}\left[\mathrm{Erfc}\left\{(s_1 - s_2)/\sqrt{2}\right\} - 1\right] = \frac{[1 - \epsilon + \epsilon \exp\{-(s_1 - s_2)^2/2\}](y_m - s_1)}{\sqrt{2\pi}}, \tag{29}$$

and then

$$y_m(\theta) \approx s_1 + \frac{\epsilon\sqrt{\pi/2}[\mathrm{Erfc}\,\{(s_1 - s_2)/\sqrt{2}\} - 1]}{[1 - \epsilon + \epsilon \exp\{-(s_1 - s_2)^2/2\}]}. \tag{30}$$

Now using

$$\mathrm{Erfc}\,\{(s_1 - s_2)/\sqrt{2}\} = 2\Phi\{(s_2 - s_1)/\sqrt{2}\}, \tag{31}$$

and $\exp\{-(s_1 - s_2)^2/2\} = \sqrt{2\pi}\phi(s_1 - s_2)$, we can write

$$y_m(\theta) \approx s_1 + \frac{\epsilon\sqrt{\pi/2}(2\Phi\{(s_2 - s_1)/\sqrt{2}\} - 1)}{1 - \epsilon - \sqrt{2\pi}\epsilon\phi(s_2 - s_1)}. \tag{32}$$

For small ϵ the denominator is close to 1. From Eqs (21) and (22), we can write:

$$s_2 - s_1 = (v_{21} - v_{11})\cos\theta - (v_{22} - v_{12})\sin\theta = \delta_1\cos\theta - \delta_2\sin\theta, \tag{33}$$

where $\delta_1 = v_{21} - v_{11}$ and $\delta_2 = v_{22} - v_{12}$. Thus

$$\begin{aligned} y_m(\theta) \quad &\approx \{(v_{11} - \mu_1)\cos\theta - (v_{12} - \mu_2)\sin\theta\} \\ &+ \frac{\epsilon\sqrt{\pi/2}[2\Phi\{(\delta_1\cos\theta - \delta_2\sin\theta)/\sqrt{2}\} - 1]}{1 - \epsilon - \sqrt{2\pi}\epsilon\phi(\delta_1\cos\theta - \delta_2\sin\theta)}. \end{aligned} \tag{34}$$

According to Eq (17), our job is to find the optimal $\boldsymbol{\mu}^* = (\mu_1^*, \mu_2^*)^T$, which minimises

$$M = \int_0^{2\pi} y_m^2(\theta)\, d\theta. \tag{35}$$

The integrand involves the standard normal distribution function, which is tricky to handle analytically. Hence, we use the approximation, $\phi(z) \approx (1 + \cos z)/2\pi$, for $-\pi < z < \pi$, for the standard normal density [11], which enables the following proposition.

Proposition. *Let* $\boldsymbol{X}_1 = (X_{11}, X_{12})^T$ *and* $\boldsymbol{X}_2 = (X_{21}, X_{22})^T$. *Suppose that* $\mathbf{X}_1 \sim \mathcal{N}_2(\boldsymbol{\nu}_1, \Sigma_1)$ *and* $\mathbf{X}_2 \sim \mathcal{N}_2(\boldsymbol{\nu}_2, \Sigma_2)$ *independently, where* $\boldsymbol{\nu}_1 = (\nu_{11}, \nu_{12})^T$ *and* $\boldsymbol{\nu}_2 = (\nu_{21}, \nu_{22})^T$, *respectively. Let the mixture, W, of* $\boldsymbol{X}_1$ *and* $\boldsymbol{X}_2$ *be*

$$f_W(w_1, w_2) = (1 - \epsilon) f_{\mathbf{X}_1}(w_1, w_2) + \epsilon f_{\mathbf{X}_2}(w_1, w_2),$$

where $\epsilon \in [0, 1]$ *is considered small.*

An approximation of the yamm estimator, $\boldsymbol{\mu}^* = (\mu_1^*, \mu_2^*)$, *is*

$$\begin{aligned} \mu_1^* &= \nu_{11} + \pi^{-1/2} R\epsilon (1 - R^2/32 + R^4/1536) \cos\alpha, \\ \mu_2^* &= \nu_{12} + \pi^{-1/2} R\epsilon (1 - R^2/32 + R^4/1536) \sin\alpha, \end{aligned} \tag{36}$$

where $R^2 = (\delta_1^2 + \delta_2^2)$, $\delta_1 = \nu_{21} - \nu_{11}$, $\delta_2 = \nu_{22} - \nu_{12}$ *and* $\alpha = \arctan(\delta_2/\delta_1)$. *The approximation we use is valid whenever* $|R\cos(\theta + \alpha)| < \sqrt{2}\pi$, *where* θ *is the projection direction when computing yamm. This inequality is true for all* θ *whenever* $R < \sqrt{2}\pi$.

Intuitively, the approximation in the Proposition works whenever the two cluster means are close enough together, i.e. when $R^2 = \delta_1^2 + \delta_2^2 < 2\pi^2$.

In particular, when $\nu_{11} = \nu_{21}$ or $\nu_{12} = \nu_{22}$ (i.e. when one of the $\delta_i = 0$, $i = 1, 2$), we can form a more accurate approximation. This is because the approximation for the standard normal distribution function, $\phi(z) \approx (1 + \cos z)/2\pi$, is no longer required to find the optimal $\boldsymbol{\mu}^* = (\mu_1^*, \mu_2^*)^T$ minimising Eq (35). Without loss of generality, let $\nu_1 = (\nu_{11}, \nu_{12})^T = (0, 0)^T$ and $\nu_2 = (\nu_{21}, \nu_{22})^T = (0, d)^T$, we obtain the yamm estimator as follows

$$\begin{aligned} \mu_1^* &= 0, \\ \mu_2^* &= 2^{-1/2} \epsilon d\, e^{-\frac{d^2}{8}} (\text{BesselI}[0, d^2/8] + \text{BesselI}[1, d^2/8]), \end{aligned} \tag{37}$$

where BesselI$[n, z]$ is the modified Bessel function of the first kind, sometimes denoted $I_n(z)$. For the proof of the proposition, see S2 Appendix.

2.3.4 The yamm influence curve on the mixture. This section numerically computes and plots yamm for the case where $\epsilon = 0.05$, $\mathbf{X}_1 \sim \mathcal{N}_2(\boldsymbol{\nu}_1, I_2)$ and $\mathbf{X}_2 \sim \mathcal{N}_2(\boldsymbol{\nu}_2, I_2)$, with $\boldsymbol{\nu}_1 = (0, 0)^T$ and $\boldsymbol{\nu}_2 = (0, d)^T$ for $d \in \mathbb{R}$. We explore how yamm varies as d increases from 0 to 10 in steps of 0.2. If yamm is robust, then it should increase with d, but plateau beyond a certain point.

For each value d we estimate yamm as the mean over five hundred bivariate mixture realizations, with two thousand projections involved for each yamm computation, using methods described below in Section 2.4. The numerically computed crosses in Fig 2 show that, for this setup, yamm plateaus somewhere between $d = 2$ and $d = 4$.

The solid red line in Fig 2 shows our theoretical approximation of the yamm influence curve with the more specific setup, where $\boldsymbol{\mu}^*$ follows Eq (37). Under this approximation, the influence curve closely follows the numerically computed crosses. On the other hand, the solid blue line is the approximation of the yamm under the more general setting of Eq (36), which exhibits poor approximation after $d > 4.5$, although it performs reasonably well when the inter-cluster mean distance $0 < d < 4.5$, and does not plateau.

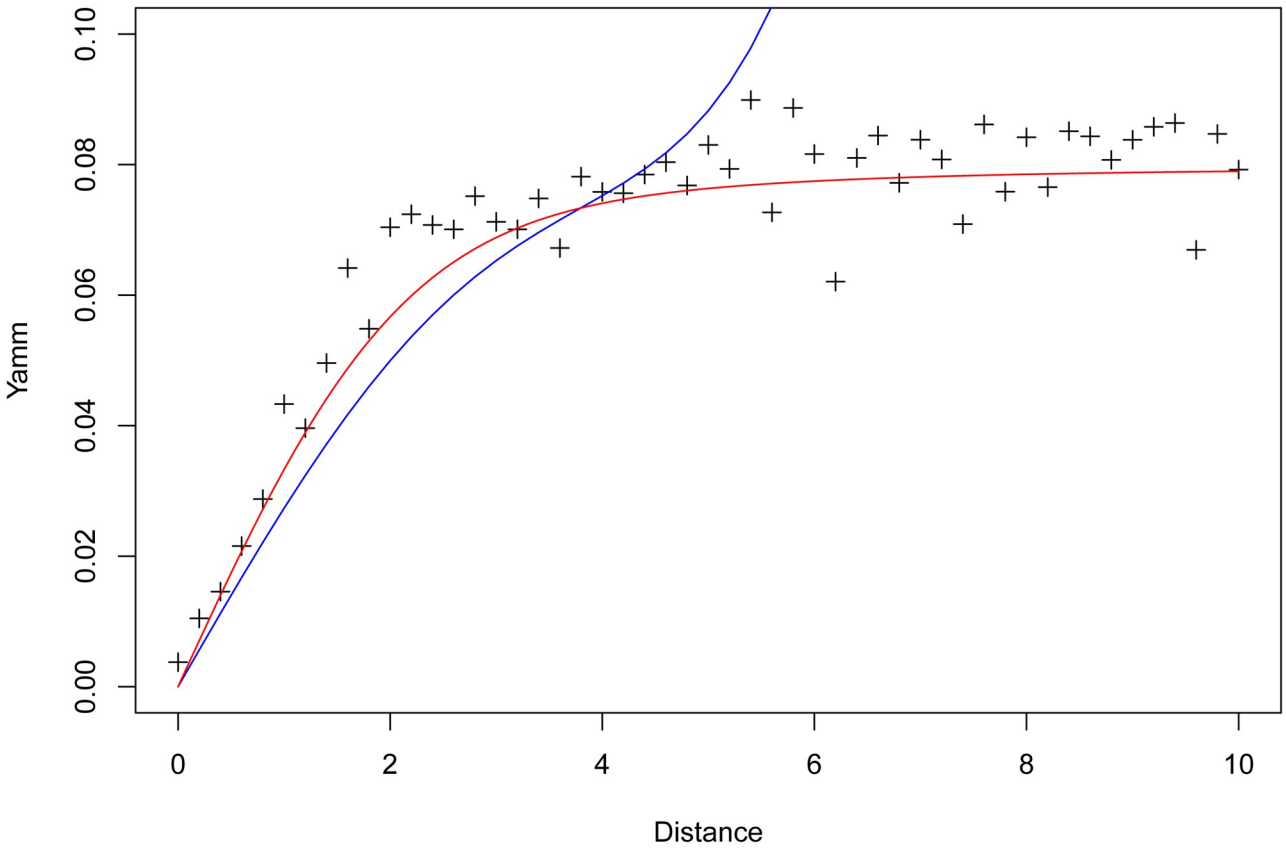

Fig 2. Yamm computed on simulated setup, increasing the distance between two bivariate normals. *Crosses*: numerically computed values; *Solid blue line*: approximation computed for general $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$; *Solid red line*: approximation computed when $\boldsymbol{v}_1 = (0, 0)^T$ and $\boldsymbol{v}_2 = (0, d)^T$.

https://doi.org/10.1371/journal.pone.0229845.g002

This is because, in the setup, $\delta_1 = d$, $\delta_2 = 0$, and $d > 4.5$ implies $R^2 = \delta_1^2 = d^2 > 2\pi^2$. However, the specific setup approximation of yamm obviously does not work for arbitrary values of $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, whereas the general approximation gives a good theoretical idea of the yamm influence curve when the two means of the clusters are close enough together.

2.4 Projection median and yamm computation

2.4.1 Projection median computation. A simple Monte Carlo integration [12] can be used to compute an approximation of the projection median by

$$\hat{M}_P(\mathbf{X}) = nJ^{-1}\sum_{j=1}^{J}\text{med}(\mathbf{X}_{\mathbf{a}_j}), \tag{38}$$

where J represents the number of projections used, and $\{\mathbf{a}_j\}_{j=1}^{J}$ is a set of random, independently-drawn, unit length n-vectors over $\mathbf{X}^{n-1}$.

Calculating approximation of Eq (38) is relatively straightforward, but a large value of J is required to ensure accuracy. Another approach computes the projection median directly from the definition in Eq (12), using the spherical coordinates illustrated in Eq (13), where the integral can be obtained by the trapezoidal rule. For example, in the two-dimensional case, we apply the trapezoidal rule once on Eq (11). In the three-dimensional case, we have to apply the

trapezoidal rule twice for the double integral, and so on. This direct approach is easy to implement when our dataset has a low dimension, but excessive work is required in not that many higher dimensions, even with, e.g. $n = 10$.

2.4.2 Computing yamm. To compute an approximation to yamm, we can also use Monte Carlo integration together with an optimiser. Let $J \in \mathbb{N}$ be the number of projections, $\{\mathbf{a}_j\}_{j=1}^{J}$ be a set of independent random unit length n-vectors, an estimator for $M_{\mathbf{X},\,m}(\boldsymbol{\mu})$ is given by

$$\hat{M}_{\mathbf{X},m}(\boldsymbol{\mu}) = J^{-1}\sum_{j=1}^{J} m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a}_j)^2. \tag{39}$$

We then numerically minimise $\hat{M}_{\mathbf{X},m}(\boldsymbol{\mu})$ over $\boldsymbol{\mu}$ to obtain our estimated location measure, using the BFGS optimization method [13–16]. BFGS is a quasi-Newton algorithm searching for a stationary point of a function via local quadratic approximation. Parallel versions such as `optimParallel` exist as easy to use packages in `R`.

With reasonable starting values, such as the mean or other multivariate medians, yamm typically provides accurate results with a considerably smaller number of projections than used by the Monte Carlo projection median method mentioned above.

In conclusion, projection median computation via the trapezoidal rule is fast and accurate in low dimensions, but increasingly onerous in higher dimensions, as progressively more multidimensional integration is required. For higher dimensions, we prefer the Monte Carlo method and prefer yamm over the projection median as it does not require such a large number of projections, particularly if the optimiser is given a good starting solution.

Overall, approximating the projection median by the trapezoidal rule is a good choice in $\mathbb{R}^2$ and $\mathbb{R}^3$, and either of the other two methods can be used in higher dimensions.

3 Empirical performance for different medians

This section reviews the theoretical computational complexity for a variety of medians and computes some running times for real implementations of several medians computed in R. We then present some results for accuracy of estimation for these medians.

3.1 Computational complexity and empirical speed

For a dataset in $\mathbb{R}^n$ with k observations, the computational complexity for the Spatial median is $O(nk)$ [17], which is the same for the exact computation of the component-wise median. The projection median can be obtained in $O(k^{4/3}\log^{1+\epsilon} k)$ time in $\mathbb{R}^2$ [9], and $O(k^{5/2+\epsilon})$ time in $\mathbb{R}^3$ [10]. In $\mathbb{R}^n$, with $n > 3$, Basu *et al.* showed that $O[k^{n\{1-\delta_n/(n+1)\}+\epsilon}]$ time is required to compute the projection median, where $\delta_n = (4n - 3)^{-n}$ and ϵ is a fixed small constant. Several algorithms for other multivariate medians have been developed or the bivariate case. The current best algorithms for Oja's and Liu's medians require $O(k \log^3 k)$ and $O(k^4)$ time, respectively [18], whereas that for the fastest bivariate Tukey median is $O(k \log^3 k)$ [19]. The calculation of these three multivariate medians in higher dimensions is more complicated and approximate computation is often preferred/required.

To provide empirical assessment of the real computation speed, we apply several R software medians to simulated data. There are several R functions using different algorithms to compute one median. For example, `spatial.median` from the library `ICSNP` estimates the median with the algorithm developed by Vardi and Zhang [20], while `Gmedian` developed by Cardot *et al.* [21] is faster but, perhaps, less accurate. In addition, `l1median` [22] from library `pcaPP` and `med` from `depth` also provide opportunities to compute the spatial median.

Table 1. R functions used for analysing different multivariate medians.

Median	Function	Package	Source
Spatial	`l1median`	`pcaPP`	[22]
CWmed	`med`	`depth`	—
Liu's	`med`	`depth`	[23]
Tukey's	`med`	`depth`	[24] [25]
Oja's	`ojaMedianEvo`	`OjaNP`	[26]
Projection	`PmedTrapz`	`Yamm`	Ours

https://doi.org/10.1371/journal.pone.0229845.t001

Hence, after some experiments, we choose the best function (evaluated in terms of speed and accuracy) for each multivariate median in $\mathbb{R}^2$ and $\mathbb{R}^3$ shown in Table 1. Much of the software for multivariate medians in R only works in low numbers of dimensions.

The `med` function can only calculate the bivariate Liu's median, which is considerably more challenging in higher dimensions. The calculation of Tukey's median is exact in one and two dimensions, and approximate in higher dimensions. We use the approximate Tukey's median computation in the `med` function, due to numerical errors that sometimes surface when using the exact algorithm. For Oja's median, the approximate method (evolutionary algorithm) is used instead of the exact one, as it is faster and can deal with high dimensions.

Table 2 displays mean computation times and their standard deviations across 1000 simulated datasets from the two-dimensional Laplace distribution with different numbers of observations (k) for each set. The results are produced by running R on a single core of an Intel i7-8750h processor with 2.20 GHz base clock using 16Gb RAM. For small k, Liu's median is fastest, but its speed is not as fast as others for higher k. In this experiment, Oja's median is the slowest for small k values, but its speed does not appear to be particularly sensitive to k. Hence, its speed is faster than Tukey's median when $k = 200$. The projection median is one of the quickest when k is below 100, while for large k values, the component-wise median and the Spatial median are faster.

The results in Table 2 are produced by only one possible R function for one median. However, other functions can be used. For example, the `med` function from the `depth` package can also be used to calculate the spatial median and provides accurate answers. It is extremely

Table 2. Mean and standard deviation (s.d.) of the operation time ($\times 10^{-5}$) in seconds for data in $\mathbb{R}^2$.

Median		$k = 10$	$k = 25$	$k = 50$	$k = 100$	$k = 200$
Spatial	mean	27	28	30	29	28
	s.d.	44	45	57	45	45
Component-wise	mean	24	21	25	25	24
	s.d.	42	41	43	43	43
Liu's	mean	3	6	14	49	190
	s.d.	18	24	35	66	250
Tukey's	mean	67	210	510	970	1890
	s.d.	47	28	40	56	100
Oja's	mean	1430	1400	1460	1410	1410
	s.d.	410	190	270	190	160
Projection	mean	7	12	18	31	60
	s.d.	26	32	39	46	49

https://doi.org/10.1371/journal.pone.0229845.t002

Table 3. Mean squared error ($\times 10^{-2}$) for data as in Table 2.

Location Estimator	$k = 10$	$k = 25$	$k = 50$	$k = 100$	$k = 200$
Spatial	67	21	9.7	4.6	2.3
Component-wise	74	26	12.0	5.7	2.9
Liu's	110	31	14.0	6.3	3.2
Tukey's	73	21	10.0	4.8	2.3
Oja's	75	22	11.0	5.6	3.2
Projection	66	21	9.8	4.7	2.3
Mean	110	39	20.0	9.9	5.0

https://doi.org/10.1371/journal.pone.0229845.t003

fast for small k and lower dimensions, but it becomes slower than `l1median` for larger k. Hence, we use `l1median` to compute the spatial median, whose performance for small k is also good.

3.2 Mean squared error for some medians

We assess the accuracy of some of the medians via empirical mean squared error. If $\hat{\mathbf{X}}$ is an estimator in $\mathbb{R}^n$ with respect to the unknown parameter $\boldsymbol{\mu} \in \mathbb{R}^n$, then the mean squared error is

$$\text{MSE}\,(\hat{\mathbf{X}}\,) = n^{-1}E(||\hat{\mathbf{X}} - \boldsymbol{\mu}||_2^2), \tag{40}$$

where $n^{-1}||\hat{\mathbf{X}} - \boldsymbol{\mu}||_2^2$ represents the squared Euclidean distance between $\hat{\mathbf{X}}$ and $\boldsymbol{\mu}$, normalized by the vector length. Smaller $\text{MSE}\,(\hat{\mathbf{X}})$ values are better.

Table 3 shows MSE results based on the same simulations as used for Table 2. Not surprisingly, for this long-tailed data, all medians perform better than the sample mean. The spatial median and the projection median have smaller mean squared error, the latter performing better for small k values. On the other hand, Liu's median always produces a very high mean squared error.

Conclusion. Based on these simulations, for the `R` functions listed in Table 1, the spatial and projection medians always have the lowest mean squared error, but also fast running speeds. Although Liu's median has the shortest computation time, for small k, it is the most inaccurate, and its computation time becomes long for large datasets. Similarly, the component-wise median is fast, even when k increases, but it has a large mean squared error. Hence, the spatial and projection medians are good choices when computing two-dimensional robust measures of location in this case, and the latter is preferred for small datasets. The computational results for high-dimensional simulations ($n = 3, 5, 10$) can be found in S1 Table.

3.3 2D projection median computation functions

The R package `DurocherProjectionMedian` can be downloaded from Github at https://github.com/12ramsake/DurocherProjectionMedian.

The `DurocherProjectionMedian` package provides functions to compute the projection median via the Monte Carlo integration method using `projectionMedianMC`) [27] and an exact method for two dimensions proposed by Ramsay [28] using `projectionMedian2D`. Tables 4 and 5 show the performance of the different functions computing the two-dimensional projection median of 1000 simulated datasets from the Laplace distribution with different k.

Table 4. Mean and standard deviation (s.d.) of the operation time ($\times 10^{-5}$) in seconds for different R functions to produce the projection median.

		k				
R Function		**10**	**25**	**50**	**100**	**200**
`PmedTrapz`	mean	7	12	18	31	60
	s.d.	26	32	39	46	49
`projectionMedian2D`	mean	320	1020	3930	11640	44830
	s.d.	50	99	420	970	2690
`PmedMCInt`	mean	250	320	490	870	1670
	s.d.	40	39	33	50	58
`projectionMedianMC`	mean	930	970	1010	1130	1280
	s.d.	49	57	65	60	55

https://doi.org/10.1371/journal.pone.0229845.t004

For the Monte Carlo Integration method, when k is small (e.g. under 150 in $\mathbb{R}^2$), the computation time of `projectionMedianMC` is longer than our `PmedMCInt` under the same number of projections in both $\mathbb{R}^2$ and high dimensions, whereas both implementations have almost the same MSE.

Although the `projectionMedian2D` provides a slightly smaller MSE, its running time is slow. Our `PmedTrapz` is faster and its MSE performance is comparable to `projectionMedian2D`, and, hence, the former might be recommended as the best choice for $\mathbb{R}^2$.

4 The yamm R package

Our `Yamm` R package provides users with functions to compute the projection median according to the different methods mentioned in section 2.4. `PmedMCInt` computes the projection median using the Monte Carlo approximation; `PmedTrapz` uses the trapezoidal rule and currently, it is only valid in two and three dimensions; `yamm` computes the projection median using the Monte Carlo approximation to find the shift vector $\boldsymbol{\mu}$ minimising our objective function `yamm.obj`. The package also includes functions `Plot2dMedian` and `Plot2dMedian` to plot different multivariate medians for data in both $\mathbb{R}^2$ and $\mathbb{R}^3$. Most functions in our package are implemented internally using C code. This section provides some brief illustrations of the use of `Yamm`.

4.1 `Yamm` projection medians

The function `PmedMCInt` computes the projection median for any multivariate data, `x`, by invoking

```
PmedMCInt(x, nprojs = 20000)
```

Since this function uses Monte Carlo integration, we need to choose the number of projections J, which has a default value of 20000. Typically, a large J is required to obtain a stable

Table 5. Mean squared error ($\times 10^{-3}$) for 1000 sets of data in $\mathbb{R}^2$ generated from Laplace distribution.

	k				
R Function	**10**	**25**	**50**	**100**	**200**
`PmedTrapz`	656	207	98.2	47.1	23.2
`projectionMedian2D`	656	206	97.4	46.9	22.9
`PmedMCInt`	659	205	97.8	47.0	23.0
`projectionMedianMC`	659	205	97.6	47.0	23.0

https://doi.org/10.1371/journal.pone.0229845.t005

answer, which means the result will not change much if recomputed under the same conditions. This function returns the projection median estimate vector.

The function `PmedTrapz` computes the projection median in $\mathbb{R}^2$ and $\mathbb{R}^3$ and is invoked by

```
PmedTrapz(x, no.subinterval)
```

`PmedTrapz` applies the trapezoidal rule once in $\mathbb{R}^2$ and twice in $\mathbb{R}^3$ on each entry of the vector med($\mathbf{X}_a$), mentioned in section 2.1.2, and returns a vector of the projection median estimate.

The argument `no.subinterval` determines the number of subintervals for the trapezoidal rule. For the bivariate case the `no.subinterval` argument is a single number that controls the number of subdivisions for the one-dimensional integration; for the trivariate case the argument is a vector of length two that controls the number of subdivisions for the two integrals. In general, it is better to use at least 36 subintervals, which typically produces accurate results without excessive running time.

More subintervals may be appropriate for more complex datasets. For some unusual data sets it would be ideal to have a high resolution of the interval of integration in one particular region, and a relatively low resolution elsewhere, but this is beyond the scope of the current research. A small number of partitions, e.g. below 15, is not recommended for reasons of accuracy.

The `yamm` function is valid for data of any dimension. It uses an optimiser to provide another method to compute the projection median. The arguments are

```
yamm(x, nprojs = 2000, reltol = 1e-06,
xstart = l1median(x), opt.method = "BFGS",
doabs = 0, full.results = FALSE).
```

The `yamm` function is a wrapper to minimise the the objective function `yamm.obj`, which uses the Monte Carlo method to approximate the squared or absolute value of the univariate median of the projection of the shifted data matrix. The `nprojs` argument controls the number of projections in the Monte Carlo approximation and `doabs` is an indicator, where 1 uses the absolute value of the univariate median and 0 forces the use of the squared value. The arguments `reltol`, `xstart`, `opt.method` are supplied directly to the R optimisation function `optim`: `reltol` is the tolerance for the optimiser, with default value of 10^{-6}. Usually, we set a larger value (e.g. 10^{-3}) to this argument, which will reduce the running time, whilst maintaining accuracy. The argument `opt.method` controls the selection of optimisation methods, which can be chosen from any of the four options, "`BFGS`", "`Nelder-Mead`" [29], "`CG`" [30], "`L-BFGS-B`" [31], and "`SANN`" [32]. The default choice "`BFGS`" is relatively fast and stable in our case. See the help page of the function `optim` in R for further details about the different optimisation methods. The `xstart` argument provides the initial value for the parameters to optimise over, which plays an important role in the function `yamm`. A good starting point will reduce the running time and provide a more accurate result, so we use the spatial median as the default value. Other multivariate medians could be used, but they need to be fast. If `full.results = TRUE`, the output of this function involves a list with components obtained from the `optim` function, otherwise, it returns a vector containing the multivariate median estimate.

4.2 Some real examples

We now exhibit results for the projection medians applied to some real datasets. Our plots show different multivariate medians and the sample mean value for two simulated datasets in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively, allowing the methods to be compared.

4.2.1 Beetle data. The famous `beetle` data [33] takes six measurements on 74 flea-beetles, with each belonging to one of three different species. We apply `yamm` and obtain the following output:

```
yamm(beetle, nprojs = 1000, reltol = 1e-3, doabs = 0,
full.results = TRUE)
[1] 180.19194 123.73920 49.97819 135.87913 13.62603 95.49062
$value
[1] 5.585139
$counts
function gradient
90 4
$convergence
[1] 0
$message
NULL
```

The `yamm` results show that the optimiser executed 90 calls to the objective function `yamm.obj` and constructed 4 gradients. The `par` component contains the estimate of the yamm for the beetle data. These results are not that different from the output generated by `PmedMCInt`, which is

```
PmedMCInt(beetle, nprojs = 100000)
[1] 179.54428 124.72128 50.56934 137.47363 13.23372 94.80188
```

For the beetle data, we chose the number of projections in `yamm` to be 1000, while many more projections were required (e.g. 100000) in `PmedMCInt` to obtain a similar and consistent result; although `yamm` requires optimisation. Fewer projections for the function `PmedMCInt` may lead inaccurate results for some components of the multivariate median. `PmedTrapz` is not valid in this six-dimensional case, but we will show that it has a similar output when computing projection median in two- and three-dimensions.

4.2.2 Simulated Data in $\mathbb{R}^2$ with three clusters. We now use the function `Plot2dMedian` in the package `Yamm` to generate and display different multivariate medians for the simulated data set `clusters2d`. This set contains three clusters, which are generated randomly from different independent normal distributions, and two outliers.

Here, we display the three different estimates of the projection median. When computing other multivariate medians, we use functions from R packages listed in section 3.1. The actual data points is plotted with grey dots. The first plot in Fig 3 is producing excluding the two outliers, whilst the second one includes them. The projection medians produced with different estimators are very close to each other, and not far from the other median estimators also. Fig 3 also shows that the multivariate medians are not particularly affect by the outliers, whilst the mean value is.

4.2.3 Simulated data in $\mathbb{R}^3$ with four clusters. The function `Plot3dMedian` in `Yamm` plots the three-dimensional medians. The dataset `clusters3d` has four clusters, each generated from different independent normal distributions, as well as five outliers. Fig 4 is produced with the dataset `clusters3d`, whose outliers have been removed. It shows that apart from the Oja's median, the other medians are located close to each other. Again, the three approximations of the projection median almost coincide in every component.

4.3 The muqie plot and some examples

As well as obtaining a robust location measure, we can use projections to provide information on the spread and configuration of the data. Obtaining true multivariate quantiles can be

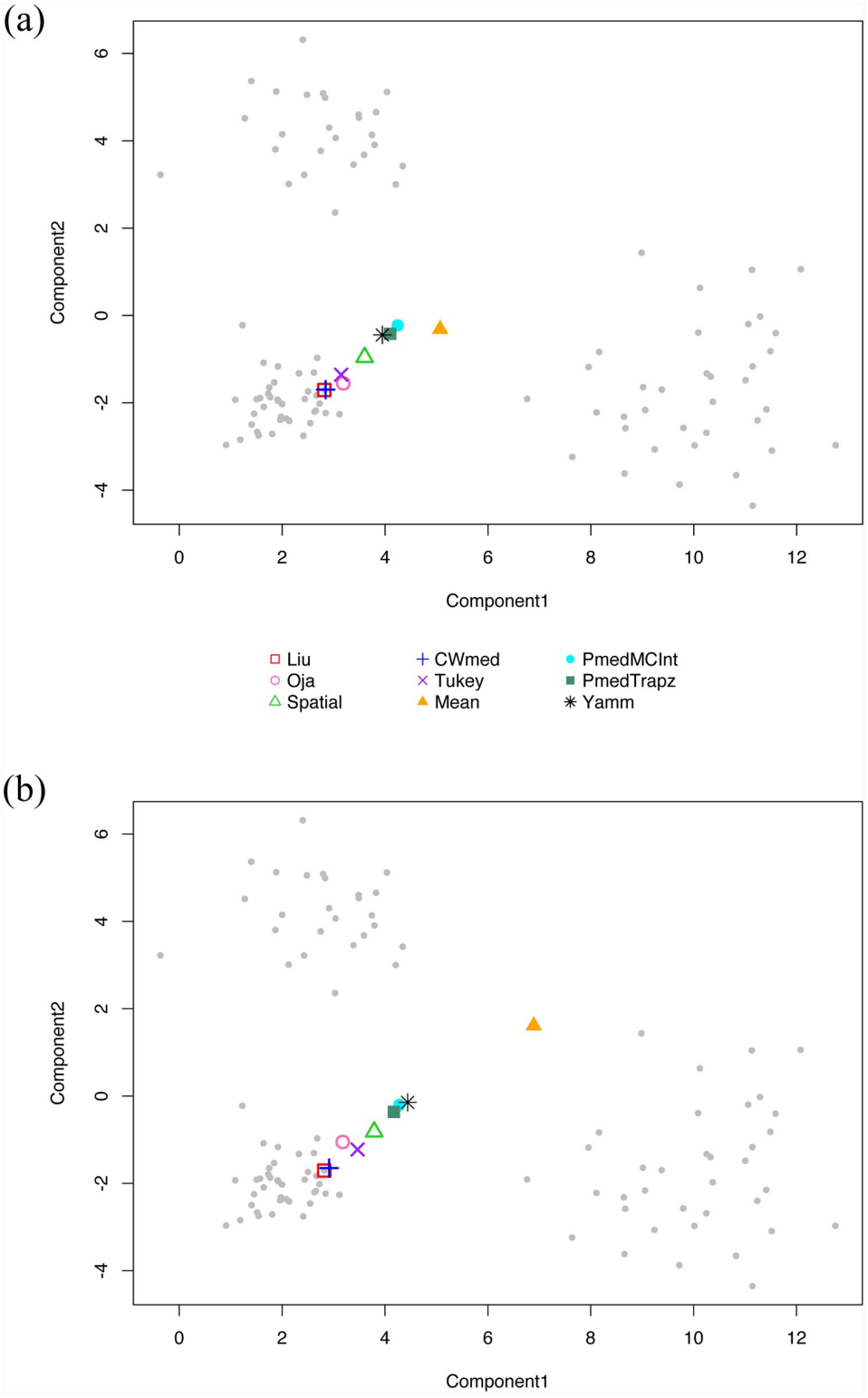

Fig 3. Bivariate medians and mean for three cluster two-dimensional set. *Top*: without outliers; *Bottom*: with outliers (out of plot area).

https://doi.org/10.1371/journal.pone.0229845.g003

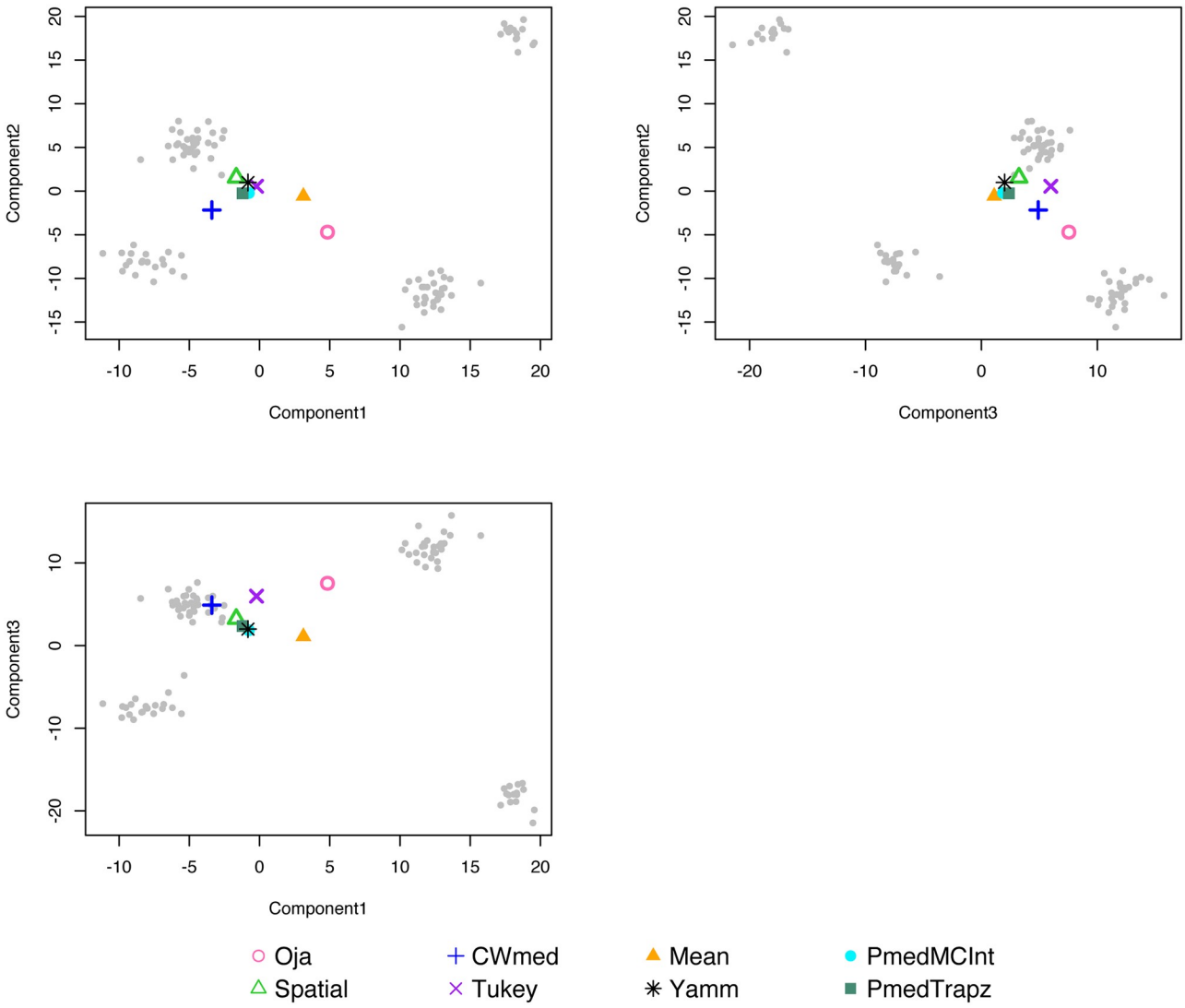

Fig 4. Trivariate medians & mean for four cluster three-dimensional set.

https://doi.org/10.1371/journal.pone.0229845.g004

computationally challenging, and what we produce are not true multivariate quantiles, but they do enable us to gain useful understanding about multivariate data. The muqie (MUltivariate QuantIlE) plots are constructed as follows.

First choose a unit-length direction vector, u. Then project our yamm-centred multivariate data onto u to obtain a univariate set. The muqie point, $Q(\alpha, u)$, is merely the vector u rescaled to have length equal to the α-quantile of the univariate set. A muqie plot is the collection of all muqie points, $Q(\alpha, u)$ over all unit-length direction vectors u. In practice, we construct our plot by choosing a number of directions and joining the points. The basic concept, and plots, are not new, Section 2 of Fraiman and Pateiro-Lopez [34] introduces the concept based on mean-centred data and is related to ideas in [35]. Our main addition to this body of work is to (i) centre using yamm, or other robust median and (ii) presenting the muqie plots as dynamic videos of increasing α.

Fig 5 shows two muqie plots for $\alpha = 0.4$ and $\alpha = 0.8$. The latter indicates the three cluster nature. Surprisingly, this also shows up clearly in the $\alpha = 0.4$ plot with the 0.4 quantile for, e.g.

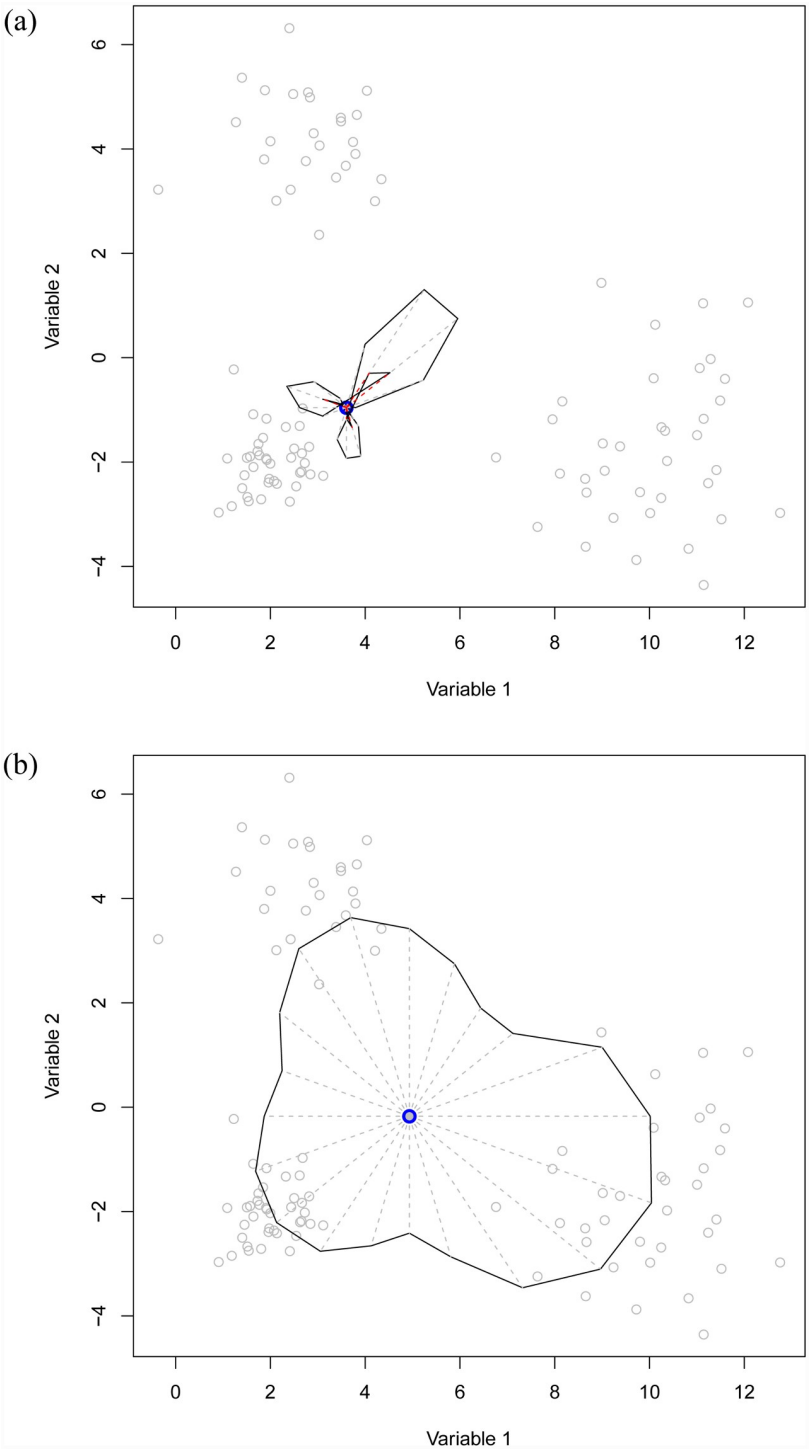

Fig 5. Muqie plot for the three cluster two-dimensional data set without outliers. The figures are produced for different values of pseudo-quantile α. The centre point (in blue) in each plot is the yamm median. Left: $\alpha = 0.4$, Right: $\alpha = 0.8$.

https://doi.org/10.1371/journal.pone.0229845.g005

the bottom-left cluster appearing in a "north-easterly" direction and coloured red in our plot. The movie `Animation` shows an animated plot, which includes both the plots in Fig 5 and many of the others for increasing values of α.

These plots were produced by the `muqie()` function in the `Yamm` package. For the animated plot, the package includes the `makeplot()` function, which calls `muqie()` for multiple values of α. Then we use the CRAN package `animation` to produce an animated GIF using

```
saveGIF(makeplot(clusters2d[,-c(102,103)], nprojs = 4000),
diff.col = 3, interval = 0.1, width = 500, height = 500).
```

The movie `beetle` shows a three-dimensional Muqie plot using three variables from the beetle data. The R commands used were:

```
saveGIF(makeplot3D(beetle, dm = c(1,3,6)), diff.col = 3,
interval = 0.2, width = 500, height = 500)
```

5 Conclusions and discussions

We have introduced a new method, yamm, to compute the projection median, for data in $\mathbb{R}^n$ with $n \geq 2$. We have proved the theoretical equivalence of yamm and the projection median. Through theoretical and numerical investigations we demonstrate the robustness of yamm on a simple, but illuminating, bivariate setup.

Then, we illustrated three computation methods for the projection median, which can be best deployed in different situations. Approximating the projection median by the Monte Carlo method is valid in any dimensions but requires a large number of projections to ensure accuracy, while using the trapezoidal rule is computationally fast and accurate in two and three dimensions, but requires more integration on the projection vector in the higher dimensions, which becomes rapidly more complex. The yamm approximation can also compute the median in any dimensions. Its computational speed is not as quick as the other two, under the same conditions (e.g. the number of projections). However, thanks to the optimiser, a small number of the projections can be chosen to obtain an accurate median with a reasonable starting point (e.g. other multivartiate medians or mean value), which can be a distinct advantage.

Our research also documents the simulated empirical performance for different medians in terms of the computation time and the mean squared error. Using different R functions to calculate different multivariate medians, we find that the spatial median and the projection median are always accurate with relatively fast speed using the existing R functions. The performance of other multivariate medians either exhibits slow speed or large mean squared error.

Finally, we introduce our R package, `Yamm`, that contains our three methods to compute the projection median. We show that our methods coincide with each other in $\mathbb{R}^2$ and $\mathbb{R}^3$, and all multivariate medians are not affected by the outliers in the dataset, but the location of the mean value varies a lot. Currently, the function `PmedTrapz` in the R package is only valid in $\mathbb{R}^2$ and $\mathbb{R}^3$, further investment can be conducted on extending this function to higher dimensions.

The `Yamm` package also introduces our Muqie plots, which are capable of producing animated plots of two- and three-dimensional sets' projected quantiles. The animated 'growth' of these "quantile" plots give a vivid picture of the extent, spread and configuration of data in the sets.

The `Yamm` package is available on the CRAN archive.

Supporting information

S1 Appendix.
(PDF)

S2 Appendix.
(PDF)

S1 Table. Simulation performance for high-dimensional medians.
(PDF)

S1 File.
(GZ)

S1 Video.
(MP4)

S2 Video.
(MP4)

Author Contributions

Conceptualization: Guy Nason.

Data curation: Fan Chen, Guy Nason.

Formal analysis: Fan Chen, Guy Nason.

Funding acquisition: Guy Nason.

Investigation: Fan Chen, Guy Nason.

Methodology: Fan Chen, Guy Nason.

Project administration: Fan Chen, Guy Nason.

Resources: Fan Chen, Guy Nason.

Software: Fan Chen, Guy Nason.

Supervision: Guy Nason.

Validation: Fan Chen, Guy Nason.

Visualization: Fan Chen, Guy Nason.

Writing – original draft: Fan Chen, Guy Nason.

Writing – review & editing: Fan Chen, Guy Nason.

References

1. Hayford J. W. (1902). What is the center of an area or the center of a population. *Journal of the American Statistical Association*, 8(58):47–58. https://doi.org/10.2307/2276137
2. Weber A. (1909). *Über den Standort der Industrien*. Mohr.
3. Weber A. (1929). *Theory of the Location of Industries*. The University of Chicago Press.
4. Tukey J. W. (1975). Mathematics and the picturing of data. *In Proceedings of the International Congress of Mathematicians*, 2:523–531.
5. Oja H. (1983). Descriptive statistics for multivariate distributions. *Statistics & Probability Letters*, 1:327–332. https://doi.org/10.1016/0167-7152(83)90054-8
6. Small C. G. (1990). A survey of multidimensional medians. *International Statistical Review*, 58(3):263–277. https://doi.org/10.2307/1403809

7. Chaudhuri P. and Sengupta D. (1993). Sign tests in multidimension: Inference based on the geometry of data cloud. *Journal of the American Statistical Association*, 88:1363–1370. https://doi.org/10.1080/01621459.1993.10476419
8. Oja H. (2013). Multivariate median. In Becker C., Fried R., and Kuhnt S., editors, *Robustness and Complex Data Structures*, chapter 1, pages 3–16. Springer, Berlin.
9. Durocher S. and Kirkpatrick D. G. (2005). The projection median of a set of points in $\mathbb{R}^2$. *Journal of Computational Geometry*, 42:364–375. https://doi.org/10.1016/j.comgeo.2008.06.006
10. Basu R., Bhattacharya B. B., and Talukdar T. (2012). The projection median of a set of points in $\mathbb{R}^d$. *Discrete and Computational Geometry*, 47(2):329–346. https://doi.org/10.1007/s00454-011-9380-6
11. Johnson N. L., Kotz S., and Balakrishnan N. (1995). *Continuous univariate distributions*. Number v.2 in Wiley series in probability and mathematical statistics: Applied probability and statistics. Wiley & Sons.
12. Robert C. P. and Casella G. (2005). *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg.
13. Broyden C. G. (1970). The convergence of a class of double-rank minimization algorithms. *The Institute of Mathematics and Its Applications*, 6:76–90. https://doi.org/10.1093/imamat/6.1.76
14. Fletcher R. (1970). A new approach to variable metric algorithms. *Computer Journal*, 13(3):317–322. https://doi.org/10.1093/comjnl/13.3.317
15. Goldfarb D. (1970). A family of variable metric updates derived by variational means. *Journal of the Mathematics of Computation*, 24(109):123–126.
16. Shanno D. F. (1970). Conditioning of quasi-newton methods for function minimization. *Journal of the Mathematics of Computation*, 24(111):647–656. https://doi.org/10.1090/S0025-5718-1970-0274029-X
17. Bose P., Maheshwari A., and Morin P. (2003). Fast approximations for sums of distances, clustering and the Fermat-Weber problem. *Computational Geometry: Theory and Applications*, 24(3):135–146. https://doi.org/10.1016/S0925-7721(02)00102-5
18. Aloupis G., Langerman S., Soss M., and Toussaint G. T. (2003). Algorithms for bivariate medians and a Fermat-Torricelli problem for lines. *Computational Geometry*, 26:69–79. https://doi.org/10.1016/S0925-7721(02)00173-6
19. Langerman S. and Steiger W. (2003). *Optimization in Arrangements*. Springer Berlin Heidelberg, Berlin, Heidelberg.
20. Vardi Y. and Zhang C. (2000). The multivariate l1-median and associated data depth. *Proceedings of the National Academy of Sciences*, 97(4):1423–1426. https://doi.org/10.1073/pnas.97.4.1423
21. Cardot H., Cénac P., and Zitt P. A. (2013). Efficient and fast estimation of the geometric median in Hilbert spaces with an averaged stochastic gradient algorithm. *Bernoulli Society for Mathematical Statistics and Probability*, 19(1):18–43.
22. Croux C., Filzmoser P., and Oliveira M. R. (2006). Algorithms for projection-pursuit robust principal component analysis. *KU Leuven Working Paper No. KBI 0624*, 19(1):18–43.
23. Rousseeuw P. J. and Ruts I. (1996). Algorithm AS 307: Bivariate location depth. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 45(4):516–526.
24. Rousseeuw P. J., Ruts I., and Tukey J. W. (1999). The bagplot: A bivariate boxplot. *The American Statistician*, 53(4):382–387. https://doi.org/10.2307/2686061
25. Struyf A. and Rousseeuw P. J. (2000). High-dimensional computation of the deepest location. *Computational Statistics & Data Analysis*, 34(4):415–426. https://doi.org/10.1016/S0167-9473(99)00112-7
26. Fischer D., Mosler K., Möttönen J., Nordhausen K., Pokotylo O., and Vogel D. (2016). Computing the Oja median in R: The package OjaNP. *ArXiv*, pages 1–36.
27. Durocher S., Leblanc A., and Skala M. (2017). The projection median as a weighted average. *Journal of Computational Geometry*, 8:78–104.
28. Ramsay K. (2017). Computable, robust multivariate location using integrated univariate ranks.
29. Nelder J. A. and Mead R. (1965). A simplex algorithm for function minimization. *Computer Journal*, 7:308–313.
30. Fletcher R. and Reeves C. M. (1964). Function minimization by conjugate gradients. *Computer Journal*, 7:148–154. https://doi.org/10.1093/comjnl/7.2.149
31. Nocedal J. and Wright S. J. (1999). *Numerical Optimization*. Springer, first edition.
32. Byrd R. H., Lu P., Nocedal J., and Zhu C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16:1190–1208. https://doi.org/10.1137/0916069
33. Lubischew A. A. (1962). On the use of discriminant functions in taxonomy. *Biometrics*, 18:455–477. https://doi.org/10.2307/2527894

34. Fraiman R. and Pateiro-Lopez B. (2012). Quantiles for finite and infinite dimensional data. *Journal of Multivariate Analysis*, 108:1–14. https://doi.org/10.1016/j.jmva.2012.01.016

35. Kong L. and Mizera I. (2012). Quantile tomography: using quantiles with multivariate data. *Statistica Sinica*, 22:1589–1610.