Journal of Statistical Software

December 2025, Volume 115, Issue 4. *doi: 10.18637/jss.v115.i04*

watson: An R Package for Fitting Mixtures of Watson Distributions

Lukas Sablica
WU Wirtschafts-
universität Wien

Kurt Hornik
WU Wirtschafts-
universität Wien

Josef Leydold
WU Wirtschafts-
universität Wien

Abstract

In this paper we present and showcase the R package **watson** which provides a computational framework for fitting and random sampling of the Watson distribution on a p-dimensional sphere. We first introduce the random sampling scheme of the package, which offers two sampling algorithms that are based of the results of Sablica, Hornik, and Leydold (2025). What is more, the package offers a smart tool to combine these two methods, and based on the selected parameters, it approximates the relative sampling speed for both methods and picks the faster one. In addition, we describe the main fitting function for the mixtures of Watson distribution which uses the expectation-maximization (EM) algorithm. Special features are the possibility to use multiple variants of the E-step and M-step, sparse matrices for the data representation and a control parameter which will dynamically eliminate small clusters with overall contribution smaller than this parameter. Moreover, we discuss the numerical issues of the whole fitting procedure and describe how this is handled and solved in the package. Finally, we demonstrate the package on multiple examples involving misspecified simulation study, estimation of the New Zealand earthquake data and depth image clustering.

Keywords: Watson distribution, spherical data, Kummer's function, directional distribution.

1. Introduction

In the history of directional statistics clustering on the sphere has gained a lot of popularity, while still being a difficult task mostly when it comes to estimation. The troubles usually come from the difficulty to evaluate the normalizing constants associated with the directional distributions. For the most commonly employed distribution the normalizing terms can be mathematically conveniently represented in terms of infinite hypergeometric series, which however are hard to evaluate numerically using standard floating point computations.

Probably the most natural choice of the directional distribution is the von Mises-Fisher (vMF) distribution. The vMF distribution has concentric contour lines similar to the multivariate normal distribution which allows to model data concentrated around these directions. Here the estimating procedure has been solved by introducing tight bounds on the normalizing constant (Hornik and Grün 2014a) that allowed to build a fitting framework for mixtures of such distributions (Hornik and Grün 2014b).

While the vMF distribution is the first choice for most of the directional data, this distribution can be inappropriate if the data contains some additional structure. An example of such data is axially symmetric data (i.e., $\boldsymbol{x}$ and $-\boldsymbol{x}$ are indistinguishable) used for example in structural geology or rock magnetism areas. An essential choice in these cases is the Watson distribution (Watson 1965) whose mixture modeling will be of the main focus in this paper.

Recently, mostly thanks to the renewed attention in directional data modeling due to machine learning and sentiment analysis, the Watson distribution and its application in mixture modeling was discussed in Bijral, Breitenbach, and Grudic (2007) and Sra and Karp (2013). The approximation of the maximum likelihood estimation using continued fractions shows anomalous convergence for a major set of values producing the loss of numerical precision, for more details see Gautschi (1977). On the other hand, Bijral *et al.* (2007) derived a maximum likelihood estimation using useful approximations but without any mathematical justification. A brilliant and mathematically justified approach is to use the derived bounds from Sra and Karp (2013) to approximate the true values during the estimation, which however again comes with the price of an approximation error.

Further insights to this problem were recently presented in Sablica and Hornik (2022), where the authors derived a convergent sequence of bounds that can be used during the estimation procedure. These bounds only after two iterations provide state of the art boundaries and allow to estimate the likelihood with arbitrary precision. This and also other numerical techniques and applications from Sablica and Hornik (2024) were used and adjusted to the modeling of the axial data using Watson distribution which is offered as the R package **watson** (Sablica, Hornik, and Leydold 2026) described in this paper.

This paper is organized as follows: Firstly, in Section 2 we introduce the Watson distribution together with the available sampling and estimation methods. Then, Section 3 presents modeling with finite mixtures of Watson distributions. In Section 4 we then introduce the main functions of the **watson** package and their use is demonstrated. Section 5 summarizes the main numerical difficulties and their solutions that are being applied under the hood of the package, when estimation or sampling is requested. Finally, the package is demonstrated in Section 6 with multiple simulated and real data examples.

2. The Watson distribution

Following Mardia and Jupp (2009), let $\mathbb{S}^{p-1} = \{\boldsymbol{x} \in \mathbb{R}^p, \|\boldsymbol{x}\|_2 = 1\}$ be the $(p-1)$-dimensional unit sphere. The density of a unit vector in $\mathbb{R}^p$ with Watson distribution is then of the form

$$f(\boldsymbol{x}|\kappa, \mu) = M\left(\frac{1}{2}, \frac{p}{2}, \kappa\right)^{-1} e^{\kappa(\boldsymbol{x}^\top \mu)^2}, \tag{1}$$

where $\kappa \in \mathbb{R}$ is the concentration parameter, $\mu \in \mathbb{S}^{p-1}$ is the mean direction parameter and

$$M(a, b, z) = {}_1F_1(a; b; z) = \sum_{n=0}^{\infty} \frac{(a)_n}{(b)_n} \frac{z^n}{n!} \tag{2}$$

is the confluent hypergeometric function of the first kind (National Institute of Standards and Technology, Equation 13.2.E2), also known as Kummer's function. Clearly, the density is symmetric about 0, so the Watson distribution can also be taken as a distribution on the projective space $\mathbb{P}^{p-1}$ of directions in $\mathbb{R}^p$, such that the vectors $\pm\boldsymbol{x} \in \mathbb{S}^{p-1}$ are equivalent.

For the case where $\kappa > 0$, the distribution is bipolar with density attaining maxima at $\pm\mu$. In the case of $\kappa < 0$, the distribution is a symmetric girdle distribution with data concentrated around the great circle orthogonal to μ. In the case of $\kappa = 0$ the distribution simplifies to a uniform distribution on the sphere and as $\kappa \to \pm\infty$ the distribution becomes more and more concentrated around the above mentioned shapes. Finally, it is easy to see that for orthogonal Q, such that $Q\mu = \mu$, we have $\mu^\top(Q\boldsymbol{x}) = \mu^\top Q^\top Q\boldsymbol{x} = \mu^\top\boldsymbol{x}$, showing the rotational symmetry about μ. For more details, see Mardia and Jupp (2009).

2.1. Simulation of the Watson distribution

Firstly, assume $\kappa = 0$. The Watson distribution then simplifies to the uniform or isotropic distribution on sphere $\mathbb{S}^{p-1}$ (Bingham 1974), which can be easily generated using p standard normal random variables. Assume i.i.d. $X_i \sim N(0, 1)$ and $X = (X_1, \ldots, X_p)$, then $Y = X/\|X\|_2$ has a uniform distribution on $\mathbb{S}^{p-1}$. To see this note that $X \sim N_p(0, I_p)$ is invariant under rotations, i.e., $QX \overset{d}{=} X$ for any orthogonal Q, and since $\|Y\|_2 = 1$, Y must be uniformly distributed on the sphere.

For the cases of non-zero κ, the current literature that considers directly the Watson distribution is, according to our best knowledge, equipped mostly only with the cases where $p \leq 3$, for example see Best and Fisher (1986) or Li and Wong (1993). The only exception is Sablica *et al.* (2025) which offers two algorithms that can perform such a task for any set of parameters. Exactly the two main algorithms of this reference are also offered by the **watson** package. While having two algorithms that can perform the same task might seem redundant on the first look, each of the algorithms excels in different settings making them together a very balanced combination that ensures efficient sampling across a wide range of settings. What is more, the users can select to use an automated selection of the sampling methods, where for every component of a mixture distribution the sampling algorithm is automatically selected based on the parameters and settings. The way this is handled is that the package stores a grid of measured run times for both of the algorithms and uses linear interpolation to approximate the runtimes for any selection of parameters κ, μ, dimension and number of draws. Based on this the code then selects a faster method. For more details see Section 4.1.

The first algorithm adapts the rejection sampling algorithm from Kent, Ganeiber, and Mardia (2018), which originally samples Bingham distribution using angular central Gaussian (ACG) envelopes. Sablica *et al.* (2025) showed that such an adaptation allows to obtain closed form expression for the parameters that maximize the efficiency and furthermore avoids the computational burden associated with high-dimensional matrix inversion by using a smart matrix inversion technique. This allowed to further analyze the efficiency of the algorithm

Figure 1: Relative differences of ACG sampler and Tinflex sampler for $n = 10$ and $n = 10000$, with reference value being the smaller of the two values. Negative and positive numbers (toned into red and blue color respectively) indicate the dominance of ACG sampler and Tinflex sampler respectively. Cases where $p > 3$ and $\kappa \geq (p-3)/2$ (i.e., where the log-density of projected distribution is neither concave nor convex on $(0, 1)$) are further annotated with a thick black border (Sablica *et al.* 2025).

and the asymptotic behavior of the rejection constant $\mathcal{M}_p(\kappa)$ resulting in

$$\lim_{\kappa \to -\infty} \mathcal{M}_p(\kappa) \approx \sqrt{\frac{p}{p-1}}, \qquad \text{and} \qquad \lim_{\kappa \to \infty} \mathcal{M}_p(\kappa) \approx \sqrt{\frac{pe}{2}}, \tag{3}$$

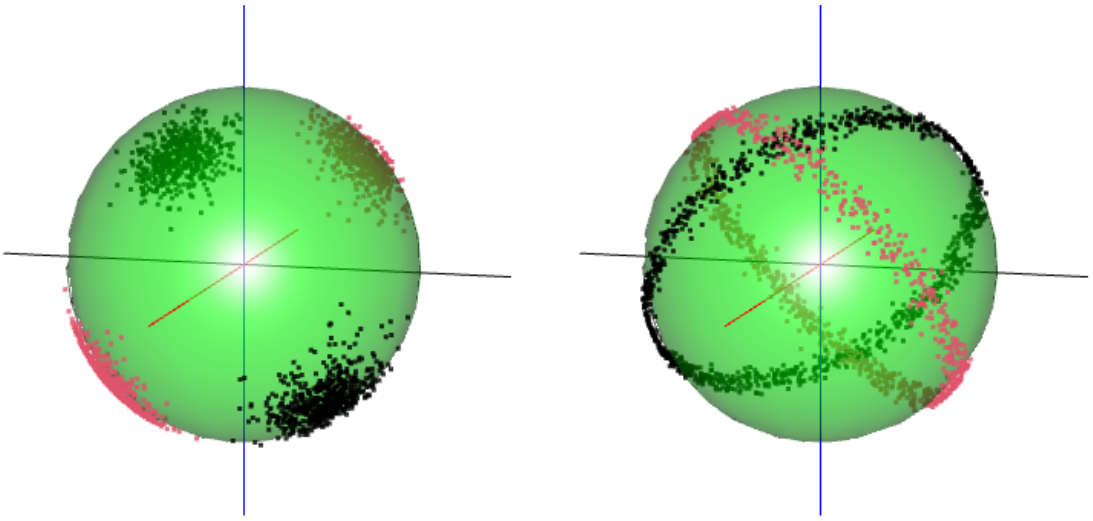
for p large enough. Moreover, the authors showed that these two asymptotic results can be seen as worst case scenarios and a general bounds for the efficiency of the algorithm, because they bound the efficiency from above for all $\kappa \in (-\infty, 0)$ and $\kappa \in (0, \infty)$, respectively.

The second algorithm uses the projection results for the Saw distribution family (Saw 1978). This offers to reduce the whole problem to a sampling from a univariate distribution, for which it has been shown that it fulfills the properties needed by the **Tinflex** algorithm (Leydold, Botts, and Hörmann 2019; Botts, Hörmann, and Leydold 2012). In this algorithm the condition of a log-concave density as required in the standard adaptive rejection sampler by Gilks and Wild (1992) has been dropped. Only a very rough estimate of the inflection point of the log-density has to be provided. Since **Tinflex** requires some initial time for the setup period, it is only natural to expect it to be slower when only small amounts of random draws are needed. On the other side, thanks to the univariate form of the marginal distribution and bounded rejection constant it dominates the first introduced algorithm for n large enough and p small enough.

Both algorithms were already compared in Sablica *et al.* (2025), where the resulting runtimes are visualized in Figure 1.

2.2. Estimation of the Watson distribution

Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{S}^{p-1}$ be i.i.d. sample from the Watson distribution with parameters κ and μ



(a) $\kappa = (20, 20)$ (b) $\kappa = (-200, -200)$

Figure 2: Resulting samples for two-component mixtures with positive (left) and negative (right) concentration parameters. The code for generating these samples can be found in Section 4.1.

and define $\mathbf{X}$ to be the design matrix with $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ as rows. The log-likelihood with respect to the uniform distribution on the sphere is then given by

$$\ell\left(\kappa, \mu | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\right) = n(\kappa \mu^\top \mathbf{S} \mu - \log\left(M(1/2, p/2, \kappa)\right)), \tag{4}$$

where $\mathbf{S}$ is the scatter matrix $\mathbf{S} = \mathbf{X}^\top \mathbf{X}/n$. Since $\mathbf{S}$ is symmetric, $\mu^\top \mathbf{S} \mu = R(\mathbf{S}, \mu)$ is a Rayleigh quotient of matrix $\mathbf{S}$ and thus satisfies $\lambda_1 \leq R(\mathbf{S}, \mu) \leq \lambda_p, \forall \mu \in \mathbb{S}^{p-1}$, where $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_p$ are the ordered eigenvalues of $\mathbf{S}$. More precisely, it holds that $R(\mathbf{S}, s_p) = \lambda_p$ and $R(\mathbf{S}, s_1) = \lambda_1$, where $s_1, \ldots, s_p$ are the normalized eigenvectors corresponding to $\lambda_1, \ldots, \lambda_p$. Hence, maximizing Equation 4 with respect to μ obviously gives

$$\hat{\mu} = \begin{cases} s_1, & \text{if } \hat{\kappa} < 0, \\ s_p, & \text{if } \hat{\kappa} > 0. \end{cases} \tag{5}$$

Setting the first order derivative with respect to κ to zero gives

$$g\left(\frac{1}{2}, \frac{p}{2}, \hat{\kappa}\right) = \frac{M'(\frac{1}{2}, \frac{p}{2}, \hat{\kappa})}{M(\frac{1}{2}, \frac{p}{2}, \hat{\kappa})} = \mu^\top \mathbf{S} \mu = R(\mathbf{S}, \mu) = r, \tag{6}$$

with $0 \leq \lambda_1 \leq r \leq \lambda_p \leq 1, \ \forall \mu \in \mathbb{S}^{p-1}$ and

$$g(a, b, \kappa) = \frac{\mathrm{d} \log(M(a, b, \kappa)))}{\mathrm{d}\kappa} = \frac{M(a, b, \kappa)'}{M(a, b, \kappa)} = \frac{a}{b} \frac{M(a+1, b+1, \kappa)}{M(a, b, \kappa)}, \tag{7}$$

as defined in Sra and Karp (2013) or Sablica and Hornik (2022), as the problem that has to be solved. Clearly, the maximizations are not independent, however, if $\hat{\kappa} > 0$, s_p and λ_p are optimal and if $\hat{\kappa} < 0$, the solution is given by s_1 and λ_1. For $\hat{\kappa} = 0$, which is optimal only when

$\lambda_p = \lambda_1$, any μ is optimal and thus w.l.o.g. $\hat{\mu} = s_1$. Otherwise, we solve $g(\frac{1}{2}, \frac{p}{2}, \hat{\kappa}^{(1)}) = \lambda_1$ and $g(\frac{1}{2}, \frac{p}{2}, \hat{\kappa}^{(2)}) = \lambda_p$ to obtain $\hat{\kappa}^{(1)}, \hat{\mu}^{(1)} = s_1$ and $\hat{\kappa}^{(2)}, \hat{\mu}^{(2)} = s_p$, respectively. We then compute the logarithm of the likelihood ratio

$$\ell(\hat{\kappa}^{(1)}, \hat{\mu}^{(1)} | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) - \ell(\hat{\kappa}^{(2)}, \hat{\mu}^{(2)} | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \tag{8}$$

and select $(\hat{\kappa}^{(1)}, \hat{\mu}^{(1)})$ if the ratio is larger than zero and $(\hat{\kappa}^{(2)}, \hat{\mu}^{(2)})$ otherwise.

To sum up, we are interested in the solution κ of the highly-nonlinear problem

$$g\left(a, b, \kappa\right) = r, \quad \text{where} \quad 0 < a < b, \text{ and } 0 \leq r \leq 1. \tag{9}$$

It can be shown (see Sra and Karp 2013) that $g(a, b, z)$, for $0 < a < b$, as is the case for the Watson distribution (where $a = 1/2$ and $b = p/2$), is a strictly increasing function which maps the interval $(-\infty, \infty)$ onto the interval $(0, 1)$. Even though $g(a, b, z)$ admits a continued fraction representation, possible approximation using continued fractions shows anomalous convergence for a major set of values producing the loss of numerical precision, for more details see Gautschi (1977).

Sra (2007) suggested the ad-hoc approximation

$$\kappa \approx (a + b - 1)\left(\frac{1}{1 - r} - \frac{a}{(b - 1)r}\right) \tag{10}$$

based on the approximation of $M(a, b, z) \approx M(a, b + 1, z)$.

Bijral *et al.* (2007) at around the same time offered another approximation,

$$\kappa \approx \frac{br - a}{r\left(1 - r\right)}, \tag{11}$$

which they observed to be accurate for the Watson case. The equation was derived from the assumption $g(a, b, \kappa) \approx g(a + 1, b + 1, \kappa)$ and was also offered together with a correction term ("determined empirically"). The final form was presented as

$$\kappa \approx \frac{br - a}{r\left(1 - r\right)} + \frac{r}{2b\left(1 - r\right)}. \tag{12}$$

A huge step further was then accomplished in Sra and Karp (2013), where the authors derived tight bounds for the inverse of $g(a, b, \kappa)$. Defining

$$\begin{aligned} L(r) =& \frac{rb - a}{r(1 - r)}\left(1 + \frac{1 - r}{b - a}\right), \\ B(r) =& \frac{rb - a}{2r(1 - r)}\left(1 + \sqrt{1 + \frac{4\left(b + 1\right)r\left(1 - r\right)}{a\left(b - a\right)}}\right), \\ U(r) =& \frac{rb - a}{r(1 - r)}\left(1 + \frac{r}{a}\right), \end{aligned} \tag{13}$$

it has been shown that the solution of equation Equation 9 satisfies

$$\begin{aligned} L(r) < \kappa < B(r) < U(r) \quad & \text{for } 0 < r < a/b, \\ L(r) < B(r) < \kappa < U(r) \quad & \text{for } a/b < r < 1, \end{aligned} \tag{14}$$

where all bounds are additionally exact at $\kappa = 0$, i.e., $r = a/b$. To have a unanimous decision rule which bound to use, the authors suggest the following rule-of-thumb:

$$\kappa \approx \begin{cases} U(r), & \text{if} \quad 0 < r < \frac{a}{2b}, \\ B(r), & \text{if} \quad \frac{a}{2b} \leq r < \frac{2a}{\sqrt{b}}, \\ L(r), & \text{if} \quad \frac{2a}{\sqrt{b}} \leq r < 1. \end{cases} \tag{15}$$

Furthermore, Sra and Karp (2013) introduced a closed form Newton algorithm to solve Equation 9, given the assumption that $g(a, b, \kappa)$ can be easily evaluated

$$\kappa_{n+1} = \kappa_n - \frac{g(a, b, \kappa_n) - r}{g'(a, b, \kappa_n)} = \kappa_n - \frac{g(a, b, \kappa_n) - r}{(1 - b/\kappa_n)\, g\left(a, b, \kappa_n\right) + (a/\kappa_n) - \left(g\left(a, b, \kappa_n\right)\right)^2}. \tag{16}$$

Observe that the iteration can be performed only with one evaluation of the ratio $g(a, b, \kappa)$.

This finally leads to the contribution by Sablica and Hornik (2022), where the authors have derived iterative bounds to evaluate $g\left(a, b, \kappa\right)$ for the cases where $0 < a < b$. The resulting bounds offer a cheap and efficient way to evaluate the necessary function and show a fast convergence with asymptotically correct behavior. Additionally, it is demonstrated that already the first iteration of the bounds defines the same bounds as in Sra and Karp (2013) offering a better approximation for $g\left(a, b, \kappa\right)$ if more than one iteration is performed. This suggests to combine the techniques from Sablica and Hornik (2022) with the Newton method derived by Sra and Karp (2013) to solve Equation 9.

What is more, since the first iterations are still defined in closed form, one can start the Newton procedure with the mid-value of the bounds and perform a bracketed type of Newton algorithm, which combines derivative-based and bisection steps with the starting brackets given by the bounds Equation 13. With some small adjustments, this is also the to-go and default implementation in package **watson**.

A final possibility is to add the logarithm to the whole procedure and to solve

$$\log(g\left(a, b, \kappa\right)) = \log(r), \quad \text{where} \quad 0 < a < b, \text{ and } 0 \leq r \leq 1. \tag{17}$$

One can show that the Newton step is then defined as

$$\begin{aligned} \kappa_{n+1} &= \kappa_n - \frac{\log(g(a, b, \kappa_n)) - \log(r)}{\log\left(g(a, b, \kappa_n)\right)'} \\ &= \kappa_n - \frac{\log(g(a, b, \kappa_n)) - \log(r)}{(1 - b/\kappa) + \left(a/\left(\kappa_n g\left(a, b, \kappa_n\right)\right)\right) - g\left(a, b, \kappa_n\right)}, \end{aligned} \tag{18}$$

again requiring only one evaluation of $g(a, b, \kappa)$ per iteration. Following the results of Sablica and Hornik (2022), since it holds

$$g(a, b, \kappa) = 1 - g(b - a, b, -\kappa), \tag{19}$$

w.l.o.g. one can assume $\kappa < 0$, and therefore the above method can numerically help in the cases where the previous derivative is numerically equal to zero, i.e., with extremely small or large values of κ.

All of the mentioned cases are implemented in the **watson** package and the usage will be discussed more in Section 4.

3. Finite mixture modeling

Finite mixture modeling is a popular statistical tool in many research areas. These models allow to cluster observations by assuming that for each observation there exists a suitable parametric distribution, which defines a subgroup of the data. The mixture distribution is then a convex combination of the corresponding components, where the weight are usually specified by the affiliation of the data to a given component.

Mixture models have attracted a lot of popularity also in directional statistics. The EM algorithms for the Watson distribution are for example given in Bijral *et al.* (2007) and Sra and Karp (2013). In the following, for the purpose of clarity, we will stick more to the latter, and hence the matrix notation.

We note that, while the main area of applications for mixture models is the clustering in the unsupervised settings, where the associated labels of each data point are not available for training, mixtures can be used also in the supervised settings, where the associated labels are specified by the user and the parameters of the parametric distributions are estimated conditionally on these assignments. Examples for both of these applications are provided in Section 6.

3.1. Estimating the parameters of mixtures of Watson distributions

Suppose we have $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{S}^{p-1}$ i.i.d. sample and as before define $\mathbf{X}$ to be the design matrix with $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ as rows. We are interested in the modeling of the data into K multivariate Watson distributions, that together form a mixture distribution. Let $W_p(\boldsymbol{x}|\mu_j, \kappa_j)$ be the density of the j-th component, and let π_j be the corresponding weight, with $\sum_{j=1}^{K} \pi_j = 1$. Then the density for a given observation $\boldsymbol{x}_i$ is

$$f(\boldsymbol{x}_i|\pi_1, \mu_1, \kappa_1 \ldots, \pi_k, \mu_k, \kappa_k) = \sum_{j=1}^{K} \pi_j W_p\left(\boldsymbol{x}_i|\mu_j, \kappa_j\right), \tag{20}$$

such that the log-likelihood for the whole data is given by

$$\ell(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n|\pi_1, \mu_1, \kappa_1 \ldots, \pi_k, \mu_k, \kappa_k) = \sum_{i=1}^{n} \log\left(\sum_{j=1}^{K} \pi_j W_p\left(\boldsymbol{x}_i|\mu_j, \kappa_j\right)\right). \tag{21}$$

The EM algorithm for fitting mixtures of Watson distributions consists of the following steps:

1. Initialization: Assign the probabilities of the component memberships to each observation. This can either be done randomly or by giving user-defined values. Such initialization allows further preprocessing as for example the diametrical clustering, which will be discussed later in this section. Finish the initialization using the M-step which assigns the starting parameters of the mixture distribution using maximum likelihood techniques.

   We note that an EM-algorithm is also possible to initialize with the given components parameters and continue with the E-step, however, since it is considered much easier to have a prior knowledge about the classification rather than the components parameters, this initialization is not offered in **watson**.

2. Repeat the following procedure until the convergence or the maximum number of iterations is reached:

E-step: First construct a lower bound for the log-likelihood $\ell(\cdot)$ given by

$$\ell(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n | \pi_1, \mu_1, \kappa_1 \ldots, \pi_k, \mu_k, \kappa_k) \geq \sum_{i,j} \beta_{ij} \log \frac{\pi_j W_p\left(\boldsymbol{x}_i | \mu_j, \kappa_j\right)}{\beta_{ij}}, \tag{22}$$

where β_{ij} are the posteriors defined as

$$\beta_{ij} = \frac{\pi_j W_p\left(\boldsymbol{x}_i | \mu_j, \kappa_j\right)}{\sum_{k=1}^{K} \pi_k W_p\left(\boldsymbol{x}_i | \mu_k, \kappa_k\right)}. \tag{23}$$

This calculates the probabilities of belonging to a component conditional on the observed values, and defines the so-called soft-E-step.
From the numerical perspective, it is safer to write this as

$$\log \beta_{ij} = \log \pi_j + \kappa_j (\boldsymbol{x}_i^\top \mu_j)^2 - \log M\left(\frac{1}{2}, \frac{p}{2}, \kappa_k\right) - \log\left(\sum_{k=1}^{K} \pi_k W_p\left(\boldsymbol{x}_i | \mu_k, \kappa_k\right)\right), \tag{24}$$

and hence

$$\begin{aligned} \log \beta_{ij} = & \log \pi_j + \kappa_j (\boldsymbol{x}_i^\top \mu_j)^2 - \log M\left(\frac{1}{2}, \frac{p}{2}, \kappa_k\right) + m \\ & - \log\left(\sum_{k=1}^{K} \exp\left(\log \pi_k + \kappa_j (\boldsymbol{x}_k^\top \mu_k)^2 - \log M\left(\frac{1}{2}, \frac{p}{2}, \kappa_k\right) - m\right)\right), \end{aligned} \tag{25}$$

where $m = \arg\max_j \ \log \pi_j + \log W_p\left(\boldsymbol{x}_i | \mu_j, \kappa_j\right)$.

A further possibility is using a hard assignment step (see Sra and Karp 2013), also called categorical step, where:

$$\beta_{ij} = \begin{cases} 1 & \text{if} \quad j = \arg\max_{j'} \ \log \pi_{j'} + \log W_p\left(\boldsymbol{x}_i | \mu_{j'}, \kappa_{j'}\right), \\ 0 & \text{otherwise.} \end{cases} \tag{26}$$

If the maximum is not unique, the category is assigned randomly to one of the leading categories.

A final choice is to use the so-called stochastic step (Celeux and Govaert 1992), or S-step, where the category is assigned at random for each observation i to one component j, with probability equal to its posterior probability $\beta_{i,j}$.

M-step: The M-step is defined by the maximization of the expected log-likelihood by determining, separately for each cluster j, the optimal parameters μ_j and κ_j, i.e., by selecting the pair (κ_j, μ_j) that maximizes the likelihood from $\left(\kappa_j^{(1)}, \mu_j^{(1)}\right)$ and $\left(\kappa_j^{(2)}, \mu_j^{(2)}\right)$, where:

$$\begin{aligned} \mathbf{S}^j = \frac{\sum_{i=1}^{n} \beta_{ij} \boldsymbol{x}_i \boldsymbol{x}_i^\top}{\sum_{i=1}^{n} \beta_{ij}}, \qquad \pi_j = \frac{1}{n} \sum_{i=1}^{n} \beta_{ij}, \\ \kappa_j^{(1)} = g^{-1}\left(1/2, p/2, \lambda_1^j\right), \qquad \kappa_j^{(2)} = g^{-1}\left(1/2, p/2, \lambda_p^j\right), \\ \mu_j^{(i)} = s_p^j \quad \text{if } \kappa_j^{(i)} > 0, \qquad \mu_j^{(i)} = s_1^j \quad \text{if } \kappa_j^{(i)} \leq 0, \quad \text{for } i = 1, 2, \end{aligned} \tag{27}$$

and $s_1^j, \ldots, s_p^j$ are the eigenvectors of $\mathbf{S}^j$ ordered such that the appertaining eigenvalues satisfy $\lambda_1^j \leq \lambda_2^j \leq \cdots \leq \lambda_p^j$.

Convergence check: Stop if the relative absolute change in the log-likelihood is smaller than a given threshold. Note that the calculation of log-likelihood is strongly connected with the E-step calculations, and thus the convergence is assessed during this procedure.

Diametrical clustering which acknowledges the unit norm and sign invariance

3.2. Diametrical clustering

Directional clustering, in particular the algorithm proposed in Dhillon, Marcotte, and Roshan (2003), is a well-known k-means type clustering algorithm in bioinformatics. The algorithm uses a non-parametric method and similarly as the EM algorithm, it is based on first picking the category that maximizes the squared scalar product (E-step), and then defining new concentration directions for each cluster (M-step). Compared to the usual k-means clustering algorithm, diametrical clustering takes into account the unit norm and is invariant to signs.

Sra and Karp (2013) showed that this algorithm is equivalent to the Watson EM algorithm with $\kappa_j \to \infty$, $\forall j = 1, \ldots, K$, which implies that $\beta_{ij} \to \{0, 1\}$. Equivalently, the authors showed that this can be also seen as a hard-assignment EM algorithm, where all κ parameters are equal and hence can be ignored. This suggests that it is natural to expect a better performance with the Watson mixture modeling (as the diametrical clustering can be seen as a special case), however, thanks to the robustness of the algorithm one may still use the diametrical clustering in the initialization phase as preprocessing before the main part of the EM starts.

The **watson** package offers such preprocessing of the model using the `init_iter` parameter in the `watson()` function, which will be discussed more in the next section. Alternatively, the user may also directly apply only directional clustering to the data using the `diam_clus()` function. This function takes a data matrix `x`, the number of clusters `k`, and the number of iterations `niter` as arguments, as shown in the following definition:

```
diam_clus(x, k, niter = 100)
```

3.3. Illustrative example

To illustrate the use of the Watson distribution to model data on the sphere we use the household data set from package **HSAUR3** (Hothorn and Everitt 2023). These data have been already estimated in directional statistics using the mixture of von Mises-Fisher distributions in Hornik and Grün (2014b). The point of this illustration is to show that if the data are distributed only in the positive orthant, then the Watson distribution is capable of doing the same work as a more specified von Mises-Fisher distribution. The use of the package to analyze axial data will be demonstrated in Section 6.

The data consist of the expenditures on four commodity groups of 20 single men and 20 single women. In the following, similarly as in Hornik and Grün (2014b), we will focus only on the expenditure on housing, foodstuffs and services, in order to be able to visualize the data and

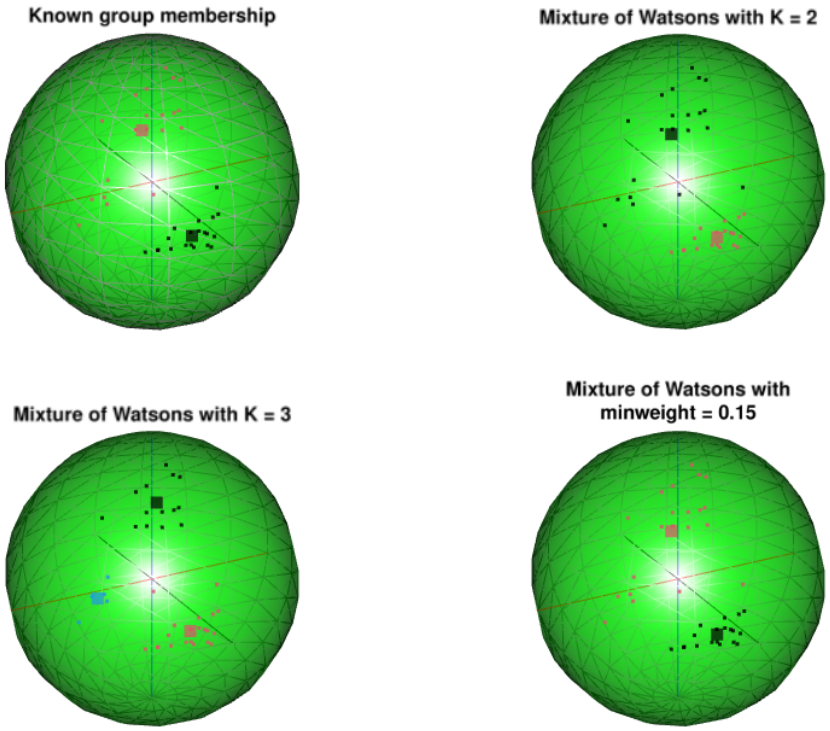


Figure 3: Household expenses data with gender indicated by color after projection to the sphere together with the estimated μ parameters at the top left, estimated mixtures of watson distributions with $K = 2$ and $K = 3$ at the top right and bottom left, respectively and results of an estimation if `minweight` is set to 0.15 at the bottom right.

	π	Housing	Food	Service	κ	BIC
$K = 2$	0.53	0.66	0.64	0.40	10.21	−144.49
	0.47	0.95	0.13	0.27	57.44	
$K = 3$	0.13	0.67	0.31	0.68	91.58	−156.04
	0.35	0.59	0.76	0.28	32.34	
	0.52	0.95	0.15	0.27	42.70	

Table 1: Estimated results of finite mixtures of Watson distributions to the household data.

the results. The data points are projected onto the sphere by normalizing them, and thus, in the following analysis we are interested in finding groups of households which have similar proportion of expenses rather than households that differ in their total absolute expenses.

During the fitting procedure, the gender information is not used and it is investigated if the finite mixtures can recognize the groups just by the angular similarities between the expenses.

If the model is estimated with the assumption of two underlying unobserved groups, the results deliver only one misclassification. This is the same result as in Hornik and Grün (2014b). The same holds for the situation where $K = 3$, which was also with the mixture of Watson distributions observed to be the model with the lowest BIC (see for example McLachlan and Peel 2000), with only $K = 1, \ldots, 4$ considered. Finally, if the model is estimated under the

condition that the every group must contain at least 15% of the data (`minweight = 0.15`) and the estimation starts with $K = 6$, the best model coincides with the one estimated using $K = 2$. The estimated parameters and the BIC value are given in Table 1. The R code for reproducing these results is provided in Section 4.3 after introducing package **watson**.

4. Software

4.1. `rmwat()`

The software implementation of the random variate generation methods from Section 2.1 can be obtained using the `rmwat()` function from the **watson** package.

```
rmwat(n, weights, kappa, mu, method = "acg", b = -10, rho = 1.1)
```

The main arguments are `n` for the desired sample size, `kappa` and `mu` for the concentration and mean direction parameters, respectively, and `method` to select the sampling algorithm. Additionally, `b` and `rho` control the performance of the ACG and Tinflex samplers. For a detailed description of all arguments, see the function documentation.

The samples from a two component mixture model with components having positive ($\kappa = (20, 20)$) and negative ($\kappa = (-200, -200)$) concentration parameters can be obtained as follows. The resulting samples are visualized in Figure 2 in Section 2.1.

```
R> library("watson")
R> sample1 <- rmwat(n = 2000, weights = c(0.5, 0.5), kappa = c(20, 20),
+    mu = matrix(c(1, 1, 1, -1, 1, 1), nrow = 3))
R> sample2 <- rmwat(n = 2000, weights = c(0.5, 0.5), kappa = c(-200, -200),
+    mu = matrix(c(1, 1, 1, -1, 1, 1), nrow = 3))
```

4.2. `watson()`

The main function of the **watson** package is the `watson()` function for fitting mixtures of Watson distributions. It can be called as:

```
watson(x, k, control = list(), ...)
```

The `watson()` function fits a mixture of Watson distributions to data using the EM algorithm. It takes a data matrix `x` and the number of components `k` as input. Several control parameters can be specified via the `control` argument, allowing customization of the estimation process. These include the type of E-step to use (`E`), the method for estimating concentration parameters in the M-step (`M`), the minimum prior probability for a component (`minweight`), and the number of EM runs to perform (`nruns`). For a comprehensive list of all control parameters and their descriptions, refer to the function documentation.

The object returned by `watson()` is of the '`watfit`' class with available methods `print()`, `coef()`, `logLik()` and `predict()` (yields either the component assignments or the matrix of a-posteriori probabilities). Finally, the function `diamclus()` for diametrical clustering is available.

4.3. Illustrative example: Household expenses

This section contains the code for reproducing the results of the presented example in the Section 3.3. First the data are loaded and the columns housing, foodstuff and service are extracted and stored in the variable `x`. Additionally, the gender classification is extracted and used to estimate the mean direction for both genders separately.

The function `watson()` is then used to estimate the mixture of Watson distributions with the number of components varying from 1 to 5 and the BIC values are reported. The last estimation sets the `minweight` parameter to 0.15, in order to avoid clusters with only insignificant number of elements is grouped together. For this estimation, the fitting is repeated 100 timed, to avoid sub-optima where EM algorithm could be trapped in some local optimum. This model is in addition printed out, with an achieved log-likelihood of 85.158 with respect to the uniform distribution on the sphere.

```
R> data("household", package = "HSAUR3")
R> x <- household[, c("housing", "food", "service")]
R> gender <- household$gender
R> wat <- lapply(1:4, function(K) watson(x, k = K))
R> sapply(wat, BIC)

[1] -111.2910 -144.4939 -156.0443 -147.1691

R> (watt <- watson(x, k = 6, minweight = 0.15, nruns = 100))

Fitted 2-components Watson mixture:

Weights: 0.4689717 0.5310283
Kappa: 57.43703 10.21159

Mu:
        clus_1    clus_2
[1,] 0.9545064 0.6639429
[2,] 0.1260827 0.6367097
[3,] 0.2702234 0.3921488

Log-likelihood: 85.15802,  Average log-likelihood: 2.12895

R> table(predict(watt), gender)

 gender
  female male
1     19    0
2      1   20
```

5. Numerical issues

In the following, we will use the notation from Section 2 and write

$$g(a,b,z) = \frac{\mathrm{d}\log(M(a,b,z)))}{\mathrm{d}z} = \frac{M(a,b,z)'}{M(a,b,z)} = \frac{a}{b}\frac{M(a+1,b+1,z)}{M(a,b,z)}. \tag{28}$$

As shown in Sections 2 and 3, computing ML estimation of concentration parameters for Watson distributions on $\mathbb{R}^p$ necessitates the solution κ of

$$g\left(a,b,\kappa\right) = r, \quad \text{where} \quad 0 < a < b, \text{ and } 0 \leq r \leq 1. \tag{29}$$

Furthermore, a computation of log-likelihoods or the a-posteriori probabilities in the mixture modeling requires to evaluate the expressions as $\log(M(a,b,z))$, where $0 < a < b$. Because $M(a,b,z) \to \infty$ as $z \to \infty$, a direct evaluation of $M(\cdot)$ with further logarithm or quotient function application is clearly not a good idea, mostly if we know that the fraction satisfies $0 < g(a,b,z) < 1$. In general, it can be observed that the Kummer's function easily over- or underflows for very general set of parameters just because of the geometric series structure the function embraces.

The above discussed problems were also the main reasons to develop the EM algorithm for **watson** from scratch, rather than rely on the existing general frameworks for mixture distributions, as for example **flexmix** (Grün and Leisch 2008). Having our own implementation allows us to focus on the numerical stability when working with the special functions to avoid possible overflows/underflows, while offering the high-performance of a C++-based implementation using **RcppArmadillo** (Eddelbuettel and Sanderson 2014).

5.1. Approximation of the Kummer's ratio

We assume $b > a > 0$, $z < 0$ and follow the approach used in Sablica and Hornik (2022). Define the sequences of real numbers through the recursive relation as

$$l^{(0)}_{a+n,b+n}(z) = \frac{2\left(a+n\right)}{\sqrt{\left(z-(b+n)\right)^2 + 4\left(a+n\right)z} - z + (b+n)}, \tag{30}$$

$$u^{(0)}_{a+n,b+n}(z) = 1 - \frac{2\left(b-a\right)}{\sqrt{\left(z+(b+n)+1\right)^2 - 4(b-a+1)z} + z + (b+n) - 1}, \tag{31}$$

with

$$l^{(m)}_{a+n,b+n}(z) = \frac{a+n}{b+n-z+zl^{(m-1)}_{a+n+1,b+n+1}(z)}, \quad u^{(m)}_{a+n,b+n}(z) = \frac{a+n}{b+n-z+zu^{(m-1)}_{a+n+1,b+n+1}(z)}. \tag{32}$$

The sequences $\left(l^{(0)}_{a,b}(z), l^{(1)}_{a,b}(z), l^{(2)}_{a,b}(z), \ldots\right)$ and $\left(u^{(0)}_{a,b}(z), u^{(1)}_{a,b}(z), u^{(2)}_{a,b}(z), \ldots\right)$ converge monotonically to $g(a,b,z)$ from below and above, respectively. For the proof see Sablica and Hornik (2022).

Using the equality $g(a,b,z) = 1 - g(b-a,b,-z)$ (another result from Sablica and Hornik 2022), the evaluation routine can be easily described by Algorithm 1.

The convergence of the irrational bounds for the values $a = 0.5$, $b = 50$ and $a = 99.5$, $b = 100$ is visualized in Figure 4. Note that the parameters were specifically chosen to be either close

Algorithm 1 Kummer's function algorithm

procedure KUMMER'S(a, b, z, N = number of iterations)
  if $z = 0$ **then return** a/b
  if $z < 0$ **then return** $\left(l_{a,b}^{(N)}(z) + u_{a,b}^{(N)}(z)\right)/2$
  if $z > 0$ **then return** $1 - \left(l_{b-a,b}^{(N)}(-z) + u_{b-a,b}^{(N)}(-z)\right)/2$

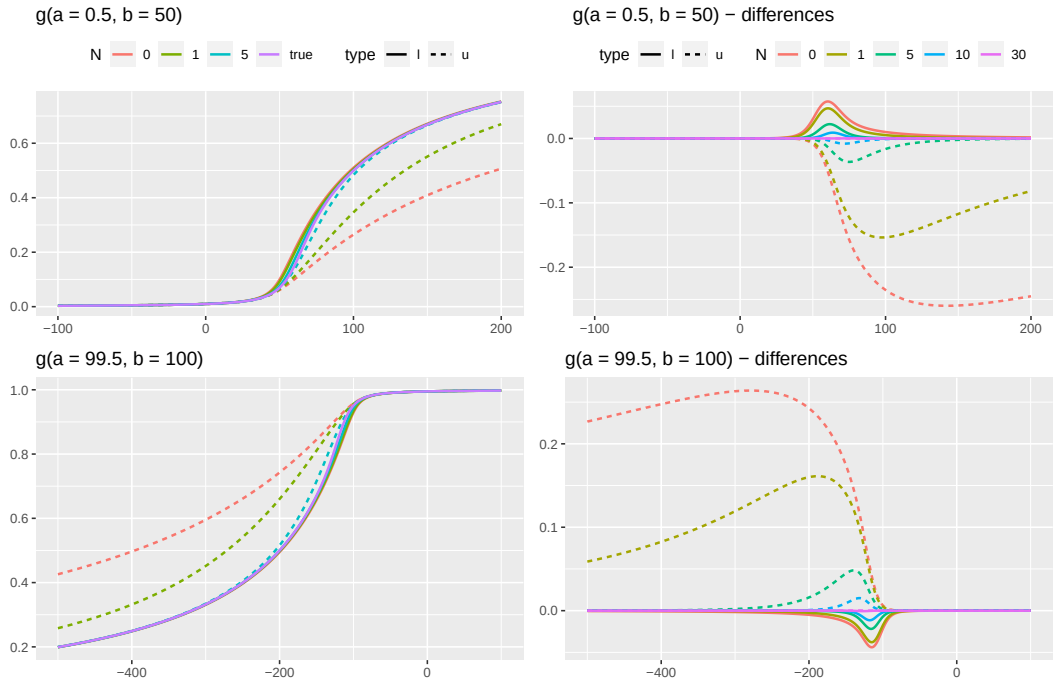


Figure 4: The convergence of the irrational bounds (left) and their relative differences from $g(a, b, z)$ (right) for the values $a = 0.5$, $b = 50$ (upper row) and $a = 99.5$, $b = 100$ (bottom row) Sablica and Hornik (2022).

or far away from each other as this makes the approximation more difficult, because the crossing point $z = 0$, where the bounds are exact, is either after or before the steepest part of the function, respectively. The "true" values were calculated using the Mathematica software (Wolfram Research, Inc. 2022) for higher precision. The plot with the differences indicates that even with the chosen parameters, after 5 iterations the irrational bounds are decently close and after 30 iterations almost exact. Furthermore, it should be pointed out that the iterations are composed only by four very simple arithmetic operations, thus in a compiled programming languages as for example C or C++ such estimation requires almost no time even for big N.

To solve the inverse of $g(a, b, z)$ one combines this evaluation with a bracketed type of Newton algorithm from Section 2. Note that

$$u_{a,b}^{(0)^{-1}}(r) = L(r), \quad u_{a,b}^{(1)^{-1}}(r) = B(r), \quad l_{a,b}^{(1)^{-1}}(r) = U(r) \tag{33}$$

(see Proposition 3 of Sablica and Hornik 2022) offers exactly such brackets for the starting value.

Even better, Sablica and Hornik (2024) showed that for $r > \frac{w_{a,b}^2 a + w_{a,b}(b-a)a}{(b+1)(b-a)+w_{a,b}^2 a}$, where $w_{a,b} = \frac{\sqrt{16ab+8a+1}+4a+1}{8a}$ the bound

$$L_2^{-1}(a,b,r) = \frac{rb-a}{r\,(1-r)} \frac{a\left(\gamma_{a,b}^\flat + \bar{\eta}_{a,b}(\gamma_{a,b}^\flat)\right) + br\left(\gamma_{a,b}^\flat - \bar{\eta}_{a,b}(\gamma_{a,b}^\flat)\right)}{2ab}, \tag{34}$$

with $\gamma_{a,b}^\flat = b + w_{a,b}$ and $\bar{\eta}_{a,b}(\gamma_{a,b}^\flat) = \frac{\gamma_{a,b}^\flat(a+b)-2ab}{b-a}$ is a better upper bound than ${u_{a,b}^{(1)}}^{-1}(r)$. Since the iteration is done on the negative reals, i.e., $r < a/b$, this motivates the following result.

Proposition 1. *Let $b > a > 0$ and define $L_2^{-1}(a,b,r)$ as in Equation 34. Then for $Q^{-1}(a,b,r) = -L_2^{-1}(b-a,b,1-r)$*

$$g^{-1}(a,b,r) > Q^{-1}(a,b,r) > {u_{a,b}^{(1)}}^{-1}(r) \quad \textit{for} \quad r < \frac{(b+1)a - w_{b-a,b}(b-a)a}{(b+1)a + w_{b-a,b}^2(b-a)}. \tag{35}$$

The proof is given in the appendix A.

Note that $Q^{-1}(a,b,r)$ is a lower bound on the whole $r < \frac{a}{b}$, however only for sufficiently small r dominates ${u_{a,b}^{(1)}}^{-1}(r)$.

Thus if r is sufficiently small or big, the bounds $Q^{-1}(a,b,r)$ and $L_2^{-1}(a,b,r)$ provide better bounds for the starting bracket. Exactly this idea is implemented in the `watson()` and is employed if `M` is set either to `"bisection"`, `"lognewton"` or `"newton"`.

5.2. Approximation of the Kummer's function logarithm

Since $g(a,b,z)$ is the logarithmic derivative of the Kummer's function $M(a,b,z)$, it can be also directly used to calculate its logarithm. For the following let $m(a,b,z) = \log(M(a,b,z))$ and $0 < a < b$. Since $M(a,b,0) = 1 \;\forall\; a,b$, the general formula can be written as

$$m(a,b,z) = m(a,b,0) + \int_0^z g(a,b,z)dz = \int_0^z g(a,b,z)dz, \tag{36}$$

which simplifies the problem to the integration of $g(a,b,z)$ on a compact set.

We note that, while this allows for simpler solutions, as for example the numerical integration of $g(\cdot)$, **watson** employs more controlled approaches, which will be discuses in the following sections.

Bounds for Kummer's function logarithm

This approach is based on the families of bounds $S_{\gamma,\eta}^+(a,b,l)$ and $S_{\gamma,\eta}^-(a,b,l)$ from Sablica and Hornik (2024) that can be obtained by integrating the family of possible bounds for $g(a,b,z)$ given as

$$B_{\gamma,\eta}(a,b,z) = \frac{\frac{a}{b}(\gamma+\eta)}{\sqrt{z^2 + 2\left(\frac{a}{b}(\gamma+\eta)-\eta\right)z + \gamma^2} - z + \eta}, \qquad \text{with } \gamma > 0, -\gamma < \eta \leq \gamma\frac{a+b}{b-a}. \tag{37}$$

For more details see Sablica and Hornik (2024).

This method defines easily accessible bounds for $m(a, b, z)$. What is more, the authors showed that if for both (upper and lower) bounds the subfamily with $\eta = (\gamma(a + b) - 2ab)/(b - a)$ is used, the error is at most $\frac{(\gamma_L - \gamma_U)}{2}\left(1 + \frac{a \log(\frac{a}{b})}{b-a}\right)$, where γ_L and γ_U are the values used for γ in the case of lower and upper bound, respectively. Evaluating this with the two most accurate bounds of the reference (i.e., $U(a, b, z)$ and $L_2(a, b, z)$) gives an absolute error at most equal to

$$\frac{\sqrt{16ab + 8a + 1} - 4a + 1}{16a}\left(1 + \frac{a \log(\frac{a}{b})}{b - a}\right). \tag{38}$$

With the bracketed term being smaller than 1, the overall term can get large only if b is relatively much larger than a. For the case of Watson distribution with $a = 1/2$ and $b = p/2$, the problematic term is equal to

$$\frac{\sqrt{16ab}}{16a} = \frac{\sqrt{b}}{4\sqrt{a}} = \frac{\sqrt{p}}{4}, \tag{39}$$

which gives a limiting error at most 5 even for 400-dimensional problems. We further note that as $z \to \infty$, the corresponding relative approximation error tends to zero.

This error can be even improved by integrating first over bounds that are more precise close to 0 and switching to the above bounds when they become superior. Such an approach additionally improves the maximal error by the difference of the bounds up to the crossing value. For more details see the bound $R_{max}(a, b, l)$ in Sablica and Hornik (2024). Note that while this form seems complicated, for a computer is this a trivial task as all the integrals attain a closed form.

The package **watson** uses the above defined techniques, more specifically the mean of bounds based on $U(a, b, z)$ for the upper bound and $R_{max}(a, b, z)$ for the lower bound to approximate $m(a, b, z)$, while controlling for the error. If the difference between the bounds is relatively "too big", the package will continue with the following ad-hoc method.

Ad-hoc method for the Kummer's function logarithm

Another method is to use the properties of $g(a, b, z)$ and rewrite $m(a, b, z)$ as a sum of parts which form it. This would still require one evaluation of the Kummer's function, but one can choose the parameters for which $M(a, b, z)$ either does not underflow or overflow. An example of such a case is, e.g., $M(a, b, z)$ where $a \leq 1$, $b < 2$ and z is negative. For such a set of parameters the evaluation using the GNU Scientific Library (GSL) (Gough 2009) does not underflow even for cases as $z = -1 \times 10^{300}$ and additionally the approximation error is negligible. This behavior is presented because the rising factorials of $M(a, b, z)$ do not increase fast enough. For the following recall that $a = 0.5$, the derivation for a more general case can be found in Sablica and Hornik (2022). Formally,

$$m(a, b, z) = z + m(b - a, b, -z), \tag{40}$$

where we applied the Kummer's identity $M(a, b, z) = e^z M(b - a, b, -z)$. Thus,

$$\begin{aligned} m(a, b, z) = z &+ \log(g(b - a - 1, b - 1, -z)) + \log(b - 1) - \log(b - a - 1) \\ &+ m(b - a - 1, b - 1, -z). \end{aligned} \tag{41}$$

Recursively rewriting the above expression yields

$$= z + \underbrace{\sum_{i=1}^{\lfloor b-a \rfloor_s} \left(\log(g(b-a-i, b-i, -z)) + \log(b-i) - \log(b-a-i)\right)}_{S} + m(b-a-\lfloor b-a \rfloor_s, b-\lfloor b-a \rfloor_s, -z), \tag{42}$$

where $\lfloor x \rfloor_s = \max_{m \in \mathbb{Z}} m < x$ is the strict floor operator. Hence

$$m(a,b,z) = \begin{cases} S + z + m(b-a-\lfloor b-a \rfloor_s, b-\lfloor b-a \rfloor_s, -z), & \text{if } z > 0, \\ S + m\left(a, b-\lfloor b-a \rfloor_s, z\right), & \text{if } z < 0, \end{cases} \tag{43}$$

where we again applied the Kummer's identity for the negative case. A performance of such procedure is visualized in Sablica and Hornik (2022).

This further allows to combine the previous method together with this one, where the only one required evaluation of $m(\cdot)$ in evaluated using the bounds presented in the previous section. Clearly out of the possible cases that can appear under the Watson settings, the worse scenario in terms of the error would require to evaluate $m(1/2, 1, r)$ with the limiting error Equation 38 at most ≈ 0.1.

This method is also implemented in the **watson** as the last case scenario. First using the algorithms and asymptotic formula implemented in GSL, the code tries to evaluate the function directly. If the resulting error value is not 0 (i.e., `GSL_SUCCESS`), the code tries to evaluate the transformed version using the Kummer's identity. If also this method fails (this is usually only for very large values, or parameters), the values are calculated using the presented bounds. Finally if the difference between the bounds is bigger than $2\log(1.02)$ (i.e., the true value of $M(a,b,z)$ could appear outside of the interval $[\theta/1.02, 1.02\theta]$, where θ is the calculated value), the above method is used which has not been observed to fail so far.

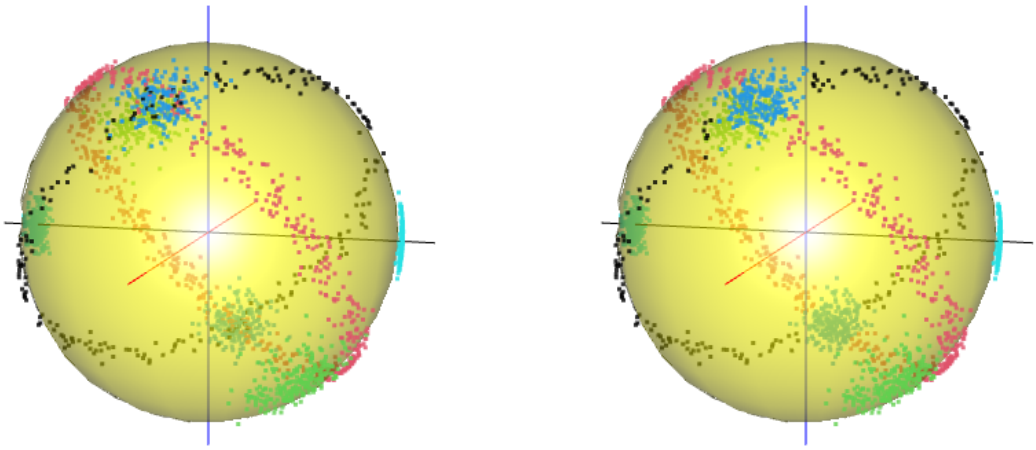
6. Application

This section demonstrates the functionality of the **watson** package through several examples. First, a simulation study is conducted to assess the performance of the EM algorithm in a controlled setting. Second, the package is applied to the New Zealand earthquake data to illustrate its use in a supervised setting with real-world axial data. Finally, the package is used for unsupervised clustering of depth images, showcasing its application in computer vision tasks. These examples highlight the versatility and practical utility of the **watson** package for analyzing spherical data.

6.1. Simulation study

In this section we present and illustrate the package and its EM algorithm on simulated data. Again we will concentrate on the case that can be visualized and so on the data with $p = 3$. These are simulated using the `rmwat()` function,

```
R> d <- rmwat(n = 2000, weights = c(0.1, 0.3, 0.2, 0.2, 0.2),
+    kappa = c(-200, -200, 30, 50, 100),
+    mu = matrix(c(1, 1, 1, -1, 1, 1, -1, -1, -1, 0, 1, -1, 1, 0, 0),
+      nrow = 3))
```



(a) True (b) Estimated

Figure 5: Estimation results.

where the parameters are chosen such that the clusters are overlapped while all of them being unique in the sense of the κ and μ parameter. As can be observed from the function call, the true simulated mixture consists of 5 component distributions, where 2 components are of the great circle shape (i.e., with $\kappa < 0$) and 3 with are concentrated in their mean direction.

```
R> model <- watson(d, 7, minweight = 0.02, nruns = 20)
R> model

Fitted 5-components Watson mixture:

Weights: 0.1965 0.1669577 0.1050926 0.213298 0.3181517
Kappa: 95.46819 51.58479 -186.3339 31.16424 -189.2505

Mu:
           clus_1       clus_2    clus_3    clus_4     clus_5
[1,] -0.999986605 -0.004850995 0.5707955 0.5779898 -0.5786764
[2,]  0.005164531 -0.701141010 0.5910836 0.5669366  0.5793756
[3,] -0.000343014  0.713006137 0.5699234 0.5869503  0.5739839

Log-likelihood: 3346.933,  Average log-likelihood: 1.673466
```

The model is estimated with the `watson()` function, where the number of clusters is on purpose misspecified to 7. To allow the algorithm to converge to the true constellation, the `minweight` parameter is set to 2%, removing dynamically components with weight smaller than the given value. The procedure is repeated 20 times with different random initial values and the best model is stored. Finally, the results are printed.

Despite the misspecification, similarly as in the first example, the algorithm converged to the true number of components. What is more, comparing the estimated parameters, one can

easily recognize the simulated distribution. The fitted model consists of two clusters with huge negative κ close to -200 and three components with positive κ close to 30, 50 and 100, respectively, which are exactly the parameters used to simulate the distribution. The achieved likelihood is 3346.933 with the average likelihood per data-point 1.67. Finally, the visualized results follow in Figure 5.

6.2. Supervised settings: New Zealand earthquake data

To illustrate the usage of **watson** with real data in the supervised setting (i.e., under the assumption of the associated labels of each data point being available for training), we consider the New Zealand earthquake data, which were recently analyzed by Arnold and Jupp (2013) and Fallaize and Kypraios (2016). Tectonic stress that occurs during the earthquake gives rise to a rupture on a fault plane (planar surface across which relative motion occurs, see Stein and Wysession (2009)). It is of the interest of the geologists and geophysicists to compare these faults, and by doing so to compare the earthquakes at different times or locations to analyze the possible similarities. The geometry of these faults is commonly described by three angles: strike, dip and slip (Stein and Wysession 2009), which can be transformed into a triplet of orthogonal axes known as the compressional, P, tensional, T, and null, B. This forms an orthogonal axial frame (set of r orthogonal axes $\{u_1, u_2, \ldots, u_r\}$ in $\mathbb{R}^p$) with $p = r = 3$ describing the main directions of the focal mechanics. We note, that the frame is of an axial structure, because of the freedom in the choice of the reference wall. While such data can be treated as a 3-dimensional axial frame as in Arnold and Jupp (2013), similarly as in Fallaize and Kypraios (2016), we will dedicate our analysis only to the null axis, i.e., we will consider the data of the form $p = 3$, $r = 1$.

The dataset can be composed by merging three smaller datasets, each containing data from a different place or time. The first two datasets contain 50 observation each, near Christchurch which were observed before and after an earthquake on 22 February 2011 (labeled as CE and CL). The third dataset consists of 32 observation obtained from the South Island. Geophysicists are interested weather the event on 22 February 2011 changed the pattern of the earthquakes close to Christchurch and weather this structure is similar to the one found in South Island.

Considering now the code of the analysis, the dataset is firstly created by merging the smaller data frames and the labels indicating the location of the earthquakes are extracted. The angles are then transformed into the strike, dip and slip angles (Arnold and Townend 2007), which are then used to generate the null axis for all the measurements (Stein and Wysession 2009). These are then stored in `null.RData`. The code for the data preprocessing can be found in the supplementary material. Finally, the estimation for all three groups is performed and the extracted labels, indicating the allocation for every data point to one of the clusters, are given using the `ids` parameter. We note that the estimation can be performed also in an unsupervised setting, however because of the strong overlap that is present in the data, the estimation procedure generates clusters with much higher likelihood as the true assignments with however no connection to them.

```
R> load("null.RData")
R> gg <- watson(b, ids = classif)
R> gg
```

```
Fitted 3-components Watson mixture:

Weights: 0.3787879 0.3787879 0.2424242
Kappa: 4.349961 3.772625 1.376732

Mu:
          clus_1      clus_2    clus_3
[1,]  0.00802186  0.04389806 0.3119964
[2,] -0.05382264 -0.01207179 0.3210766
[3,]  0.99851829  0.99896308 0.8941857

Log-likelihood: 75.79279,  Average log-likelihood: 0.5741878
```

This model can be then also compared with the model where the same structure of all three earthquakes is assumed. This is estimated again using the `ids` parameter, where all three datasets are assigned to the same cluster.

```
R> one <- watson(b, ids = rep("one", length(classif)))
R> one

Fitted 1-components Watson mixture:

Weights: 1
Kappa: 3.147332

Mu:
          clus_1
[1,] 0.062097858
[2,] 0.002016874
[3,] 0.998068028

Log-likelihood: 63.0042,  Average log-likelihood: 0.4773045
```

This allows to compare the observed log-likelihood differences to the distribution of the log-likelihood differences under the assumption of null-hypothesis that the three datasets arise from the same distribution. This can be easily estimated by using the parametric bootstrap and hence the `rmwat()` function, where we first sample from the simple model with only 1 cluster and estimate the log-likelihood differences for such a data set. This is then repeated 10000 times and the observed difference is compared with the simulated results.

```
R> B <- 10000
R> samples <- sapply(1:B, function(x) {
+    sample1 <- rmwat(132, 1, one$kappa_vector, one$mu_matrix)
+    model3 <- watson(sample1, ids = classif)
+    model1 <- watson(sample1, ids = rep("a", length(classif)))
+    logLik(model3) - logLik(model1)
+  })
R> sum(samples > logLik(gg) - logLik(one))/B

[1] 4e-04
```

The results suggest similar observations as were obtained in the previous literature. Comparing the results from the Christchurch clusters with the South Island results indicates a much stronger difference than between the Christchurch datasets. The concentration of the earthquakes is much smaller in the case of South Island and the euclidean distance between the mean directions of first Christchurch cluster and South Island cluster is ≈ 0.49. The authors in Arnold and Jupp (2013) obtained in their $p = r = 3$ setting from a test on the equality p-value smaller than 0.001, which coincides with the estimated p-value (0.0004) from our experiments.

The clusters next to Christchurch do not show any strong evidence in the difference of the estimated parameters. Both clusters have a concentration parameter κ close to 4 with the Euclidean norm of the mean direction difference equal to ≈ 0.055. For this comparison, Arnold and Jupp (2013) obtained a p-value 0.89, which again agrees with the results that can be obtained using **watson** (0.83). The whole analysis for only these two datasets and the resulting p-value is available in appendix B.

6.3. Unsupervised settings: Depth image clustering

The expanding use of cameras in daily life and also in many scientific disciplines has attracted a decent amount of attention to the research areas like image processing, robotics or computer vision. These interests have even gone bigger together with the capabilities of cameras to produce a high quality color images. However, while the quality of images is still on an increasing path, the amount of information that can be extracted from such a projection of a 3D space on a 2D space has its natural bound. To overcome this limitations, some cameras (mostly in the gaming and scientific industries, e.g., Microsoft Kinect sensors) have integrated sensors to explore the geometric structure of the surrounding using the so called depth images. The depth image assigns to every pixel the values that represent the distance of the object from a viewpoint. While the amount of information that these images contain is also limited, depth images excel in recognition of long planar surfaces, which is many times a difficult task if only an RGB picture is analyzed, see Hasnat (2014). This allows to use depth images and the features extracted from it as additional components in the analysis of the color images, which together forms the RGB-D image analysis.

The recognition of the planar surfaces is commonly deployed using the surface normal (see, e.g., Silberman, Hoiem, Kohli, and Fergus (2012)), which is a 3-dimensional unit vector representing the normal to the pixel of interest and the pixels in the neighbourhood that fall under the prespecified threshold when comparing the depth values. The sample space of these normals is a 3D sphere, which makes the distributions from the directions statistics a great choice when analyzing these vectors. Even better, due to its axial symmetry, the Watson distribution can better handle the noise in the surface normals (Rusu 2013), making it a superior choice over the usual choices as for example the von Mises-Fisher distribution.

To illustrate the usage of **watson** in the unsupervised setting, we consider the NYU Depth Dataset V2 (Silberman *et al.* 2012), which consists of a collection of color and depth images obtained using Microsoft Kinect cameras. The surface normals were extracted using the toolbox of the NYU database. In fact, the Watson distribution has been already used to cluster this database (see Hasnat, Alata, and Trémeau 2014), however the model were estimated with a slightly different procedure with hierarchical clustering applied to generate the submodels. In this example we illustrate how easily the surface normals can be clustered using the **watson** package, allowing the R users to process these types of images just by using few lines of code.

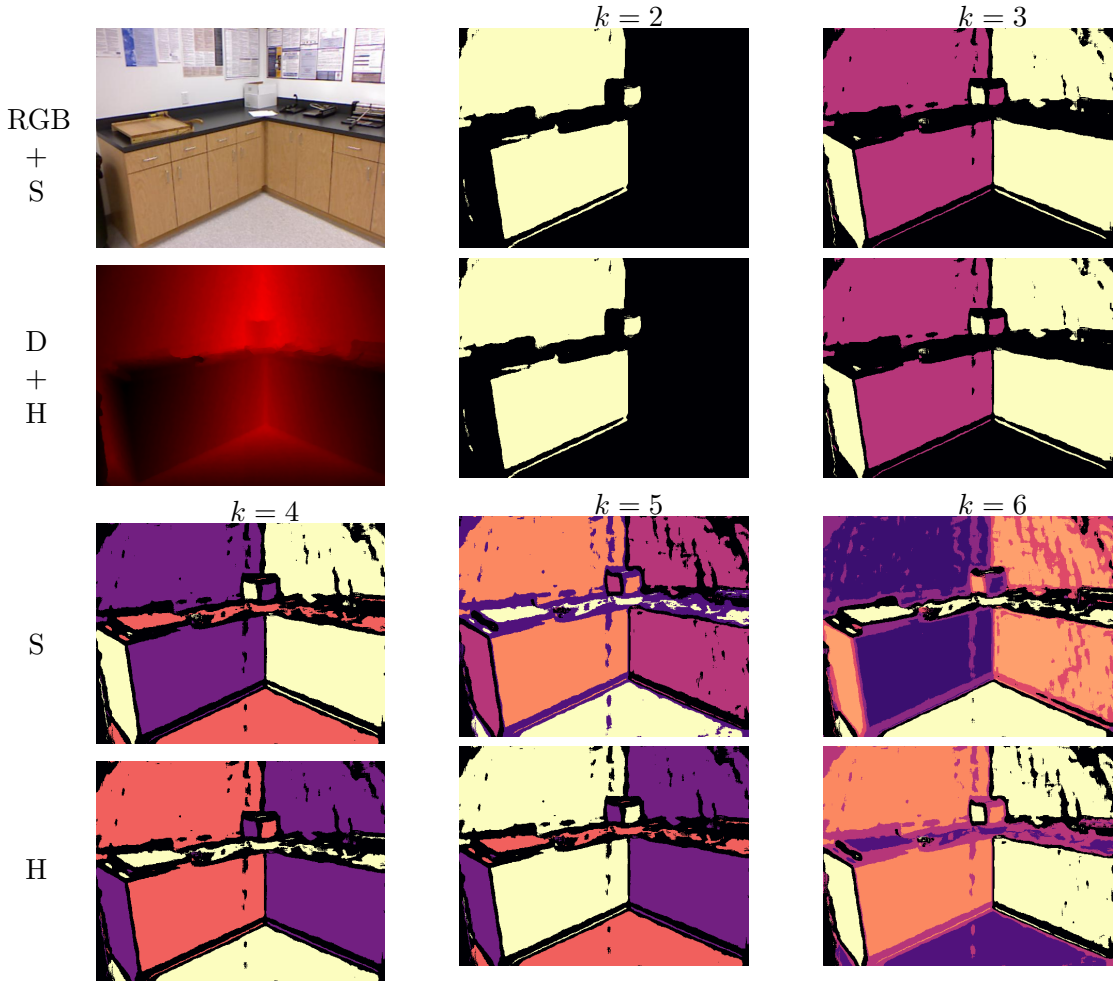


Figure 6: The figure shows the estimate results. First two pictures show the true RGB picture (not used in the analysis) and the depth picture from which the surfaces were estimated. The black color has been always assigned to the components with the smallest concentration in the absolute value, i.e., the component that collects the elements that do not belong to the dominant clusters. For more analyzed images, see Appendix C.

First all additional packages are loaded. Packages **grid** (R Core Team 2025) and **viridis** (Garnier 2024) will be used to visualize the results while the **parallel** (R Core Team 2025) package is loaded in order to estimate the models with different component numbers in parallel. After the data are loaded, the `grid.raster()` function is used to generate images representing both the RGB image and the depth image on a red scale. The RGB image is rendered using appropriate color channels, while the depth image is scaled to a red gradient. Each picture from the NYU Depth Dataset V2 consists of 427×561 pixels, giving together 239547 surface normals per picture to analyze.

The surface normals are then clustered in parallel using 3 parallel units for models with component numbers ranging from 2 to 7. Each estimation performs 100 runs, after which the run with best likelihood is returned. Additionally, the parameter `minweight` is again specified to avoid clusters with weight smaller than 0.05. After the estimation, the data-set

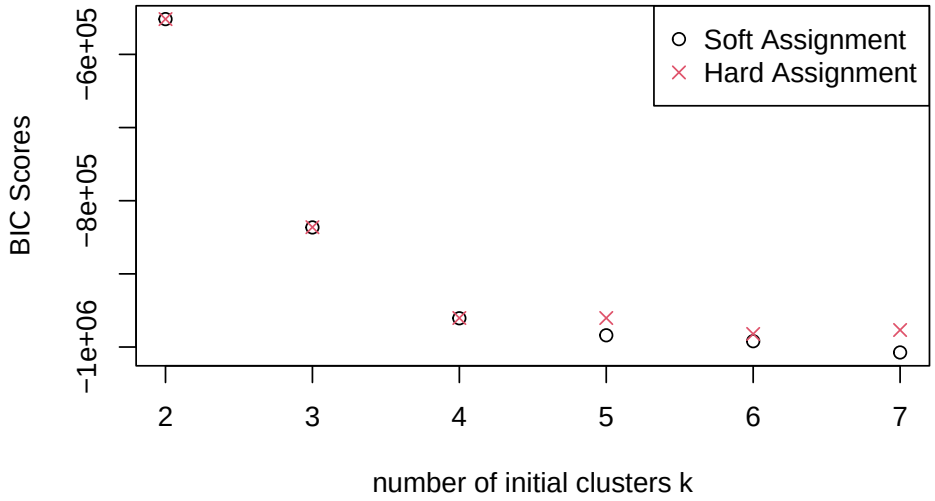


Figure 7: BIC scores of the models ranging from $k = 2$ to $k = 7$. Circles indicate the scores in the case of the soft-assignment and crosses in the case of hard-assignment.

is erased from all the models to avoid overflow on memory if multiple models for multiple pictures are stored. Next the categories for all pixels are predicted, the results are visualized and BIC scores are extracted. This procedure is then repeated with the E argument equal to `"hardmax"`, indicating the algorithm to perform the hard clustering. Finally, the plot with BIC scores is stored. The code for the analysis, including data preprocessing and visualization, is available in the replication materials. The key parts of the analysis are shown below:

```
R> watrun <- function(i, a) {
+     w <- watson(a, k = i, minweight = 0.05, nruns = 100, verbose = TRUE)
+     w$data <- NULL
+     w
+  }
R> watrunhard <- function(i, a) {
+    w <- watson(a, k = i, E = "hardmax", minweight = 0.05,
+      nruns = 100, verbose = TRUE)
+    w$data <- NULL
+    w
+  }
R> cl <- makeCluster(3, outfile = "progress.txt")
R> clusterExport(cl = cl, list("watson"))
R> parallel::clusterSetRNGStream(cl = cl, 1)
R> B <- parLapply(cl, 2:7, watrun, a)
R> stopCluster(cl)
R> cl <- makeCluster(3, outfile = "progress.txt")
R> clusterExport(cl = cl, list("watson"))
R> parallel::clusterSetRNGStream(cl = cl, 1)
R> B <- parLapply(cl, 2:7, watrunhard, a)
R> stopCluster(cl)
```

The picture indicates unsurprising behavior. With a small number of components the algorithm tends to pick the dominant surfaces in the picture and concentrate on them. In our observations, the resulting parameter setup always tends to have one cluster with small concentration parameter, which thus behaves as a uniform distribution and hence clusters the elements that have not been assigned to the dominant clusters in some specific directions. This results has been also observed in Hasnat (2014). In our analysis the black color was always assigned to this cluster, while the other colors were assigned by random.

The BIC scores can be then again used to detect the optimal number of components using the kink plot. Similarly as with other pictures the hard and soft clustering tends to agree on likelihood up to the point where the models start to overfit. Here by observing the pictures the hard-assignment performs slightly better as it tends to use more the ability to remove non-significant clusters and hence to climb down to the true number of planar surfaces. For the image analyzed above the BIC scores and also the attached pictures indicate that after $k = 4$ the models tends to overfit and fails to perfectly recognize the planar surfaces as one element. For more discussion on the selection of the optimal k, see Hasnat (2014).

7. Conclusion

In this paper we presented and showcased the R package **watson**. We first introduced the random sampling function for (mixtures of) Watson distributions that is provided by the package in the form of two algorithms developed in Sablica *et al.* (2025). In addition to these two methods, the package also provides an automated selection of the faster algorithm and hyper-parameters.

The second significant contribution of the **watson** package is the `watson()` function, which provides an EM algorithm to fit a mixture of Watson distributions to a given data set on the sphere. Special focus has been given on numerical problems that arise from the evaluation of the likelihood as well as from the ML estimation of the concentration parameter.

All this was then demonstrated on multiple examples. In particular, we showed how the `minweight` parameter can be used to recognize the true amount of clusters even in a misspecified model. Just by using a few simple calls, we estimated the mixture of the Watson distributions to the axial New Zealand earthquake data to recognize similar results as has been observed in the literature. Finally, we showed how **watson** can be just in a few extra lines of code adapted to an algorithms for clustering of depth images and for the recognition of the surface normals.

A possible extension for package **watson** would be to extend the scope to a more general distributions, as for example the Bingham distribution (Bingham 1974). However, this would require solutions to numerical problems substantially more complicated than the ones for the Watson distribution presented here.

Acknowledgments

The authors are grateful to Richard Arnold and Peter Jupp for providing the New Zealand earthquake data.

References

Arnold R, Jupp PE (2013). "Statistics of Orthogonal Axial Frames." *Biometrika*, **100**(3), 571–586. doi:10.1093/biomet/ast017.

Arnold R, Townend J (2007). "A Bayesian Approach to Estimating Tectonic Stress from Seismological Data." *Geophysical Journal International*, **170**(3), 1336–1356. doi:10.1111/j.1365-246x.2007.03485.x.

Best D, Fisher N (1986). "Goodness-of-Fit and Discordancy Tests for Samples from the Watson Distribution on the Sphere." *Australian Journal of Statistics*, **28**, 13–31. doi:10.1111/j.1467-842x.1986.tb00580.x.

Bijral AS, Breitenbach M, Grudic G (2007). "Mixture of Watson Distributions: A Generative Model for Hyperspherical Embeddings." In M Meila, X Shen (eds.), *AISTATS 2007: Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2, pp. 35–42. San Juan. URL http://jmlr.org/proceedings/papers/v2/bijral07a/bijral07a.pdf.

Bingham C (1974). "An Antipodally Symmetric Distribution on the Sphere." *The Annals of Statistics*, **2**(6), 1201–1225. doi:10.1214/aos/1176342874.

Botts C, Hörmann W, Leydold J (2012). "Transformed Density Rejection With Inflection Points." *Statistics and Computing*, **23**(2), 251–260. doi:10.1007/s11222-011-9306-4.

Celeux G, Govaert G (1992). "A Classification EM Algorithm for Clustering and Two Stochastic Versions." *Computational Statistics & Data Analysis*, **14**(3), 315–332. doi:10.1016/0167-9473(92)90042-e.

Dhillon IS, Marcotte EM, Roshan U (2003). "Diametrical Clustering for Identifying Anti-Correlated Gene Clusters." *Bioinformatics*, **19**(13), 1612–1619. doi:10.1093/bioinformatics/btg209.

Eddelbuettel D, Sanderson C (2014). "**RcppArmadillo**: Accelerating R with High-Performance C++ Linear Algebra." *Computational Statistics & Data Analysis*, **71**, 1054–1063. doi:10.1016/j.csda.2013.02.005.

Fallaize CJ, Kypraios T (2016). "Exact Bayesian Inference for the Bingham Distribution." *Statistics and Computing*, **26**(1–2), 349–360. doi:10.1007/s11222-014-9508-7.

Garnier S (2024). ***viridis**: Colorblind-Friendly Color Maps for R*. doi:10.32614/CRAN.package.viridis. R package version 0.6.5.

Gautschi W (1977). "Anomalous Convergence of a Continued Fraction for Ratios of Kummer Functions." *Mathematics of Computation*, **31**(140), 994–999. doi:10.2307/2006129.

Gilks WR, Wild P (1992). "Adaptive Rejection Sampling for Gibbs Sampling." *Applied Statistics*, **41**(2), 337–348. doi:10.2307/2347565.

Gough B (2009). *GNU Scientific Library Reference Manual*. 3rd edition. Network Theory.

Grün B, Leisch F (2008). "FlexMix Version 2: Finite Mixtures with Concomitant Variables and Varying and Constant Parameters." *Journal of Statistical Software*, **28**(4), 1–35. doi:10.18637/jss.v028.i04.

Hasnat A (2014). *Unsupervised 3D Image Clustering and Extension to Joint Color and Depth Segmentation*. Ph.D. thesis, Jean Monnet University Saint-Etienne.

Hasnat MA, Alata O, Trémeau A (2014). "Unsupervised Clustering of Depth Images Using Watson Mixture Model." *2014 22nd International Conference on Pattern Recognition*, pp. 214–219. doi:10.1109/icpr.2014.46.

Hornik K, Grün B (2014a). "On Maximum Likelihood Estimation of the Concentration Parameter of Von Mises-Fisher Distributions." *Computational Statistics*, **29**(5), 945–957. doi:10.1007/s00180-013-0471-0.

Hornik K, Grün B (2014b). "**movMF**: An R Package for Fitting Mixtures of Von Mises-Fisher Distributions." *Journal of Statistical Software*, **58**(10), 1–31. doi:10.18637/jss.v058.i10.

Hothorn T, Everitt BS (2023). ***HSAUR3**: A Handbook of Statistical Analyses Using R (3rd Edition)*. doi:10.32614/CRAN.package.HSAUR3. R package version 1.0-14.

Kent JT, Ganeiber AM, Mardia KV (2018). "A New Unified Approach for the Simulation of a Wide Class of Directional Distributions." *Journal of Computational and Graphical Statistics*, **27**(2), 291–301. doi:10.1080/10618600.2017.1390468.

Leydold J, Botts C, Hörmann W (2019). ***Tinflex**: A Universal Non-Uniform Random Number Generator*. doi:10.32614/CRAN.package.Tinflex. R package version 1.5.

Li KH, Wong CKF (1993). "Random Sampling from the Watson Distribution." *Communications in Statistics – Simulation and Computation*, **22**(4), 997–1009. doi:10.1080/03610919308813139.

Mardia KV, Jupp PE (2009). *Directional Statistics*, volume 494. John Wiley & Sons.

McLachlan GJ, Peel D (2000). *Finite Mixture Models*. John Wiley & Sons, New York.

National Institute of Standards and Technology (2023). "NIST Digital Library of Mathematical Functions." Version 1.0.19, URL https://dlmf.nist.gov/.

R Core Team (2025). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. doi:10.32614/R.manuals. URL https://www.R-project.org/.

Rusu RB (2013). *Semantic 3D Object Maps for Everyday Robot Manipulation*. Springer-Verlag. doi:10.1007/978-3-642-35479-3.

Sablica L, Hornik K (2022). "On Bounds for Kummer's Function Ratio." *Mathematics of Computation*, **91**, 887–907. doi:10.1090/mcom/3690.

Sablica L, Hornik K (2024). "Family of Integrable Bounds for the Logarithmic Derivative of Kummer's Function." *Journal of Mathematical Analysis and Applications*, **537**(1), 128262. doi:10.1016/j.jmaa.2024.128262.

Sablica L, Hornik K, Leydold J (2025). "Efficient Sampling from the Watson Distribution in Arbitrary Dimensions." *Journal of Computational and Graphical Statistics*, **34**(3), 923–933. doi:10.1080/10618600.2024.2416521.

Sablica L, Hornik K, Leydold J (2026). ***watson**: Fitting and Simulating Mixtures of Watson Distributions*. doi:10.32614/CRAN.package.watson. R package version 1.0.0.

Saw JG (1978). "A Family of Distributions on the m-Sphere and Some Hypothesis Tests." *Biometrika*, **65**(1), 69–73. doi:10.2307/2335278.

Silberman N, Hoiem D, Kohli P, Fergus R (2012). "Indoor Segmentation and Support Inference from RGBD Images." In *European Conference on Computer Vision*, pp. 746–760. Springer-Verlag.

Sra S (2007). *Matrix Nearness Problems in Data Mining*. Ph.D. thesis, The University of Texas at Austin. AAI3277669.

Sra S, Karp D (2013). "The Multivariate Watson Distribution: Maximum-Likelihood Estimation and Other Aspects." *Journal of Multivariate Analysis*, **114**, 256–269. doi:10.1016/j.jmva.2012.08.010.

Stein S, Wysession M (2009). *An Introduction to Seismology, Earthquakes, and Earth Structure*. John Wiley & Sons.

Watson GS (1965). "Equatorial Distributions on a Sphere." *Biometrika*, **52**(1/2), 193–201. doi:10.2307/2333824.

Wolfram Research, Inc (2022). "Mathematica, Version 12.0." Champaign, URL https://www.wolfram.com/mathematica/.

A. Proofs

Proof of Proposition 1. Let $r' > \frac{w_{a',b}^2 a' + w_{a',b}(b-a')a'}{(b+1)(b-a')+w_{a',b}^2 a'}$, then it holds that

$$g^{-1}(a', b, r') < L_2^{-1}(a', b, r') < {u_{a',b}^{(1)}}^{-1}(r') \tag{44}$$

(Sablica and Hornik 2024). Thus let $a = b - a'$ and $r = 1 - r'$, from which

$$-g^{-1}(b-a, b, 1-r) > Q^{-1}(a, b, r) = -L_2^{-1}(b-a, b, 1-r) > -{u_{b-a,b}^{(1)}}^{-1}(1-r) \tag{45}$$

$$\text{for} \quad r < 1 - \frac{w_{b-a,b}^2(b-a) + w_{b-a,b}(b-a)a}{(b+1)a + w_{b-a,b}^2(b-a)} = \frac{(b+1)a - w_{b-a,b}(b-a)a}{(b+1)a + w_{b-a,b}^2(b-a)}. \tag{46}$$

Additionally, from the form of

$${u_{a,b}^{(1)}}^{-1}(r) = \frac{br - a}{2r(1-r)} \left(1 + \sqrt{1 + \frac{4\,(b+1)\,r\,(1-r)}{a\,(b-a)}}\right), \tag{47}$$

it is easy to observe that ${u_{a,b}^{(1)}}^{-1}(r) = -{u_{b-a,b}^{(1)}}^{-1}(1-r)$, which is true also for $g(\cdot)$, as it holds that $g(a, b, z) = 1 - g(b-a, b, -z)$ (see Sablica and Hornik 2022) and thus $g^{-1}(a, b, r) = -g^{-1}(b-a, b, 1-r)$. This completes the proof. □

B. New Zealand earthquake data

```
R> c <- b[classif == "CE" | classif == "CL",]
R> classifi <- classif[classif == "CE" | classif == "CL"]
R> gg2 <- watson(c, ids = classifi)
R> one2 <- watson(c, ids = rep("one", length(classifi)))
R> samples2 <- sapply(1:B, function(x) {
+    sample1 <- rmwat(100, 1, one2$kappa_vector, one2$mu_matrix)
+    model3 <- watson(sample1, ids = classifi)
+    model1 <- watson(sample1, ids = rep("one", length(classifi)))
+    logLik(model3) - logLik(model1)
+  })
R> sum(samples2 > logLik(gg2) - logLik(one2))/B

[1] 0.8305
```

C. Depth image clustering: Results

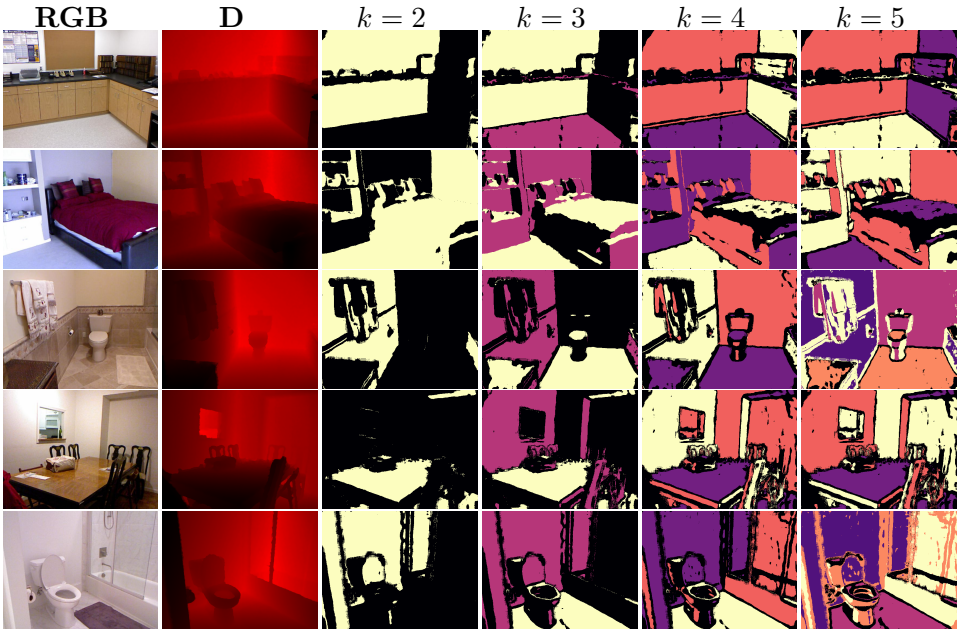


Figure 8: Estimated results using the hard-clustering for the other selected pictures.

Affiliation:

Lukas Sablica, Kurt Hornik, Josef Leydold
Institute for Statistics and Mathematics
WU Wirtschaftsuniversität Wien
1020 Vienna, Austria
E-mail: Lukas.Sablica@wu.ac.at, Kurt.Hornik@wu.ac.at, Josef.Leydold@wu.ac.at

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/
December 2025, Volume 115, Issue 4 *Submitted:* 2024-11-20
doi:10.18637/jss.v115.i04 *Accepted:* 2025-02-24