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

Fan Chen1☯, Guy Nason2☯*

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

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 \( x = (x_1, \ldots, x_k) \) 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

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

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

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...
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 $X = (x_1, \ldots, x_k)^T$ be an $n$-dimensional i.i.d. sample with distribution function $F : \mathbb{R}^n \rightarrow \mathbb{R}$. 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, $M_C(X) \in \mathbb{R}^n$ minimises

\[
k^{-1} \sum_{i=1}^k \{ (|x_i - m_i|) + \cdots + |x_m - m_n| \} - (|x_i| + \cdots + |x_m|),
\]

over $m \in \mathbb{R}^n$, where $|m|^2 = \sum_{i=1}^n m_i^2$ is the (squared) Euclidean norm. The corresponding population functional, $M_C(F)$, for the vector of population medians minimises

\[
E\{ (|x_i - m_1|) + \cdots + |x_n - m_n| \} - (|x_i| + \cdots + |x_n|).
\]

### 1.2 Spatial median

The spatial median $M_S(X)$, also known as the $L_1$ median, minimises

\[
k^{-1} \sum_{i=1}^k \{ |x_i - m| - |x_i| \}
\]

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

\[
E\{ |x - m| - |x| \}.
\]

### 1.3 Oja’s median

Let $X = (x_1, \ldots, x_k)^T$ be an i.i.d. sample in $\mathbb{R}^n$ with distribution function $F : \mathbb{R}^n \rightarrow \mathbb{R}$. The volume of the $n$-variate simplex determined by the $n + 1$ vertices $(m_1, \ldots, m_{n+1})$ is

\[
V(m_1, \ldots, m_{n+1}) = \frac{1}{n!} \left| \det \begin{pmatrix} 1 & \cdots & 1 \\ m_1 & \cdots & m_{n+1} \end{pmatrix} \right|.
\]

The Oja median, $M_O(X)$, minimises

\[
\left( \begin{array}{c} k \\ \binom{n}{k} \end{array} \right) \sum_{i_1 < \cdots < i_k} V(x_{i_1}, \ldots, x_{i_k}, m),
\]

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

\[
E_F\{ V(x_{i_1}, \ldots, x_{i_k}, m) \}.
\]
1.4 Tukey’s median

Let \( X = (x_1, \ldots, x_k)^T \) 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

\[
\hat{F}(H) = n^{-1} \sum_{i=1}^{k} \mathbb{I}(x_i \in H),
\]

where \( \mathbb{I} \) is the usual indicator function. Then, define the depth, \( D(\mu) \), of a point \( \mu \in \mathbb{R}^n \) within the dataset, to be the infimum of \( \hat{F}(H) \), that is taken over all closed half spaces \( \mu \in H \). Tukey’s median is defined as the set of points \( \mu \) 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 \( X \) be a multiset of points in \( \mathbb{R}^2 \) and \( \theta \in [0, 2\pi) \) be an angle. Let \( X_\theta \) denote the multiset defined by the projection of \( X \) onto the unit vector \( u_\theta = (\cos \theta, \sin \theta) \), so

\[
X_\theta = \{ u_\theta(x, u_\theta) \mid x \in X \},
\]

where \( \langle \cdot \rangle \) denotes the usual inner product.

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

\[
M_p(X) = \pi^{-1} \int_0^{2\pi} \text{med}(X_\theta) \, d\theta,
\]

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

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

\[
M_p(X) = n \frac{\int_{X^{n-1}} \text{med}(X_a) \, da}{\int_{X^{n-1}} \, da} = n \int_{X^{n-1}} \text{med}(X_a) \, df(a),
\]

where \( X^{n-1} = \{ x \in \mathbb{R}^n \mid ||x|| = 1 \} \) is the unit \( n \)-dimensional hypersphere, \( \text{med}(X_a) \) is the median of the projection of \( X \) onto the line through the origin parallel to \( a \), and \( f \) is the normalised uniform measure over \( X^{n-1} \). Hence, for a point \( x = (x_1, x_2, \ldots, x_n) \in X^{n-1} \), the \( n \)-
dimensional spherical coordinates are given by
\[
\begin{align*}
    x_1 &= \cos \theta_1 \\
    x_2 &= \sin \theta_1 \cos \theta_2 \\
    x_3 &= \sin \theta_1 \sin \theta_2 \cos \phi_3 \\
    &\vdots \\
    x_{n-1} &= \sin \theta_1 \cdots \sin \theta_{n-2} \cos \theta_{n-1} \\
    x_n &= \sin \theta_1 \cdots \sin \theta_{n-2} \sin \theta_{n-1},
\end{align*}
\]
(13)
where each angle $\theta_1, \theta_2, \ldots, \theta_{n-2}$ has a range of $\pi$ and $\theta_{n-1}$ has range of $2\pi$. Also, the normalised uniform measure $f$ over $X^{n-1}$ is given by
\[
df = \frac{d^{n-1}V}{\int_0^\pi \int_0^\pi \cdots \int_0^\pi d^{n-1}V},
\]
(14)
where $d^{n-1}V = \sin^{n-2}\theta_1 \sin^{n-3}\theta_2 \cdots \sin \theta_{n-2} \, d\theta_1 d\theta_2 \cdots d\theta_{n-1}$ 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 $X = (x_1, \ldots, x_k)^T \in \mathbb{R}^{k \times n}$ be a random sample of size $k \in \mathbb{N}, x_i \in \mathbb{R}^n$. Let $a$ be a $n \times 1$ projection vector of unit length, $1_k$ be the $k \times 1$ vector of ones and $\mu$ a shift vector of length $n$. Let $y$ be the projection of $X$ onto $a$ after $X$ has been shifted by $\mu$:
\[
y = (X - 1_k \mu^T) a,
\]
(15)
where $y \in \mathbb{R}^k$. The univariate median $m$ of the projected points $y$ is
\[
m_X(\mu, a) = m(y).
\]
(16)

Now define the integral
\[
M_{X,m}(\mu) = \int_{\{|a'a'|=1\}} m_X(\mu, a)^2 \, da.
\]
(17)
The yamm estimator of location for $X$ is
\[
\hat{\mu} = \text{yamm}(X) = \arg\min_{\mu} M_{X,m}(\mu).
\]
(18)
Eqs (17) and (18) illustrate the rationale behind yamm. Intuitively, if the shift vector $\mu$ is far away from the true ‘middle’ of the dataset, then the magnitude of $m_X(\mu, a)$, as well as the integral $M_{X,m}(\mu)$, will be large. By contrast, a smaller $m_X(\mu, a)$ can be obtained when the $\mu$ is moving closer to the true ‘middle’ of the data set.

Instead of computing the squared value of $m_X(\mu, a)$ 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 $m_X(\mu, a)$, when $\mu$ 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 $\mu$ is set to be $(2.2, 8)$ and $(2.75)$ respectively, and for each plot, two thousand random projections were used to calculate the univariate median $m_X(\mu, a)$, using methods to be explained in
Section 2.4. Fig 1 shows that when $\mu$ is near the Tukey’s median, the magnitude of each $m_X(\mu, a)$ is less than 0.65, while a larger value, ranging from 0 to 1.2, is shown in the figure when $\mu$ is far away from the median. Overall, when integrated the quantity involving the $\mu$ 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 $\text{med}(X_\mu)$ directly over the unit hypersphere in $\mathbb{R}^n$, whereas yamm minimises the objective function $M_{x,m}(\mu) \in \mathbb{R}$ over the shift vector $\mu$. Despite these differences, the following theorem shows that the projection median and yamm are identical.
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.

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.

2.3.1 Bivariate mixture setup. Let \(X_1 \sim \mathcal{N}_2(v_1, \Sigma_1)\) and \(X_2 \sim \mathcal{N}_2(v_2, \Sigma_2)\) be independent bivariate normal random variables, where \(X_1 = (X_{11}, X_{12})^T, X_2 = (X_{21}, X_{22})^T\) with mean vector \(v_1 = (v_{11}, v_{12})^T\) and \(v_2 = (v_{21}, v_{22})^T\). Let \(R(\theta)\) be a rotation matrix with angle \(\theta\) given by

\[
R(\theta) = \begin{pmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{pmatrix}.
\]

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

\[
Y_i \sim \mathcal{N}_2 \{R(v_i - \mu), R\Sigma_i R^T\}, \quad \text{for } i = 1, 2.
\]

Denote \(Y_i = (Y_{i1}, Y_{i2})^T, Y_{i1}\) is the first entry of \(Y_i\) for \(i = 1, 2\). Then, it is immediate that

\[
s_1 = (v_{i1} - \mu_1) \cos \theta - (v_{i2} - \mu_2) \sin \theta \quad \text{and} \quad \sigma_1^2 = (R\Sigma_i R^T)_{1,1},
\]

\[
s_2 = (v_{i2} - \mu_2) \cos \theta - (v_{i2} - \mu_2) \sin \theta \quad \text{and} \quad \sigma_2^2 = (R\Sigma_i R^T)_{1,1}.
\]

The mixture distribution that we study is

\[
f_m(w_1, w_2) = (1 - \epsilon)f_{X_1}(w_1, w_2) + \epsilon f_{X_2}(w_1, w_2),
\]

where \(f_{X_i}\) is the density of \(X_i\) and \(\epsilon \in [0, 1]\), is typically small. Here, \(f_{X_i}\) is considered to be the bulk of the distribution and \(f_{X_2}\) the outlier.

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

\[
f_{\gamma}(y) = (1 - \epsilon)\phi_{s_1, \sigma_1^2}(y) + \epsilon\phi_{s_2, \sigma_2^2}(y),
\]

where \(s_1, s_2, \sigma_1^2, \sigma_2^2\) are as above and \(\phi\) is the standard normal density.

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

\[
F_{\gamma}(y) = (1 - \epsilon)\Phi_{s_1, \sigma_1^2}(y) + \epsilon\Phi_{s_2, \sigma_2^2}(y),
\]

where \(\Phi\) is the standard normal distribution function. We require the median of the projected distribution, i.e. find

\[
y_m(\epsilon, \theta, s_1, s_2, \Sigma_1, \Sigma_2) \text{ such that } F_{\gamma}(y_m) = 1/2.
\]
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 $R(\theta)$ is an orthogonal matrix, this means that $\sigma_2^T = \sigma_3^T = 1$ and Eq (25) becomes

$$F_s(y) = (1 - \epsilon) \Phi(y - s_1) + \epsilon \Phi(y - s_2).$$

(27)

For small $\epsilon$, we know that the median should be close to the median of the bulk, so the median of $F_s$ 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 $F_s(y)$ around $s_1$, the quantity we know is close to our median:

$$F_s(y) \approx [1 + \epsilon - \epsilon \text{ Erfc} \{(s_1 - s_2)/\sqrt{2}\}] / 2$$

$$+ (2\pi)^{-1/2}[1 + \epsilon - \epsilon \exp \{- (s_1 - s_2)^2/2\}]/(y - s_1)$$

$$+ O\{(y - s_1)^2\},$$

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

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

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

(29)

and then

$$y_m(\theta) \approx s_1 + \frac{\epsilon \sqrt{\pi/2} \left[ \text{Erfc} \left\{ \frac{(s_1 - s_2)/\sqrt{2}}{\sqrt{2}} \right\} - 1 \right]}{1 - \epsilon + \epsilon \exp \{- (s_1 - s_2)^2/2\}}.$$  

(30)

Now using

$$\text{Erfc} \left\{ \frac{(s_1 - s_2)/\sqrt{2}}{\sqrt{2}} \right\} = 2\Phi \left\{ \frac{(s_2 - s_1)/\sqrt{2}}{\sqrt{2}} \right\},$$

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

$$y_m(\theta) \approx s_1 + \frac{\epsilon \sqrt{\pi/2} \left[ 2\Phi \left\{ \frac{(s_2 - s_1)/\sqrt{2}}{\sqrt{2}} \right\} - 1 \right]}{1 - \epsilon + \sqrt{2\pi} \phi(s_2 - s_1)}.$$  

(31)

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

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

(33)

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

$$y_m(\theta) \approx \{(v_{11} - \mu_1) \cos \theta - (v_{12} - \mu_2) \sin \theta\}$$

$$+ \frac{\epsilon \sqrt{\pi/2} \left[ 2\Phi \left\{ \delta_1 \cos \theta - \delta_2 \sin \theta \right\} / \sqrt{2} \right] - 1}{1 - \epsilon - \sqrt{2\pi} \phi(\delta_1 \cos \theta - \delta_2 \sin \theta)}.$$  

(34)
According to Eq (17), our job is to find the optimal \( \mathbf{\mu}^* = (\mu^*_1, \mu^*_2)^T \), which minimises
\[
M = \int_0^{2\pi} \psi^2_m(\theta) \, d\theta.
\] (35)

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

**Proposition.** Let \( X_1 = (X_{11}, X_{12})^T \) and \( X_2 = (X_{21}, X_{22})^T \). Suppose that \( X_1 \sim N_2(v_1, \Sigma_1) \) and \( X_2 \sim N_2(v_2, \Sigma_2) \) independently, where \( v_1 = (v_{11}, v_{12})^T \) and \( v_2 = (v_{21}, v_{22})^T \), respectively. Let the mixture, \( W \), of \( X_1 \) and \( X_2 \) be
\[
f_w(w_1, w_2) = (1 - \epsilon)f_{X_k}(w_1, w_2) + \epsilon f_{X_k}(w_1, w_2),
\]
where \( \epsilon \in [0, 1] \) is considered small.

An approximation of the yamm estimator, \( \mathbf{\mu}^* = (\mu^*_1, \mu^*_2) \), is
\[
\begin{align*}
\mu^*_1 &= v_{11} + \pi^{-1/2}\text{Re}(1 - R^2/32 + R^4/1536) \cos \alpha, \\
\mu^*_2 &= v_{12} + \pi^{-1/2}\text{Re}(1 - R^2/32 + R^4/1536) \sin \alpha,
\end{align*}
\] (36)

where \( R^2 = (\delta_1^2 + \delta_2^2) \), \( \delta_1 = v_{21} - v_{11}, \delta_2 = v_{22} - v_{12} \) and \( \alpha = \text{arctan}(\delta_2/\delta_1) \). The approximation we use is valid whenever \( |R \cos (\theta + \alpha)| < \sqrt{2\pi} \), where \( \theta \) is the projection direction when computing yamm. This inequality is true for all \( \theta \) 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 \( v_{11} = v_{21} \) or \( v_{12} = v_{22} \) (i.e. when one of the \( \delta_i \) is \( 0, i = 1, 2 \)), we can form a more accurate approximation. This is because the approximation for the standard normal distribution function, \( \psi(z) \approx (1 + \cos z)/2\pi \), is no longer required to find the optimal \( \mathbf{\mu}^* = (\mu^*_1, \mu^*_2)^T \) minimising Eq (35). Without loss of generality, let \( v_1 = (v_{11}, v_{12})^T = (0, 0)^T \) and \( v_2 = (v_{21}, v_{22})^T = (0, d)^T \), we obtain the yamm estimator as follows
\[
\begin{align*}
\mu^*_1 &= 0, \\
\mu^*_2 &= 2^{-1/2}d \epsilon \frac{d^2}{8} \left( \text{BesselI}[0, d^2/8] + \text{BesselI}[1, d^2/8] \right),
\end{align*}
\] (37)

where \( \text{BesselI}[n, z] \) 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 \( \epsilon = 0.05, X_1 \sim N_2(v_1, I_1) \) and \( X_2 \sim N_2(v_2, I_2) \), with \( v_1 = (0, 0)^T \) and \( v_2 = (0, d)^T \) 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 \( \mathbf{\mu}^* \) 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.
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 \( \nu_1 \) and \( \nu_2, \) 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

\[
\hat{M}_P(X) = n J^{-1} \sum_{j=1}^{J} \text{med}(X_{a_j}),
\]

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

Calculating approximation of \( \text{Eq (38)} \) 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 \( \text{Eq (12)}, \) using the spherical coordinates illustrated in \( \text{Eq (13)}, \) where the integral can be obtained by the trapezoidal rule. For example, in the two-dimensional case, we apply the trapezoidal rule once on \( \text{Eq (11)}. \) 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, $\{a_j\}_{j=1}^J$ be a set of independent random unit length $n$-vectors, an estimator for $M_{x, m}(\mu)$ is given by

$$
\hat{M}_{x, m}(\mu) = f^{-1} \sum_{j=1}^J m_x(\mu, a_j)^2. \tag{39}
$$

We then numerically minimise $\hat{M}_{x, m}(\mu)$ over $\mu$ 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 $O(k^{d_3}(\log k)^{1+\epsilon})$ 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 $\epsilon$ 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 $O(k \log k)$ 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.
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 1. R functions used for analysing different multivariate medians.

<table>
<thead>
<tr>
<th>Median</th>
<th>Function</th>
<th>Package</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatial</td>
<td><code>lmedian</code></td>
<td>pcaPP</td>
<td>[22]</td>
</tr>
<tr>
<td>CWmed</td>
<td><code>med</code></td>
<td>depth</td>
<td>—</td>
</tr>
<tr>
<td>Liu’s</td>
<td><code>med</code></td>
<td>depth</td>
<td>[23]</td>
</tr>
<tr>
<td>Tukey’s</td>
<td><code>med</code></td>
<td>depth</td>
<td>[24] [25]</td>
</tr>
<tr>
<td>Oja’s</td>
<td><code>ojaMedianEvo</code></td>
<td>OjaNP</td>
<td>[26]</td>
</tr>
<tr>
<td>Projection</td>
<td><code>PmedTrapz</code></td>
<td>Yamm</td>
<td>Ours</td>
</tr>
</tbody>
</table>

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

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

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

https://doi.org/10.1371/journal.pone.0229845.t002
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 $\hat{X}$ is an estimator in $\mathbb{R}^n$ with respect to the unknown parameter $\mu \in \mathbb{R}^n$, then the mean squared error is

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

where $n^{-1}||\hat{X} - \mu||_2^2$ represents the squared Euclidean distance between $\hat{X}$ and $\mu$, normalized by the vector length. Smaller $\text{MSE}(\hat{X})$ 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 $\text{projectionMedianMC}$ [27] and an exact method for two dimensions proposed by Ramsay [28] using $\text{projectionMedian2D}$. 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 3. Mean squared error ($\times10^{-2}$) for data as in Table 2.

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

https://doi.org/10.1371/journal.pone.0229845.t003
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 $\mu$ 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

$$PmedMCInt(x, \text{nprojs} = 20000)$$

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 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.

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

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

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

<table>
<thead>
<tr>
<th>R Function</th>
<th>$k$</th>
<th>10</th>
<th>25</th>
<th>50</th>
<th>100</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>PmedTrapz</td>
<td>656</td>
<td>207</td>
<td>98.2</td>
<td>47.1</td>
<td>23.2</td>
<td></td>
</tr>
<tr>
<td>projectionMedian2D</td>
<td>656</td>
<td>206</td>
<td>97.4</td>
<td>46.9</td>
<td>22.9</td>
<td></td>
</tr>
<tr>
<td>PmedMCInt</td>
<td>659</td>
<td>205</td>
<td>97.8</td>
<td>47.0</td>
<td>23.0</td>
<td></td>
</tr>
<tr>
<td>projectionMedianMC</td>
<td>659</td>
<td>205</td>
<td>97.6</td>
<td>47.0</td>
<td>23.0</td>
<td></td>
</tr>
</tbody>
</table>

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 \( \text{PmedTrapz} \) computes the projection median in \( \mathbb{R}^2 \) and \( \mathbb{R}^3 \) and is invoked by

\[
\text{PmedTrapz}(x, \text{no.subinterval})
\]

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

The argument \( \text{no.subinterval} \) determines the number of subintervals for the trapezoidal rule. For the bivariate case the \( \text{no.subinterval} \) 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 \( \text{yamm} \) function is valid for data of any dimension. It uses an optimiser to provide another method to compute the projection median. The arguments are

\[
\text{yamm}(x, \text{nprojs} = 2000, \text{reltol} = 1e-06, \text{xstart} = \text{l1median}(x), \text{opt.method} = \text{“BFGS”}, \text{doabs} = 0, \text{full.results} = \text{FALSE}).
\]

The \( \text{yamm} \) function is a wrapper to minimise the the objective function \( \text{yamm.obj} \), 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 \( \text{nprojs} \) argument controls the number of projections in the Monte Carlo approximation and \( \text{doabs} \) is an indicator, where 1 uses the absolute value of the univariate median and 0 forces the use of the squared value. The arguments \( \text{reltol}, \text{xstart}, \text{opt.method} \) are supplied directly to the \( \text{R} \) optimisation function \( \text{optim} \); \( \text{reltol} \) 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 \( \text{opt.method} \) 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 \( \text{optim} \) in \( \text{R} \) for further details about the different optimisation methods. The \( \text{xstart} \) argument provides the initial value for the parameters to optimise over, which plays an important role in the function \( \text{yamm} \). 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 \( \text{full.results} = \text{TRUE} \), the output of this function involves a list with components obtained from the \( \text{optim} \) 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:

```r
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

```r
PmedMCInt(beetle, nprojs = 100000)
```

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
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 \( \alpha \)-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 \( \alpha \).

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

https://doi.org/10.1371/journal.pone.0229845.g004
Fig 5. Muqie plot for the three cluster two-dimensional data set without outliers. The figures are produced for different values of pseudo-quartile $\alpha$. The centre point (in blue) in each plot is the yann 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 $\alpha$.

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 $\alpha$. Then we use the CRAN package animation to produce an animated GIF using

```r
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:

```r
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 multivariate 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


