Journal of Statistical Software

October 2021, Volume 99, Issue 11. doi: 10.18637/jss.v099.i11

BAMBI: An R Package for Fitting Bivariate Angular Mixture Models

Saptarshi Chakraborty
State University of New York at Buffalo

Samuel W. K. Wong
University of Waterloo

Abstract

Statistical analyses of directional or angular data have applications in a variety of fields, such as geology, meteorology and bioinformatics. There is substantial literature on descriptive and inferential techniques for univariate angular data, with the bivariate (or more generally, multivariate) cases receiving more attention in recent years. More specifically, the bivariate wrapped normal, von Mises sine and von Mises cosine distributions, and mixtures thereof, have been proposed for practical use. However, there is a lack of software implementing these distributions and the associated inferential techniques. In this article, we introduce **BAMBI**, an R package for analyzing bivariate (and univariate) angular data. We implement random data generation, density evaluation, and computation of theoretical summary measures (variances and correlation coefficients) for the three aforementioned bivariate angular distributions, as well as two univariate angular distributions: the univariate wrapped normal and the univariate von Mises distribution. The major contribution of **BAMBI** to statistical computing is in providing Bayesian methods for modeling angular data using finite mixtures of these distributions. We also provide functions for visual and numerical diagnostics and Bayesian inference for the fitted models. In this article, we first provide a brief review of the distributions and techniques used in **BAMBI**, then describe the capabilities of the package, and finally conclude with demonstrations of mixture model fitting using **BAMBI** on the two real data sets included in the package, one univariate and one bivariate.

Keywords: angular data, mixture models, bivariate data, von Mises distribution, wrapped normal distribution, R, Hamiltonian Monte Carlo, MCMC, Gibbs sampler.

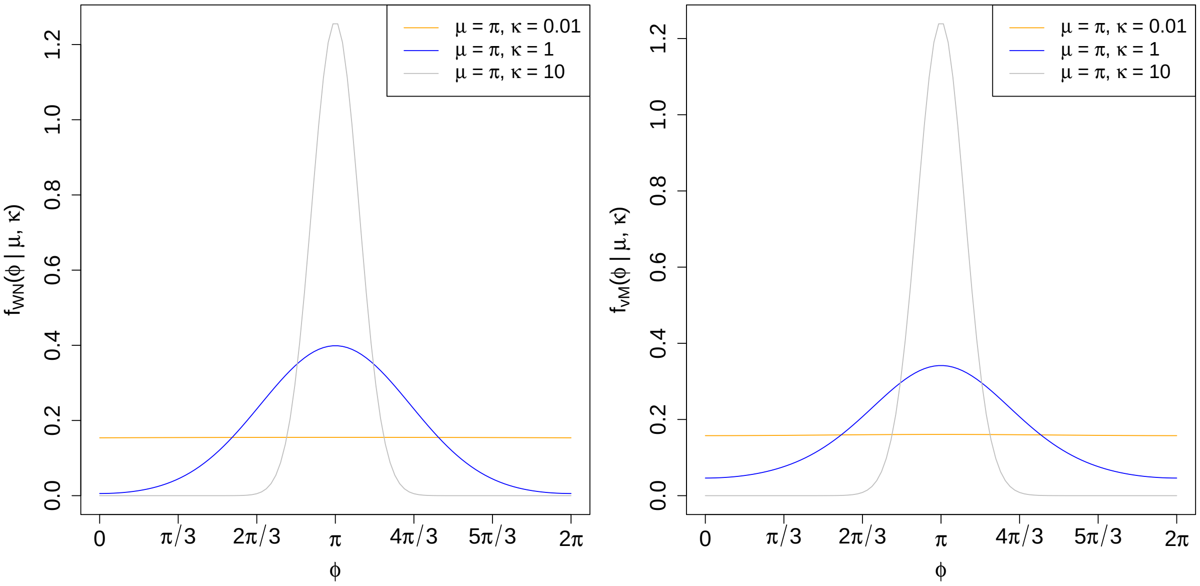
1. Introduction

Statistical analyses of angular or directional data have found applications in a variety of fields, such as geology (Earth's magnetic poles), meteorology (wind directions) and bioinformatics (backbone structures of proteins). Directional data can be univariate or multivariate, and one way of representing such data is via angles measured on a circle $[0, 2\pi)$ (element-wise when multivariate), and hence the name *angular*. Angular methods are also applicable to any interval that wraps around (e.g., $[0, L)$ or $[-L/2, L/2)$ for some $L > 0$) when transformed to the circle $[0, 2\pi)$. The wraparound condition on the support invalidates direct applicability of many standard statistical methods. There is substantial literature devoted to the development of descriptive and inferential techniques for directional data (see, e.g., Mardia and Jupp (2009); Mardia (1972); Fisher (1995)), with the traditional univariate case as the primary focus, although the bivariate case is gaining increasing interest (Singh, Hnizdo, and Demchuk 2002; Mardia, Taylor, and Subramaniam 2007) along with the emergence of new applications. Bivariate angular data can now be found in a variety of modern scientific problems, with many notable applications arising from the field of computational biology (Mardia *et al.* 2007; Boomsma, Mardia, Taylor, Ferkinghoff-Borg, Krogh, and Hamelryck 2008; Lennox, Dahl, Vannucci, and Tsai 2009; Bhattacharya and Cheng 2015). A major area of research in protein bioinformatics involves modeling and predicting protein 3-D structures, which requires proper handling of the paired backbone torsion angles. Formal analyses of these bivariate angle pairs thus require rigorous statistical techniques and models.

A unique feature in the modeling of directional data is the use of angular probability distributions, or mixtures thereof (see Section 1.4), which are inherently different from their linear (Euclidean) counterparts because of the wraparound nature of their supports. Bayesian methods provide flexible tools for analyzing and modeling such data. First, one may incorporate prior information, if available, into modeling. Second, one may use powerful computational methods, i.e., Markov chain Monte Carlo (MCMC, see Section 2.2) for sampling from the posterior, to fit such models and assess the fitted models. Third, one may readily compute posterior quantities of interest while coherently accounting for uncertainty in the model parameters. Within this context, this package was developed for fitting **B**ivariate **A**ngular **M**ixtures using **B**ayesian **I**nference (hence, the package name **BAMBI**). In **BAMBI** we implement the two most popular angular distributions, namely the wrapped normal (or Gaussian) and the von Mises distributions, and consider both univariate and bivariate versions of these. **BAMBI** provides functionality for modeling univariate and bivariate angular data using these distributions, and for fitting finite mixture models of these distributions. Package **BAMBI** (Chakraborty and Wong 2021) is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=BAMBI`. We first introduce the basics of these distributions and mixture models. It should be noted that the bivariate distributions considered in this paper have support $[0, 2\pi)^2$ (i.e., on a *torus*), which are distinct from those defined on the surface of the *unit sphere*, such as the von Mises-Fisher distribution.

1.1. Wrapped normal distributions

For univariate continuous data, the angular analogue of the normal distribution on the real line is the wrapped normal distribution obtained by *wrapping* a normal random variable around the unit circle (see, e.g., Jona-Lasinio, Gelfand, and Jona-Lasinio (2012)). Formally, let X be a normal random variable with mean μ and variance $\sigma^2 > 0$. Then the distribution of $\psi = X \mod 2\pi$ is called the *wrapped normal distribution* with mean μ and variance σ^2



(a) Wrapped normal density.

(b) Von Mises density.

Figure 1: Univariate wrapped normal density $f_{\mathrm{WN}}(\phi|\mu,\kappa)$ and univariate von Mises density $f_{\mathrm{vM}}(\phi|\mu,\kappa)$ with $\mu = \pi$ and different κ's.

and is denoted by $\mathrm{WN}(\mu, \sigma^2)$. The density of $\psi \sim \mathrm{WN}(\mu, \sigma^2)$ is given by:

$$f_{\mathrm{WN}}(\psi|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \sum_{\omega \in \mathbb{Z}} \exp\left[-\frac{1}{2\sigma^2}(\psi - \mu - 2\pi\omega)^2\right]; \quad \psi \in [0, 2\pi) \tag{1}$$

where $\mathbb{Z}$ denotes the set of all integers. Since the density contains a summation over entire $\mathbb{Z}$, without loss of generality, we let $\mu \in [0, 2\pi)$ to ensure identifiability. Figure 1(a) displays the univariate wrapped density with $\mu = \pi$ and $\kappa = 0.01, 1$ and 10, which shows that the density is symmetric around μ and becomes more concentrated as κ increases.

The multivariate generalization of the above distribution is straightforward (Jona-Lasinio *et al.* 2012). The distribution of a random vector $\boldsymbol{\psi} = (\psi_1, \ldots, \psi_p)^\top$ with probability density

$$\frac{1}{\sqrt{|\Sigma|(2\pi)^p}} \sum_{\boldsymbol{\omega} \in \mathbb{Z}^p} \exp\left[-\frac{1}{2}\left(\boldsymbol{\psi} - \boldsymbol{\mu} - 2\pi\boldsymbol{\omega}\right)^\top \Sigma^{-1}\left(\boldsymbol{\psi} - \boldsymbol{\mu} - 2\pi\boldsymbol{\omega}\right)\right]; \quad \boldsymbol{\psi} \in [0, 2\pi)^p \tag{2}$$

with $\boldsymbol{\mu} \in [0, 2\pi)^p$ and Σ positive definite, is called the p-variate wrapped normal distribution with mean vector $\boldsymbol{\mu}$ and variance matrix Σ, denoted by $\mathrm{WN}_p(\boldsymbol{\mu}, \Sigma)$. Although (1) and (2) are the most common parameterizations of the wrapped normal distributions found in the literature, to facilitate comparability with the von Mises distribution (defined in Section 1.2), we shall use the equivalent representation(s) obtained through the re-parameterization(s) $\kappa = 1/\sigma^2$ and $\Delta = \Sigma^{-1}$. **BAMBI** handles the univariate and bivariate cases, namely $p = 1$ and $p = 2$. Thus, the form of the univariate wrapped normal density we use is

$$f_{\mathrm{WN}}(\psi|\mu,\kappa) = \sqrt{\frac{\kappa}{2\pi}} \sum_{\omega \in \mathbb{Z}} \exp\left[-\frac{\kappa}{2}(\psi - \mu - 2\pi\omega)^2\right]; \quad \psi \in [0, 2\pi) \tag{3}$$

with $\mu \in [0, 2\pi)$ and $\kappa > 0$; and that of the bivariate density is

$$\begin{aligned} & f_{\mathrm{WN}_2}(\psi_1, \psi_2 | \mu_1, \mu_2, \kappa_1, \kappa_2, \kappa_3) \\ = & \frac{\sqrt{\kappa_1\kappa_2 - \kappa_3^2}}{2\pi} \sum_{(\omega_1,\omega_2)\in\mathbb{Z}^2} \exp\left[-\frac{1}{2} \left\{ \kappa_1(\psi_1 - \mu_1 - 2\pi\omega_1)^2 + \kappa_2(\psi_2 - \mu_2 - 2\pi\omega_2)^2 \right.\right. \\ & \left.\left. + 2\kappa_3(\psi_1 - \mu_1 - 2\pi\omega_1)(\psi_2 - \mu_2 - 2\pi\omega_2) \right\} \right] \end{aligned} \tag{4}$$

where $\psi_1, \psi_2, \mu_1, \mu_2 \in [0, 2\pi)$, $\kappa_1, \kappa_2 > 0$ and $\kappa_3^2 \leq \kappa_1\kappa_2$, obtained by letting $\boldsymbol{\mu} = (\mu_1, \mu_2)^\top$ and

$$\Delta = \begin{pmatrix} \kappa_1 & \kappa_3 \\ \kappa_3 & \kappa_2 \end{pmatrix}.$$

Similarly, the bivariate wrapped normal density is also symmetric around (μ_1, μ_2) and becomes more concentrated as κ_1 and/or κ_2 increases, while the parameter κ_3 regulates the association between the random coordinates. This can be visualized from Figure 2 displaying the surfaces of the density created via **BAMBI** function `surface_model`, for different parameter combinations (the code for generating these plots can be found in the replication script for this paper). The upper panels of Figure 2 show how the density becomes more concentrated when κ_1 and κ_2 are increased (while keeping κ_3 fixed). In contrast, the lower panels of Figure 2 display density surfaces showing how the association between the random coordinates changes (from positive to negative), when κ_3 is changed (from negative to positive, since κ_3 is the diagonal element of the *inverse* covariance matrix) while keeping κ_1 and κ_2 fixed.

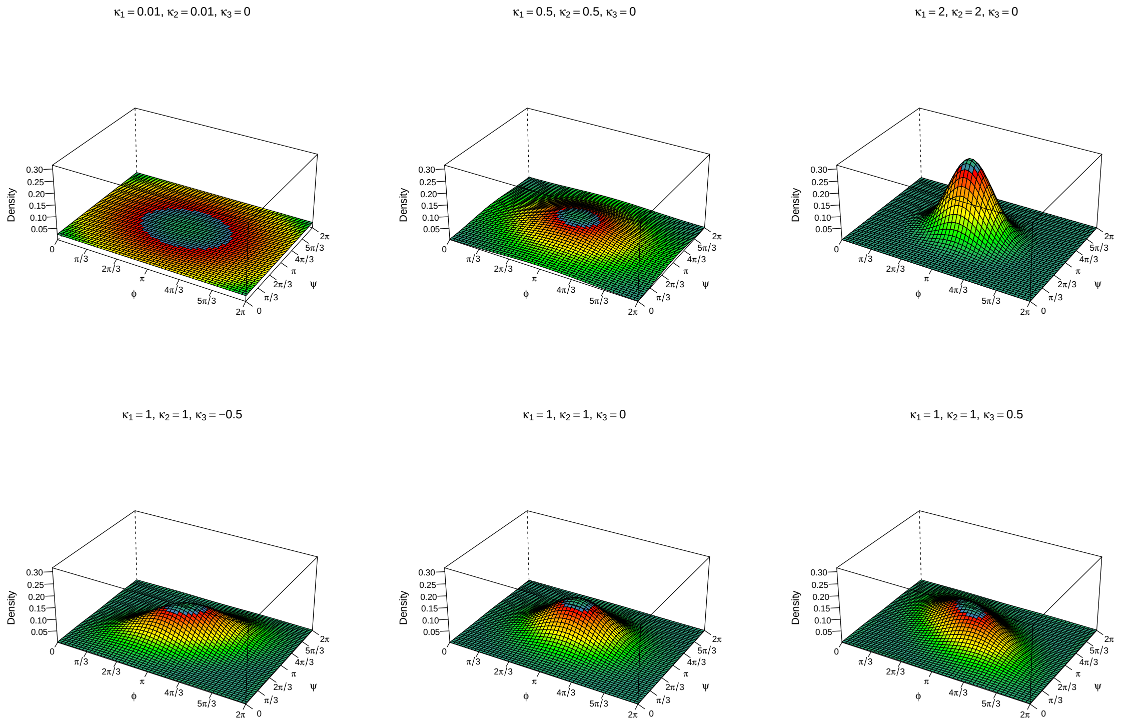
Note that when $\kappa \to 0$ (or $\Delta \to 0_{2\times2}$) then the distribution of $\psi = X \mod 2\pi$ converges to the uniform distribution over $[0, 2\pi)$ (or $[0, 2\pi)^2$). Hence, we shall include the cases $\kappa = 0$ and $\kappa_1 = \kappa_2 = \kappa_3 = 0$ in the support of these parameters, and define the associated densities by their limits.

The precision parameter κ (κ_1, κ_2 in the bivariate case) is (are) conceptually similar to the concentration parameters in the von Mises distribution (see Section 1.2). Therefore to aid comparability, we shall call κ (κ_1 and κ_2) the concentration parameter(s) of the univariate (bivariate) wrapped normal model. In **BAMBI**, evaluation of univariate and bivariate wrapped normal densities are implemented through the function `dwnorm` and `dwnorm2` respectively. Random data from these models can be generated using `rwnorm` and `rwnorm2` respectively.

1.2. Von Mises distributions

Wrapped normal models have a high computational cost in practice. Although the sum over $\mathbb{Z}$ in the expression for the density can be well-approximated by a sum over the set $A = \{-3, -2, -1, 0, 1, 2, 3\}$ (i.e., 3 integer displacements, covering ± 3 standard deviations from the mean), it can be seen that the number of terms in the sum grows exponentially as the dimension increases. For instance, in the bivariate case, even if $\mathbb{Z}$ is approximated by set A, the (double) sum in the density consists of 49 terms.

Because of this difficulty, the von Mises distribution is an alternative that is widely used; it is able to approximate the wrapped normal while being less computationally intensive (Mardia and Jupp 2009, p. 36). A random variable ψ is said to follow the von Mises distribution (also called the circular normal distribution, Jammalamadaka and Sengupta (2001)) with mean



$\kappa_1 = 0.01$, $\kappa_2 = 0.01$, $\kappa_3 = 0$

$\kappa_1 = 0.5$, $\kappa_2 = 0.5$, $\kappa_3 = 0$

$\kappa_1 = 2$, $\kappa_2 = 2$, $\kappa_3 = 0$

$\kappa_1 = 1$, $\kappa_2 = 1$, $\kappa_3 = -0.5$

$\kappa_1 = 1$, $\kappa_2 = 1$, $\kappa_3 = 0$

$\kappa_1 = 1$, $\kappa_2 = 1$, $\kappa_3 = 0.5$

Figure 2: Bivariate wrapped normal density for $\mu_1 = \mu_2 = \pi$ and various κ_1, κ_2, κ_3.

parameter μ and concentration parameter κ, denoted $\psi \sim \text{vM}(\mu, \kappa)$, if ψ has the density

$$f_{\text{vM}}(\psi \mid \mu, \sigma) = \frac{1}{2\pi I_0(\kappa)} \exp(\kappa \cos(\psi - \mu)); \quad \psi \in [0, 2\pi) \tag{5}$$

where $\mu \in [0, 2\pi)$, $\kappa \geq 0$ and $I_r(\cdot)$ denotes the modified Bessel function of the first kind and order r. Letting $\kappa = 0$ makes (5) the uniform density over $[0, 2\pi)$, and when $\kappa \to \infty$, (5) converges to a normal density. An intuitive explanation of the latter result follows from the fact that when the concentration parameter κ is large, $\psi - \mu \approx 0$, so that $\cos(\psi - \mu) \approx 1 - (\psi - \mu)^2/2$, which makes the exponent in the density (5) approximately proportional to the $\text{N}(\mu, (1/\sqrt{\kappa})^2)$ density. A formal proof can be found in Jammalamadaka and Sengupta (2001, Proposition 2.2).

Figure 1(b) plots the von Mises densities with $\mu = \pi$ and $\kappa = 0.01, 1$ and 10, which shows that the density is symmetric around μ and becomes more concentrated as κ increases, and that the density is broadly similar to the associated univariate wrapped normal density.

A multivariate generalization for the univariate von Mises distribution is however not as straightforward as the wrapped normal distribution, as there is not a unique way of defining a multivariate distribution with univariate von Mises-like marginals. In the bivariate case, two versions of the bivariate von Mises distribution have been suggested for practical use, namely the sine model (Singh *et al.* 2002) and the cosine model (Mardia *et al.* 2007). They are comparable to the bivariate normal model both in terms of number of parameters (five), and the interpretability of those parameters. Other generalizations with more parameters have been studied theoretically (Mardia 1975; Rivest 1988).

Let $\boldsymbol{\psi} = (\psi_1, \psi_2)^\top$ be a random vector on $\mathbb{R}^2$ with support $[0, 2\pi)^2$. Then $\boldsymbol{\psi}$ is said to follow the (bivariate) von Mises sine distribution with mean parameters μ_1, μ_2, concentration parameters κ_1, κ_2, and association parameter κ_3, denoted $\boldsymbol{\psi} \sim \text{vM}_2^\text{s}(\mu_1, \mu_2, \kappa_1, \kappa_2, \kappa_3))$, if $\boldsymbol{\psi}$ has the probability density

$$\begin{aligned} &f_{\text{vM}_2^\text{s}}(\psi_1, \psi_2 \mid \mu_1, \mu_2, \kappa_1, \kappa_2, \kappa_3) \\ &= C_s(\kappa_1, \kappa_2, \kappa_3) \exp[\kappa_1 \cos(\psi_1 - \mu_1) + \kappa_2 \cos(\psi_2 - \mu_2) + \kappa_3 \sin(\psi_1 - \mu_1) \sin(\psi_2 - \mu_2)] \quad (6)\end{aligned}$$

where $\kappa_1, \kappa_2 \geq 0$, $-\infty < \kappa_3 < \infty$, $\mu_1, \mu_2 \in [0, 2\pi)$ and the normalizing constant is given by

$$C_s(\kappa_1, \kappa_2, \kappa_3)^{-1} = 4\pi^2 \sum_{m=0}^{\infty} \binom{2m}{m} \left(\frac{\kappa_3^2}{4\kappa_1\kappa_2}\right)^m I_m(\kappa_1) I_m(\kappa_2). \quad (7)$$

In contrast, $\boldsymbol{\psi}$ is said to follow the (bivariate) von Mises cosine distribution with mean parameters μ_1, μ_2, concentration parameters κ_1, κ_2, and association parameter κ_3, denoted $\boldsymbol{\psi} \sim \text{vM}_2^\text{c}(\mu_1, \mu_2, \kappa_1, \kappa_2, \kappa_3)$, if $\boldsymbol{\psi}$ has the probability density [1]

$$\begin{aligned} &f_{\text{vM}_2^\text{c}}(\psi_1, \psi_2 \mid \mu_1, \mu_2, \kappa_1, \kappa_2, \kappa_3) \\ &= C_c(\kappa_1, \kappa_2, \kappa_3) \exp[\kappa_1 \cos(\psi_1 - \mu_1) + \kappa_2 \cos(\psi_2 - \mu_2) + \kappa_3 \cos(\psi_1 - \mu_1 - \psi_2 + \mu_2)]. \quad (8)\end{aligned}$$

Here, similar to the sine model, $\kappa_1, \kappa_2 \geq 0$, $-\infty < \kappa_3 < \infty$, $\mu_1, \mu_2 \in [0, 2\pi)$ and the normalizing constant is given by

$$C_c(\kappa_1, \kappa_2, \kappa_3)^{-1} = 4\pi^2 \left\{ I_0(\kappa_1) I_0(\kappa_2) I_0(\kappa_3) + 2 \sum_{m=0}^{\infty} I_m(\kappa_1) I_m(\kappa_2) I_m(\kappa_3) \right\}. \quad (9)$$

From (6) and (8) it is easy to see that when $\kappa_3 = 0$, both the von Mises sine and cosine densities become products of univariate von Mises densities, implying independence between the two random coordinates. In addition, when κ_1 and κ_2 are also zero, both densities become uniform over $[0, 2\pi)^2$. Singh *et al.* (2002) and Mardia *et al.* (2007) provide explicit forms for the marginal and conditional distributions in the sine and cosine models; the conditional distributions in both sine and cosine models are univariate von Mises, whereas the marginal distributions, although not von Mises, are symmetric around μ_1 and μ_2.

One key difference between the bivariate wrapped normal model and the bivariate von Mises models is that κ_3^2 is not required to be bounded above by $\kappa_1\kappa_2$ in the latter, and thus can take any value in $(-\infty, \infty)$. Consequently, the densities can be bimodal; Mardia *et al.* (2007) show that the sine (cosine) joint density is unimodal if $\kappa_3^2 < \kappa_1\kappa_2$ ($\kappa_3 \geq -\kappa_1\kappa_2/(\kappa_1 + \kappa_2)$), and bimodal otherwise. This flexibility gives the two bivariate von Mises distributions richer sets of possible contour plots and the ability to model a larger class of angular data.

Figures 3 and 4 display the surfaces of the von Mises sine and von Mises cosine densities respectively with $\mu_1 = \mu_2 = \pi$, $\kappa_1 = \kappa_2 = 1$ and various κ_3's. From Figure 3, it can be seen that the density is bimodal when $\kappa_3 = \pm 2$ (or more generally for $|\kappa_3| \geq 1$ when

[1] Mardia *et al.* (2007) define the density with $-\kappa_3$ instead of κ_3 in the exponent. However, that makes the normalizing constant equal to $C_c(\kappa_1, \kappa_2, -\kappa_3)$ in our current notation (i.e., in the form shown in (9)) and not $C_c(\kappa_1, \kappa_2, \kappa_3)$ as given in the paper. See Appendix A for a proof.

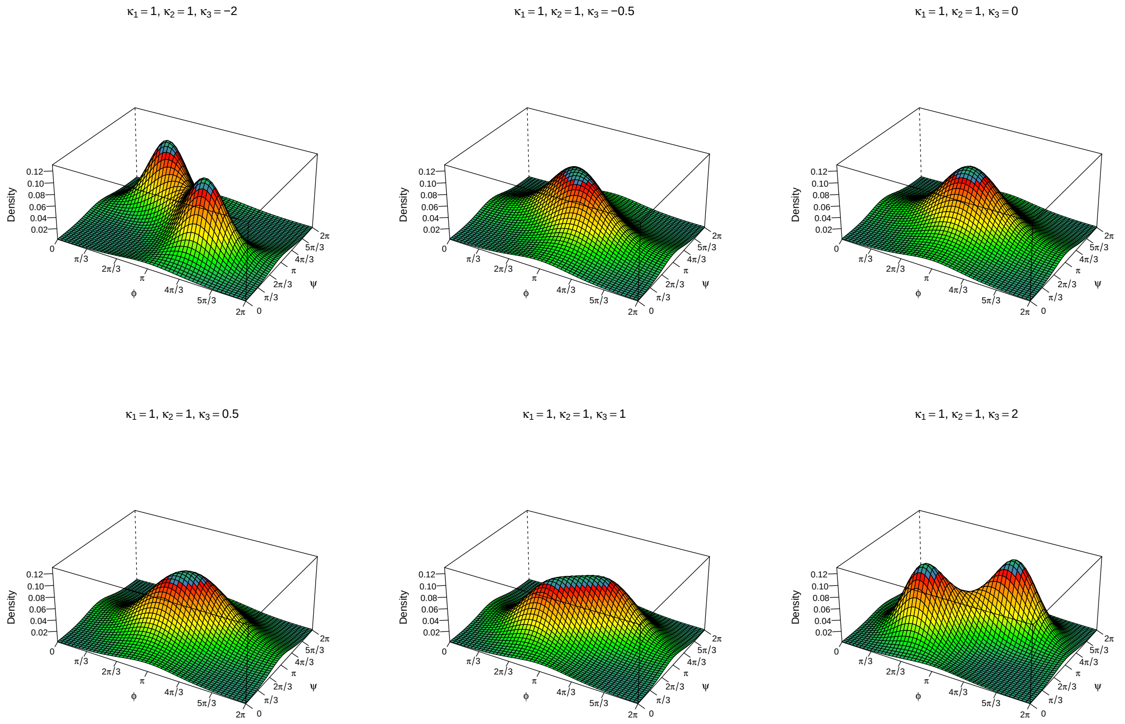


Figure 3: Von Mises sine density for $\mu_1 = \mu_2 = \pi$, $\kappa_1 = \kappa_2 = 1$ and various κ_3.

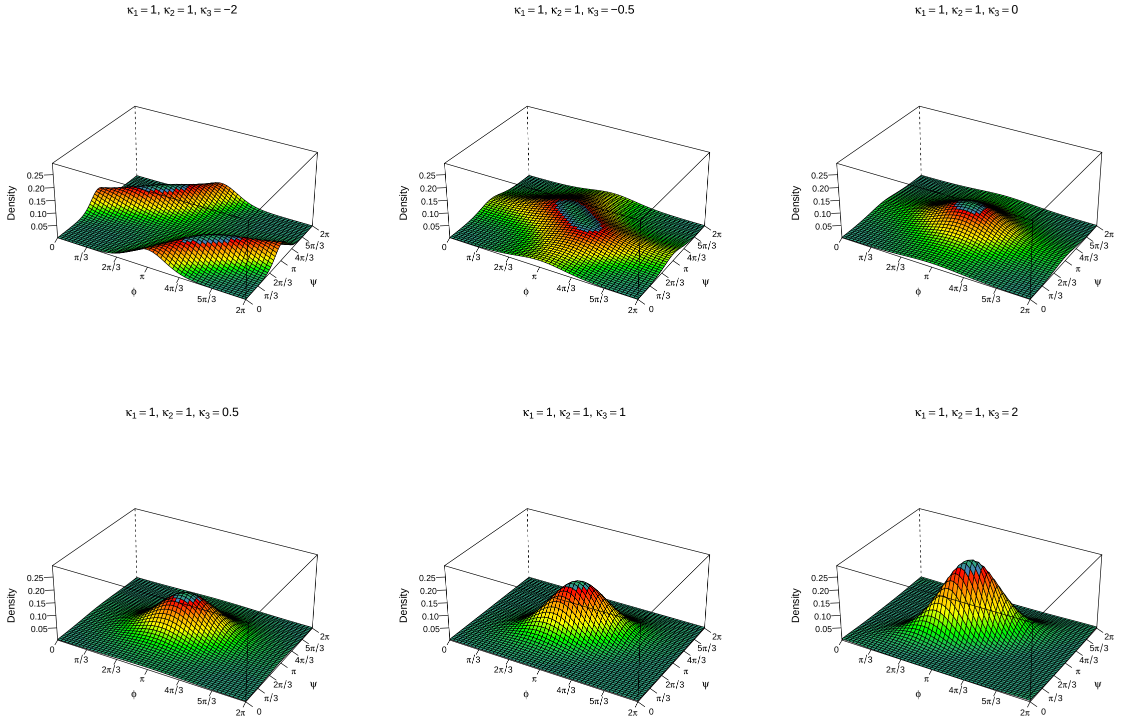


Figure 4: Von Mises cosine density for $\mu_1 = \mu_2 = \pi$, $\kappa_1 = \kappa_2 = 1$ and various κ_3.

$\kappa_1 = \kappa_2 = 1$), and unimodal when $|\kappa_3| < 1$. It can also be seen that the density surface (or the contours) of a sine model with $\kappa_3 = \xi$ is essentially a mirror image of that with $\kappa_3 = -\xi$, for any $\xi \in (-\infty, \infty)$; see, e.g., the upper-left and the lower-right panels of Figure 3. Such is however, not the case for the cosine density, as depicted in Figure 4. The cosine density is bimodal when κ_3 is very negative ($\kappa_3 \leq -0.5$ when $\kappa_1 = \kappa_2 = 1$, see, e.g., the upper-left and upper-middle panels of Figure 4), and is unimodal otherwise. Moreover, flipping the sign of κ_3 does not yield density surfaces (or contours) that are mirror images of each other.

An interesting feature of both sine and cosine densities is that they both approximate the regular bivariate normal density (on $\mathbb{R}^2$) when the concentration parameters κ_1 and κ_2 are large, and the densities are unimodal (Singh *et al.* 2002, Section 2, Mardia *et al.* 2007, Theorem 1). This property is analogous to the univariate von Mises distribution. A heuristic explanation of this result again follows from the fact that when the distributions are unimodal and κ_1, κ_2 are large, then ϕ_1 and ϕ_2 are highly concentrated around μ_1 and μ_2. This means $\phi_i - \mu_i \approx 0$ so that $\sin(\phi_i - \mu_i) \approx (\phi_i - \mu_i)$ and $\cos(\phi_i - \mu_i) \approx 1 - (\phi_i - \mu_i)^2/2$ for $i = 1, 2$.

1.3. Summary measures for univariate and bivariate angular distributions

Circular summary measures are useful for describing various aspects of angular distributions. The circular mean or mean direction (see Jammalamadaka and Sengupta 2001) of an angular random variable ψ is defined as

$$E_c(\psi) = \arctan\left[\frac{E(\sin\psi)}{E(\cos\psi)}\right]$$

and the circular variance of ψ is given by

$$\mathrm{Var}_c(\psi) = 1 - E[\cos(\psi - E_c(\psi))].$$

Note that $0 \leq \mathrm{Var}_c(\psi) \leq 1$.

When considering the joint distribution of paired angular random variables (ϕ, ψ), their association can be measured using circular correlation. Multiple parametric circular correlation coefficients have been proposed in the literature, and here we consider two of them. Let μ_1 and μ_2 be the circular means of ψ_1 and ψ_2 respectively. Then the Jammalamadaka-Sarma (JS) circular correlation coefficient (Jammalamadaka and Sarma 1988) is defined as

$$\rho_{\mathrm{JS}}(\psi_1, \psi_2) = \frac{E\left[\sin(\psi_1 - \mu_1)\sin(\psi_2 - \mu_2)\right]}{\sqrt{E\left[\sin^2(\psi_1 - \mu_1)\right] E\left[\sin^2(\psi_2 - \mu_2)\right]}}. \tag{10}$$

Now let $(\psi_1^{(1)}, \psi_2^{(1)})$ and $(\psi_1^{(2)}, \psi_2^{(2)})$ be independent and identically distributed (IID) copies of (ψ_1, ψ_2). Then the Fisher-Lee (FL) circular correlation coefficient (Fisher and Lee 1983) is defined by

$$\rho_{\mathrm{FL}}(\psi_1, \psi_2) = \frac{E\left[\sin\left(\psi_1^{(1)} - \psi_1^{(2)}\right)\sin\left(\psi_2^{(1)} - \psi_2^{(2)}\right)\right]}{\sqrt{E\left[\sin^2\left(\psi_1^{(1)} - \psi_1^{(2)}\right)\right] E\left[\sin^2\left(\psi_2^{(1)} - \psi_2^{(2)}\right)\right]}}. \tag{11}$$

Both ρ_{JS} and ρ_{FL} have properties similar to the ordinary correlation coefficient. In particular, $\rho_{\mathrm{JS}}, \rho_{\mathrm{FL}} \in [-1, 1]$ and they are equal to 1 (-1) under perfect positive (negative) toroidal-linear (*T-linear*) relationship (Fisher and Lee 1983; Jammalamadaka and Sarma 1988).

Note that all distributions considered in **BAMBI** have circular mean(s) equal to the respective mean parameter(s). For the univariate models, the circular variances are just functions of the associated concentration parameter (see Mardia and Jupp 2009). In particular, if $\psi \sim \mathrm{WN}(\mu, \kappa)$ then $\mathrm{Var}_c(\psi) = 1 - \exp(-\sigma^2/2)$ with $\sigma^2 = 1/\kappa$, and for $\psi \sim \mathrm{vM}(\mu, \kappa)$, $\mathrm{Var}_c(\psi) = 1 - I_1(\kappa)/I_0(\kappa)$. For a bivariate wrapped normal model with $\Sigma = (\Sigma_{ij}) = \Delta^{-1}$, the marginal circular variance of the first coordinate is $1 - \exp(-\Sigma_{11}/2)$, $\rho_{\mathrm{FL}} = \sinh(2\Sigma_{12})/\sqrt{\sinh(2\Sigma_{11})\sinh(2\Sigma_{22})}$ and $\rho_{\mathrm{JS}} = \sinh(\Sigma_{12})/\sqrt{\sinh(\Sigma_{11})\sinh(\Sigma_{22})}$ (Fisher and Lee 1983; Jammalamadaka and Sarma 1988), where sinh denotes the hyperbolic sine function. For bivariate von Mises models (both sine and cosine forms), these expressions, provided in Appendix B, are much more complicated, and involve infinite series of product of modified Bessel functions (see Singh *et al.* 2002; Chakraborty and Wong 2018). In **BAMBI** we implement circular variances and correlation coefficients for all the three bivariate models considered in this article. In addition, a function for calculating sample circular correlation coefficients is also provided, where the sample analogs of ρ_{JS} and ρ_{FL}, along with two other non-parametric circular correlation coefficients are considered (see Section 3.3.3).

1.4. Mixture models

Mixture models are convex combinations (*mixtures*) of two or more probability distributions, and provide a semi-parametric approach to modeling complex data sets with multiple noticeably distinct clusters. Mixture models of both univariate and multivariate (non-wrapped) normal distributions are well studied in the literature (e.g., see Lindsay 1995), and implemented in many statistical packages, such as the R (R Core Team 2021) packages **mixtools** (Benaglia, Chauveau, Hunter, and Young 2009), **mclust** (Fraley, Raftery, Murphy, and Scrucca 2012; Fraley and Raftery 2002), and **Rmixmod** (Langrognet, Lebret, Poli, Iovleff, Auder, and Iovleff 2019). However, these are not applicable to mixture models for angular data. This is a key motivation for our creation of **BAMBI**, which considers finite mixture models of univariate and bivariate angular distributions (the single function `fit_angmix` handles the fitting of all such models; see Section 3.3).

Let K denote the number of components (where K is finite), $\{f(\cdot \mid \boldsymbol{\theta}_j) : j = 1, \ldots, K\}$ denote the component densities (f can be univariate or bivariate) with $\boldsymbol{\theta}_j$ denoting the parameter vector associated with the j-th component, and let $\boldsymbol{p} = (p_1, \ldots, p_K)^\top$ denote the vector of mixing proportions (or weights) with $p_j \geq 0$ and $\sum_{j=1}^K p_j = 1$. Then the mixture density is defined as

$$\tilde{f}(\cdot \mid \boldsymbol{p}; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K) = \sum_{j=1}^{K} p_j f(\cdot \mid \boldsymbol{\theta}_j) \tag{12}$$

In practice, the number of components K necessary to fit the data is usually unknown, and thus should be estimated on the basis of the data itself. (See Section 2.8 for a discussion on number of components estimation.)

An important special case of the general mixture model (12) is the mixture of product components, also called a conditional independence model. Here, one assumes each multivariate component density $f(\cdot \mid \boldsymbol{\theta}_j)$ to be a product of univariate densities; specifically for the bivariate angular models considered in **BAMBI**, this is achieved by letting $\kappa_3 = 0$ in each component. Note that a mixture of product components *does not* imply independence in the final mixture density. In fact, such a model can reasonably approximate a wide class of more complicated models, while being computationally less involved (see Grim 2017); however, one often needs a larger K compared to a general (non-product) mixture model to achieve similar results, thus offsetting some of the potential computational gains. In **BAMBI** a product component mixture can be fitted via `fit_angmix` by setting the argument `cov.restrict = "ZERO"` (see Section 3.3).

It is also noteworthy to mention the aspect of bimodality of bivariate von Mises distributions in the context of mixture modeling. In practice, often each component of a mixture model is used to represent one single (unimodal) cluster in data. However, as discussed in Section 1.2, both von Mises sine and cosine models can be bimodal depending on the values of the concentration and association parameters. When bimodality is present in some of the component specific densities, the final mixture model can be harder to interpret. To avoid this issue, it is possible to restrict the parameter spaces associated with the concentration and association parameters (by letting $\kappa_3^2 < \kappa_1\kappa_2$ in the sine model, and $\kappa_3 \geq -\kappa_1\kappa_2/(\kappa_1 + \kappa_2)$ in the cosine model) in these angular models to force unimodality in each component specific density. Consequently, a larger K may be needed to achieve similar results, which increases model complexity. In **BAMBI** we provide an option of having only unimodal von Mises component densities. This is achieved by setting the logical argument `unimodal.component = TRUE` in `fit_angmix` (defaults to `FALSE`). See the discussion in Section 3.3.

1.5. Related work and motivation for BAMBI

Literature

Several papers have addressed inferential problems relating to mixtures of bivariate angular distributions. Mardia *et al.* (2007) consider the mixture of bivariate von Mises cosine distributions, and suggest an expectation–maximization (EM) algorithm for frequentist estimation of the associated parameters. Their approach is used in Boomsma *et al.* (2008) in the context of modeling protein backbone angles. In other work, Lennox *et al.* (2009) consider a Bayesian non-parametric model involving an infinite mixture of von Mises sine distributions. In **BAMBI** we focus on classical finite mixtures, providing a unifying framework for Bayesian estimation of all three bivariate angular models presented earlier.

Software

To the best of our knowledge, no previous packages or libraries handle finite mixture modeling for univariate or bivariate angular data, whether in R or otherwise. In fact, the only available software (as of the time of writing this manuscript) that has functionality for bivariate von Mises models is the C++ library **mocapy++** (Paluszewski, Frellsen, and Hamelryck 2010) in the context of Dynamic Bayesian Networks (Paluszewski and Hamelryck 2010; Mardia *et al.* 2007). However, **mocapy++** does not implement bivariate wrapped normal models.

The overarching goal of **BAMBI** is to create a unified platform that implements descriptive and inferential statistical tools required to analyze bivariate and univariate angular data. First, **BAMBI** provides functions for density evaluation, computation of various summary measures (such as circular mean, variance and correlation coefficient), and random data generation from bivariate and univariate angular models and their mixtures. Second, it has functions for fitting these models to real angular data using Bayesian methods. Third, it implements a number of post-processing steps required in any Bayesian statistical analysis. For example, visual and numerical assessment of the goodness of fits can be done using a number of native **BAMBI** functions, as well as **coda** (Plummer, Best, Cowles, and Vines 2006) package functions, which are applicable on **BAMBI** outputs ('`angmcmc`' objects) through a convenient `as.mcmc.list` method. Furthermore, **BAMBI** has functions for model selection as well as random data generation and density evaluation from fitted models, which are useful in posterior predictive analyses.

It is to be noted that while it is possible to use general-purpose MCMC samplers such as Stan (Carpenter *et al.* 2017), **JAGS** (Plummer 2003) and **WinBUGS** (Lunn, Thomas, Best, and Spiegelhalter 2000) for fitting the angular mixture models considered in **BAMBI**, there are important motivations for developing specialized implementations for these models. First, special care needs to be taken while handling the normalizing constants in the von Mises sine and cosine densities, which contain infinite series of product of Bessel functions that can be numerically unstable for some ranges of parameter values; such cases are handled in **BAMBI** via (quasi) Monte Carlo approximations. Second, computations for Bayesian mixture modeling benefit from using a latent allocation structure, as done in **BAMBI** (see Section 2.3), which allows independent sampling of the component specific parameters. Such an approach cannot be used in Stan due to the discreteness of the allocation (p. 79, Section 6.2 of the reference manual v2.18.0); instead Stan requires marginalizing out the latent allocation variables. In contrast, **JAGS**/**WinBUGS** allows incorporation of discrete latent allocation; however, their sampling techniques do not make use of the gradient of the target (log) posterior density. As discussed in Section 2.5, Hamiltonian Monte Carlo uses the gradient and hence is typically more efficient for sampling from intractable distributions. Finally, the analytic gradients necessary for efficient MCMC sampling in these models are built into **BAMBI**.

1.6. Organization of the paper

The remainder of this article is organized as follows. In Section 2, we review Bayesian methods for fitting angular mixture models to data. In Section 3 we describe the capabilities of **BAMBI**, by describing all functions and data sets available in **BAMBI**, and providing brief overviews on their usage. Following, in Section 4 we illustrate angular mixture modeling on data sets included in **BAMBI**. The paper concludes with a brief summary and possible directions for future development in Section 5. A derivation for the von Mises cosine model normalizing constant, formulas for circular variances and correlation coefficients in the von Mises sine and cosine models, analytic forms of gradients needed for efficient MCMC sampling (discussed in Section 2.5), and MCMC parameter traceplots associated with one of the examples considered in Section 4 are provided in the Appendices.

2. Methods

2.1. Overview

We adopt a Bayesian approach for fitting angular mixture models to data. Let $\boldsymbol{\Psi}^\top = (\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_n)$ be the data matrix (or data vector in the univariate case) with each $\boldsymbol{\psi}_i$ being a bivariate vector of angles (or a univariate angle) $[0, 2\pi)^2$ (or in $[0, 2\pi)$). We are interested in fitting a mixture density of the form (12) for a given number of components K. For example, in bivariate wrapped normal mixtures, the density for the j-th component is given by $f_j \equiv f_{\mathrm{WN}_2}(\cdot \mid \boldsymbol{\theta}_j) =: f_{\mathrm{WN}_2,j}$, where $\boldsymbol{\theta}_j^\top = (\kappa_{1j}, \kappa_{2j}, \kappa_{3j}, \mu_{1j}, \mu_{2j})$ denotes the vector of (model) parameters for the j-th component, $j = 1, \ldots, K$, and the mixture density is given by $\tilde{f}_{\mathrm{WN}_2} = \sum_{j=1}^{K} p_j f_{\mathrm{WN}_2,j}$. For a specified K, our objective is to estimate the parameter vector $\boldsymbol{\eta}^\top = (\boldsymbol{\theta}^\top, \boldsymbol{p}^\top)$, which consists of the model parameters $\boldsymbol{\theta}^\top = (\boldsymbol{\theta}_1^\top, \ldots, \boldsymbol{\theta}_K^\top)$ and the mixing proportions $\boldsymbol{p}^\top = (p_1, \ldots, p_K)$, based on $\boldsymbol{\Psi}$. Often, K itself will also need to be estimated. In the following, we review some commonly used techniques in Bayesian mixture model fitting.

2.2. Bayesian mixture modeling

Under a Bayesian framework a prior distribution must be specified for the parameter vector, which can be non-informative (or diffuse) if *a priori* information is unavailable. Let $\pi(\boldsymbol{\theta}, \boldsymbol{p})$ denote the joint prior density for $\boldsymbol{\eta}$. Often the prior distributions of $\boldsymbol{\theta}$ and $\boldsymbol{p}$ are assumed to be independent so that (with a slight abuse of notation; here $\pi(y)$ stands for the appropriate prior density of the random variable y) $\pi(\boldsymbol{\theta}, \boldsymbol{p}) = \pi(\boldsymbol{\theta})\pi(\boldsymbol{p})$. Moreover, parameters from different components are often assumed to be independent, so that $\pi(\boldsymbol{\theta}) = \prod_{j=1}^{K} \pi(\boldsymbol{\theta}_j)$. Let $L(\boldsymbol{\Psi} \mid \boldsymbol{\theta}, \boldsymbol{p}) = \prod_{i=1}^{n} \tilde{f}(\psi_i \mid \boldsymbol{\theta}, \boldsymbol{p})$ denote the likelihood function of the data. Then the posterior density of $\boldsymbol{\eta}$ given the data is

$$\pi(\boldsymbol{\theta}, \boldsymbol{p} \mid \boldsymbol{\Psi}) \propto L(\boldsymbol{\Psi} \mid \boldsymbol{\theta}, \boldsymbol{p})\, \pi(\boldsymbol{p}) \prod_{j=1}^{K} \pi(\boldsymbol{\theta}_j), \tag{13}$$

which is the basis for Bayesian inference on $\boldsymbol{\eta}$. It is to be noted that the prior densities $\pi(\boldsymbol{\theta}_j)$'s all need to be proper in order to ensure that the posterior density $\pi(\boldsymbol{\theta}, \boldsymbol{p} \mid \boldsymbol{\Psi})$ is proper (see, e.g., Diebolt and Robert 1994, Section 2.2). Specific comments about the choice of priors used in the current setting are provided in Section 2.7. Note that the associated posterior mean, median or mode, commonly used as point estimates of the parameters, are not available in closed form for our distributions of interest. Additionally, $\pi(\boldsymbol{\theta}, \boldsymbol{p} \mid \boldsymbol{\Psi})$ is intractable for directly simulating IID samples, and thus some kind of Markov chain Monte Carlo (MCMC) technique is used in practice as an alternative. Starting from some initial point, an MCMC algorithm generates a Markov chain which has the target posterior density $\pi(\boldsymbol{\theta}, \boldsymbol{p} \mid \boldsymbol{\Psi})$ as the invariant distribution. Various summary measures of the posterior distributions – such as mean, mode (known as the *maximum a posteriori* or MAP parameter value), and quantiles – can then be approximated based on the MCMC realizations. In practice, the MCMC algorithm must be run long enough for the Markov chain to converge, so that the realizations approximately follow the target posterior distribution. For this purpose the chain is given a *burn-in* period, where the initial iterations are discarded.

In **BAMBI** the function `fit_angmix` fits a Bayesian angular mixture model with a specified number of components, and the function `fit_incremental_angmix` fits angular mixtures with incremental number of components to determine an optimum number of components. In the following we briefly review the MCMC generation techniques *Gibbs sampler* (GS), *Metropolis Hastings* and *Hamiltonian Monte Carlo* (HMC), and describe how they are used for sampling from the posterior distributions of model parameters and mixing proportions in these two **BAMBI** functions.

2.3. Gibbs sampler (GS)

The Gibbs sampler (GS) (Geman and Geman 1984; Gelfand and Smith 1990) breaks the Markov chain updates for the parameter vector into blocks. For example, when $\boldsymbol{\eta} = (\boldsymbol{\eta}_1, \boldsymbol{\eta}_2)$

the GS generates the N-th state of the Markov chain $(\boldsymbol{\eta}_1^{(N)}, \boldsymbol{\eta}_2^{(N)})$ from the previous state $(\boldsymbol{\eta}_1^{(N-1)}, \boldsymbol{\eta}_2^{(N-1)})$ with the steps

1. Generate $\boldsymbol{\eta}_1^{(N)}$ from $\pi(\boldsymbol{\eta}_1 \mid \boldsymbol{\eta}_2^{(N-1)}, \text{data})$.

2. Generate $\boldsymbol{\eta}_2^{(N)}$ from $\pi(\boldsymbol{\eta}_2 \mid \boldsymbol{\eta}_1^{(N)}, \text{data})$.

The GS is most effective when it is easy to sample from the (full) conditional posterior densities $\pi(\boldsymbol{\eta}_1 \mid \boldsymbol{\eta}_2, \text{data})$ and $\pi(\boldsymbol{\eta}_2 \mid \boldsymbol{\eta}_1, \text{data})$. Note that when $\boldsymbol{\eta}_1$ and $\boldsymbol{\eta}_2$ are vectors, this is sometimes called the blocked Gibbs sampler.

For mixture models, an efficient Gibbs sampling step for the mixing proportions $\boldsymbol{p}$ (when $K > 1$) can be obtained by adopting a so-called Data Augmentation scheme, where one introduces ("augments") unobserved data to make the conditional distributions simpler (Diebolt and Robert 1994). Here, we introduce (hidden) component indicators $\boldsymbol{\zeta}_i^\top = (\zeta_{i1}, \ldots, \zeta_{iK})$ corresponding to each observation $\boldsymbol{\psi}_i$ where ζ_{ij} is 1 if the i-th observation comes from the j-th component, and 0 otherwise, for $i = 1, \ldots, n$ and $j = 1, \ldots, K$. Thus, given $\zeta_{ij} = 1$, the density of $\boldsymbol{\psi}_i$ is simply $f\left(\boldsymbol{\psi}_i \mid \boldsymbol{\theta}_j\right)$, i.e., the density in the j-th component evaluated at $\boldsymbol{\psi}_i$. Moreover, $n_j := \sum_{i=1}^n \zeta_{ij}$ is the total number of observations coming from this density. It is customary to assume a Dirichlet$(\boldsymbol{\alpha})$ prior for $\boldsymbol{p}$, where $\boldsymbol{\alpha}^\top = (\alpha_1, \ldots, \alpha_K)$ with $\alpha_j > 0$ for all j, so that $\pi(\boldsymbol{p}) \propto \prod_{j=1}^K p_j^{\alpha_j - 1}$. Note that $\alpha_j = 1$ for all j represents the uniform prior. Let $Z^\top = (\boldsymbol{\zeta}_1^\top, \ldots, \boldsymbol{\zeta}_n^\top)$ and let $\boldsymbol{\theta}^{(N-1)}$, $\boldsymbol{p}^{(N-1)}$ and $Z^{(N-1)}$ be the $(N-1)$-th MCMC realizations of $\boldsymbol{\theta}$, $\boldsymbol{p}$ and Z respectively. Then the N-th realization of $\boldsymbol{p}$ (and Z) are obtained as follows:

1. For $i = 1, \ldots, n$, generate $\boldsymbol{\zeta}_i^{(N)}$ from Multinomial$\left(1; \tilde{p}_{i1}^{(N-1)}, \ldots, \tilde{p}_{iK}^{(N-1)}\right)$ independently, and define $n_j^{(N)} := \sum_{i=1}^n \zeta_{ij}^{(N)}$, where

$$\tilde{p}_{ij}^{(N-1)} = \frac{p_j^{(N-1)} f\left(\boldsymbol{\psi}_i \mid \boldsymbol{\theta}_j^{(N-1)}\right)}{\sum_{h=1}^K p_h^{(N-1)} f\left(\boldsymbol{\psi}_i \mid \boldsymbol{\theta}_h^{(N-1)}\right)} \tag{14}$$

   are the posterior membership probabilities.

2. Generate $\boldsymbol{p}^{(N)}$ from Dirichlet$\left(\alpha_1 + n_1^{(N)}, \ldots, \alpha_K + n_K^{(N)}\right)$.

Thus when $K > 1$, the latent allocation $\boldsymbol{\zeta}_i$'s generated during the Gibbs sampling step for $\boldsymbol{p}$ leads to simplifications that reduce the computational burden substantially. Note that, conditional on $\boldsymbol{\zeta}_i$'s, all $\boldsymbol{\psi}_i$'s have independent single component densities $f(\cdot \mid \boldsymbol{\theta}_{j_i})$, with j_i being the non-zero position of $\boldsymbol{\zeta}_i$. Thus, given $\boldsymbol{\zeta}_i$'s, all $\boldsymbol{\theta}_j$'s are independent with only data points coming from component j contributing to the respective likelihoods. Consequently $\boldsymbol{\theta}_j$'s can be sampled independently from their (component-specific) conditional posterior densities.

To complete the GS scheme for the mixture model, it remains to sample $\boldsymbol{\theta}_j$'s from $\pi(\boldsymbol{\theta}_j \mid Z, \boldsymbol{\Psi}, \boldsymbol{p})$. As these distributions are still intractable for direct IID simulation, we use a Markov chain simulation technique for sampling, and then combine this step with the GS updates for $\boldsymbol{p}$ and Z. In the following we describe two such Markov chain simulation techniques, and how they are used in **BAMBI**.

2.4. Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm (Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller 1953; Hastings 1970) is simple and widely-used for Markov chain simulation. Formally, let x be the current state of a Markov chain Φ with stationary density q. Let $\tilde{q}(\cdot \mid x)$ be a proposal density defined on the state space of Φ that is easy to sample from. Then the next state x' of the Markov chain Φ is obtained as follows:

1. Generate x^* from $\tilde{q}_x$.
2. Define $r(x^*, x) = \min\left\{1, \frac{q(x^*)}{q(x)} \frac{\tilde{q}(x|x^*)}{\tilde{q}(x^*|x)}\right\}$, and define the next state x' equal to x^* with probability $r(x^*, x)$ and equal to x with probability $1 - r(x^*, x)$.

The random walk variant of Metropolis-Hastings (RWMH) uses a proposal density $\tilde{q}(\cdot \mid x)$ that is symmetric about x; e.g., by taking $\tilde{q}(\cdot \mid x)$ to be the density of $Y_x = x + Y_0$, where Y_0 is a normal random variable with mean zero. Under RWMH, $\tilde{q}(x \mid x^*) = \tilde{q}(x^* \mid x)$, and hence $r(x^*, x) = \min\left\{1, \frac{q(x^*)}{q(x)}\right\}$, thus simplifying computations. In **BAMBI**, RWMH is implemented with independent normal proposals.

Note that the variance of the density $\tilde{q}(\cdot \mid x)$ strongly affects the acceptance probabilities $r(x^*, x)$. Convergence of the Markov chain will be slow if the variance of $\tilde{q}(\cdot \mid x)$ is too large or too small. Roberts, Rosenthal *et al.* (2001) suggest maintaining an acceptance rate of 20-30% as a general rule-of-thumb. In **BAMBI** we provide an auto-tuning feature that implements adaptive tuning during the burn-in period. Briefly, the acceptance rate and scale of the sampled parameters are monitored at regular intervals, and the proposal variances are adjusted accordingly (see the documentation of `fit_angmix` for details). We limit adaptation to the burn-in period, so that the desired properties of the final MCMC samples are retained.

2.5. Hamiltonian/Hybrid Monte Carlo (HMC)

Simple RWMH can become quite inefficient in multi-dimensional problems. A powerful alternative to RWMH when the gradient of the posterior density has an analytical form is Hamiltonian (also called *Hybrid*) Monte Carlo (HMC) (Duane, Kennedy, Pendleton, and Roweth 1987; Neal 1996). HMC makes use of the gradient of the log posterior density and an auxiliary random variable, and incorporates tools from molecular dynamics to furnish proposal states coming from high posterior density regions. This allows a much faster exploration of the state space than a RWMH scheme. A gentle and detailed introduction to HMC with applications to statistical problems can be found in Neal (2011). Briefly, in HMC first an auxiliary random variable $\boldsymbol{r}$ called *momentum* is considered along with the variable of interest (vector of model parameters $\boldsymbol{\theta}$ in our case), which is classically called the *position* in physical problems, denoted by $\boldsymbol{q}$[2]. Furthermore, two energy functions $U(\boldsymbol{q})$ and $K(\boldsymbol{r})$ are introduced, followed by a Hamiltonian function $H(\boldsymbol{q}, \boldsymbol{r})$ which is usually the sum of those two energies, i.e., $H(\boldsymbol{q}, \boldsymbol{r}) = U(\boldsymbol{q}) + K(\boldsymbol{r})$. $U(\boldsymbol{q})$, called the *potential energy*, is defined as the negative log posterior density of $\boldsymbol{q}$ (plus any fixed constant) in MCMC applications, and $K(\boldsymbol{r})$, called the *kinetic energy*, is usually defined as $K(\boldsymbol{r}) = \boldsymbol{r}^\top M^{-1} \boldsymbol{r}$ for some fixed positive definite matrix M. This form for $K(\boldsymbol{r})$ corresponds to the negative log density (plus a constant) of the zero-mean normal distribution with variance matrix M. In practice, M is typically taken to be

[2] In classical HMC literature, the auxiliary variable is denoted by $\boldsymbol{p}$; however, we will keep that notation for mixing proportions.

diagonal, often the identity matrix (as used in **BAMBI**), or a scalar multiple of the identity matrix. Let $\nabla U(\boldsymbol{q})$ denote the gradient vector of $U(\boldsymbol{q})$ with respect to $\boldsymbol{q}$. Further, let $\epsilon > 0$ be a small real number, called the *step-size*, and $L \geq 2$, a positive integer, called the number of *leapfrog steps.* Then one step of HMC that updates (via *leapfrog* method) the current state $\boldsymbol{q}$ to the next state $\boldsymbol{q}'$ can be described as follows:

1. Generate $\boldsymbol{r}$ from $\mathrm{N}(\mathbf{0}, M)$ and let $\boldsymbol{q}^{(0)} = \boldsymbol{q}$ and $\boldsymbol{r}^{(0)} = \boldsymbol{r} - (\epsilon/2)\nabla U(\boldsymbol{q}^{(0)})$.
2. For $t = 1, \ldots, L$ define $\boldsymbol{q}^{(t)} = \boldsymbol{q}^{(t-1)} + \epsilon\, \boldsymbol{r}^{(t-1)}$ and $\boldsymbol{r}^{(t)} = \boldsymbol{r}^{(t-1)} - (\epsilon/\gamma_l)\nabla U(\boldsymbol{q}^{(t)})$, where $\gamma_l = 1$ for $l = 1, \ldots, L-1$ and $\gamma_L = 2$.
3. Let $\boldsymbol{q}^* = \boldsymbol{q}^{(L)}$, $\boldsymbol{r}^* = -\boldsymbol{r}^{(L)}$, and define $\beta(\boldsymbol{q}^*, \boldsymbol{r}^*; \boldsymbol{q}, \boldsymbol{r}) = \min\{1, \exp[H(\boldsymbol{q}^*, \boldsymbol{r}^*) - H(\boldsymbol{q}, \boldsymbol{r})]\}$.
4. Finally, define the new state $\boldsymbol{q}'$ equal to $\boldsymbol{q}^*$ with probability $\beta(\boldsymbol{q}^*, \boldsymbol{r}^*; \boldsymbol{q}, \boldsymbol{r})$, and equal to $\boldsymbol{q}$ with probability $1 - \beta(\boldsymbol{q}^*, \boldsymbol{r}^*; \boldsymbol{q}, \boldsymbol{r})$.

Special care needs to be taken for the cases where the variables being sampled are constrained: for our angular models, μ_i's are angles in $[0, 2\pi)$, and the (raw) concentration parameters are positive. See Neal (2011, Section 5.5.1.5) for more details.

Since HMC approximates the dynamics by discretization, the step-size ϵ needs to be sufficiently small for the proposals to have a high acceptance rate. However, if ϵ is too small, convergence of the Markov chain will be slow. Thus, ϵ requires tuning to obtain a reasonable acceptance rate (∼40-90%, with 65% being optimal, as suggested by Neal 2011). In **BAMBI** we provide an auto-tune feature for ϵ similar to the one for the proposal standard deviation in RWMH (see Section 2.4), which adaptively tunes ϵ during burn-in to ensure a reasonable acceptance rate (60-90% by default).

Care is required for choosing the number of leapfrog steps L, since a L that is too large or too small can lead to poor convergence. While setting an appropriate L can be challenging for high dimensional parameter vectors, here the independence of components $\pi(\boldsymbol{\theta}_j \mid Z, \boldsymbol{\Psi}, \boldsymbol{p})$ means that only two (for univariate models) or five (for bivariate models) parameters need to be sampled at a time. Thus, the default $L = 10$ used in **BAMBI**, which works well empirically, suffices for mixtures with any number of components. As suggested in Neal (2011) and Mackenze (1989), randomly choosing ϵ and L from some fairly small interval at the beginning of every HMC step may improve convergence of the chain. In **BAMBI** ϵ is by default randomly chosen at each iteration from an interval of the form $(\epsilon_0(1-\delta), \epsilon_0(1+\delta))$ for a fixed $\epsilon_0 > 0$ (can be auto-tuned in **BAMBI**) and a given $\delta \in (0, 1)$, while L is kept fixed. However, these settings can be changed; in particular, L can also be randomly chosen from the set of integers contained in an interval $(L_0/\exp(A), L_0\exp(A))$ for some given $L_0 > 0$ and $A > 0$, or both ϵ and L can be specified to be non-random. See the documentation of `fit_angmix` for more details.

When properly tuned, HMC can achieve faster convergence and better exploration of the target density than RWMH, for a similar computational cost. Note that the computational cost for each HMC iteration is higher due to L gradient evaluations, however, HMC usually requires fewer iterations to reach stationarity and successive samples have lower autocorrelation. Hence, HMC is our recommended sampling approach in **BAMBI**. HMC, while powerful, does not solve all the challenges associated with MCMC sampling algorithms; in particular, both RWMH and HMC can get trapped in local modes. One possible remedy is to use multiple independent chains, see Section 2.10.

By default, **BAMBI** uses HMC to sample $\boldsymbol{\theta}$. All angular densities considered here, both univariate and bivariate, admit analytic gradients for efficient programming implementation. Expressions for the conditional log posterior density and its gradients are provided in the following section.

2.6. Using RMWH or HMC for angular mixture models

Consider the mixture model (12) with density $f(\cdot \mid \boldsymbol{\theta}_j)$ for the j-th component, $j = 1, \ldots, K$. It follows that given the component indicators Z, information on $\boldsymbol{p}$ is superfluous, and the complete-data (i.e., given $\boldsymbol{\Psi}$ and Z) likelihood for $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)$ is given by:

$$\text{likelihood}(\boldsymbol{\theta} \mid Z, \boldsymbol{\Psi}) \propto \prod_{i=1}^{n} \prod_{j=1}^{K} f(\boldsymbol{\psi}_i \mid \boldsymbol{\theta}_j)^{\zeta_{ij}}.$$

Recall that the joint prior density of $\boldsymbol{\theta}$ is $\prod_{j=1}^{K} \pi(\boldsymbol{\theta}_j)$. Hence, the complete-data posterior density of $\boldsymbol{\theta}$ is given by:

$$\pi(\boldsymbol{\theta} \mid Z, \boldsymbol{\Psi}) \propto \left\{ \prod_{i=1}^{n} \prod_{j=1}^{K} f(\boldsymbol{\psi}_i \mid \boldsymbol{\theta}_j)^{\zeta_{ij}} \right\} \prod_{j=1}^{K} \pi(\boldsymbol{\theta}_j).$$

Therefore, by taking the logarithm, the complete-data log posterior density (LPD) for $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)$ given the component indicators Z is obtained as

$$\begin{aligned}
\tilde{l}_{\text{complete-data}}(\boldsymbol{\theta}) := \log \pi(\boldsymbol{\theta} \mid Z, \boldsymbol{\Psi}) &= C + \sum_{i=1}^{n} \sum_{j=1}^{K} \zeta_{ij} \log f(\boldsymbol{\psi}_i \mid \boldsymbol{\theta}_j) + \sum_{j=1}^{K} \log \pi(\boldsymbol{\theta}_j) \\
&= C + \sum_{j=1}^{K} \left\{ \sum_{i=1}^{n} \zeta_{ij} \log f(\boldsymbol{\psi}_i \mid \boldsymbol{\theta}_j) + \log \pi(\boldsymbol{\theta}_j) \right\} \\
&= C + \sum_{j=1}^{K} \left\{ \sum_{i:\, \zeta_{ij}=1} \log f(\boldsymbol{\psi}_i \mid \boldsymbol{\theta}_j) + \log \pi(\boldsymbol{\theta}_j) \right\} \qquad (15)
\end{aligned}$$

where C is a constant free of $\boldsymbol{\theta}$. The above expression shows that conditional on Z, $\boldsymbol{\theta}_j$'s are independent, and that the complete-data log posterior density of $\boldsymbol{\theta}_j$ is of the form

$$\tilde{l}_j(\boldsymbol{\theta}_j) = C_j + \sum_{i:\, \zeta_{ij}=1} \log f(\boldsymbol{\psi}_i \mid \boldsymbol{\theta}_j) + \log \pi(\boldsymbol{\theta}_j) \qquad (16)$$

where C_j's are constants (free of $\boldsymbol{\theta}$). Given the current GS draw of Z, samples from the conditional posterior density $\tilde{l}_j$ in (16) can therefore be drawn independently for all $j = 1, \ldots, K$. For each $j \geq 1$, we let $\tilde{l}_j$ play the role of the target density q (see Section 2.4) in RWMH, or let $-\tilde{l}_j$ play the role of the potential energy U (see Section 2.5) in HMC; the gradient of U with respect to $\boldsymbol{\theta}_j$, ∇U, is therefore the negative of the gradient $\nabla \tilde{l}_j$. From (16), it follows that

$$\begin{aligned}
\nabla \tilde{l}_j(\boldsymbol{\theta}) &= \left(\sum_{i:\, \zeta_{ij}=1} \frac{\partial \log f(\boldsymbol{\psi}_i \mid \boldsymbol{\theta}_j)}{\partial \boldsymbol{\theta}_j} \right) + \frac{\partial \log \pi(\boldsymbol{\theta}_j)}{\partial \boldsymbol{\theta}_j} \qquad (17a) \\
&= \left(\sum_{i:\, \zeta_{ij}=1} \frac{1}{f(\boldsymbol{\psi}_i \mid \boldsymbol{\theta}_j)} \cdot \frac{\partial f(\boldsymbol{\psi}_i \mid \boldsymbol{\theta}_j)}{\partial \boldsymbol{\theta}_j} \right) + \frac{\partial \log \pi(\boldsymbol{\theta}_j)}{\partial \boldsymbol{\theta}_j} \qquad (17b)
\end{aligned}$$

For the von Mises distributions (both univariate and bivariate), form (17a) is easier to work with, whereas form (17b) is more useful for the wrapped normal distributions. Full analytic expressions for all model specific gradients are provided in Appendix C.

Note that parameters with a non-negative support are often sampled more efficiently on the log scale; we use this strategy for sampling the concentration parameters κ (in univariate models) and κ_1, κ_2 (in bivariate models).

2.7. Choice of priors

Selection of prior constitutes an important step in Bayesian analyses, as they play a key role in the final inference. This is comparatively more standard for the component-specific model parameters $\boldsymbol{\theta}$. As discussed, proper prior distributions for the model parameters are required to ensure posterior propriety. For the mean parameters μ (in univariate models) and μ_1, μ_2 (in bivariate models), their prior distributions can be taken to be a member of the same family of the distribution which are being used in the mixture model (e.g., von Mises sine prior for (μ_1, μ_2) in a von Mises sine mixture model) to aid conjugacy. Lennox *et al.* (2009) use this conjugate prior for the mean parameters in their von Mises sine (infinite) mixture model. Note that conjugacy for the mean parameter is not achievable except in trivial cases in the wrapped normal distributions (both univariate and bivariate). In **BAMBI** we set a uniform prior over $[0, 2\pi)$ (if univariate) or $[0, 2\pi)^2$ (if bivariate) for the mean parameter(s), which can be viewed as a special case of the von Mises and wrapped normal distributions (see Sections 1.1 and 1.2). Conjugacy is also possible for the concentration and association parameters, e.g., Lennox *et al.* (2009) consider such a family for von Mises sine model. However, that approach does not aid sampling, as the resulting unnormalized densities involve infinite sums of products of modified Bessel functions. As a simple alternative, we suggest using independent normal distributions with zero mean as the prior for the association parameter κ_3, as well as for the log of the concentration parameters κ, κ_1, and κ_2 (i.e., the prior for concentration parameters are log normal). These prior distributions can be made informative or diffuse through appropriate choices of the variance hyper-parameter. Priors are assigned independently to each parameter, and truncation is performed to reflect any specified constraints in the model (such as $\kappa_3^2 < \kappa_1\kappa_2$ in a bivariate wrapped normal model, and a von Mises sine model with unimodal density).

Care is required in the selection of prior for the mixing proportions $\boldsymbol{p}$, as an ill-chosen prior may result in very poor fits. This is particularly true when K is too large (i.e., the mixture is overfitted). Note that overfitting is a necessary step when the true number of components is unknown and needs to be estimated, see Section 2.8 for more details. It is customary to assume a Dirichlet$(\boldsymbol{\alpha})$ prior for $\boldsymbol{p}$, where $\boldsymbol{\alpha}^\top = (\alpha_1, \ldots, \alpha_K)$ with $\alpha_j > 0$, often with the special case $\alpha_j = \alpha_0$ for all j. When the mixture is overfitted, the asymptotic results in Rousseau and Mengersen (2011) show that α_j's strongly influence how the spurious mixture components are handled by the limiting posterior density. In particular, if $\max_{j=1,\ldots,K} \alpha_j < d/2$, where $d = \dim \boldsymbol{\theta}_j$, then the spurious components vanish asymptotically. On the other hand, if $\min_{j=1,\ldots,K} \alpha_j > d/2$, then the spurious components asymptotically get superimposed on some of the existing components with positive mixing proportions (Frühwirth-Schnatter 2011, Section 10.3.1). The subsequent estimation of K depends on which way the overfitting is handled by the posterior density (see Section 2.8); thus α_j's all need to be appropriately either small or large (Frühwirth-Schnatter 2011, Section 10.3.2). A uniform prior with $\alpha_j = \alpha_0 = 1$ is a rather poor choice in this regard. In **BAMBI** estimation of K is done assuming the use of $\alpha_j > d/2$ for all j in conjunction with a model selection criterion; our default is $\alpha_j = \alpha_0 = (r + r(r+1)/2)/2 + 3$ as used in Frühwirth-Schnatter (2011, Section 10.3.4), where r denotes the dimension of the data, i.e., r is 1 or 2 according as whether the model is univariate or bivariate (and consequently, all α_j's are either 4 or 5.5).

2.8. Estimating the number of components K from data

Suppose the data were generated from a mixture of K_{true} (non-empty, non-identical) components. In practice, K_{true} will not be known, and therefore mixture modeling requires estimating the appropriate number of components from the data.

In the Bayesian setting, the estimation of K_{true} requires an overfitted mixture model, i.e., one that has spurious or superfluous components. There are two ways of introducing superfluous components to overfit a mixture model, and the subsequent estimation of K_{true} should reflect which way is taken. First, the superfluous components can be arbitrarily introduced at regions with no data points ("leave some groups empty"), and assigned zero mixing proportions. Then, the number of non-empty components in the fitted mixture provides a good estimate of K_{true}. Second, the spurious components can be superimposed on some of the existing components ("let two component-specific parameters be identical"), and assigned positive mixing proportions. Here, the total number of components in the fitted mixture, after accounting for model complexity (via some model selection criterion), provides a reasonable estimate for K_{true}. Note that the prior distribution of the mixing proportion $\boldsymbol{p}$ affects the way overfitting is handled by the posterior, and hence the associated prior hyper-parameters need to be wisely chosen (see Section 2.7). A detailed discussion on the estimation of the number of components can be found in Frühwirth-Schnatter (2011, Section 10.3.1).

In **BAMBI** we assume that the superfluous components are introduced in the second ("let two component-specific parameters be identical") way. Consequently, K_{true} is estimated by first incrementally fitting the data with one additional component (starting from $K = 1$), until a model with $K + 1$ component fails to improve upon the previous fit with K component (as determined through a model selection criterion); that value of K is then used as an estimate of K_{true}. There exist multiple model selection criteria in the literature; we review six such criteria implemented in **BAMBI** and comment on their applicability in MCMC simulations. In the following, $\boldsymbol{\eta} = (\boldsymbol{\theta}, \boldsymbol{p})$ denotes the entire parameter vector, and $\boldsymbol{y} = (y_1, \ldots, y_n)$ is the vector/matrix of n independent observations.

1. *Watanabe-Akaike Information Criterion (WAIC)* (Watanabe (2013); Gelman, Hwang, and Vehtari (2014)). Given the dataset $y_1, \ldots, y_n$, the Markov chain realizations $\{\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_N\}$ of the parameter vector, and the pointwise densities $\{p(y_i \mid \boldsymbol{\eta}_s) : i = 1, \ldots, n;\ s = 1, \ldots, N\}$, define the computed log pointwise posterior predictive density

$$\text{LPPD} = \sum_{i=1}^{n} \log \left(\frac{1}{N} \sum_{s=1}^{N} p(y_i \mid \boldsymbol{\eta}_s) \right).$$

Then WAIC is defined as

$$\text{WAIC} = \text{LPPD} - p_W$$

where p_W is a correction term to adjust for effective number of parameters. Two forms for the adjustment terms are proposed in the literature, both being approximations based on Bayesian cross validation. In the first approach, (computed) p_W is defined as

$$p_W = 2\sum_{i=1}^{n}\left[\log\left(\frac{1}{N}\sum_{s=1}^{N} p(y_i \mid \boldsymbol{\eta}_s)\right) - \frac{1}{N}\sum_{s=1}^{N}\log\, p(y_i \mid \boldsymbol{\eta}_s)\right]$$

whereas, in the second approach, (computed) p_W is defined by $p_W = \sum_{i=1}^{n} \widehat{\text{var}} \log p(y_i \mid \boldsymbol{\eta})$, where for $i = 1, \ldots, n$, $\widehat{\text{var}} \log p(y_i \mid \boldsymbol{\eta})$ denotes the estimated variance of $p(y_i \mid \boldsymbol{\eta})$ based on the realizations $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_N$.

2. *Leave One Out Cross Validation Information Criterion (LOOIC)* (Vehtari, Gelman, and Gabry 2017). Under the same set-up as WAIC, the LOOIC is defined as

$$\text{LOOIC} = \sum_{i=1}^{n}\log\left(\frac{\sum_{s=1}^{N} w_i^s p(y_i \mid \boldsymbol{\eta}_s)}{\sum_{s=1}^{N} w_i^s}\right)$$

   where for each $s = 1, \ldots, N$, $\boldsymbol{w}^s = (w_1^s, \ldots, w_n^s)$ is a vector of importance sampling weights, typically calculated via the Pareto smoothed importance sampling method (PSIS; Vehtari, Gelman, and Gabry 2015). Because of the importance sampling weights, LOOIC can be more stable in practice than WAIC. See Vehtari *et al.* (2017) for a gentle and thorough introduction to both WAIC and LOOIC, including applications and case studies.

   Because both WAIC and LOOIC are based on the mixture likelihood and do not explicitly depend on the sampled model parameters (thus, remain unaffected by the presence of multiple permutation and non-permutation modes), in **BAMBI** we recommend using either of these two criteria for selecting the number of mixture components. Both WAIC and LOOIC are made available in **BAMBI** via their implementations in the R package **loo** (Vehtari, Gabry, Magnusson, Yao, Bürkner, Paananen, and Gelman 2020), which also provides a `compare()` function for comparing WAICs/LOOICs based on estimated difference in expected log predictive density (ELPD). In **BAMBI**, during an incremental model fitting via `fit_incremental_angmix` with `crit = "WAIC"` or `crit = "LOOIC"`, a test of hypothesis H_{0K} : ELPD for the fitted model with K components $\geq$ ELPD for the fitted model with $K + 1$ components, is performed at every $K \geq 1$. The test statistic used for the test is an approximate z score based on the normalized estimated ELPD difference between the two models (obtained from `compare()`, which provides estimated ELPD difference along with its standard error). The incremental fitting stops if H_{0K} cannot be rejected at a level `alpha` (defaults to 0.05, adjusted for multiplicity) for some $K \geq 1$; this K is then regarded as the optimum number of components.

3. *Marginal Likelihood (ML).* Marginal likelihood is arguably the most natural and intuitive model selection criterion that is used in the Bayesian paradigm. As the name suggests, marginal likelihood is the likelihood obtained by integrating out the parameters from the joint density of the data and parameters, and provides a natural way of measuring the "marginal" effect of data. In the context of Bayesian model selection, marginal likelihood provides a way of selecting an optimum model in that the model with largest marginal likelihood provides the best fit. Given the likelihood $L(\boldsymbol{y} \mid \boldsymbol{\eta})$ and prior density $\pi(\boldsymbol{\eta})$, marginal likelihood is the constant (dependent only on the data):

$$m(\boldsymbol{y}) = \int_{\mathcal{E}} L(\boldsymbol{y} \mid \boldsymbol{\eta})\pi(\boldsymbol{\eta})\, d\boldsymbol{\eta}$$

where $\mathcal{E}$ denotes the support of the parameter vector $\boldsymbol{\eta}$. Note that $m(\boldsymbol{y})$ is the reciprocal of the normalizing constant required to define the posterior density. Evaluation of the marginal likelihood $m(\boldsymbol{y})$ in practice however is typically challenging, as it tends to be a high-dimensional intractable integral (as in our case). Multiple estimation techniques based on samples from the posterior density $\pi(\boldsymbol{\eta} \mid \boldsymbol{y})$ have been proposed in the literature; in **BAMBI** we implement bridge sampling (Meng and Wong 1996; Meng and Schilling 2002). Briefly, the key idea is to first write $m(\boldsymbol{y})$ as

$$m(\boldsymbol{y}) = \frac{\int_{\mathcal{E}} h(\boldsymbol{\eta}) L(\boldsymbol{y} \mid \boldsymbol{\eta}) \pi(\boldsymbol{\eta}) g(\boldsymbol{\eta}) \, d\boldsymbol{\eta}}{\int_{\mathcal{E}} h(\boldsymbol{\eta}) g(\boldsymbol{\eta}) \pi(\boldsymbol{\eta} \mid \boldsymbol{y}) \, d\boldsymbol{\eta}} = \frac{E_g \left[h(\boldsymbol{\eta}) L(\boldsymbol{y} \mid \boldsymbol{\eta}) \pi(\boldsymbol{\eta})\right]}{E_{\pi(\cdot \mid \boldsymbol{y})} \left[h(\boldsymbol{\eta}) g(\boldsymbol{\eta})\right]}$$

where g is a density, called the *proposal density*, and h is a function, called the *bridge function*. Then one approximates the above ratio by

$$\hat{m}(\boldsymbol{y}) = \frac{\frac{1}{n_2} \sum_{j=1}^{n_2} h(\boldsymbol{\eta}_j^*) L(\boldsymbol{y} \mid \boldsymbol{\eta}_j^*) \pi(\boldsymbol{\eta}_j^*)}{\frac{1}{n_1} \sum_{j=1}^{n_1} h(\boldsymbol{\eta}_j^\dagger) g(\boldsymbol{\eta}_j^\dagger)}$$

where $\boldsymbol{\eta}_1^\dagger, \ldots, \boldsymbol{\eta}_{n_1}^\dagger$ are MCMC samples from the posterior density $\pi(\cdot \mid \boldsymbol{y})$ and $\boldsymbol{\eta}_1^*, \ldots, \boldsymbol{\eta}_{n_2}^*$ are samples from the proposal density g. Note that h and g play crucial roles in the estimation of $m(\boldsymbol{y})$, and must be optimally chosen for accurate results. See Gronau *et al.* (2017) for a gentle and detailed tutorial on bridge sampling. In **BAMBI**, marginal likelihood can be used to select the optimal number of mixture component in a `fit_incremental_angmix` run, by specifying `crit = "LOGML"`. This will ensure computation of the log marginal likelihood via bridge sampling for every mixture model during the incremental run, and the model attaining the first minimum negative log marginal likelihood will be treated as the optimum model.

It should however be noted that for mixture models, optimal selection of h and g is typically difficult due to the multi-modality of the posterior density; see Frühwirth-Schnatter (2006, Chapter 5), for a review of some of the available methods. In **BAMBI**, marginal likelihood is computed by leveraging the function `bridge_sampler` from the R package **bridgesampling** (Gronau, Singmann, and Wagenmakers 2020), and the authors of **bridgesampling** warn against the use of `bridge_sampler` in mixture models (see the discussion section in Gronau *et al.* 2020). As such, use of this method in **BAMBI** is not recommended, even though the functionality is available.

4. *Akaike Information Criterion (AIC, Akaike 1974).* Let $\hat{L}$ be the maximum value of the likelihood function for the model and let m be the number of estimated parameters in the model. Then AIC is defined as

$$\text{AIC} = -2 \log \hat{L} + 2m.$$

5. *Bayesian Information Criterion (BIC, Schwarz 1978).* Under the same setup, if n denotes the number of data points, BIC is defined by

$$\text{BIC} = -2 \log \hat{L} + m \log(n).$$

Observe that both AIC and BIC depend on the maximum value $\hat{L}$, which, in general, is not directly available in MCMC simulations. A possibly suboptimal estimate of the global maximum is given by the maximum value of the likelihood function computed at the MCMC samples.

During model selection, the model with minimum AIC (or BIC) can be treated as the optimal model. In **BAMBI** AIC/BIC can be used for selecting optimum number of components in an `fit_incremental_angmix` run by specifying `crit = "AIC"` or `crit = "BIC"`. This will ensure computation of AIC/BIC of every mixture model fitted during the incremental fitting; the model attaining the first minimum AIC/BIC will be treated as the optimum model.

It should to be noted, however, that AIC and BIC are both based on asymptotic normality results that *do not* hold for mixture models with multiple modes, and hence their use in selecting the number of mixture components may lead to inconsistent results. Thus, though implemented, using AIC or BIC is not recommended in **BAMBI**.

6. *Deviance Information Criterion (DIC, Spiegelhalter, Best, Carlin, and Van der Linde 2002).* DIC is another model selection criterion, which, similar to AIC and BIC, is based on an asymptotic result for large samples. Let $D(\boldsymbol{\eta}) = -2 \log p(\text{data} \mid \boldsymbol{\eta})$ denote the deviance, where $\boldsymbol{\eta}$ denotes the vector of all parameters in the model and $p(\text{data} \mid \boldsymbol{\eta})$ denotes the likelihood. Let $\{\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_N\}$ denote the MCMC realizations of the parameters. Define (estimated) effective number of parameters p_D by $p_D = \bar{D}(\boldsymbol{\eta}) - D(\bar{\boldsymbol{\eta}})$, where $\bar{D}(\boldsymbol{\eta}) = N^{-1} \sum_{s=1}^{N} D(\boldsymbol{\eta}_s)$ and $\bar{\boldsymbol{\eta}} = N^{-1} \sum_{s=1}^{N} \boldsymbol{\eta}_s$. Another commonly used form for p_D is given by $p_D = \widehat{\text{var}}\, D(\boldsymbol{\eta})/2$, (Gelman, Carlin, Stern, Dunson, Vehtari, and Rubin 2013) where $\widehat{\text{var}}\, D(\boldsymbol{\eta})$ denotes the estimated variance of $D(\boldsymbol{\eta})$ based on $\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_N$. Then DIC is defined as

$$\text{DIC} = p_D + \bar{D}(\boldsymbol{\eta}) = D(\bar{\boldsymbol{\eta}}) + 2p_D.$$

In **BAMBI** AIC/BIC can be used for selecting optimum number of components in an `fit_incremental_angmix` run by specifying `crit = "DIC"`. This will ensure computation of DIC of every mixture model fitted during the incremental fitting; the model attaining the first minimum DIC will be treated as the optimum model.

It should be noted that use of DIC can be unstable in practice. For example, if the first form of p_D, i.e., $p_D = \bar{D}(\boldsymbol{\eta}) - D(\bar{\boldsymbol{\eta}})$) is used, DIC becomes heavily dependent on the plug-in estimator $\bar{\boldsymbol{\eta}}$. However, in Bayesian mixture modeling the posterior mean is not always a suitable plug-in estimator for the parameter vector as it may lie between different modes of the posterior density (Plummer 2008). The problem is exacerbated by the presence of label switching in the MCMC samples (see Section 2.9). Moreover, depending on how the information on latent component indicators are handled, multiple versions of DIC can be constructed here. Celeux, Forbes, Robert, and Titterington (2006) consider no less than eight variants, but are unable to recommend any of them for practical use. Likewise we caution against the use of DIC, although the functionality is available in **BAMBI**.

2.9. Label switching

Label switching is a fundamental aspect of Bayesian mixture modeling that requires proper care. Briefly, when exchangeable priors $\pi(\boldsymbol{\theta}_j)$'s are placed on the model parameters $\boldsymbol{\theta}_j$'s, the resulting posterior distribution becomes invariant to permutation in the component label j's. As a result, the posterior density consists of symmetric or permutation modes that are identical up to permutation of component labels. A well mixing MCMC algorithm will explore these permutation modes, causing the component labels to switch over the course of an MCMC simulation. This phenomenon is called label switching, and is required in MCMC-based Bayesian mixture modeling to justify convergence. A fundamental limitation of MCMC-based Bayesian mixture modeling is that the chains may become trapped at local modes, rather than fully exploring the symmetric modes. One possible remedy is to run multiple independent chains to improve exploration of the posterior. Alternatively, one may embed a deliberate random relabeling step into the sampler, i.e., adopt a so-called *permutation sampling* scheme (Frühwirth-Schnatter 2001): after each draw of the random allocation, components are relabeled according to a random permutation of $1, \ldots, K$. This is, in fact, a specific example of a *sandwich algorithm* (Meng and Van Dyk 1999; Yu and Meng 2011; Hobert, Roy, and Robert 2011), where a computationally inexpensive step (of drawing a random permutation of $\{1, \ldots, K\}$, and then relabeling the components according to that random permutation) is *sandwiched* in between the two steps (drawing the allocation vector, and drawing the component-specific parameters) of a Data Augmentation algorithm. Sandwich algorithms often converge faster than the original Data Augmentation algorithm; in the case of permutation sampling, the chain is forced to visit the permutation modes (and potentially the non-permutation modes) more frequently. These potential improvements in convergence would not be achieved by simply randomly switching labels of the MCMC samples post-hoc. In permutation sampling, inclusion of the random label switching step results in a modified MCMC algorithm that is theoretically proven to be at least as good as the original Data Augmentation algorithm in terms of convergence rates (Khare and Hobert 2011). However, care must be taken if RWMH or HMC updates for the component specific parameters are adaptively tuned according to the scales and variabilities of the sampled model parameters (to do so properly requires keeping track of each component label). In **BAMBI** permutation sampling can be done after burn-in, by setting `perm_sampling = TRUE` (defaults to `FALSE`) in a `fit_angmix` call.

Although label switching is required for MCMC convergence in Bayesian mixture modeling, its presence in MCMC samples makes inference on the different components via posterior means or quantiles challenging (note that MAP estimation is not affected). A number of techniques have been proposed to handle this problem; see, e.g., Jasra, Holmes, and Stephens (2005) and Rodríguez and Walker (2014). The available methods either need to be applied during MCMC sampling (*on-line*) or after simulating the entire chain (*post processing*).

Several post-processing techniques that undo label switching are implemented in the useful R package **label.switching** (Papastamoulis 2016), and in **BAMBI**, we provide a wrapper called `fix_label` for the main `label.switching` function from that package. All the methods available in `label.switching` are appropriately implemented in `fix_label`, which takes an '`angmcmc`' object (see Section 3.1) as input, and may require additional user inputs depending on the method used. The Kullback-Leibler divergence based method by Stephens (2000) (`method = "STEPHENS"`) is used by default if the permutation sampling is performed during original MCMC run; otherwise, the default method is the data-based algorithm of Rodríguez and Walker (2014) (`method = "DATA-BASED"`); neither requires any additional input other than an '`angmcmc`' object.

2.10. Initialization of the parameters, and the use of multiple chains

MCMC algorithms can converge faster if the initial values are chosen well. The function `fit_angmix`, when called without supplying starting parameter values, will automatically initialize the latent allocation to the mixture components. The default (and recommended) option is initialization via a k-means algorithm: toroidal angle pairs are first projected onto the surface of a unit sphere, and then Cartesian coordinates of the projected spherical points are clustered. Random initial allocation is also provided as an option, but is not recommended as it may lead to slow convergence. Once an initial allocation is obtained, component specific parameters are estimated via method of moments (see Jammalamadaka and Sengupta 2001; Singh *et al.* 2002; Mardia *et al.* 2007, for more details on these estimators), and the mixing proportions are estimated by the sample proportions.

When explicit starting values of the model parameters and mixing proportions are provided to `fit_angmix`, no initial allocation is necessary. This is particularly useful for estimating the number of components, when mixture models are being fitted incrementally (e.g., via `fit_incremental_angmix`). Under incremental model fitting the parameters of a $K+1$ component mixture can be initialized directly from the parameter estimates from a K component mixture; the extra component is simply taken as a "copy" of an existing component (preferably the one with the largest mixing proportion), and the associated mixing proportion is distributed equally between the two identical components. This method is expected to work well when the posterior density handles overfitting in the "let two component-specific parameters be identical" way (see Sections 2.7 and 2.8), which is the approach taken in **BAMBI**. As such, this is the default method of initializing parameters in `fit_incremental_angmix` when $K > 2$; however k-means allocation followed by moment estimation can also be used, by setting `prev_par = FALSE`.

Finally, we note that even with good initial values, MCMC samplers can still get trapped in local modes for a large number of iterations, rather than fully exploring the posterior density. One possible remedy is to run multiple independent chains to improve exploration of the posterior, which is implemented in **BAMBI**. The argument `n.chains` (set to 3 by default) specifies the number of independent chains to run in `fit_angmix`. These chains can be run in parallel, see Section 3.3 for more details.

3. BAMBI Package

This section overviews the functionalities of **BAMBI**. At the core of the package is the '`angmcmc`' object, which is created when a model fitting function is used. In the following we first describe the '`angmcmc`' objects, then describe the data sets included in **BAMBI**, and finally discuss the functions available in **BAMBI** and comment on their usability. However, this is not an exhaustive manual; all functions in **BAMBI** include R documentations, which should serve as the definitive resources.

3.1. '`angmcmc`' objects

'`angmcmc`' objects are classified lists belonging to the S3 class '`angmcmc`' that are created when the function `fit_angmix` is used. An '`angmcmc`' object contains a number of elements, including the dataset and its dimension (i.e., univariate or bivariate), the model being fitted, the tuning parameters used, MCMC samples of the parameter vector, and at each iteration the (hidden) component indicators for data points, log-likelihood and log posterior density values (up to additive constants). When printed, an '`angmcmc`' object returns a brief summary of the function arguments used and the acceptance rate of the proposal states (in HMC and RWMH). An '`angmcmc`' object can be used as an argument for the diagnostic and post-processing functions available in **BAMBI** for making further inferences.

3.2. Data sets

BAMBI contains two illustrative data sets, namely `wind` (univariate) and `tim8` (bivariate), each measured in the radian scale $[0, 2\pi)$.

`wind` This dataset consists of 239 observations on wind direction (originally measured in 10s of degrees, and then converted into radians) measured at Saturna Island, British Columbia, Canada during October 1–10, 2016 (obtained from Environment Canada website). There was a severe storm during October 4-7 in Saturna Island, which caused significant fluctuations in wind direction.

`tim8` This consists of 490 pairs of backbone dihedral angles (ϕ, ψ) for *Triose Phosphate Isomerase* (8TIM). The three dimensional structure of 8TIM is available from the protein data bank (PDB). The protein is an example of a TIM barrel, a common type of protein fold exhibiting alternating α-helices and β-sheets. The backbone angles for this protein were obtained by using the DSSP software (Touw, Baakman, Black, Te Beek, Krieger, Joosten, and Vriend (2015); Kabsch and Sander (1983)) on the PDB file for 8TIM, and then converted into radians.

3.3. Functions

In **BAMBI**, all five models described in Section 1, namely the univariate von Mises (`vm`), univariate wrapped normal (`wnorm`), bivariate von Mises sine (`vmsin`), bivariate von Mises cosine (`vmcos`) and bivariate wrapped normal (`wnorm2`), and their (within same model) mixtures are implemented. The functions in **BAMBI** can be classified into six major categories.

Density and random samples from an angular distribution

The functions `dvm`, `dwnorm`, `dvmsin`, `dvmcos` and `dwnorm2` evaluate the density and the functions `rvm`, `rwnorm`, `rvmsin`, `rvmcos` and `rwnorm2` generate random samples from the models `vm`, `wnorm`, `vmsin`, `vmcos` and `wnorm2` respectively. The parameters of the models are specified as arguments; otherwise, default values (zero means, unit concentrations, and zero association) are used.

Density evaluations require computation of the normalizing constants, which for the `vmcos` model requires proper care, especially when κ_1, κ_2 or $|\kappa_3|$ is large. This is because the analytic expression involves infinite (alternating if $\kappa_3 < 0$) series of product of modified Bessel functions, which become numerically unstable when these parameters are large. As such, when $\kappa_3 < -5$ or $\max(\kappa_1, \kappa_2, |\kappa_3|) > 50$, the reciprocal of the integral for the normalizing constant is evaluated numerically using a (quasi) Monte Carlo method. By default, `n_qrnd` `=` 10^4 pairs of Sobol numbers are used for this purpose; however, `n_qrnd`, or a two-column matrix `qrnd` containing random/quasi-random numbers between 0 and 1 can be supplied for this approximation[3]. For `vmsin` model, evaluation of the constant via its analytic form is much more stable, as the associated infinite series consists only of non-negative terms. For univariate and bivariate wrapped normal models, the default absolute integer displacement for approximating the wrapped normal sum is 3, which can be changed to any value in $\{1, 2, 3, 4, 5\}$, through the argument `int.displ`. Note that `int.displ` regulates how many terms would be used to approximate the infinite sum present in the univariate and bivariate wrapped normal densities in (3) and (4). For example, `int.displ` $= M$ implies that the infinite sum in the univariate wrapped normal density will be approximated by a finite sum of $2M + 1$ values, with the summation index ω ranging over $\{0, \pm 1, \ldots, \pm M\}$. For a bivariate wrapped normal density, setting `int.displ` $= M$ will ensure that the infinite double sum is approximated by a finite double sum, with the paired summation index (ω_1, ω_2) ranging over $\{0, \pm 1, \ldots, \pm M\}^2$.

Random data generation from the von Mises models (both univariate and bivariate) is done via rejection sampling. In the univariate case, the von Mises random deviates are efficiently generated using a rejection sampling scheme from a wrapped Cauchy distribution (Best and Fisher 1979; Mardia and Jupp 2009). For the bivariate models, two forms of random samplings are implemented. In the first method, random deviates are generated via a naive bivariate rejection sampler with uniform proposal density (the majorization constant is numerically evaluated). In the second method (proposed in the web appendix of Mardia *et al.* (2007)), random deviates are first generated from the marginal distribution of one coordinate, then the other coordinate is drawn from the corresponding conditional distribution (which is von Mises in both models). The authors note that this latter scheme has a typical efficiency rate of at least 60%. It is to be noted that while this scheme is usually more efficient than the naive rejection sampler (especially when the concentration is high), it does have an often substantial overhead due to the numerical computations required for determining appropriate proposal density parameters. These overheads often outweigh efficiency gains, especially if the sample size and/or concentration parameters are small. In **BAMBI**, therefore, the naive rejection sampler is used by default when the sample size is moderate or small (< 100), or when the concentration parameters are small (< 0.1)[4]. For wrapped normal distributions (both univariate and bivariate) a random deviate is easily obtained by sampling from the unwrapped normal distribution (using `rnorm` if univariate, and `rvmnorm` from package **mvtnorm** (Genz, Bretz, Miwa, Mi, and Hothorn 2021) if bivariate), and then wrapping into $[0, 2\pi)$.

Density and random samples from a finite mixture model

Analogous to the functions for single component densities, the functions `dvmmix`, `dwnormmix`, `dvmsinmix`, `dvmcosmix` and `dwnorm2mix` evaluate the density and the functions `rvmmix`, `rwnormmix`, `rvmsinmix`, `rvmcosmix` and `rwnorm2mix` generate random samples from mixtures of `vm`, `wnorm`, `vmsin`, `vmcos` and `wnorm2` respectively. All model parameters and mixing proportions must be provided as input arguments.

[3] The user may perform a Monte Carlo approximation for the normalizing constant even when numerical evaluation of the analytic formula is stable, by changing `force_approx_const` to `TRUE` from its default value `FALSE`.

[4] When the concentration parameters are large, the density becomes concentrated in (a) very narrow region(s). As such, efficiency of the naive sampler, which draws proposal random deviates from a uniform density over the entire support, can be 15-20% or less. However, even then the overall run times of the naive method are often still comparable to Mardia *et al.* (2007)'s method when the sample size is moderate or small.

Visualizing and summarizing bivariate models

To visualize the density for any of the three bivariate angular mixture models (with specified parameters and number of components) considered in this paper, the functions `surface_model` and `contour_model` can be used, which respectively plot the surface and the contour of a mixture density. To compute summary statistics for a single bivariate angular distribution, the function `circ_varcor_model` can be used, which calculates the circular variance and correlation coefficients (both Jammalamada-Sarma and Fisher-Lee forms, see Section 1.3). However, summarizing angular mixture models via circular variances and correlations is not recommended, as interpretations of the results can be challenging when multiple clusters are present in the data[5]. The function `circ_cor` implements the sample Jammalamada-Sarma and Fisher-Lee circular correlation coefficients, as well as two forms of Kendall's tau (Fisher and Lee 1983; Zhan, Ma, Liu, and Shimizu 2019) as non-parametric measures. The sample circular variance can be computed using the `var.circular` function from R package **circular** (Agostinelli and Lund 2017).

Fitting a single component model or a finite mixture model using MCMC

Given a dataset, using methods discussed in Section 2, the function `fit_angmix` generates MCMC samples for parameters in an angular mixture model with a specified number of components. Available models for bivariate input data (which must be supplied as a two-column matrix or data frame) are `vmsin`, `vmcos` and `wnorm2`, and for univariate data are `vm` and `wnorm`. The argument `ncomp` specifies number of components in the mixture model, with `ncomp = 1` representing the single component case (i.e., fitting a single density). A Gibbs sampler is used to generate latent component indicators, and conditional on this allocation the model parameters are sampled either by HMC (default), or by RWMH (can be specified through the argument `method`). A permutation sampling step can be added after burn-in by setting the logical argument `perm_sampling` to `TRUE`. The tuning parameters `epsilon` and `L` in HMC, and `propscale` in RWMH have pre-specified default values, and there is an auto-tuning feature for `epsilon` and `propscale` which is used by default, but can be turned off by setting the logical argument `autotune = FALSE`. The burn-in proportion can be specified through the argument `burnin.prop`, which is set to 0.5 by default. For HMC, the option to use random `epsilon` and `L` at each iteration is specified via the logical arguments `epsilon.random` and `L.random` respectively. In practice, using multiple chains is recommended, and the argument `n.chains` specifies the number of chains to be used. These chains can be run in parallel, if the logical argument `chains_parallel` is set to `TRUE`. The parallelization is implemented using `future_lapply` from R package **future.apply** (Bengtsson 2021b); an appropriate `future::plan()` must be set in advance to ensure that the chains run in parallel (otherwise the chains will run sequentially), see Section 4 for an example. To retain reproducibility while running multiple chains in parallel, the *same* RNG state is passed at the beginning of each chain. This is done by specifying `future.seed = TRUE` in `future.apply::future_lapply` call. Then at the beginning of the i-th chain, before drawing any parameters, i-many Uniform(0, 1) random numbers are generated using `runif(i)` (and then thrown away). This ensures that the RNG states across chains prior to sampling of the parameters are different (but reproducible), and hence, no two chains can become identical, even if they have the same starting and tuning parameters. This however, creates a differ-

[5] To calculate the circular variances and correlations for a mixture density, one can simulate from the density first, and then approximate the population quantities by their sample analogs on the basis of the simulated data.

ence between a `fit_angmix` call with multiple chains which is run sequentially by setting `chains_parallel = FALSE`, and another call which is run sequentially because of a sequential `plan()` (or no `plan()`), with `chains_parallel = TRUE`. In the former, `base::lapply` instead of `future_lapply` is used, which means that *different* RNG states are passed at the initiation of each chain.

There are options for choosing prior hyperparameters. The prior for the association parameter κ_3 in bivariate models, and the log of the concentration parameters κ (in univariate models), κ_1 and κ_2 (in bivariate models) are taken to be the normal distribution (i.e., the priors for $\kappa, \kappa_1, \kappa_2$ are log normal), all with zero mean. The default variance for the normal prior is 1000, which provide diffuse priors, although they can be set by the user via the argument `norm.var`. A fixed non-informative Uniform$(0, 2\pi)$ prior is used for the mean parameters. The Dirichlet prior parameters α_j's for the mixing proportions p_j's can be supplied through the argument `pmix.alpha`, which can either be a positive real number (same for all α_j), or a vector of the same length as `pmix`. It is recommended that α_j's be chosen large for proper handling of overfitted mixtures; following Frühwirth-Schnatter (2011, Section 1.3.2), all α_j's default to $(r + r(r+1)/2)/2 + 3$, where r denotes dimension of the data (i.e., $r = 1$ for univariate data, and $r = 2$ for bivariate data). See Sections 2.7 and 2.8 for more details.

The argument `cov.restrict` specifies any (additional) restriction to be imposed on the component specific association parameters while fitting the model. The available choices are `"POSITIVE", "NEGATIVE", "ZERO"` and `"NONE"`. Note that when `cov.restrict = "ZERO"`, `fit_angmix` fits a mixture with product components. By default, `cov.restrict = "NONE"`, which does not impose any (additional) restriction. When `model` is `"vmsin"` or `"vmcos"`, the component densities can be bimodal. However, one can restrict these densities to be unimodal, by setting the logical argument `unimodal.component` to `TRUE` (defaults to `FALSE`). For `"wnorm"` and `"wnorm2"` models, the default absolute integer displacement for approximating the wrapped normal sum is 3, which can be changed to any value in $\{1, 2, 3, 4, 5\}$, through the argument `int.displ`. For `"vmcos"` model, the normalizing constant is numerically approximated using quasi Monte Carlo method when analytic evaluation suffers from numerical instability. The arguments `qrnd` and `n_qrnd` can be used to alter the default settings used for these approximations. See the documentation of `fit_angmix` for more details.

The function `fit_angmix` creates an '`angmcmc`' object, which can be used for assessing the fit, post processing, and estimating parameters.

Assessing the fit

Goodness of fit for MCMC-based Bayesian modeling depends on both convergence of the Markov chain and the appropriateness of the model used. **BAMBI** contains a number of functions that can be used to examine these two aspects. The functions `paramtrace` and `lpdtrace` respectively plot the parameter and log posterior density traces for visual assessment of convergence. These two functions are called together in the `plot` method for '`angmcmc`' objects. The `as.mcmc.list` method for '`angmcmc`' objects provides a convenient way of converting an '`angmcmc`' object to an '`mcmc.list`' object from package **coda**, which provides several additional functions for convergence diagnostics.

Once convergence is justified, the appropriateness of the fitted model can be visually assessed by the S3 functions `densityplot` from **lattice** (Sarkar 2021) and `contour`. The first function plots the density surface (for bivariate data) or density curve (for univariate data) of the fitted mixture model, and the second plots the associated contour of a bivariate model. Note that these plots provide visual assessment of the goodness of fit by assuming the Markov chains have converged and the parameters can be estimated on the basis of the MCMC samples. As such, convergence of the MCMC samples must be ensured prior to this step. Otherwise these visual diagnostics will lead to misleading conclusions.

The comparative goodness of fit for two mixture models can be assessed on the basis of model selection criteria implemented in **BAMBI**, namely, marginal likelihood, AIC, BIC, DIC, WAIC and LOOIC via the functions `bridge_sampler.angmcmc`, `AIC`, `BIC`, `DIC`, `waic.angmcmc` and `loo.angmcmc`. As with the diagnostic plots, one general caveat for using any of these model selection criteria is that one should ensure convergence of the associated Markov chain first; otherwise, the results may be misleading.

Post-processing and estimating parameters

BAMBI provides several post-processing functions to aid inference on the basis of the generated MCMC samples. The function `add_burnin_thin` adds additional burn-in and/or thinning to an '`angmcmc`' object and the function `select_chains` extracts a subset of chains. These two functions can be helpful if convergence diagnostics indicate that some of the chains are poor mixing and/or require additional burn-in and thinning for convergence.

As described in Section 2.9, care should be taken to ensure that there is no label switching in the MCMC output if inference is being made on the basis of posterior mean/median. If present, label switching can be fixed by applying the wrapper function `fix_label` on the '`angmcmc`' object, which will output another '`angmcmc`' object with label switching fixed.

Point estimates of the parameters are obtained using the **BAMBI** function `pointest` on a fitted '`angmcmc`' object. The function `pointest` calculates point estimates by applying `fn` on the MCMC samples, where `fn` is either a function, or a character string specifying the name of the function. Default for `fn` is `mean`, which computes posterior mean. If `fn` is `"MODE"` or `"MAP"` then the (MCMC-based approximate) MAP estimate is returned. Posterior quantiles can be estimated by (sample) quantiles of the MCMC realizations using the S3 function `quantile.angmcmc`. These quantiles can be used to construct credible sets. For example, if ξ_ζ denotes the ζ-th (sample) quantile of the MCMC observations for $0 < \zeta < 1$, the central 95% credible interval is given by $(\xi_{0.025}, \xi_{0.975})$. Both of these functions can be applied on specific parameters and/or component labels by setting the arguments `par.name` and `comp.label` accordingly. The S3 function `summary.angmcmc` prints (estimated) posterior means and the central 95% credible intervals for all the parameters.

The estimated latent allocation from an '`angmcmc`' object can be obtained using the function `latent_allocation`, which first estimates parameters via `pointest`, computes posterior membership probabilities (see (14)) for each data point, and then assigns each data point the class with largest membership probability. The (estimated) log-likelihood of an '`angmcmc`' object can be extracted as a '`logLik`' object using the S3 function `logLik.angmcmc`. Note that there are two methods of obtaining the log-likelihood from an '`angmcmc`' object. In the default method (`method = 1`), the final log-likelihood is computed by applying a function `fn` (defaults to `max`) on the iteration wise log-likelihood values obtained during the original MCMC sampling. On the other hand, if `method = 2`, first the parameters are estimated (using `pointest`), and then the log-likelihood is computed at the estimated parameters.

Density evaluations and random data generation from a fitted model can be done using the functions `d_fitted` and `r_fitted` respectively. Both functions take an '`angmcmc`' object as input and apply the appropriate model specific density evaluation and random data generation functions with the estimate $\hat{\boldsymbol{\eta}}$ of the parameter vector $\boldsymbol{\eta}$ obtained via `pointest`. The actual MCMC samples for one or more parameters in one or more components from one or more chains can be extracted via `extractsamples` on an '`angmcmc`' object for further analysis.

Incremental mixture model fitting and number of components estimation

Using the methods and model selection criteria described in Sections 2.7 and 2.8, the function `fit_incremental_angmix` fits mixture models with incremental number of components by calling `fit_angmix` at each step, and uses a Bayesian model selection criterion to determine an optimal number of components. The arguments `start_ncomp` and `max_ncomp` provide the starting and maximum number of components to be used in the incremental fitting, which are respectively set to 1 and 10 by default. The available model selection criterion to use (specified via the argument `crit`) are `"LOGML"`, `"AIC"`, `"BIC"`, `"WAIC"`, `"LOOIC"` and `"DIC"`, which is computed for every intermediate fit. The initial values of the starting model (or a model with ≤ 2 components) are obtained by default using moment estimation on k-means clusters (they can also be directly supplied by the user). For the subsequent models (with number of components ≥ 3), the initial values are by default obtained from the MCMC-based MAP parameter estimates for the previous model with one fewer component (see Section 2.8). This can be overridden by setting `prev_par = FALSE`, to use k-means clustering followed by moment estimation instead. By default, only the "best" chain, i.e., the one with maximum average log posterior density, is used for computation of model selection criterion, and parameter estimation (if `prev_par` is set to `TRUE`). This default helps safeguard against situations where some of the chains may get trapped at local optima. However, samples from all chains can be used for these computations by setting `use_best_chain = FALSE`. The function stops when `crit` achieves its first minimum, or when `max_ncomp` is reached, and returns a list with the following elements:

- `fit.best` is an '`angmcmc`' object corresponding to the optimum or *best* fit.
- `crit.all` provides a vector (list) of model selection criterion values for each incremental model fitted.
- `crit.best` is the value of the model selection criterion for the model with optimal number of components.
- `maxllik.all` is the maximum (obtained from MCMC iterations) log-likelihood for all fitted models.
- `maxllik.best` is maximum log-likelihood for the optimal model.
- `ncomp.best` is the optimal number of components associated with the "best" model.
- `fit.all` is a list consisting of '`angmcmc`' objects for all number of components fitted during the model selection process. Any element of this list can be used as an argument for any function that takes '`angmcmc`' objects as input. However, this can be very memory intensive, and as such, by default not returned (can be returned by setting `return_all = TRUE`).

The '`angmcmc`' object corresponding to the best fit and the associated value of the model selection criterion can also be extracted from the output of `fit_incremental_angmix` extracted using the convenience functions `bestmodel` and `bestcriterion` respectively.

4. Illustrations

In this section we illustrate functionalities of **BAMBI** by fitting mixture models to the angular data sets included in the package. The following command

```
R> library("BAMBI")
```

loads the package after it has been installed. For reproducibility of the results presented, the same random seed 12321 is used for all of our examples.

4.1. Fitting mixture models on the tim8 bivariate data

The `tim8` dataset consists of 490 backbone torsion angle pairs (ϕ, ψ) for the protein 8TIM. The protein is an example of a TIM barrel, a common type of protein fold exhibiting alternating α-helices and β-sheets. Its Ramachandran plot (i.e., a scatterplot of (ϕ, ψ) pairs) is generated using the following R command and shown in Figure 5.

```
R> plot(tim8, pch = 16, xlim = c(0, 2*pi), ylim = c(0, 2*pi),
+    main = "8TIM", col = scales::alpha("black", 0.6))
```

Note that this scatterplot projects the torus onto a 2D surface, which cannot show the wraparound nature of the angles. This projection is not unique and affects the appearance of the scatterplot depending on how the angles are represented, e.g., in $[-\pi, \pi)$ instead of $[0, 2\pi)$.

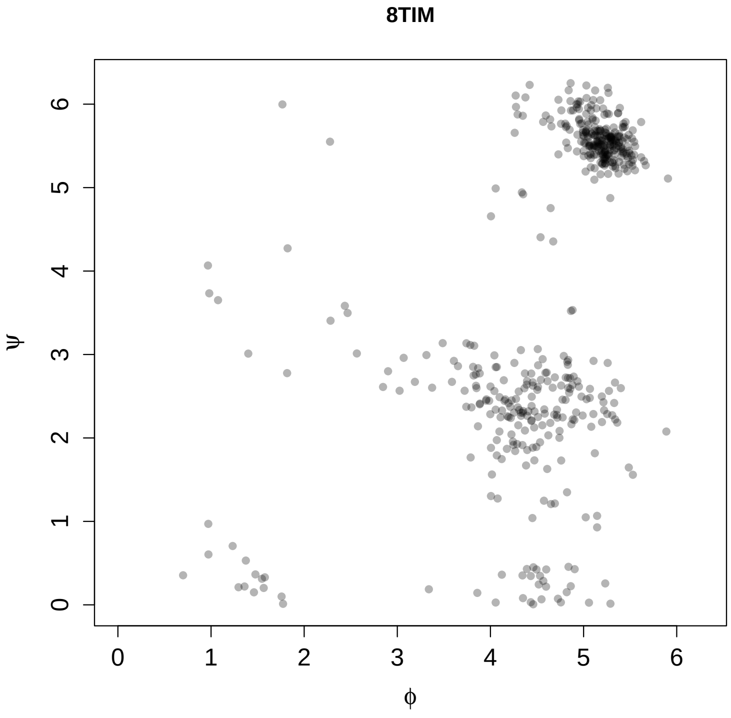


Figure 5: Ramachandran plot for 8TIM.

Moreover, one should be careful to note that the top and bottom boundaries in these plots join together, as do the left and right boundaries. In Figure 5, about 3-6 visually distinct clusters can be seen; however, we need to note that the points around (4.5, 0) and (5.5, 6) in fact may form a single cluster. Such features cannot be correctly modeled with statistical methods that ignore circularity. Thus, this is a suitable example for illustrating the need for mixtures of (bivariate) angular distributions.

To fit a bivariate mixture model with a specified number of components to the dataset, we can use the **BAMBI** function `fit_angmix` by specifying a model and the number of components to be used. For example, the following R command fits a 4 component `vmsin` mixture by generating 3 MCMC chains with 20,000 samples each for the mixture model parameters. HMC is used for sampling the model parameters, with tuning parameter `epsilon` adaptively tuned during burn-in (which is by default constituted by the first half of all iterations, i.e., first 10,000 iterations), and `L` taking its default value 10. A `fit_angmix` call creates an '`angmcmc`' object, which can then be used for various post-processing tasks, including convergence assessment, parameter estimation and visualizing goodness of fit.

```
R> set.seed(12321)
R> fit.vmsin.4comp <- fit_angmix("vmsin", tim8, ncomp = 4, n.iter = 2e4,
+    n.chains = 3)
```

Note that in order for the independent chains to be run in parallel, an appropriate `plan()` from R package **future** (Bengtsson 2021a) needs to be set first; otherwise the chains will run sequentially. For example, running the commands

```
R> library("future")
R> plan(multisession, gc = TRUE)
```

before the `fit_angmix` call will ensure that the three chains are run in parallel, provided resources are available. We suggest setting `gc = TRUE` in `plan`, to allow proper garbage collection from the parallel workers, even though it adds some overhead. This is because the parallel workers can end up leaving heavy memory footprints, especially when mixture models are being fitted incrementally.

In the previous example with `fit_angmix`, the number of components K was specified through `ncomp`. However, the "true" K generally needs to be estimated from the data. For this purpose, we use `fit_incremental_angmix` with `start_ncomp = 2`, which fits angular mixtures with incremental number of components (starting with 2 components), and uses a Bayesian model selection criterion to determine an optimal model. We use this function to fit optimal mixtures of `vmsin`, `vmcos` and `wnorm2` models separately. By specifying `n.chains = 3` and `n.iter = 20,000` in `fit_incremental_angmix`, each incremental model will be fitted using three chains with 20,000 iterations each (with first 10,000 iterations treated as burn-in, where `epsilon` in HMC is tuned).

By default, the algorithm uses MCMC-based MAP estimates from the preceding fitted model with one fewer component (if available) as starting parameter values (see Section 2.10). We use the default leave one out cross validation information criterion (`"LOOIC"`) as the model selection criterion for determining the optimal model. When the function stops, we extract the best fitted model using the function `bestmodel`, and assess convergence of the associated chains. After justifying convergence, we provide point and interval estimates of the parameters and visually examine goodness of fit.

Fitting the `vmsin` mixture model

We start with the `vmsin` model. The R commands are as follows:

```
R> set.seed(12321)
R> fit.vmsin <- fit_incremental_angmix(model = "vmsin", data = tim8,
+    crit = "LOOIC", start_ncomp = 2, max_ncomp = 10, n.iter = 2e4,
+    n.chains = 3)
```

The algorithm stops at 5 components and determines the optimal number of components to be 4 on the basis of the `"LOOIC"` values. The MCMC-based maximum log-likelihood estimates for the intermediate models are −945.4840, −853.5790, −803.3334, and −794.3692 which are steadily increasing. This is expected, since a "smaller" mixture should be nested within a "larger" mixture when properly fitted. We extract the optimum fitted model from the output of `fit_incremental_angmix` via `bestmodel`:

```
R> fit.vmsin.best <- bestmodel(fit.vmsin)
```

Before estimating parameters, we first need to assess convergence and stationarity of the Markov chains. For this purpose, we first look at the (non-normalized) log posterior density (LPD) traceplots, which can be obtained using the **BAMBI** function `lpdtrace`: 1

```
R> lpdtrace(fit.vmsin.best)
```

The resulting plot displayed in Figure 6 shows that all three chains have stabilized into similar LPD ranges after burn-in, without noticeable trends or patterns. Next, we look at the parameter traces, plotted using **BAMBI** function `paramtrace`:

```
R> paramtrace(fit.vmsin.best)
```

These traceplots are displayed in the panels of Figures 18–21 in Appendix D, which show adequate signs of convergence and stationarity for samples within each chain. Stationarity of a chain can be formally tested using Geweke's convergence diagnostic (Geweke 1991), which tests equality (of means) of the first and last part of a Markov chain using standard z scores. The test is implemented in R package **coda**, and can be applied on `fit.vmsin.best`, first by converting it into a **coda** '`mcmc.list`' object via S3 function `as.mcmc.list`:

```
R> mcmc.vmsin.best <- coda::as.mcmc.list(fit.vmsin.best)
```

and then by applying the **coda** function `geweke.diag` on the output. We apply `geweke.diag` on `mcmc.vmsin.best` by setting both `frac1` and `frac2` equal to 0.5, to test the equality of the first and second halves. This produces a list of size 3 containing z statistics for each chain. These values are displayed in Figure 7 as barplots. The R commands are as follows:

```
R> geweke_res <- coda::geweke.diag(mcmc.vmsin.best, frac1 = 0.5, frac2 = 0.5)
R> par(mfrow = c(1, 3), mar = c(5, 6, 2, 1))
R> for(j in 1:3) {
+    barplot(geweke_res[[j]]$z, horiz = TRUE, names = names(geweke_res[[j]]$z),
+    xlim = range(geweke_res[[j]]$z, -3, 3), ylab = "", xaxt = 'n', las = 2)
+    axis(1, las = 1)
+    title(main = paste("Chain", j), xlab = "geweke.diag")
+  }
```

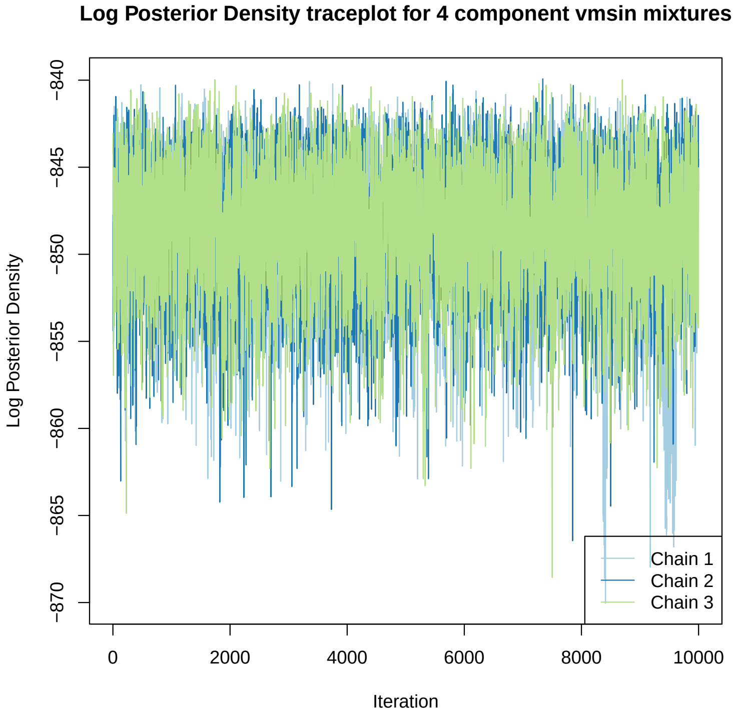


Figure 6: Log posterior traceplot for the Markov chain associated with the best fitted `vmsin` mixture model.

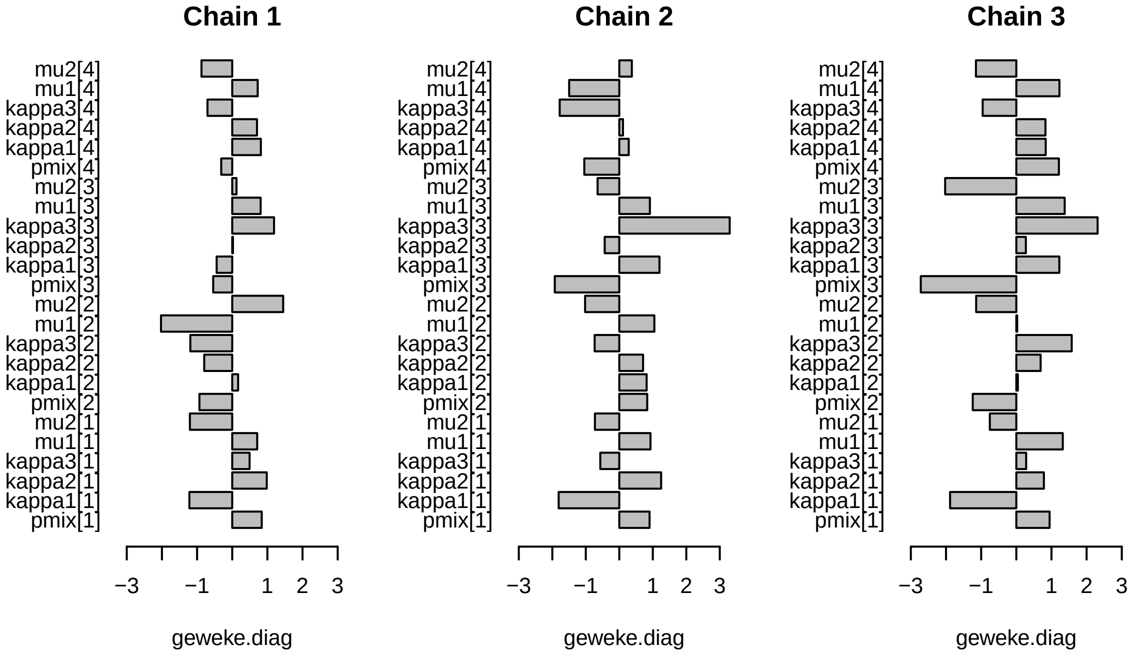


Figure 7: Geweke diagnostic z scores for the three Markov chains associated with the best (4 component) `vmsin` mixture model.

From these barplots, we see that all the z scores more or less lie between ±3, thus indicating adequate similarity between the first and second halves of the chains.

Package **coda** provides functions for a number of additional diagnostic plots and formal tests that may be used after conversion to a 'mcmc.list' object. Note that not all tests are applicable in every situation. For example, the Gelman-Rubin test (Gelman and Rubin 1992), available via **coda** function gelman.diag, assumes normality of the posterior density; this assumption is clearly violated when the posterior density is multimodal. In this example, as well as for mixture models with several components in general, multimodality in the posterior density can be commonly observed.

Posterior multimodality is evident in the parameter traceplots for the current fit: the sample values of the same parameter across the independent chains exhibit noticeable differences (see, e.g., the panels of Figure 19). Note that this can be due to both permutation (i.e., label-switching) and non-permutation modes. These modes are from similar posterior density regions, as the LPD traces show. This also indicates that various regions of the posterior density are being well explored by the three chains together.

Next, we consider parameter estimation and assessing goodness of fit. Since the combined MCMC samples are multimodal, the posterior mean point estimates from raw MCMC samples will not be meaningful, as they will lie in between modes. A simple alternative is to use the MCMC based approximate MAP estimate, which is unaffected by multimodality; proper care must be taken otherwise. The inferential difficulties associated with having permutation modes in the MCMC samples can be (potentially) solved by undoing label switching. The **BAMBI** function fix_label (with the default settings) can be used for this purpose:

```
R> fit.vmsin.best <- fix_label(fit.vmsin.best)
```

The parameter traceplots obtained (using paramtrace after undoing label switching) are displayed in Figures 22–25 in Appendix D. Compare these traceplots to the ones displayed in Figures 18–21. It can be seen that this procedure indeed removes some of the permutation modes, in that the parameter traces for the three independent chains now largely overlap. However, some non-unique modes are still present, as can be seen in the traces of κ_3 and μ_2 in component 4, displayed in panel (d) and (f) of Figure 25: e.g., κ_3 "jumps" between modes at approximately 2 and -2 over the course of the MCMC simulation. These modes might be genuine non-permutation modes, or permutation modes that fix_label is unable to resolve.

In **BAMBI**, parameter estimates are computed using the function pointest, which can find point estimates of the whole parameter vector, as well as its sub-vectors. Note that the function supports multiple methods of estimation. In particular, the argument fn in pointest specifies what function to evaluate on the MCMC samples for estimation. For example, fn = mean computes the MCMC posterior mean, while fn = "MODE" returns an MCMC based approximate MAP estimate. We use pointest to find the MAP and posterior mean estimates (after applying fix_label), and then note their differences. The R commands are as follows.

```
R> round(pointest(fit.vmsin.best, fn = "MODE"), 2)

           1    2     3     4
pmix     0.43 0.15  0.36  0.07
kappa1  33.11 8.10  4.20  4.98
kappa2  24.59 1.76  9.37  0.00
```

```
kappa3 -11.86 0.06 -1.67 -1.74
mu1      5.21 4.63  4.42  1.67
mu2      5.56 6.22  2.44  5.01
```

```
R> round(pointest(fit.vmsin.best, fn = mean), 2)
```

```
            1     2     3     4
pmix     0.43  0.17  0.34  0.07
kappa1  36.49  7.39  4.35  4.23
kappa2  29.19  2.08  8.10  0.07
kappa3 -12.78 -0.40 -1.07 -0.03
mu1      5.23  4.67  4.43  1.73
mu2      5.55  6.17  2.43  3.44
```

We note that both the approximate MAP estimate and the posterior mean estimate reasonably agree on the first three components. However, they disagree on the remaining fourth component regarding the value of `mu2` and `kappa3`. This is not surprising, since the MCMC samples for these two parameters have (possibly non-permutation) multiple modes, as we saw earlier. In this case, their posterior mean estimates lie in between modes, and hence are not good point estimates. To visualize the differences between these estimates, we plot the contours and surfaces of the corresponding fitted model densities using the S3 functions `contour` from base R **graphics** (R Core Team 2021) and `densityplot` from **lattice** (Sarkar 2021) for '`angmcmc`' objects:

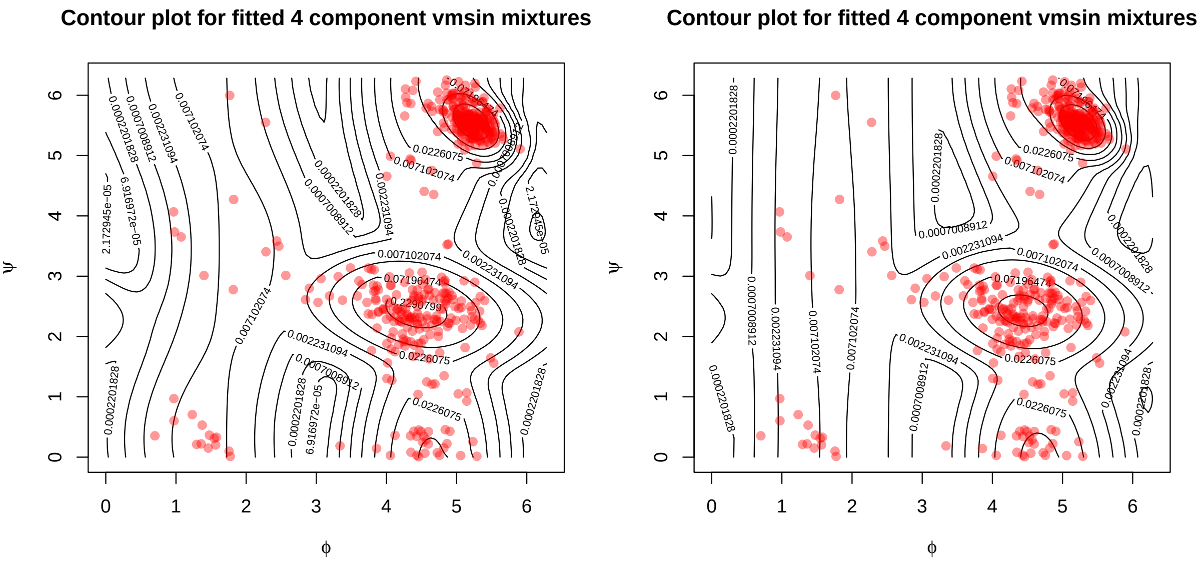
```
R> contour(fit.vmsin.best, fn = "MODE")
R> lattice::densityplot(fit.vmsin.best, fn = "MODE")
R> contour(fit.vmsin.best, fn = mean)
R> lattice::densityplot(fit.vmsin.best, fn = mean)
```

The contour plots are shown in Figures 8(a) and 8(b), and the density surfaces in Figures 9(a) and 9(b). As can be seen, these plots are visually quite similar, despite the differences in the point estimates. This is due to the fact the component mostly affected by the existence of multiple modes has a low mixing proportion. Nonetheless, in the current setting, the MAP estimate is the better of the two for the reasons described.

Finally, we compute interval estimates of the parameters. This done by the S3 function `summary` for '`angmcmc`' objects, which computes the MCMC posterior mean along with a 95% credible interval:

```
R> summary(fit.vmsin.best)
```

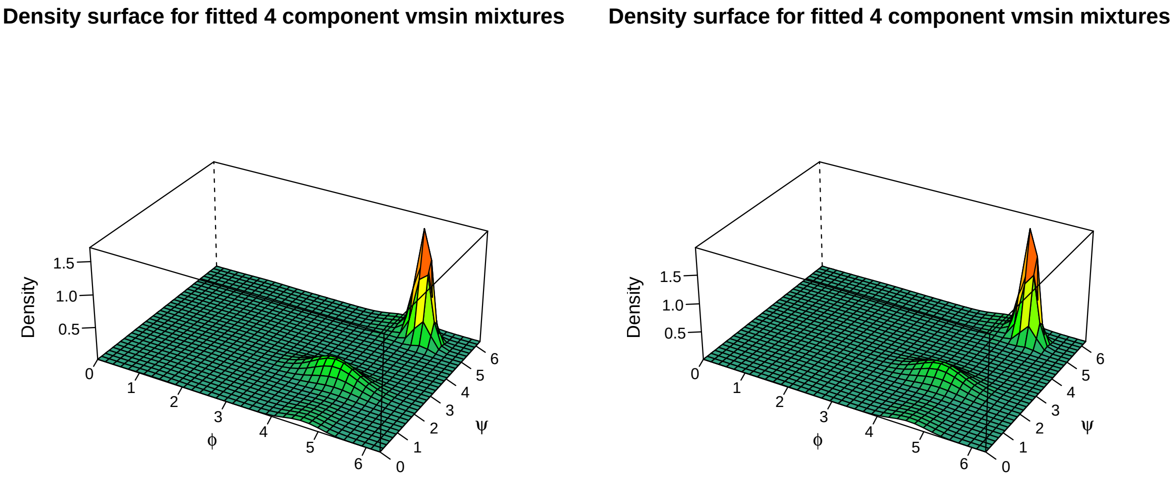
```
                             1                   2                    3
pmix         0.43 (0.38, 0.48)   0.17 (0.12, 0.22)    0.34 (0.29, 0.38)
kappa1   36.49 (28.98, 45.60)  7.39 (4.99, 10.84)    4.35 (3.44, 5.46)
kappa2   29.19 (21.90, 38.89)   2.08 (1.19, 3.24)   8.10 (5.98, 10.63)
kappa3 -12.78 (-18.50, -7.51) -0.40 (-1.81, 1.15) -1.07 (-2.26, 0.041)
mu1          5.23 (5.20, 5.26)   4.67 (4.54, 4.80)    4.43 (4.35, 4.52)
mu2          5.55 (5.51, 5.58)   6.17 (5.99, 6.28)    2.43 (2.36, 2.49)
```



(a) Approximate MAP estimate from MCMC samples.

(b) MCMC posterior mean after resolving label switching.

Figure 8: Contour plots for fitted 4 component `vmsin` mixture model with parameters estimated via MCMC-based MAP estimation (left) and posterior mean estimation (right).



(a) Approximate MAP estimate from MCMC samples.

(b) MCMC posterior mean after resolving label switching.

Figure 9: Density surfaces for fitted 4 component `vmsin` mixture model with parameters estimated via MCMC-based MAP estimation (left) and posterior mean estimation (right).

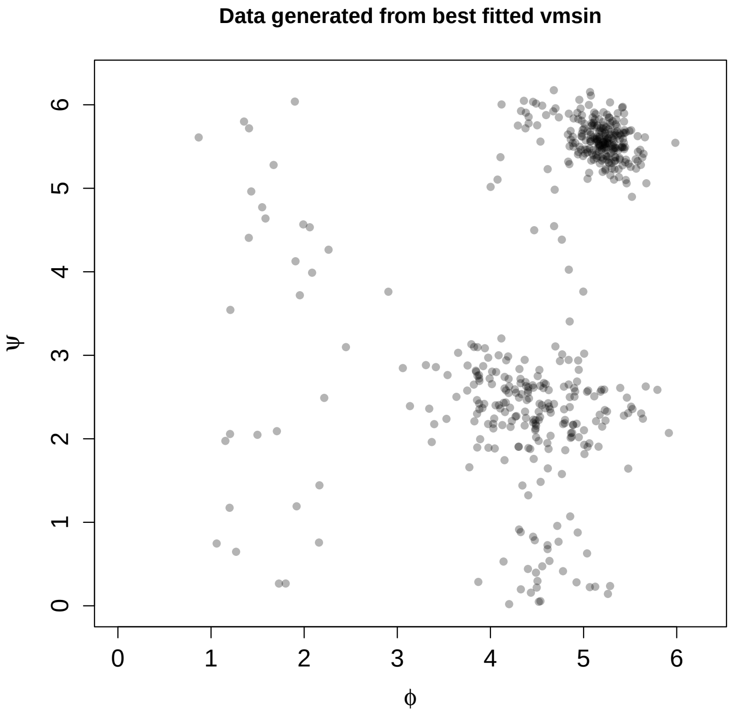


Figure 10: Ramachandran plot for the data generated from the best fitted `vmsin` mixture model.

```
                                       4
pmix    0.065 (0.044, 0.093)
kappa1     4.23 (1.31, 7.73)
kappa2 0.067 (0.00012, 0.51)
kappa3  -0.025 (-3.04, 3.04)
mu1        1.73 (1.44, 2.12)
mu2        3.44 (0.74, 5.96)
```

We also use this example to illustrate `r_fitted`, which generates random deviates from a fitted model, with parameters estimated using `pointest`. The corresponding function `d_fitted` evaluates the density. These can be useful for posterior predictive checks. We draw observations from the best (4 component) fitted `vmsin` model, construct the Ramachandran plot for the generated dataset (exhibited in Figure 10) and compare it with the original Ramachandran plot. The following are the R commands used for this purpose.

```
R> set.seed(12321)
R> vmsin.data <- r_fitted(nrow(tim8), fit.vmsin.best, fn = "MODE")
R> plot(vmsin.data, xlab = "phi", ylab = "psi", xlim = c(0, 2*pi),
+    ylim = c(0, 2*pi), pch = 16, col = scales::alpha("black", 0.6))
R> title("Data generated from best fitted vmsin")
```

Observe the similarity between Figures 5 and 10. The different clusters of the actual data points are reproduced well in the simulated data, which corroborate the goodness of fit.

Fitting the vmcos *mixture model*

Next, we fit `vmcos` mixtures to the data, by using `fit_incremental_angmix` with `model = "vmcos"`. We set `n_qrnd = 1e4` (the default used in `dvmcos`), which specifies that 10,000 pairs of quasi-random Sobol numbers would be used to approximate the `vmcos` normalizing constant in cases where its analytic computation is unstable. In a small dimensional problem with finite variance (such as ours), the Sobol sequence (or low discrepancy quasi-random sequences in general) often provides a better Monte Carlo approximation than (pseudo-) random sequences. In fact, for two dimensional problems, the rate of convergence for a Sobol sequence based Monte Carlo approximation is $O((\log N)^2/N)$ as opposed to $O(1/\sqrt{N})$ for an ordinary (pseudo-) random sequence based Monte Carlo approximation (Lemieux and Faure 2009), where N denotes the number of (quasi) random pairs used. See the documentation of `dvmcos` for examples comparing analytic, quasi Monte Carlo, and ordinary Monte Carlo approximations of the (normalizing constant of the) `vmcos` density. For `fit_incremental_angmix` (or more specifically in `fit_angmix`), Monte Carlo approximations based on 10^4 pairs of Sobol numbers typically provide reasonable approximations, while keeping the computational burden moderate.

The following are the R commands used for incrementally fitting `vmcos` mixture models.

```
R> set.seed(12321)
R> fit.vmcos <- fit_incremental_angmix(model = "vmcos", data = tim8,
+    crit = "LOOIC", start_ncomp = 2, max_ncomp = 10, n.iter = 2e4,
+    n.chains = 3, use_best_chain = FALSE, n_qrnd = 1e4)
```

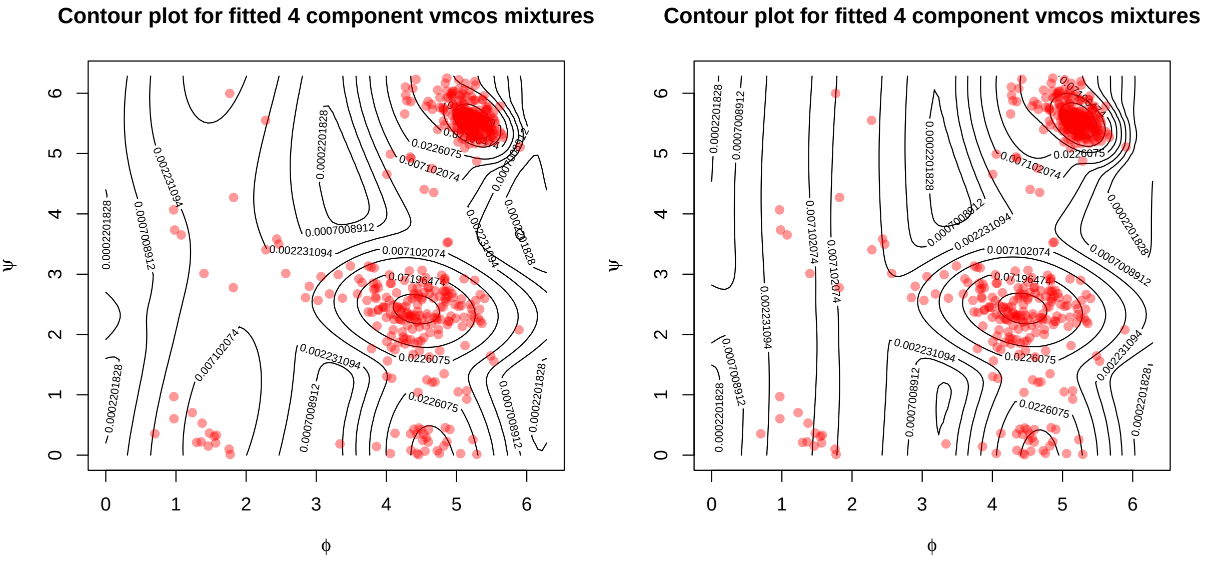
Similar to the `vmsin` case, here also the algorithm stops at 5 components and determines the optimal number of components to be 4. We first extract the "best" fitted model, via `bestmodel`:

```
R> fit.vmcos.best <- bestmodel(fit.vmcos)
```

and then plot the log posterior and parameter traces. These plots show similar convergence properties, and are omitted for brevity. For parameter estimation, we compute both (approximate) MAP and posterior mean estimates (after undoing label switching), and plot the contour and surface of the associated fitted model densities. The following are the associated R commands:

```
R> fit.vmcos.best <- fix_label(fit.vmcos.best)
R> contour(fit.vmcos.best, fn = "MODE")
R> lattice::densityplot(fit.vmcos.best, fn = "MODE")
R> contour(fit.vmcos.best, fn = mean)
R> lattice::densityplot(fit.vmcos.best, fn = mean)
```

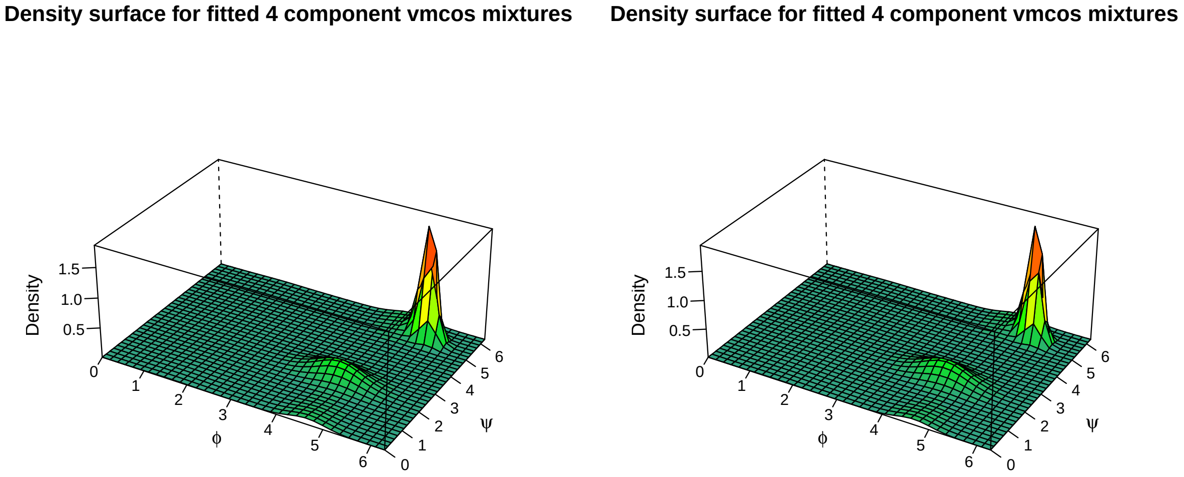
The fitted contours are displayed in Figures 11(a) and 11(b), and the density surfaces are displayed in Figures 12(a) and 12(b), are noticeably similar. They are also broadly similar to the ones obtained for the fitted `vmsin` models. Estimated posterior means along with estimated 95% credible intervals are obtained using the S3 function `summary.angmcmc` as follows.



(a) Approximate MAP estimate from MCMC samples.

(b) MCMC posterior mean after resolving label switching.

Figure 11: Contour plots for fitted 4 component `vmcos` mixture model with parameters estimated via MCMC-based MAP estimation (left) and posterior mean estimation (right)



(a) Approximate MAP estimate from MCMC samples.

(b) MCMC posterior mean after resolving label switching.

Figure 12: Density surfaces for fitted 4 component `vmcos` mixture model with parameters estimated via MCMC-based MAP estimation (left) and posterior mean estimation (right).

```
R> summary(fit.vmcos.best)

                             1                    2                    3
pmix        0.43 (0.38, 0.49)    0.16 (0.12, 0.21)    0.34 (0.29, 0.38)
kappa1   48.68 (37.97, 61.62)   7.89 (5.16, 11.92)    5.15 (3.82, 6.69)
kappa2   41.27 (30.56, 55.34) 2.00 (0.00049, 4.11)   8.77 (6.36, 11.70)
kappa3 -12.57 (-18.65, -7.29)   0.11 (-1.57, 2.24) -0.97 (-2.14, 0.094)
mu1         5.23 (5.20, 5.26)    4.67 (4.53, 4.80)    4.43 (4.34, 4.51)
mu2         5.55 (5.51, 5.58)    6.17 (5.99, 6.28)    2.43 (2.37, 2.50)
                           4
pmix   0.064 (0.043, 0.091)
kappa1    3.63 (1.19, 6.69)
kappa2 0.18 (0.00013, 1.30)
kappa3  -0.20 (-1.55, 0.91)
mu1       1.60 (1.35, 1.92)
mu2       3.06 (0.13, 6.09)
```

Fitting the `wnorm2` *mixture model*

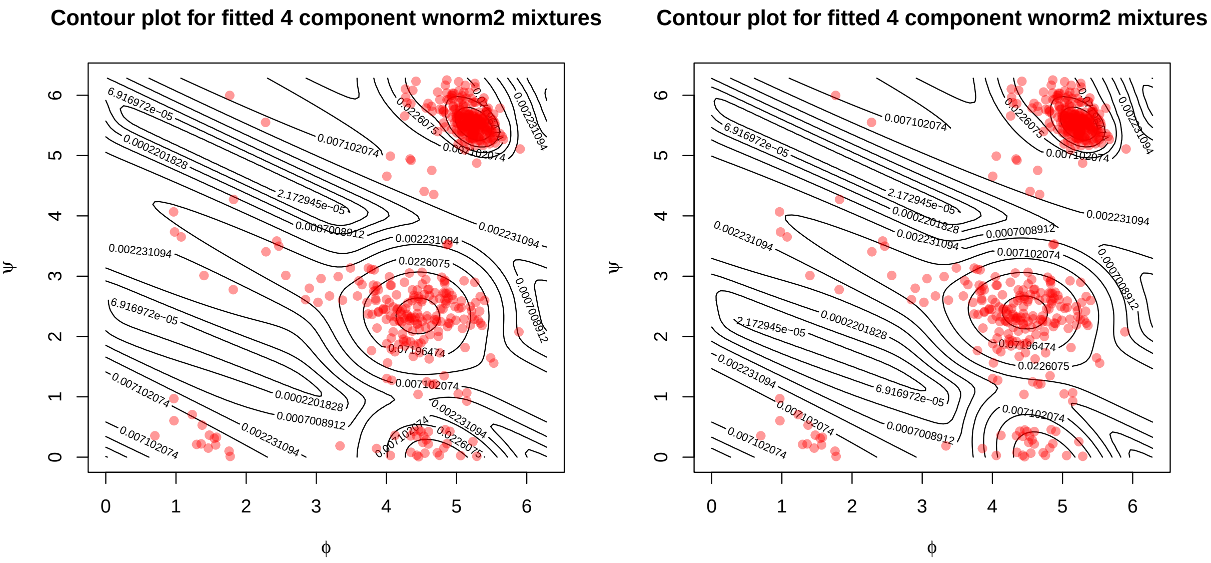
Finally, we fit `wnorm2` mixtures to the data. The R commands used are as follows:

```
R> set.seed(12321)
R> library(future)
R> plan(multiprocess(workers = 3))
R> fit.wnorm2 <- fit_incremental_angmix(model = "wnorm2", data = tim8,
+    crit = "LOOIC", start_ncomp = 2, max_ncomp = 10, n.iter = 2e4,
+    n.chains = 3, use_best_chain = FALSE)
```

Here also, the function stops at 5 components and determines the optimal number of components to be 4[6]. As done in the previous two cases, after extracting the best model, we assess convergence via trace plots (omitted for brevity). We find MCMC-based MAP and posterior mean estimates (after undoing label switching), and also find credible interval estimates. Finally we assess goodness of fit through fitted contour and density surfaces. The following are the R commands that perform these tasks.

```
R> fit.wnorm2.best <- bestmodel(fit.wnorm2)
R> lpdtrace(fit.wnorm2.best)
R> paramtrace(fit.wnorm2.best)
R> fit.wnorm2.best <- fix_label(fit.wnorm2.best)
R> contour(fit.wnorm2.best, fn = "MODE")
R> lattice::densityplot(fit.wnorm2.best, fn = "MODE")
R> contour(fit.wnorm2.best, fn = mean)
R> lattice::densityplot(fit.wnorm2.best, fn = mean)
R> summary(fit.wnorm2.best)
```

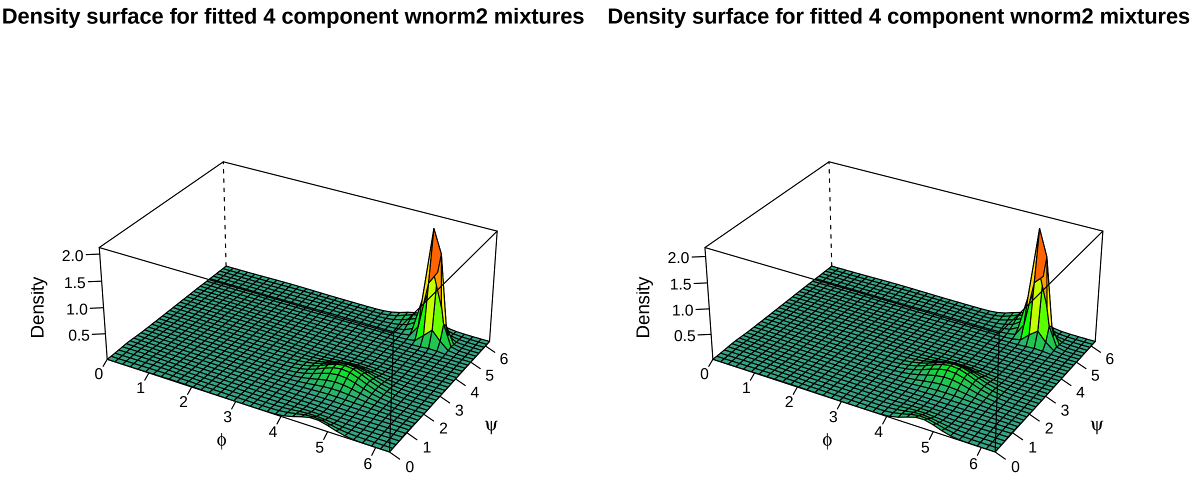
[6] It should be noted that the runtime for wrapped normal fitting is considerably longer than the von Mises sine models, due to the computational burden; see Section 1.1.



(a) Approximate MAP estimate from MCMC samples.

(b) MCMC posterior mean after resolving label switching.

Figure 13: Contour plots for fitted 4 component `wnorm2` mixture model with parameters estimated via MCMC-based MAP estimation (left) and posterior mean estimation (right).



(a) Approximate MAP estimate from MCMC samples.

(b) MCMC posterior mean after resolving label switching.

Figure 14: Density surfaces for fitted 4 component `wnorm2` mixture model with parameters estimated via MCMC-based MAP estimation (left) and posterior mean estimation (right).

The contours and density surfaces displayed in Figures 13 and 14) are noticeably similar. They are also broadly similar to the fitted `vmsin` and `vmcos` mixture model density contours and surfaces. The estimated credible intervals along with MCMC posterior means for fitted 4 component `wnorm2` mixture are obtained using the S3 function `summary.angmcmc` as follows.

```
R> summary(fit.wnorm2.best)
```

```
                           1                  2                  3
pmix      0.39 (0.31, 0.46)  0.17 (0.12, 0.24)  0.32 (0.27, 0.37)
kappa1 39.76 (30.19, 52.15) 9.41 (5.93, 13.60)  4.74 (3.57, 6.14)
kappa2 36.80 (24.43, 54.02) 6.51 (1.46, 11.82)  6.55 (4.35, 9.62)
kappa3  11.78 (4.77, 19.61) 3.39 (-0.26, 7.21) 0.28 (-0.80, 1.29)
mu1       5.24 (5.21, 5.27)  4.81 (4.63, 4.97)  4.47 (4.39, 4.56)
mu2       5.52 (5.48, 5.57)  6.09 (5.90, 6.27)  2.41 (2.33, 2.49)
                         4
pmix   0.12 (0.085, 0.16)
kappa1  2.69 (1.08, 4.42)
kappa2 8.17 (3.60, 13.08)
kappa3  4.61 (1.91, 7.51)
mu1     5.39 (4.28, 6.22)
mu2     1.53 (1.05, 2.17)
```

Comparative analysis of the three fitted models

So far, we have considered mixtures of `vmsin`, `vmcos` and `wnorm2` densities, and have fitted them to `tim8` data. The associated optimal number of components were determined via leave one out information criterion (LOOIC) in incremental fitting schemes. We then plotted the fitted density contours and surfaces to assess the goodness of fit, and noticed that these plots are broadly similar across the three optimal fitted mixture models (of `vmsin`, `vmcos` and `wnorm2` densities). It is natural to then consider the question of which among these three fitted bivariate mixture models best explains the data. This can again be answered via LOOIC. For '`angmcmc`' objects LOOIC can be conveniently computed using the S3 function `looic` from package **loo**. However, here we do not need to recompute them since they were already computed during incremental model fitting, and can be extracted via the convenience function `bestcriterion` as follows:

```
R> looic.all.bi <- list(
+   vmsin.4 = bestcriterion(fit.vmsin),
+   vmcos.4 = bestcriterion(fit.vmcos),
+   wnorm2.4 = bestcriterion(fit.wnorm2)
+ )
```

Now we compare the three models via `loo::compare` on the basis of their LOOIC's:

```
R> comp <- loo::loo_compare(x = looic.all.bi)
R> comp
```

```
         elpd_diff se_diff
wnorm2.4   0.0       0.0  
vmsin.4   -7.4       7.6  
vmcos.4  -12.3       7.3  
```

The documentation of `loo_compare` from package **loo** v2.4.1 says "When comparing two fitted models, we can estimate the difference in their expected predictive accuracy by the difference in `elpd_loo` or `elpd_waic` (or multiplied by -2, if desired, to be on the deviance scale). When using `loo_compare()`, the returned matrix will have one row per model and several columns of estimates. The values in the `elpd_diff` and `se_diff` columns of the returned matrix are computed by making pairwise comparisons between each model and the model with the largest ELPD (the model in the first row)".

Thus the above output provides a ranking among the three models based on their (estimated) expected log predictive density (ELPD) values; a higher ELPD indicates a better predictive accuracy and thus a better fit. The fitted `wnorm2` model appears to have the highest ELPD (see the column `elpd_diff` in the above output), followed by the fitted `vmsin` model and the fitted `vmcos` model. However, these ELPD's are estimates, and the variabilities of these estimates need to be considered when making comparisons. To address this, we make use of the standard errors of the differences provided in the column `se_diff` and construct approximate 95% uncertainty interval estimates of the pairwise ELPD differences (viz., `elpd_diff` ± 2 `se_diff`) for the fitted model pairs (`vmsin`, `wnorm2`) and (`vmcos`, `wnorm2`). An ELPD difference is considered to be significant (at the 95% level) if the corresponding interval estimate does not contain zero. The R commands are as follows.

```
R> find_ci <- function(x, digits = 1) {
+    round(c(lower = unname(x[1] - 2*x[2]), upper = unname(x[1] + 2*x[2])),
+    digits = digits)
+ }
R> t(apply(comp[-1, c("elpd_diff", "se_diff")], 1, find_ci))
```

```
        lower upper
vmsin.4 -22.5   7.8
vmcos.4 -26.9   2.3
```

This shows that approximate 95% interval estimates of the ELPD differences between the fitted best `vmsin` and the best `wnorm2` model, and the best `vmcos` and the best `wnorm2` model, are $(-22.5, 7.8)$ and $(-26.9, 2.3)$ respectively, both containing zero. It therefore follows that all three of the fitted (four component) mixture models are not significantly different in terms of their goodness of fit to these data.

4.2. Fitting mixture models on the wind (univariate) data

The `wind` data contains 239 observations on wind direction in radians measured at Saturna Island, British Columbia, Canada, during October 1–10, 2016. As a result of a severe storm that occurred during that period, the data shows significant variability with an interesting bi- (or possibly tri-) modality. Figure 15 shows a histogram of the data, constructed by applying the default `hist` function on `wind[, "angle"]`.

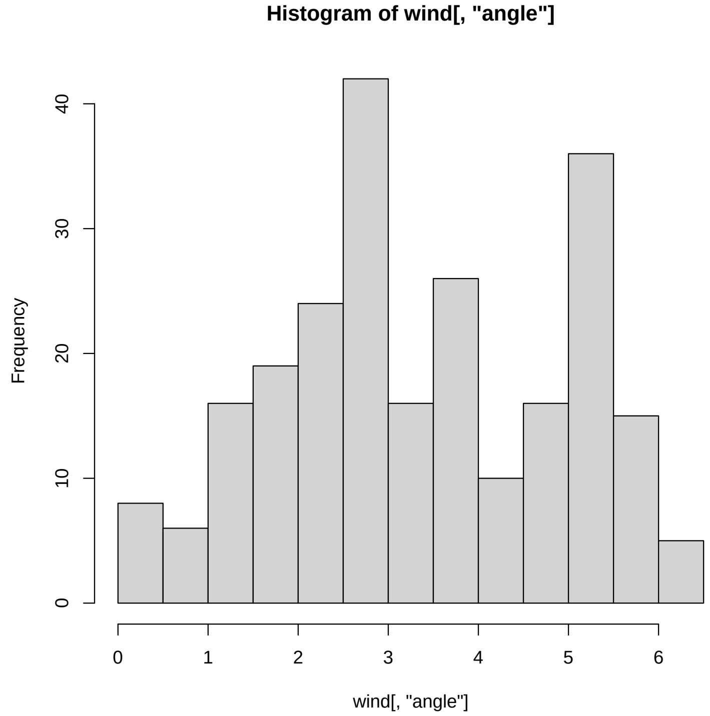


Figure 15: Histogram for the wind dataset.

Similar to the bivariate case, we use `fit_incremental_angmix` to fit mixtures of `vm` and `wnorm` separately with incremental number of components (starting at 1) and determine an optimum size in each case. To fit each mixture model, we first generate 20,000 MCMC samples for the parameters, with the (default) first half taken as burn-in. Except for `n.iter`, defaults for all other arguments are used in these examples. After generating the MCMC samples, we assess their convergence via LPD and parameter trace plots. Following, we visualize the fits via density curves constructed using the S3 function `densityplot` (which requires **lattice**). Finally, we compute point and interval estimates for each parameter using the S3 function `summary.angmcmc`.

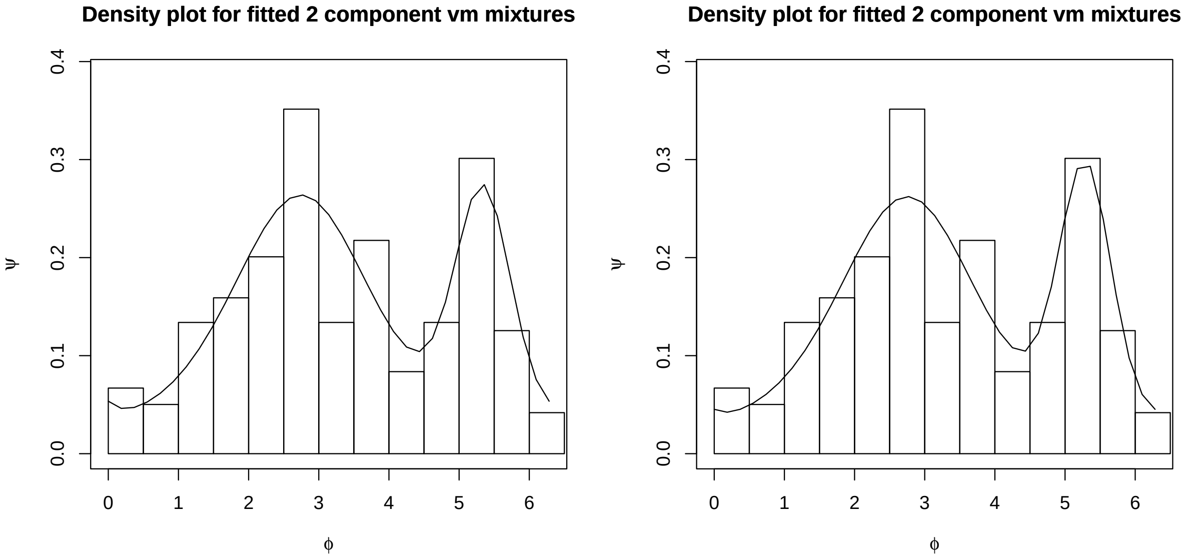
Fitting the `vm` *mixture model*

We start with `vm`. The R commands are as follows:

```
R> set.seed(12321)
R> fit.vm <- fit_incremental_angmix(model = "vm", data = wind[, "angle"],
+    crit = "LOOIC", start_ncomp = 1, max_ncomp = 10, n.iter = 2e4,
+    n.chains = 3)
```

The function stops at 3 components and determines the optimal number of components to be 2. After it stops, we extract the '`angmcmc`' object corresponding to the best model from its output, and inspect its LPD and parameter traces for convergence (omitted for brevity).

```
R> fit.vm.best <- bestmodel(fit.vm)
```



(a) Approximate MAP estimate from MCMC samples.

(b) MCMC posterior mean after resolving label switching.

Figure 16: Density curves for fitted 2 component `vm` mixture model with parameters estimated via MCMC-based MAP estimation (left) and posterior mean estimation (right).

```
R> lpdtrace(fit.vm.best)
R> paramtrace(fit.vm.best)
```

We first use `fix_label` to undo label switching, and then assess goodness of fit through density curves fitted using MAP and posterior mean estimation:

```
R> fit.vm.best <- fix_label(fit.vm.best)
R> lattice::densityplot(fit.vm.best, fn = "MODE")
R> lattice::densityplot(fit.vm.best, fn = mean)
```

The plots are displayed in Figures 16(a) and 16(b), which show noticeable similarity.

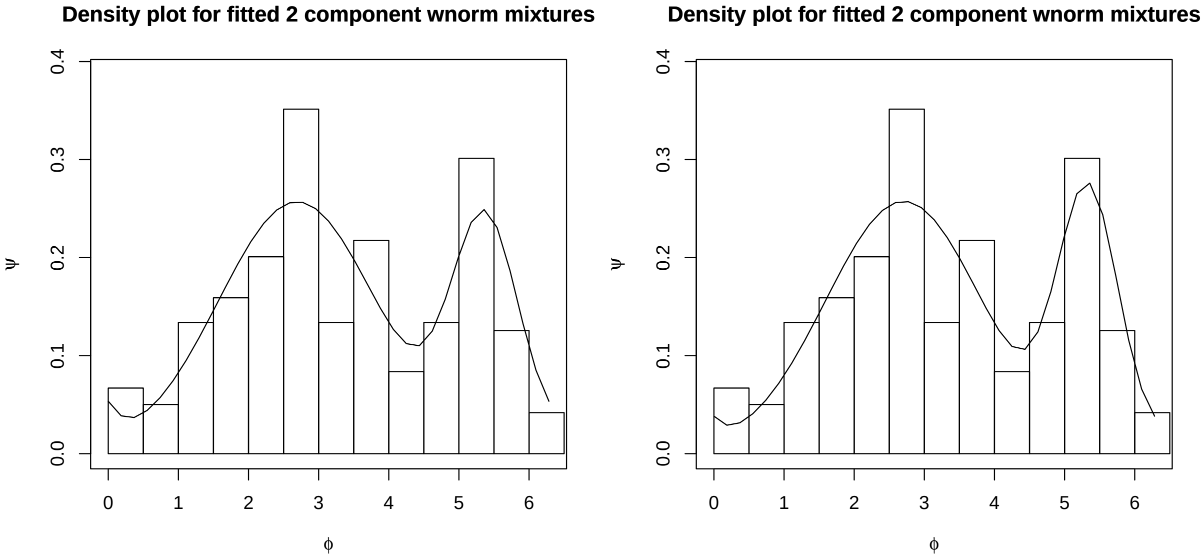
Finally, we compute MCMC posterior mean and associated 95% credible interval using S3 function `summary`:

```
R> summary(fit.vm.best)
```

```
                         1                  2
pmix   0.24 (0.13, 0.46) 0.76 (0.54, 0.87)
kappa 7.36 (1.12, 21.97) 1.03 (0.58, 1.79)
mu     5.29 (5.01, 5.49) 2.75 (2.48, 3.04)
```

Fitting the `wnorm` mixture model

Next, we do similar exercises with `wnorm` model. The following are the R codes used.



(a) Approximate MAP estimate from MCMC samples.

(b) Estimated posterior mean after resolving label switching.

Figure 17: Density curves for fitted 2 component `wnorm` mixture model with parameters estimated via MCMC-based MAP estimation (left) and posterior mean estimation (right).

```
R> set.seed(12321)
R> fit.wnorm <- fit_incremental_angmix(model = "wnorm", data = wind[, "angle"],
+    crit = "LOOIC", start_ncomp = 1, max_ncomp = 10, n.iter = 2e4,
+    n.chains = 3)
R> fit.wnorm.best <- bestmodel(fit.wnorm)
R> lpdtrace(fit.wnorm.best)
R> paramtrace(fit.wnorm.best)
R> fit.wnorm.best <- fix_label(fit.wnorm.best)
R> lattice::densityplot(fit.wnorm.best, fn = "MODE")
R> lattice::densityplot(fit.wnorm.best, fn = mean)
```

Similar to the `vm` case, here also the function stops at 3 components and determines the optimal number of components to be 2. The LPD and parameter traceplots are omitted for brevity. The density curves fitted using MAP and posterior mean estimates are shown in Figures 17(a) and 17(b) respectively, which are noticeably similar. They are also broadly similar to the plots associated with the fitted `vm` mixture densities shown in Figure 16.

Finally we compute the MCMC posterior mean and 95% credible interval using the S3 function `summary`.

```
R> summary(fit.wnorm.best)

                       1                  2
pmix   0.28 (0.15, 0.46) 0.72 (0.54, 0.85)
kappa 5.38 (0.97, 17.97) 0.81 (0.41, 1.48)
mu     5.34 (5.00, 5.54) 2.71 (2.43, 3.03)
```

Comparison between the two models

Similar to the bivariate case, we compare the fitted vm and wnorm mixture models using their LOOIC values. We first extract the LOOICs using the convenience function bestcriterion:

```
R> looic.all.uni <- list(
+   vm.2 = bestcriterion(fit.vm),
+   wnorm.2 = bestcriterion(fit.wnorm)
+ )
```

Then we compare the two models based on their estimated expected log predictive densities, by using loo::loo_compare():

```
R> loo::loo_compare(x = looic.all.uni)

        elpd_diff se_diff
vm.2     0.0       0.0
wnorm.2 -0.7       1.0
```

Clearly an approximate 95% credible interval estimate for the ELPD difference, obtained by elpd_diff ± 2 se, contains zero. This implies that the fitted vm model and the fitted wnorm model do not have a significant difference in terms of their goodness of fit to these data.

5. Concluding remarks and future work

Angular data, both univariate and bivariate, arise naturally in a variety of modern scientific problems, and their analyses require appropriate use of rigorous statistical tools and distributions specifically developed for such data. The lack of comprehensive software implementing such methods (in R or otherwise) has hindered their applicability in practice – especially for bivariate angular models and mixtures thereof.

The package **BAMBI** is our contribution to this area, providing a platform that implements a set of formal statistical tools and methods for analyzing such data, and is readily accessible to practitioners. There are various directions in which the software could be extended in future releases. Some possible features under consideration include the following.

- Implementation of additional angular distributions, such as wrapped Cauchy.
- Additional methods of density evaluation and random simulation from fitted models on the basis of MCMC samples.
- Visualizations of bivariate angles with toroidal plots.
- Use of parallel tempering or related methods during MCMC simulations for faster exploration of the posterior density.
- Proper handling of overfitting heterogeneity that takes place in high dimensional mixture models when some of the component specific parameters in two different components are identical. Frühwirth-Schnatter (2011) suggests the use of sparse priors for the component specific location parameters to deal with this problem.

Acknowledgments

The authors thank the two anonymous referees who reviewed this work, and provided valuable, thorough and constructive suggestions on both the package and the manuscript. The authors are also thankful to Anahita Nodehi from Tarbiat Modares University, Iran, for notifying them of a few typos and providing helpful suggestions on the R package documentation.

References

Abramowitz M, Stegun IA (1964). *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*, volume 55. Courier Corporation.

Agostinelli C, Lund U (2017). ***circular**: Circular Statistics*. R package version version 0.4-93, URL https://CRAN.R-project.org/package=circular.

Akaike H (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control*, **19**(6), 716–723. doi:10.1109/tac.1974.1100705.

Benaglia T, Chauveau D, Hunter DR, Young D (2009). "**mixtools**: An R Package for Analyzing Finite Mixture Models." *Journal of Statistical Software*, **32**(6), 1–29. doi:10.18637/jss.v032.i06.

Bengtsson H (2021a). ***future**: Unified Parallel and Distributed Processing in R for Everyone*. R package version 1.22.1, URL https://CRAN.R-project.org/package=future.

Bengtsson H (2021b). ***future.apply**: Apply Function to Elements in Parallel Using Futures*. R package version 1.8.1, URL https://CRAN.R-project.org/package=future.apply.

Best DJ, Fisher NI (1979). "Efficient Simulation of the Von Mises Distribution." *Journal of the Royal Statistical Society C*, **28**(2), 152–157. doi:10.2307/2346732.

Bhattacharya D, Cheng J (2015). "De Novo Protein Conformational Sampling Using a Probabilistic Graphical Model." *Scientific Reports*, **5**, 16332. doi:10.1038/srep16332.

Boomsma W, Mardia KV, Taylor CC, Ferkinghoff-Borg J, Krogh A, Hamelryck T (2008). "A Generative, Probabilistic Model of Local Protein Structure." *Proceedings of the National Academy of Sciences of the United States of America*, **105**(26), 8932–8937. doi:10.1073/pnas.0801715105.

Carpenter B, Gelman A, Hoffman M, Lee D, Goodrich B, Betancourt M, Brubaker M, Guo J, Li P, Riddell A (2017). "Stan: A Probabilistic Programming Language." *Journal of Statistical Software*, **76**(1), 1–32. doi:10.18637/jss.v076.i01.

Celeux G, Forbes F, Robert CP, Titterington DM (2006). "Deviance Information Criteria for Missing Data Models." *Bayesian Analysis*, **1**(4), 651–673. doi:10.1214/06-ba122.

Chakraborty S, Wong SWK (2018). "On the Circular Correlation Coefficients for Bivariate Von Mises Distributions on a Torus." arXiv:1804.08553, URL https://arxiv.org/abs/1804.08553.

Chakraborty S, Wong SWK (2021). ***BAMBI**: Bivariate Angular Mixture Models.* R package version 2.3.3, URL `https://CRAN.R-project.org/package=BAMBI`.

Diebolt J, Robert CP (1994). "Estimation of Finite Mixture Distributions through Bayesian Sampling." *Journal of the Royal Statistical Society B*, **56**(2), 363–375. `doi:10.1111/j.2517-6161.1994.tb01985.x`.

Duane S, Kennedy AD, Pendleton BJ, Roweth D (1987). "Hybrid Monte Carlo." *Physics Letters B*, **195**(2), 216–222. `doi:10.1016/0370-2693(87)91197-x`.

Fisher NI (1995). *Statistical Analysis of Circular Data.* Cambridge University Press.

Fisher NI, Lee AJ (1983). "A Correlation Coefficient for Circular Data." *Biometrika*, **70**(2), 327–332. `doi:10.1093/biomet/70.2.327`.

Fraley C, Raftery AE (2002). "Model-Based Clustering, Discriminant Analysis and Density Estimation." *Journal of the American Statistical Association*, **97**, 611–631. `doi:10.1198/016214502760047131`.

Fraley C, Raftery AE, Murphy TB, Scrucca L (2012). "**mclust** Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation." *Technical Report 597*, Department of Statistics, University of Washington.

Frühwirth-Schnatter S (2001). "Markov Chain Monte Carlo Estimation of Classical and Dynamic Switching and Mixture Models." *Journal of the American Statistical Association*, **96**(453), 194–209. `doi:10.1198/016214501750333063`.

Frühwirth-Schnatter S (2006). *Finite Mixture and Markov Switching Models.* Springer-Verlag. `doi:10.1007/978-0-387-35768-3`.

Frühwirth-Schnatter S (2011). *Dealing with Label Switching under Model Uncertainty*, chapter 10, pp. 213–239. John Wiley & Sons. `doi:10.1002/9781119995678.ch10`.

Gelfand AE, Smith AFM (1990). "Sampling-Based Approaches to Calculating Marginal Densities." *Journal of the American Statistical Association*, **85**(410), 398–409. `doi:10.1080/01621459.1990.10476213`.

Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB (2013). *Bayesian Data Analysis.* 3rd edition. Chapman & Hall/CRC. `doi:10.1201/b16018`.

Gelman A, Hwang J, Vehtari A (2014). "Understanding Predictive Information Criteria for Bayesian Models." *Statistics and Computing*, **24**(6), 997–1016. `doi:10.1007/s11222-013-9416-2`.

Gelman A, Rubin DB (1992). "Inference from Iterative Simulation Using Multiple Sequences." *Statistical Science*, **7**(4), 457–472. `doi:10.1214/ss/1177011136`.

Geman S, Geman D (1984). "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-6**(6), 721–741. `doi:10.1109/tpami.1984.4767596`.

Genz A, Bretz F, Miwa T, Mi X, Hothorn T (2021). ***mvtnorm**: Multivariate Normal and t Distributions*. R package version 1.1-2, URL https://CRAN.R-project.org/package=mvtnorm.

Geweke JF (1991). "Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments." *Staff Report 148*, Federal Reserve Bank of Minneapolis. URL https://ideas.repec.org/p/fip/fedmsr/148.html.

Grim J (2017). "Approximation of Unknown Multivariate Probability Distributions by Using Mixtures of Product Components: A Tutorial." *International Journal of Pattern Recognition and Artificial Intelligence*, **31**(09), 1750028. doi:10.1142/s0218001417500288.

Gronau QF, Sarafoglou A, Matzke D, Ly A, Boehm U, Marsman M, Leslie DS, Forster JJ, Wagenmakers EJ, Steingroever H (2017). "A Tutorial on Bridge Sampling." *Journal of Mathematical Psychology*, **81**, 80–97. doi:10.1016/j.jmp.2017.09.005.

Gronau QF, Singmann H, Wagenmakers EJ (2020). "**bridgesampling**: An R Package for Estimating Normalizing Constants." *Journal of Statistical Software*, **92**(10), 1–29. doi:10.18637/jss.v092.i10.

Hastings WK (1970). "Monte Carlo Sampling Methods Using Markov Chains and Their Applications." *Biometrika*, **57**(1), 97–109. doi:10.1093/biomet/57.1.97.

Hobert JP, Roy V, Robert CP (2011). "Improving the Convergence Properties of the Data Augmentation Algorithm with an Application to Bayesian Mixture Modeling." *Statistical Science*, **26**(3), 332–351. doi:10.1214/11-sts365.

Jammalamadaka SR, Sarma YR (1988). "A Correlation Coefficient for Angular Variables." *Statistical Theory and Data Analysis II*, pp. 349–364.

Jammalamadaka SR, Sengupta A (2001). *Topics in Circular Statistics*, volume 5. World Scientific.

Jasra A, Holmes CC, Stephens DA (2005). "Markov Chain Monte Carlo Methods and the Label Switching Problem in Bayesian Mixture Modeling." *Statistical Science*, **20**(1), 50–67. doi:10.1214/088342305000000016.

Jona-Lasinio G, Gelfand A, Jona-Lasinio M (2012). "Spatial Analysis of Wave Direction Data Using Wrapped Gaussian Processes." *The Annals of Applied Statistics*, **6**(4), 1478–1498. doi:10.1214/12-aoas576.

Kabsch W, Sander C (1983). "Dictionary of Protein Secondary Structure: Pattern Recognition of Hydrogen-Bonded and Geometrical Features." *Biopolymers*, **22**(12), 2577–2637. doi:10.1002/bip.360221211.

Khare K, Hobert JP (2011). "A Spectral Analytic Comparison of Trace-Class Data Augmentation Algorithms and Their Sandwich Variants." *The Annals of Statistics*, **39**(5), 2585–2606. doi:10.1214/11-aos916.

Langrognet F, Lebret R, Poli C, Iovleff S, Auder B, Iovleff S (2019). ***Rmixmod**: Classification with Mixture Modelling*. R package version 2.1.2-1, URL https://CRAN.R-project.org/package=Rmixmod.

Lemieux C, Faure H (2009). "New Perspectives on $(0, s)$-Sequences." In *Monte Carlo and Quasi-Monte Carlo Methods*, pp. 113–130. Springer-Verlag. doi:10.1007/978-3-642-04107-5_7.

Lennox KP, Dahl DB, Vannucci M, Tsai JW (2009). "Density Estimation for Protein Conformation Angles Using a Bivariate Von Mises Distribution and Bayesian Nonparametrics." *Journal of the American Statistical Association*, **104**(486), 586–596. doi:10.1198/jasa.2009.0024.

Lindsay BG (1995). *Mixture Models: Theory, Geometry and Applications*, volume 5 of *NSF-CBMS Regional Conference Series in Probability and Statistics*. Institute of Mathematical Statistics. doi:10.1214/cbms/1462106013.

Lunn D, Thomas A, Best NG, Spiegelhalter DJ (2000). "**WinBUGS** – A Bayesian Modelling Framework: Concepts, Structure, and Extensibility." *Statistics and Computing*, **10**(4), 325–337. doi:10.1023/a:1008929526011.

Mackenze PB (1989). "An Improved Hybrid Monte Carlo Method." *Physics Letters B*, **226**(3-4), 369–371. doi:10.1016/0370-2693(89)91212-4.

Mardia KV (1972). *Statistics of Directional Data*. Probability and Mathematical Statistics. Academic Press.

Mardia KV (1975). "Statistics of Directional Data." *Journal of the Royal Statistical Society B*, **37**(3), 349–371. doi:10.1111/j.2517-6161.1975.tb01550.x.

Mardia KV, Jupp PE (2009). *Directional Statistics*. Wiley Series in Probability and Statistics. John Wiley & Sons. doi:10.1002/9780470316979.

Mardia KV, Taylor CC, Subramaniam GK (2007). "Protein Bioinformatics and Mixtures of Bivariate Von Mises Distributions for Angular Data." *Biometrics*, **63**(2), 505–512. doi:10.1111/j.1541-0420.2006.00682.x.

Meng XL, Schilling S (2002). "Warp Bridge Sampling." *Journal of Computational and Graphical Statistics*, **11**(3), 552–586. doi:10.1198/106186002457.

Meng XL, Van Dyk DA (1999). "Seeking Efficient Data Augmentation Schemes via Conditional and Marginal Augmentation." *Biometrika*, **86**(2), 301–320. doi:10.1093/biomet/86.2.301.

Meng XL, Wong WH (1996). "Simulating Ratios of Normalizing Constants via a Simple Identity: A Theoretical Exploration." *Statistica Sinica*, pp. 831–860. doi:10.1214/ss/1028905934.

Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953). "Equation of State Calculations by Fast Computing Machines." *The Journal of Chemical Physics*, **21**(6), 1087–1092. doi:10.1063/1.1699114.

Neal RM (1996). "Priors for Infinite Networks." In *Bayesian Learning for Neural Networks*, pp. 29–53. Springer-Verlag. doi:10.1007/978-1-4612-0745-0_2.

Neal RM (2011). "MCMC Using Hamiltonian Dynamics." In S Brooks, A Gelman, G Jones, XL Meng (eds.), *Handbook of Markov Chain Monte Carlo*, Handbooks of Modern Statistical Methods, chapter 5, pp. 113–162. Chapman & Hall/CRC. URL http://mcmchandbook.net/HandbookChapter5.pdf.

Paluszewski M, Frellsen J, Hamelryck T (2010). ***mocapy++***. C++ library version 1.05, URL https://sourceforge.net/projects/mocapy/.

Paluszewski M, Hamelryck T (2010). "**Mocapy++** – A Toolkit for Inference and Learning in Dynamic Bayesian Networks." *BMC Bioinformatics*, **11**(1), 126. doi:10.1186/1471-2105-11-126.

Papastamoulis P (2016). "**label.switching**: An R Package for Dealing with the Label Switching Problem in MCMC Outputs." *Journal of Statistical Software, Code Snippets*, **69**(1), 1–24. doi:10.18637/jss.v069.c01.

Plummer M (2003). "**JAGS**: A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling." In K Hornik, F Leisch, A Zeileis (eds.), *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*. Technische Universität Wien, Vienna, Austria. URL https://www.R-project.org/conferences/DSC-2003/Proceedings/Plummer.pdf.

Plummer M (2008). "Penalized Loss Functions for Bayesian Model Comparison." *Biostatistics*, **9**(3), 523–539. doi:10.1093/biostatistics/kxm049.

Plummer M, Best N, Cowles K, Vines K (2006). "**coda**: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL https://CRAN.R-project.org/doc/Rnews/.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rivest LP (1988). "A Distribution for Dependent Unit Vectors." *Communications in Statistics – Theory and Methods*, **17**(2), 461–483. doi:10.1080/03610928808829634.

Roberts GO, Rosenthal JS, *et al.* (2001). "Optimal Scaling for Various Metropolis-Hastings Algorithms." *Statistical Science*, **16**(4), 351–367. doi:10.1214/ss/1015346320.

Rodríguez CE, Walker SG (2014). "Label Switching in Bayesian Mixture Models: Deterministic Relabeling Strategies." *Journal of Computational and Graphical Statistics*, **23**(1), 25–45. doi:10.1080/10618600.2012.735624.

Rousseau J, Mengersen K (2011). "Asymptotic Behaviour of the Posterior Distribution in Overfitted Mixture Models." *Journal of the Royal Statistical Society B*, **73**(5), 689–710. doi:10.1111/j.1467-9868.2011.00781.x.

Sarkar D (2021). ***lattice**: Trellis Graphics for R*. R package version 0.20-45, URL https://CRAN.R-project.org/package=lattice.

Schwarz G (1978). "Estimating the Dimension of a Model." *The Annals of Statistics*, **6**(2), 461–464. doi:10.1214/aos/1176344136.

Singh H, Hnizdo V, Demchuk E (2002). "Probabilistic Model for Two Dependent Circular Variables." *Biometrika*, **89**(3), 719–723. doi:10.1093/biomet/89.3.719.

Spiegelhalter DJ, Best NG, Carlin BP, Van der Linde A (2002). "Bayesian Measures of Model Complexity and Fit." *Journal of the Royal Statistical Society B*, **64**(4), 583–639. doi:10.1111/1467-9868.00353.

Stephens M (2000). "Dealing With Label Switching in Mixture Models." *Journal of the Royal Statistical Society B*, **62**(4), 795–809. doi:10.1111/1467-9868.00265.

Touw WG, Baakman C, Black J, Te Beek TAH, Krieger E, Joosten RP, Vriend G (2015). "A Series of PDB-Related Databanks for Everyday Needs." *Nucleic Acids Research*, **43**(D1), D364–D368. doi:10.1093/nar/gku1028.

Vehtari A, Gabry J, Magnusson M, Yao Y, Bürkner PC, Paananen T, Gelman A (2020). ***loo**: Efficient Leave-One-Out Cross-Validation and WAIC for Bayesian Models.* R package version 2.4.1, URL https://CRAN.R-project.org/package=loo.

Vehtari A, Gelman A, Gabry J (2015). "Pareto Smoothed Importance Sampling." arXiv:1507.02646 [stat.CO], URL https://arxiv.org/abs/1507.02646.

Vehtari A, Gelman A, Gabry J (2017). "Practical Bayesian Model Evaluation Using Leave-One-out Cross-Validation and WAIC." *Statistics and Computing*, **27**(5), 1413–1432. doi:10.1007/s11222-016-9696-4.

Watanabe S (2013). "A Widely Applicable Bayesian Information Criterion." *Journal of Machine Learning Research*, **14**(Mar), 867–897. doi:10.1007/s11222-014-9463-3.

Yu Y, Meng XL (2011). "To Center or Not to Center: That Is Not the Question – An Ancillarity-Sufficiency Interweaving Strategy (ASIS) for Boosting MCMC Efficiency." *Journal of Computational and Graphical Statistics*, **20**(3), 531–570. doi:10.1198/jcgs.2011.203main.

Zhan X, Ma T, Liu S, Shimizu K (2019). "On Circular Correlation for Data on the Torus." *Statistical Papers*, **60**(6), 1827–1847. doi:10.1007/s00362-017-0897-5.

A. The normalizing constant for von Mises cosine density

Proposition A.1. *The normalizing constant for the density (8) is given by*

$$C_c(\kappa_1, \kappa_2, \kappa_3) = \left[(2\pi)^2 \left\{ I_0(\kappa_1) I_0(\kappa_2) I_0(\kappa_3) + 2 \sum_{n=1}^{\infty} I_n(\kappa_1) I_n(\kappa_2) I_n(\kappa_3) \right\} \right]^{-1}$$

Proof. Without loss of generality, we first assume that the mean parameters in the density (8) are all zero, i.e., $\mu_1 = \mu_2 = 0$. Therefore, our objective boils down to evaluate the integral

$$C_c(\kappa_1, \kappa_2, \kappa_3)^{-1} = \mathcal{I} = \int_0^{2\pi} \int_0^{2\pi} \exp(\kappa_1 \cos x + \kappa_2 \cos y + \kappa_3 \cos(x - y))\, dx\, dy. \tag{18}$$

Now from Equation 9.6.34 of Abramowitz and Stegun (1964), it follows that

$$\begin{aligned} \exp(\kappa_1 \cos x) &= I_0(\kappa_1) + 2 \sum_{l=1}^{\infty} I_l(\kappa_1) \cos(lx) \\ \exp(\kappa_2 \cos y) &= I_0(\kappa_2) + 2 \sum_{m=1}^{\infty} I_m(\kappa_2) \cos(my) \\ \text{and } \exp(\kappa_3 \cos(x - y)) &= I_0(\kappa_3) + 2 \sum_{n=1}^{\infty} I_n(\kappa_3) \cos(n(x - y)). \end{aligned}$$

Therefore, the integrand in (18) can be written as

$$\begin{aligned} & I_0(\kappa_1) I_0(\kappa_2) I_0(\kappa_3) + 2\{I_0(\kappa_2) + I_0(\kappa_3)\} \sum_{l=1}^{\infty} I_l(\kappa_1) \cos(lx) \\ & + 2\{I_0(\kappa_3) + I_0(\kappa_1)\} \sum_{m=1}^{\infty} I_m(\kappa_2) \cos(my) \\ & + 2\{I_0(\kappa_1) + I_0(\kappa_2)\} \sum_{n=1}^{\infty} I_n(\kappa_3) \cos(n(x - y)) \\ & + 8 \sum_{l=1}^{\infty} \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} I_l(\kappa_1) I_m(\kappa_2) I_n(\kappa_3) \cos(lx) \cos(my) \cos(n(x - y)). \end{aligned} \tag{19}$$

Note that for any positive integer q,

$$\int_0^{2\pi} \cos(qz)\, dz = \int_0^{2\pi} \sin(qz)\, dz = 0$$

which implies, for a positive integer n,

$$\begin{aligned} \int_0^{2\pi} \int_0^{2\pi} \cos(n(x - y))\, dx\, dy = & \int_0^{2\pi} \cos(nx)\, dx \int_0^{2\pi} \cos(ny)\, dy \\ & + \int_0^{2\pi} \sin(nx)\, dx \int_0^{2\pi} \sin(ny)\, dy = 0. \end{aligned}$$

(Equality of the double and the iterative integrals are ensured by the Fubini theorem, which is applicable as the integrands and the range of integrals are all finite.)

Thus the (double) integrals of the second, third and fourth terms in (19) are all zero. Hence,

$$\begin{aligned}\mathcal{I} &= (2\pi)^2 I_0(\kappa_1) I_0(\kappa_2) I_0(\kappa_3) \\ &\quad + 8 \int_0^{2\pi}\int_0^{2\pi} \sum_{l=1}^{\infty}\sum_{m=1}^{\infty}\sum_{n=1}^{\infty} I_l(\kappa_1) I_m(\kappa_2) I_n(\kappa_3) \cos(lx)\cos(my)\cos(n(x-y))\, dx\, dy. \qquad (20)\end{aligned}$$

Now, for the second term in (20), first note that

$$\begin{aligned}&\int_0^{2\pi}\int_0^{2\pi} \sum_{l=1}^{\infty}\sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \left| I_l(\kappa_1) I_m(\kappa_2) I_n(\kappa_3) \cos(lx)\cos(my)\cos(n(x-y))\right|\, dx\, dy \\ &\le \int_0^{2\pi}\int_0^{2\pi} \sum_{l=1}^{\infty}\sum_{m=1}^{\infty}\sum_{n=1}^{\infty} I_l(\kappa_1) I_m(\kappa_2) I_n(|\kappa_3|)\, dx\, dy \\ &= \sum_{l=1}^{\infty}\sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \int_0^{2\pi}\int_0^{2\pi} I_l(\kappa_1) I_m(\kappa_2) I_n(|\kappa_3|)\, dx\, dy \quad \text{(by Fubini-Tonelli)} \\ &= (2\pi)^2 \left(\sum_{l=1}^{\infty} I_l(\kappa_1)\right)\left(\sum_{m=1}^{\infty} I_m(\kappa_2)\right)\left(\sum_{n=1}^{\infty} I_n(|\kappa_3|)\right) \\ &< \infty\end{aligned}$$

where the equality in the third line follows from the Fubini-Tonelli theorem for non-negative integrands. Therefore, the Fubini theorem for general integrands can be applied to ensure interchangeability of the sums and the integrals in the second term in (20). In particular, one can write

$$\begin{aligned}&\int_0^{2\pi}\int_0^{2\pi} \sum_{l=1}^{\infty}\sum_{m=1}^{\infty}\sum_{n=1}^{\infty} I_l(\kappa_1) I_m(\kappa_2) I_n(\kappa_3) \cos(lx)\cos(my)\cos(n(x-y))\, dx\, dy \\ &= \sum_{l=1}^{\infty}\sum_{m=1}^{\infty}\sum_{n=1}^{\infty} I_l(\kappa_1) I_m(\kappa_2) I_n(\kappa_3) \int_0^{2\pi}\int_0^{2\pi} \cos(lx)\cos(my)\cos(n(x-y))\, dx\, dy. \qquad (21)\end{aligned}$$

Now, for any positive integers l, m, n,

$$\begin{aligned}\cos(lx)\cos(my)\cos(n(x-y)) &= \cos(lx)\cos(nx)\cos(my)\cos(ny) \\ &\quad + \cos(lx)\sin(nx)\cos(my)\sin(ny).\end{aligned}$$

Observe that for any two positive integers p and q,

$$\int_0^{2\pi} \cos(pz)\cos(qz)\, dz = \pi \mathbb{1}_{\{p=q\}} \text{ and } \int_0^{2\pi} \cos(pz)\sin(qz)\, dz = 0.$$

Therefore, for any positive integers l, m, n,

$$\int_0^{2\pi}\int_0^{2\pi} \cos(lx)\cos(nx)\cos(my)\cos(ny)\, dx\, dy = \pi \mathbb{1}_{\{l=n\}} \pi \mathbb{1}_{\{m=n\}} = \pi^2 \mathbb{1}_{\{l=m=n\}}$$

and

$$\int_0^{2\pi}\int_0^{2\pi} \cos(lx)\sin(nx)\cos(my)\sin(ny)\, dx\, dy = 0.$$

which implies,

$$\int_0^{2\pi} \int_0^{2\pi} \cos(lx)\cos(my)\cos(n(x-y))\,dx\,dy = \pi^2 \mathbb{1}_{\{l=m=n\}}. \tag{22}$$

Therefore, combining (20), (21) and (22), we get

$$\begin{aligned} \mathcal{I} &= (2\pi)^2 I_0(\kappa_1) I_0(\kappa_2) I_0(\kappa_3) + 8\pi^2 \sum_{l=1}^{\infty}\sum_{m=1}^{\infty}\sum_{n=1}^{\infty} I_l(\kappa_1) I_n(\kappa_3) I_m(\kappa_2) \mathbb{1}_{\{l=m=n\}} \\ &= (2\pi)^2 \left\{ I_0(\kappa_1) I_0(\kappa_2) I_0(\kappa_3) + 2\sum_{n=1}^{\infty} I_n(\kappa_1) I_n(\kappa_2) I_n(\kappa_3) \right\} \end{aligned}$$

This completes the proof. □

B. Circular variance and correlation coefficients

B.1. Von Mises sine model

Let $(\psi_1, \psi_2) \sim \mathrm{vM}_2^s(\mu_1, \mu_2, \kappa_1, \kappa_2, \kappa_3)$. Then

1. the Fisher-Lee circular correlation coefficient (11) between ψ_1 and ψ_2 is given by

$$\rho_{\mathrm{FL}}(\psi_1, \psi_2) = \frac{\left(\frac{1}{\bar{C}_s}\frac{\partial \bar{C}_s}{\partial \kappa_3}\right)\left(\frac{1}{\bar{C}_s}\frac{\partial^2 \bar{C}_s}{\partial \kappa_1 \partial \kappa_2}\right)}{\sqrt{\left(\frac{1}{\bar{C}_s}\frac{\partial^2 \bar{C}_s}{\partial \kappa_1^2}\right)\left(1-\frac{1}{\bar{C}_s}\frac{\partial^2 \bar{C}_s}{\partial \kappa_1^2}\right)\left(\frac{1}{\bar{C}_s}\frac{\partial^2 \bar{C}_s}{\partial \kappa_2^2}\right)\left(1-\frac{1}{\bar{C}_s}\frac{\partial^2 \bar{C}_s}{\partial \kappa_2^2}\right)}}.$$

2. the Jammalamadaka-Sarma circular correlation coefficient (10) between Θ and Φ is given by

$$\rho_{\mathrm{JS}}(\psi_1, \psi_2) = \frac{\frac{1}{\bar{C}_s}\frac{\partial \bar{C}_s}{\partial \kappa_3}}{\sqrt{\left(1-\frac{1}{\bar{C}_s}\frac{\partial^2 \bar{C}_s}{\partial \kappa_1^2}\right)\left(1-\frac{1}{\bar{C}_s}\frac{\partial^2 \bar{C}_s}{\partial \kappa_2^2}\right)}}.$$

3. the circular variance for ψ_i, $i = 1, 2$ is given by

$$\mathrm{Var}(\psi_i) = 1 - \frac{1}{\bar{C}_s}\frac{\partial \bar{C}_s}{\partial \kappa_i}.$$

Here $\bar{C}_s = 1/C_s$, where C_s is the normalizing constant of the von Mises sine distribution as defined in (7). Infinite series expressions for partial derivatives of $\bar{C}_s$ constant are provided as follows.

$$
\begin{aligned}
\frac{\partial \bar{C}_s}{\partial \kappa_1} &= 4\pi^2 \sum_{m=0}^{\infty} \binom{2m}{m} \left(\frac{\kappa_3^2}{4\kappa_1\kappa_2}\right)^m I_{m+1}(\kappa_1) I_m(\kappa_2) \\
\frac{\partial \bar{C}_s}{\partial \kappa_2} &= 4\pi^2 \sum_{m=0}^{\infty} \binom{2m}{m} \left(\frac{\kappa_3^2}{4\kappa_1\kappa_2}\right)^m I_m(\kappa_1) I_{m+1}(\kappa_2) \\
\frac{\partial \bar{C}_s}{\partial \kappa_3} &= 8\pi^2 \sum_{m=1}^{\infty} m \binom{2m}{m} \frac{\kappa_3^{2m-1}}{(4\kappa_1\kappa_2)^m} I_m(\kappa_1) I_m(\kappa_2) \\
\frac{\partial^2 \bar{C}_s}{\partial \kappa_1^2} &= 4\pi^2 \sum_{m=0}^{\infty} \binom{2m}{m} \left(\frac{\kappa_3^2}{4\kappa_1\kappa_2}\right)^m \\
&\qquad \left(\frac{I_{m+1}(\kappa_1)}{\kappa_1} + I_{m+2}(\kappa_1)\right) I_m(\kappa_2) \\
\frac{\partial^2 \bar{C}_s}{\partial \kappa_2^2} &= 4\pi^2 \sum_{m=0}^{\infty} \binom{2m}{m} \left(\frac{\kappa_3^2}{4\kappa_1\kappa_2}\right)^m \\
&\qquad I_m(\kappa_1) \left(\frac{I_{m+1}(\kappa_2)}{\kappa_2} + I_{m+2}(\kappa_2)\right) \\
\frac{\partial^2 \bar{C}_s}{\partial \kappa_1 \, \partial \kappa_2} &= 4\pi^2 \sum_{m=0}^{\infty} \binom{2m}{m} \left(\frac{\kappa_3^2}{4\kappa_1\kappa_2}\right)^m I_{m+1}(\kappa_1) I_{m+1}(\kappa_2)
\end{aligned}
$$

B.2. Von Mises cosine model

Let $(\psi_1, \psi_2) \sim \text{vM}_2^c(\mu_1, \mu_2, \kappa_1, \kappa_2, \kappa_3)$. Then

1. The Fisher-Lee circular correlation coefficient (11) between ψ_1 and ψ_2 is given by

$$
\rho_{\text{FL}}(\psi_1, \psi_2) = \frac{\left(\frac{1}{\bar{C}_c}\left\{\frac{\partial \bar{C}_c}{\partial \kappa_3} - \frac{\partial^2 \bar{C}_c}{\partial \kappa_1 \partial \kappa_2}\right\}\right)\left(\frac{1}{\bar{C}_c}\frac{\partial^2 \bar{C}_c}{\partial \kappa_1 \partial \kappa_2}\right)}{\sqrt{\left(\frac{1}{\bar{C}_c}\frac{\partial^2 \bar{C}_c}{\partial \kappa_1^2}\right)\left(1 - \frac{1}{\bar{C}_c}\frac{\partial^2 \bar{C}_c}{\partial \kappa_1^2}\right)\left(\frac{1}{\bar{C}_c}\frac{\partial^2 \bar{C}_c}{\partial \kappa_2^2}\right)\left(1 - \frac{1}{\bar{C}_c}\frac{\partial^2 \bar{C}_c}{\partial \kappa_2^2}\right)}}.
$$

2. The Jammalamadaka-Sarma circular correlation coefficient (10) between Θ and Φ is given by

$$
\rho_{\text{JS}}(\psi_1, \psi_2) = \frac{\frac{1}{\bar{C}_c}\left\{\frac{\partial \bar{C}_c}{\partial \kappa_3} - \frac{\partial^2 \bar{C}_c}{\partial \kappa_1 \partial \kappa_2}\right\}}{\sqrt{\left(1 - \frac{1}{\bar{C}_c}\frac{\partial^2 \bar{C}_c}{\partial \kappa_1^2}\right)\left(1 - \frac{1}{\bar{C}_c}\frac{\partial^2 \bar{C}_c}{\partial \kappa_2^2}\right)}}.
$$

3. The circular variance for ψ_i, $i = 1, 2$ is given by

$$
\text{Var}(\psi_i) = 1 - \frac{1}{\bar{C}_c}\frac{\partial \bar{C}_c}{\partial \kappa_i}.
$$

Here $\bar{C}_c = 1/C_c$ is the reciprocal of the von Mises cosine normalizing constant, as given in

(9). Infinite series expressions for partial derivatives of $\bar{C}_c$ are given as follows.

$$\frac{\partial \bar{C}_c}{\partial \kappa_1} = 4\pi^2 \left\{I_1(\kappa_1)I_0(\kappa_2)I_0(\kappa_3) + \sum_{m=1}^{\infty} I_m(\kappa_2)I_m(\kappa_3)\left[I_{m+1}(\kappa_1) + I_{m-1}(\kappa_1)\right]\right\}$$

$$\frac{\partial \bar{C}_c}{\partial \kappa_2} = 4\pi^2 \left\{I_0(\kappa_1)I_1(\kappa_2)I_0(\kappa_3) + \sum_{m=1}^{\infty} I_m(\kappa_1)I_m(\kappa_3)\left[I_{m+1}(\kappa_2) + I_{m-1}(\kappa_2)\right]\right\}$$

$$\frac{\partial \bar{C}_c}{\partial \kappa_3} = 4\pi^2 \left\{I_0(\kappa_1)I_0(\kappa_2)I_1(\kappa_3) + \sum_{m=1}^{\infty} I_m(\kappa_1)I_m(\kappa_2)\left[I_{m+1}(\kappa_3) + I_{m-1}(\kappa_3)\right]\right\}.$$

$$\frac{\partial^2 \bar{C}_c}{\partial \kappa_1^2} = 2\pi^2 \left\{I_0(\kappa_2)I_0(\kappa_3)[I_0(\kappa_1) + I_2(\kappa_1)] + \sum_{m=1}^{\infty} I_m(\kappa_2)I_m(\kappa_3)[I_{m-2}(\kappa_1) + 2I_m(\kappa_1) + I_{m+2}(\kappa_1)]\right\}$$

$$\frac{\partial^2 \bar{C}_c}{\partial \kappa_2^2} = 2\pi^2 \left\{I_0(\kappa_1)I_0(\kappa_3)[I_0(\kappa_2) + I_2(\kappa_2)] + \sum_{m=1}^{\infty} I_m(\kappa_1)I_m(\kappa_3)[I_{m-2}(\kappa_2) + 2I_m(\kappa_2) + I_{m+2}(\kappa_2)]\right\}$$

$$\frac{\partial^2 \bar{C}_c}{\partial \kappa_1 \partial \kappa_2} = 2\pi^2 \left\{2I_1(\kappa_1)I_1(\kappa_2)I_0(\kappa_3) + \sum_{m=1}^{\infty} I_m(\kappa_3)\left[I_{m+1}(\kappa_1) + I_{m-1}(\kappa_1)\right]\left[I_{m+1}(\kappa_2) + I_{m-1}(\kappa_2)\right]\right\}$$

C. Gradients

For notational simplicity we shall omit the subscripts i and j. Note that, in the sequel, $\boldsymbol{\theta}$ stands for the parameter vector for one generic component and not the entire parameter vector of all components.

C.1. Wrapped normal models

1. *Univariate case.* Here $\boldsymbol{\theta}^\top = (\kappa, \mu)$, and

$$\frac{\partial f_{\text{WN}}(\psi|\boldsymbol{\theta})}{\partial \kappa} = \frac{1}{2\kappa^{1/2}\sqrt{2\pi}} \sum_{\omega \in \mathbb{Z}} \exp\left[-\frac{\kappa}{2}(\psi - \mu - 2\pi\omega)^2\right]\left[1 - \kappa(\psi - \mu - 2\pi\omega)^2\right]$$

$$\frac{\partial f_{\text{WN}}(\psi|\boldsymbol{\theta})}{\partial \mu} = \frac{\kappa^{3/2}}{\sqrt{2\pi}} \sum_{\omega \in \mathbb{Z}} \exp\left[-\frac{\kappa}{2}(\psi - \mu - 2\pi\omega)^2\right](\psi - \mu - 2\pi\omega).$$

2. *Bivariate case.* Here $\boldsymbol{\theta}^\top = (\kappa_1, \kappa_2, \kappa_3, \mu_1, \mu_2)$, $\boldsymbol{\psi}^\top = (\psi_1, \psi_2)$ and

$$\begin{aligned}
\frac{\partial f_{\mathrm{WN}_2}(\boldsymbol{\psi}|\boldsymbol{\theta})}{\partial \kappa_1} &= \frac{1}{4\pi\sqrt{\kappa_{12.3}}} \sum_{(\omega_1,\omega_2)\in\mathbb{Z}^2} E_{\omega_1,\omega_2}\left[\kappa_2 - \kappa_{12.3}(\psi_1 - \mu_1 - 2\pi\omega_1)^2\right] \\
\frac{\partial f_{\mathrm{WN}_2}(\boldsymbol{\psi}|\boldsymbol{\theta})}{\partial \kappa_2} &= \frac{1}{4\pi\sqrt{\kappa_{12.3}}} \sum_{(\omega_1,\omega_2)\in\mathbb{Z}^2} E_{\omega_1,\omega_2}\left[\kappa_1 - \kappa_{12.3}(\psi_2 - \mu_2 - 2\pi\omega_2)^2\right] \\
\frac{\partial f_{\mathrm{WN}_2}(\boldsymbol{\psi}|\boldsymbol{\theta})}{\partial \kappa_3} &= \frac{1}{2\pi\sqrt{\kappa_{12.3}}} \sum_{(\omega_1,\omega_2)\in\mathbb{Z}^2} E_{\omega_1,\omega_2}\left[\kappa_3 - \kappa_{12.3}(\psi_1 - \mu_1 - 2\pi\omega_1)(\psi_2 - \mu_2 - 2\pi\omega_2)\right] \\
\frac{\partial f_{\mathrm{WN}_2}(\boldsymbol{\psi}|\boldsymbol{\theta})}{\partial \mu_1} &= \frac{\sqrt{\kappa_{12.3}}}{2\pi} \sum_{(\omega_1,\omega_2)\in\mathbb{Z}^2} E_{\omega_1,\omega_2}\left[\kappa_1(\psi_1 - \mu_1 - 2\pi\omega_1) + \kappa_3(\psi_2 - \mu_2 - 2\pi\omega_2)\right] \\
\frac{\partial f_{\mathrm{WN}_2}(\boldsymbol{\psi}|\boldsymbol{\theta})}{\partial \mu_2} &= \frac{\sqrt{\kappa_{12.3}}}{2\pi} \sum_{(\omega_1,\omega_2)\in\mathbb{Z}^2} E_{\omega_1,\omega_2}\left[\kappa_3(\psi_1 - \mu_1 - 2\pi\omega_1) + \kappa_2(\psi_2 - \mu_2 - 2\pi\omega_2)\right]
\end{aligned}$$

where

$$\begin{aligned}
E_{\omega_1,\omega_2} = \exp\Big[-\frac{1}{2}\Big\{&\kappa_1(\psi_1 - \mu_1 - 2\pi\omega_1)^2 + \kappa_2(\psi_2 - \mu_2 - 2\pi\omega_2)^2 \\
&+2\kappa_3(\psi_1 - \mu_1 - 2\pi\omega_1)(\psi_2 - \mu_2 - 2\pi\omega_2)\}]
\end{aligned}$$

and $\kappa_{12.3} = \kappa_1\kappa_2 - \kappa_3^2$.

C.2. Von Mises models

1. *Univariate case.* Here $\boldsymbol{\theta}^\top = (\kappa, \mu)$ and

$$\begin{aligned}
\frac{\partial \log f_{\mathrm{vM}}(\psi|\boldsymbol{\theta})}{\partial \kappa} &= \cos(\psi - \mu) - \frac{I_1(\kappa)}{I_0(\kappa)} \\
\frac{\partial \log f_{\mathrm{vM}}(\psi|\boldsymbol{\theta})}{\partial \mu} &= \kappa \sin(\psi - \mu).
\end{aligned}$$

2. *Bivariate sine model.* Here $\boldsymbol{\theta}^\top = (\kappa_1, \kappa_2, \kappa_3, \mu_1, \mu_2)$, $\boldsymbol{\psi}^\top = (\psi_1, \psi_2)$ and

$$\begin{aligned}
\frac{\partial \log f_{\mathrm{vM}_2^s}(\boldsymbol{\psi}|\boldsymbol{\theta})}{\partial \kappa_1} &= \cos(\psi_1 - \mu_1) - \frac{\partial \bar{C}_s(\kappa_1, \kappa_1, \kappa_3)/\partial \kappa_1}{\bar{C}_s(\kappa_1, \kappa_1, \kappa_3)} \\
\frac{\partial \log f_{\mathrm{vM}_2^s}(\boldsymbol{\psi}|\boldsymbol{\theta})}{\partial \kappa_2} &= \cos(\psi_2 - \mu_2) - \frac{\partial \bar{C}_s(\kappa_1, \kappa_1, \kappa_3)/\partial \kappa_2}{\bar{C}_s(\kappa_1, \kappa_1, \kappa_3)} \\
\frac{\partial \log f_{\mathrm{vM}_2^s}(\boldsymbol{\psi}|\boldsymbol{\theta})}{\partial \kappa_3} &= \sin(\psi_1 - \mu_1)\sin(\psi_2 - \mu_2) - \frac{\partial \bar{C}_s(\kappa_1, \kappa_1, \kappa_3)/\partial \kappa_3}{\bar{C}_s(\kappa_1, \kappa_1, \kappa_3)} \\
\frac{\partial \log f_{\mathrm{vM}_2^s}(\boldsymbol{\psi}|\boldsymbol{\theta})}{\partial \mu_1} &= \kappa_1 \sin(\psi_1 - \mu_1) - \kappa_3 \cos(\psi_1 - \mu_1)\sin(\psi_2 - \mu_2) \\
\frac{\partial \log f_{\mathrm{vM}_2^s}(\boldsymbol{\psi}|\boldsymbol{\theta})}{\partial \mu_2} &= \kappa_2 \sin(\psi_2 - \mu_2) - \kappa_3 \sin(\psi_1 - \mu_1)\cos(\psi_2 - \mu_2)
\end{aligned}$$

where $\bar{C}_s(\kappa_1, \kappa_1, \kappa_3) = 1/C_s(\kappa_1, \kappa_1, \kappa_3)$ and expressions for the partial derivatives are provided in Appendix B.1.

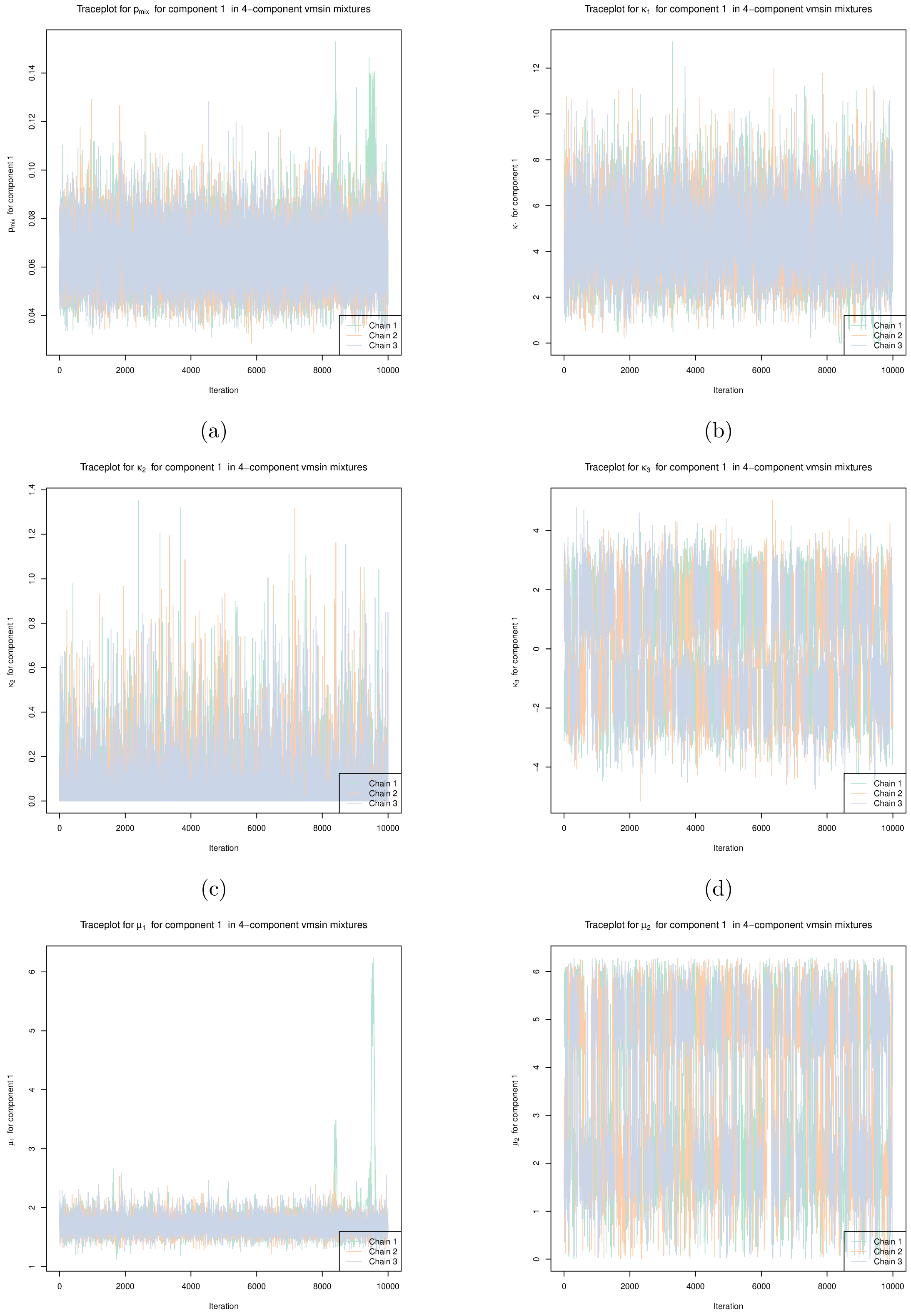
3. *Bivariate cosine model.* Here $\boldsymbol{\theta}^\top = (\kappa_1, \kappa_2, \kappa_3, \mu_1, \mu_2)$, $\boldsymbol{\psi}^\top = (\psi_1, \psi_2)$ and

$$\frac{\partial \log f_{\mathrm{vM}_2^c}(\boldsymbol{\psi}|\boldsymbol{\theta})}{\partial \kappa_1} = \cos(\psi_1 - \mu_1) - \frac{\partial \bar{C}_c(\kappa_1, \kappa_1, \kappa_3)/\partial \kappa_1}{\bar{C}_c(\kappa_1, \kappa_1, \kappa_3)}$$
$$\frac{\partial \log f_{\mathrm{vM}_2^c}(\boldsymbol{\psi}|\boldsymbol{\theta})}{\partial \kappa_2} = \cos(\psi_2 - \mu_2) - \frac{\partial \bar{C}_c(\kappa_1, \kappa_1, \kappa_3)/\partial \kappa_2}{\bar{C}_c(\kappa_1, \kappa_1, \kappa_3)}$$
$$\frac{\partial \log f_{\mathrm{vM}_2^c}(\boldsymbol{\psi}|\boldsymbol{\theta})}{\partial \kappa_3} = \cos(\psi_1 - \mu_1 - \psi_2 + \mu_2) - \frac{\partial \bar{C}_c(\kappa_1, \kappa_1, \kappa_3)/\partial \kappa_3}{\bar{C}_c(\kappa_1, \kappa_1, \kappa_3)}$$
$$\frac{\partial \log f_{\mathrm{vM}_2^c}(\boldsymbol{\psi}|\boldsymbol{\theta})}{\partial \mu_1} = \kappa_1 \sin(\psi_1 - \mu_1) + \kappa_3 \sin(\psi_1 - \mu_1 - \psi_2 + \mu_2)$$
$$\frac{\partial \log f_{\mathrm{vM}_2^c}(\boldsymbol{\psi}|\boldsymbol{\theta})}{\partial \mu_2} = \kappa_2 \sin(\psi_2 - \mu_2) - \kappa_3 \sin(\psi_1 - \mu_1 - \psi_2 + \mu_2)$$

where $\bar{C}_c(\kappa_1, \kappa_1, \kappa_3) = 1/C_c(\kappa_1, \kappa_1, \kappa_3)$ and infinite series expressions for the partial derivatives are provided in Appendix B.2.

D. Traceplots for 4 component `vmsin` models

This appendix displays traceplots of the MCMC samples for the `vmsin` model parameters in the fitted *best* (4-component) model (Section 4.1) both before and after calling `fix_label` to resolve label switching. Traces for the model parameters before and after the `fix_label` call are displayed in Figures 18–21 and 22–25, respectively. These traces demonstrate that the independent parallel chains often explore different permutation modes; see, e.g., κ_1 for component 2 in Figure 19 (b). These permutation modes are largely resolved by `fix_label`; consider, e.g., Figure 24 (b), the counterpart of Figure 19 (b) after fixing label switching.



(a)

(b)

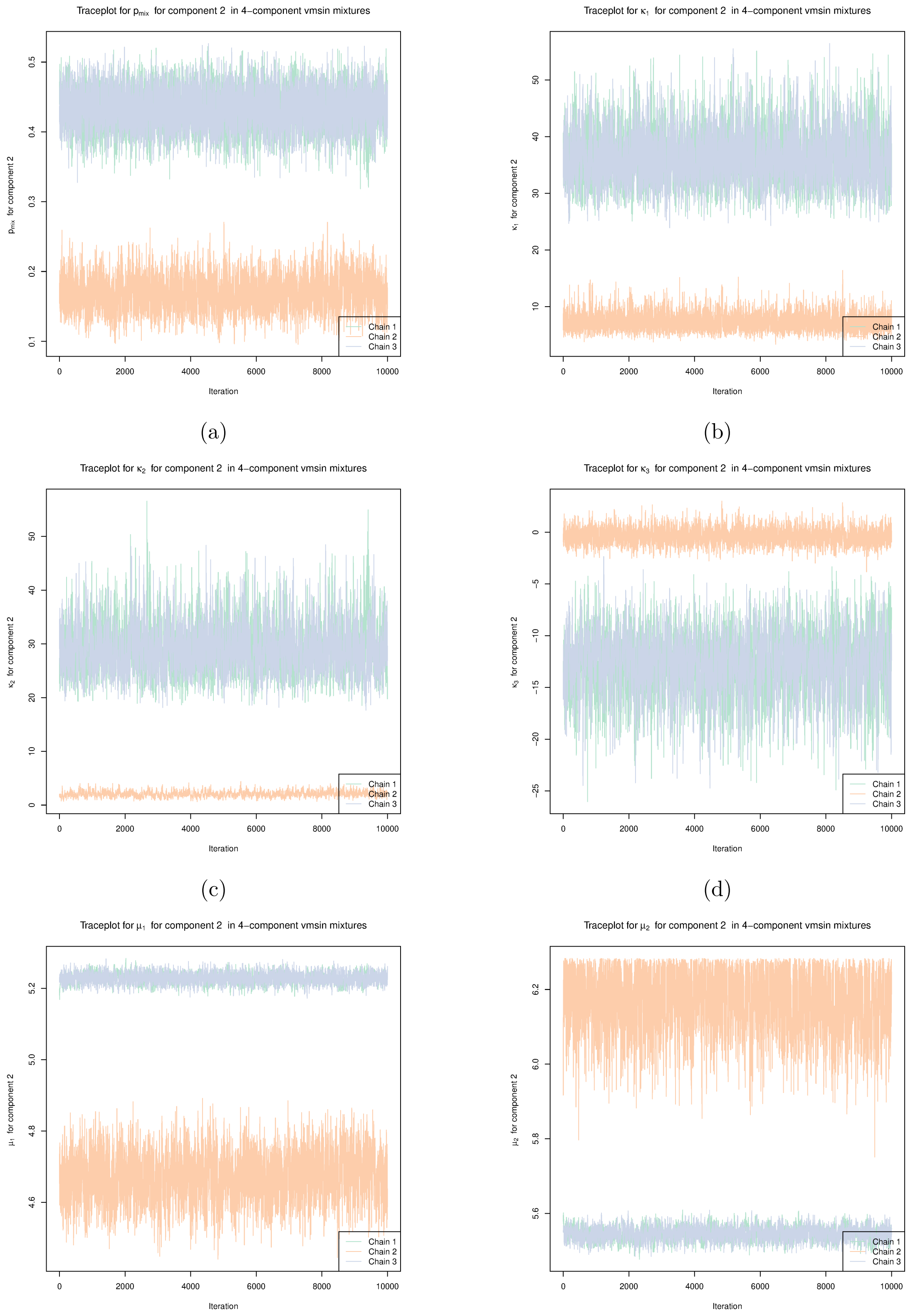
(c)

(d)

(e)

(f)

Figure 18: Traceplots for parameters in the first component for the Markov chain associated with the best fitted vmsin mixture model.



(a)

(b)

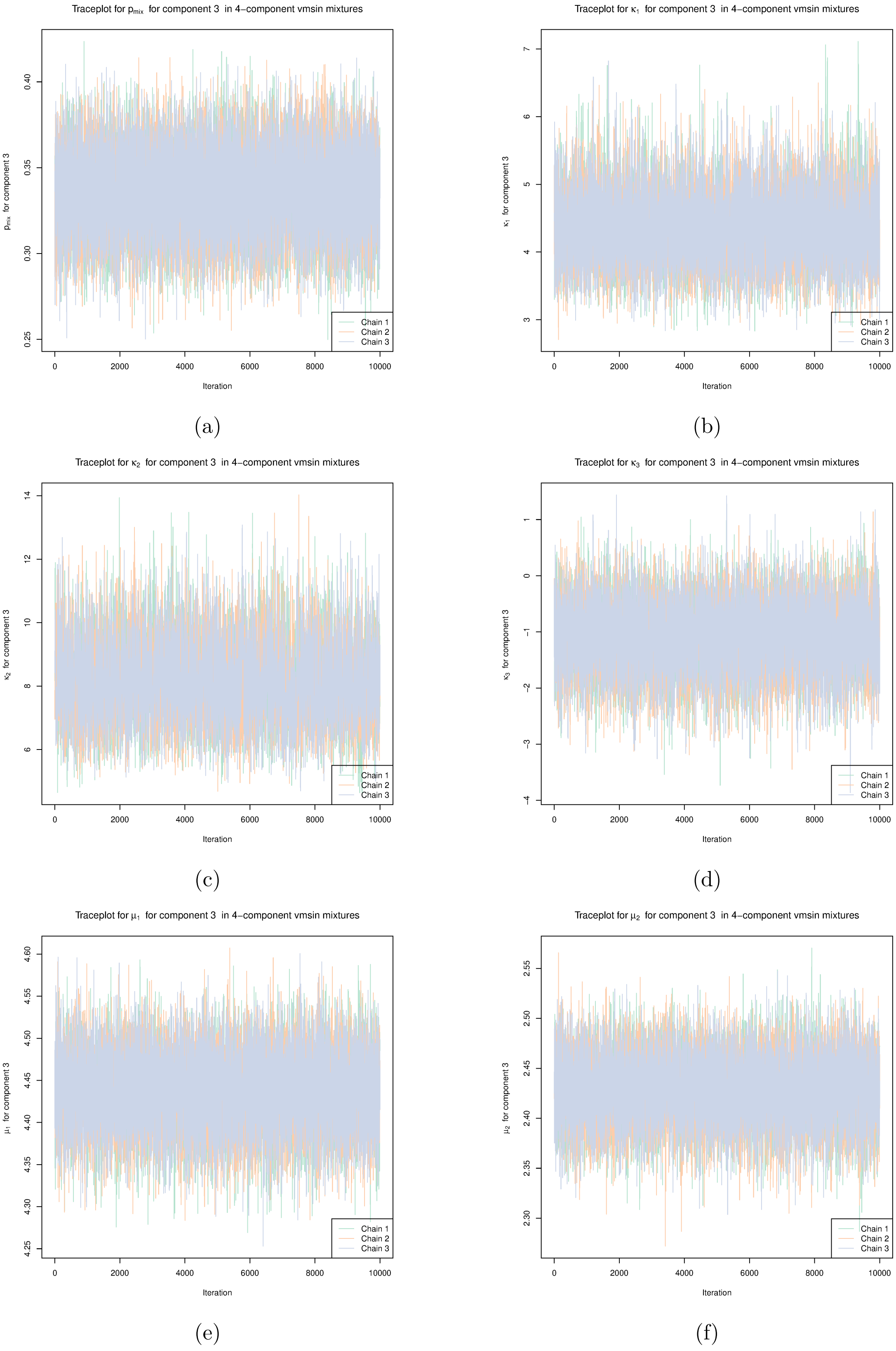
(c)

(d)

(e)

(f)

Figure 19: Traceplots for parameters in the second component for the Markov chain associated with the best fitted vmsin mixture model.

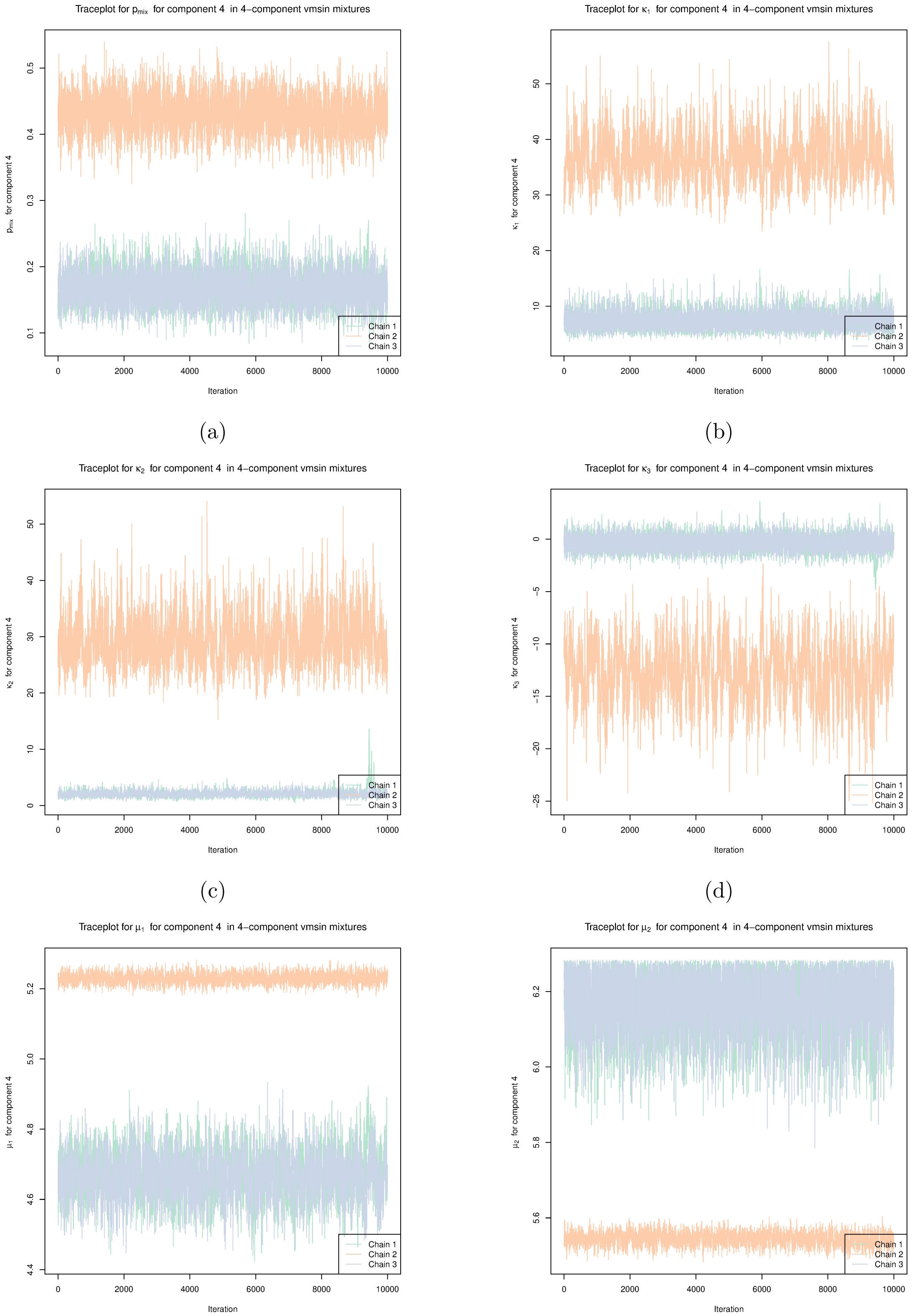


(a) (b)

(c) (d)

(e) (f)

Figure 20: Traceplots for parameters in the third component for the Markov chain associated with the best fitted vmsin mixture model.

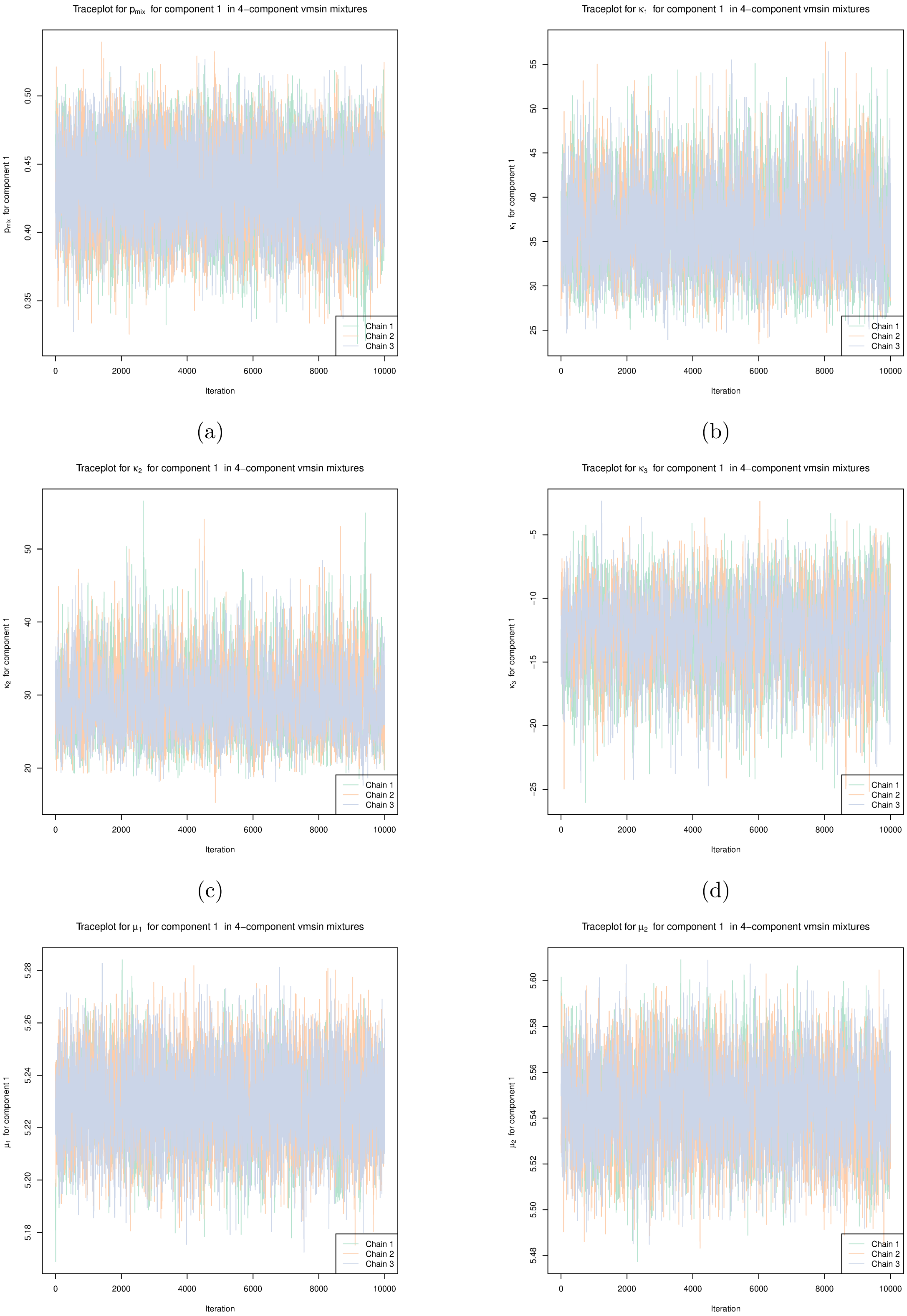


(a) (b)

(c) (d)

(e) (f)

Figure 21: Traceplots for parameters in the fourth component for the Markov chain associated with the best fitted vmsin mixture model.

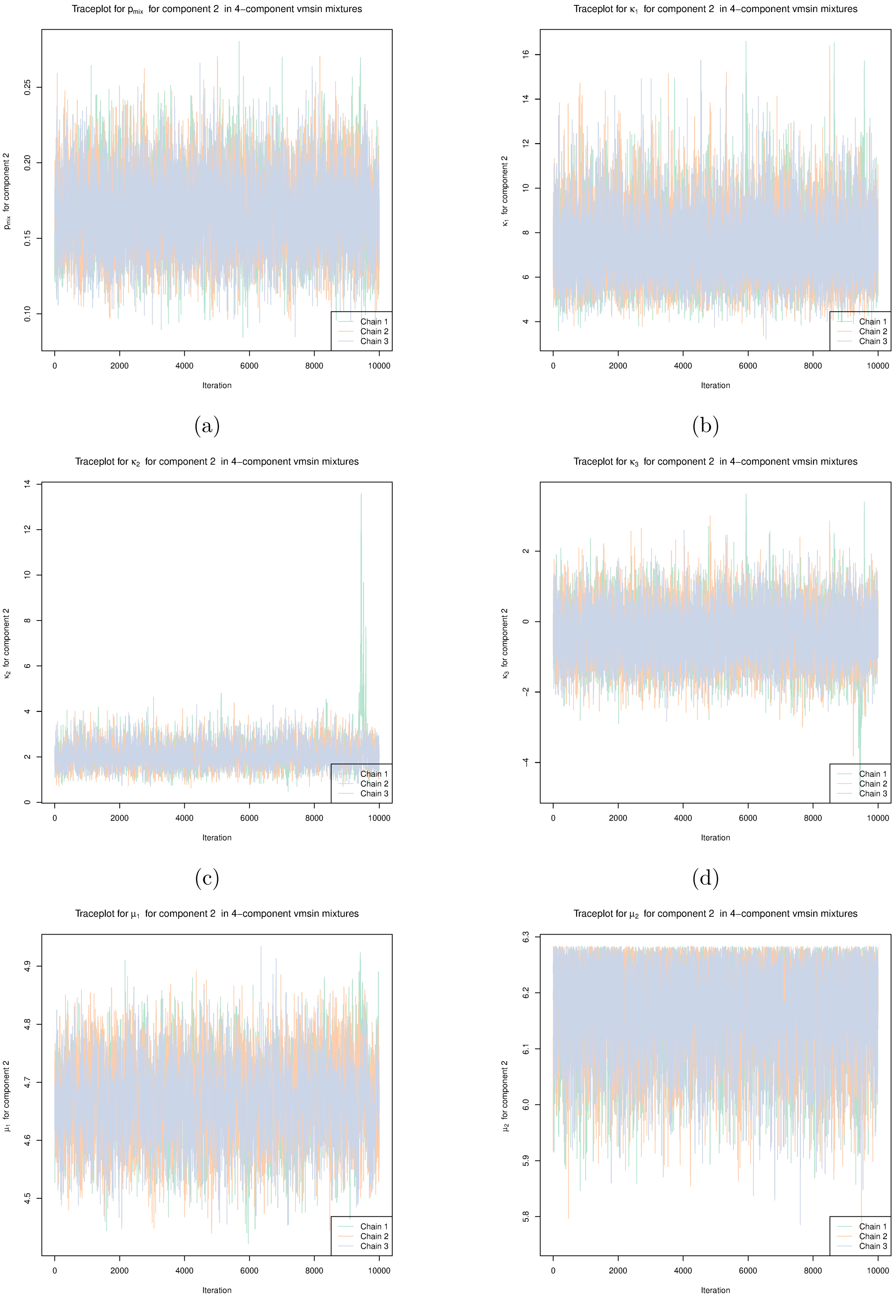


(a) (b)

(c) (d)

(e) (f)

Figure 22: Traceplots for parameters in the first component for the Markov chain associated with the best fitted vmsin mixture model, after undoing label switching.



(a)

(b)

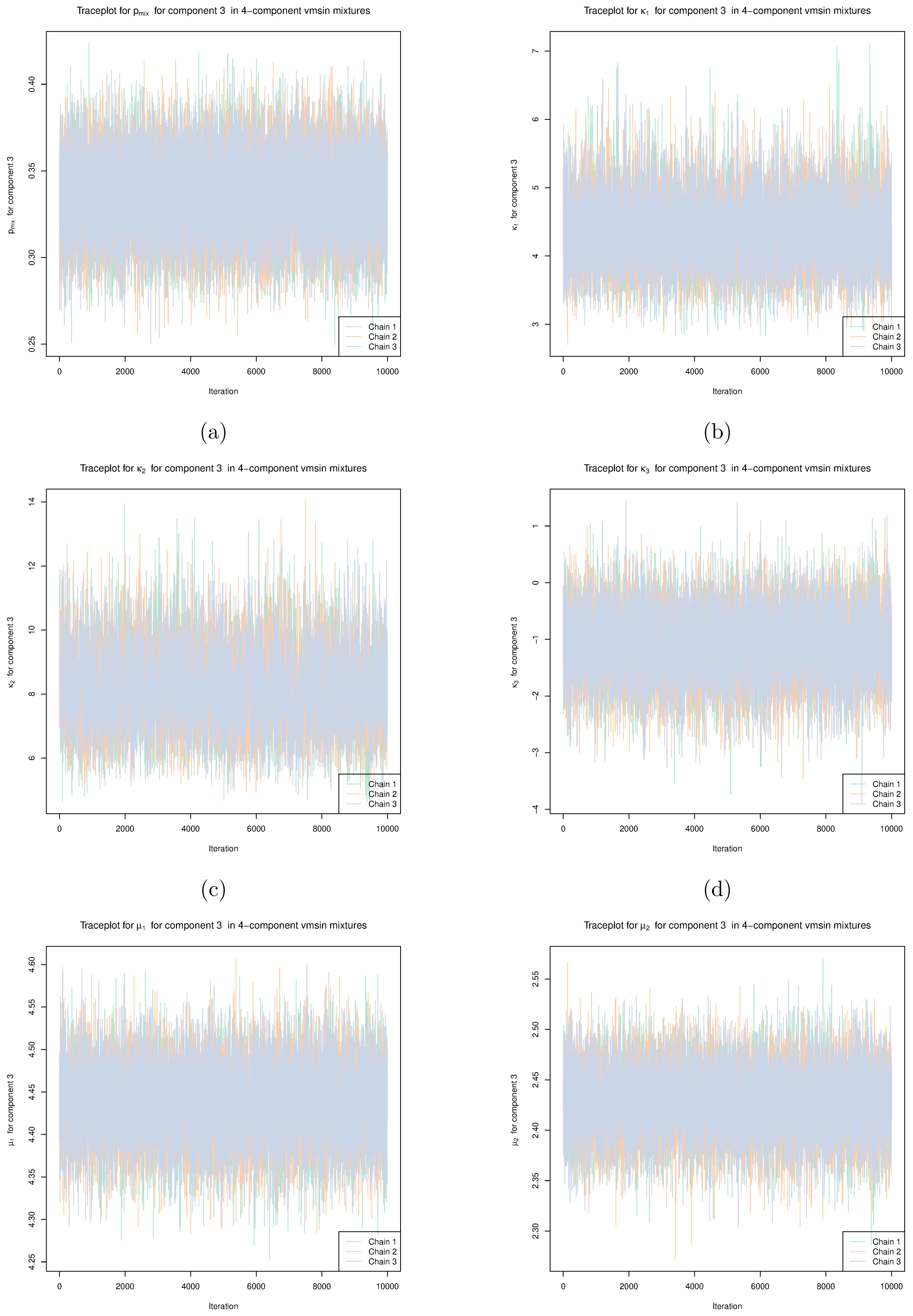
(c)

(d)

(e)

(f)

Figure 23: Traceplots for parameters in the second component for the Markov chain associated with the best fitted vmsin mixture model, after undoing label switching.



Traceplot for p_{mix} for component 3 in 4–component vmsin mixtures

Traceplot for κ_1 for component 3 in 4–component vmsin mixtures

Traceplot for κ_2 for component 3 in 4–component vmsin mixtures

Traceplot for κ_3 for component 3 in 4–component vmsin mixtures

Traceplot for μ_1 for component 3 in 4–component vmsin mixtures

Traceplot for μ_2 for component 3 in 4–component vmsin mixtures

(a)

(b)

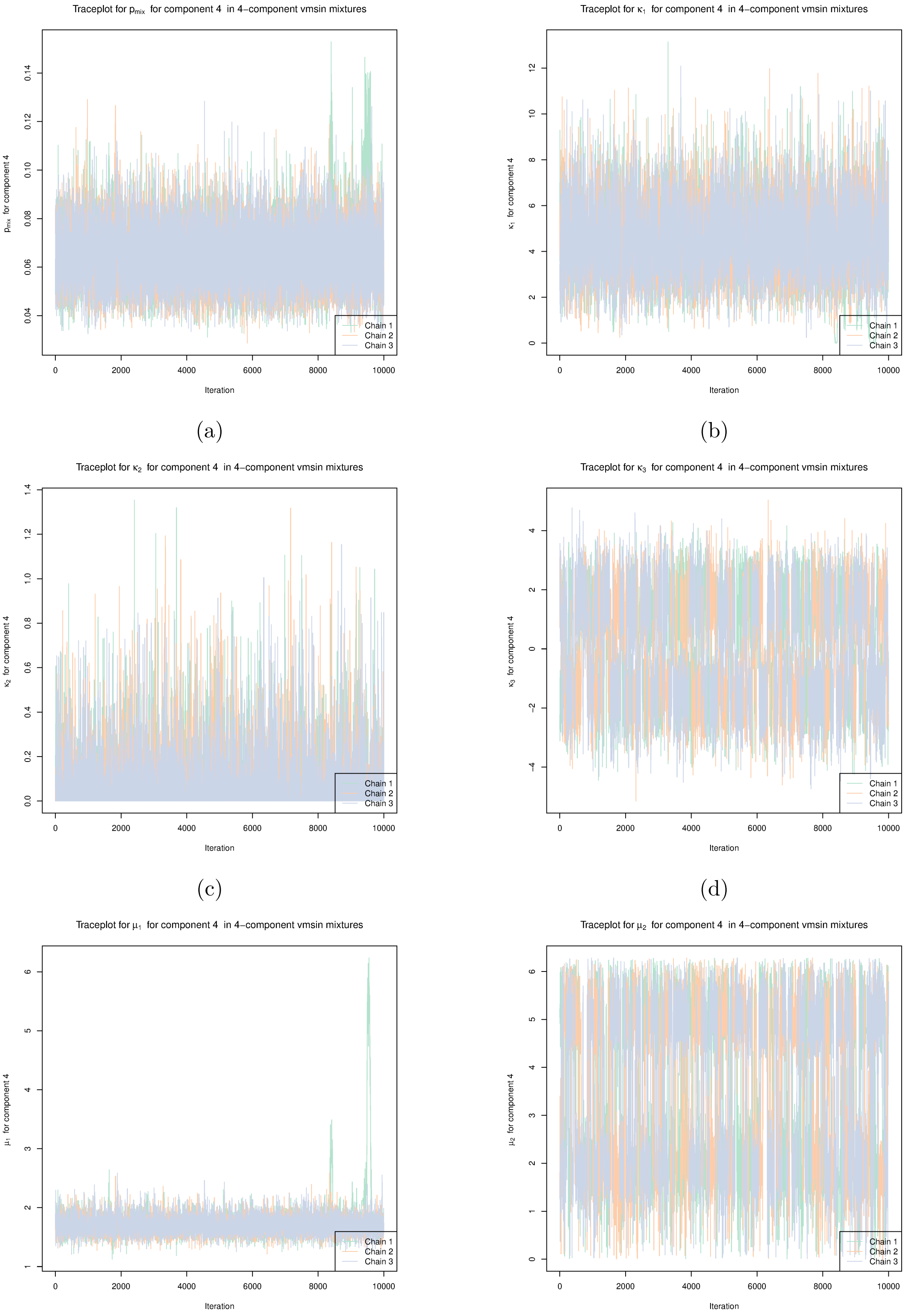
(c)

(d)

(e)

(f)

Figure 24: Traceplots for parameters in the third component for the Markov chain associated with the best fitted vmsin mixture model, after undoing label switching.


Traceplot for p_{mix} for component 4 in 4−component vmsin mixtures

Traceplot for κ_1 for component 4 in 4−component vmsin mixtures

Traceplot for κ_2 for component 4 in 4−component vmsin mixtures

Traceplot for κ_3 for component 4 in 4−component vmsin mixtures

Traceplot for μ_1 for component 4 in 4−component vmsin mixtures

Traceplot for μ_2 for component 4 in 4−component vmsin mixtures

(a) (b)

(c) (d)

(e) (f)

Figure 25: Traceplots parameters in the fourth component for the Markov chain associated with the best fitted vmsin mixture model, after undoing label switching.

Affiliation:

Saptarshi Chakraborty
Department of Biostatistics
State University of New York at Buffalo
718 Kimball Tower
Buffalo, NY 14214, United States of America
E-mail: chakrab2@buffalo.edu

Samuel W. K. Wong
Department of Statistics and Actuarial Science
University of Waterloo
200 University Ave W
Waterloo, ON N2L 3G1, Canada
E-mail: samuel.wong@uwaterloo.ca

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
October 2021, Volume 99, Issue 11 *Submitted:* 2017-01-07
doi:10.18637/jss.v099.i11 *Accepted:* 2019-12-07