ClusTorus: An R Package for Prediction and Clustering on the Torus by Conformal Prediction

by Seungki Hong and Sungkyu Jung

Abstract Protein structure data consist of several dihedral angles, lying on a multidimensional torus. Analyzing such data has been and continues to be key in understanding functional properties of proteins. However, most of the existing statistical methods assume that data are on Euclidean spaces, and thus they are improper to deal with angular data. In this paper, we introduce the package **ClusTorus** specialized to analyzing multivariate angular data. The package collects some tools and routines to perform algorithmic clustering and model-based clustering for data on the torus. In particular, the package enables the construction of conformal prediction sets and predictive clustering, based on kernel density estimates and mixture model estimates. A novel hyperparameter selection strategy for predictive clustering is also implemented, with improved stability and computational efficiency. We demonstrate the use of the package in clustering protein dihedral angles from two real data sets.

1 Introduction

Multivariate angular or circular data have found applications in some research domains including geology (e.g., paleomagnetic directions) and bioinformatics (e.g., protein dihedral angles). Due to the cyclic nature of angles, usual vector-based statistical methods are not directly applicable to such data. A p-variate angle $\theta = (\theta_1, \cdots, \theta_p)^T$ lies on the p-dimensional torus $\mathbb{T}^p = [0, 2\pi)^p$ in which the angles 0 and 2π are identified as the same point. Likewise, angles θ and $\theta \pm 2\pi$ are the same data point on the torus. Thus, statistical models and predictions on the torus should reflect this geometric constraint.

A prominent example in which multivariate angular data appear is the analysis of protein structures. As described in Branden and Tooze (1999), the functional properties of proteins are determined by the ordered sequences of amino acids and their spatial structures. These structures are determined by several dihedral angles, and thus, protein structures are commonly described on multidimensional tori. The p-dimensional torus $\mathbb{T}^p$ is the sample space we consider in this paper. Especially, for the 2-dimensional case, the backbone chain angles ϕ, ψ of a protein are commonly visualized by the Ramachandran plot, a scatter plot of dihedral angles in a 2-dimensional flattened torus $\mathbb{T}^2$ (Lovell et al., 2003; Oberholser, 2010). In Figure 1, several clustering results are visualized on the Ramachandran plot for the protein angles of SARS-CoV-2 virus, which caused the 2020-2021 pandemic (Coronaviridae Study Group of the International Committee on Taxonomy of Viruses. et al., 2020). Since the structures in protein angles are related to functions of the protein, it is of interest to analyze the scatter of the angles through, for example, density estimation and clustering. Note that the protein structure data are routinely collected and publicly available at Protein Data Bank (Berman et al., 2003) and importing such data into R is made easy by the package **bio3d** (Grant et al., 2006, 2021).

We introduce the R package **ClusTorus** (Jung and Hong, 2021) which provides various tools for handling and clustering multivariate angular data on the torus. The package provides angular adaptations of usual clustering methods such as the k-means clustering and pairwise angular distances, which can be used as an input for distance-based clustering algorithms, and implements a novel clustering method based on conformal prediction framework (Vovk et al., 2005). Also implemented in the package are the EM algorithms and an elliptical k-means algorithm for fitting mixture models on the torus, and a kernel density estimation. We will introduce various clustering tools implemented in the package, explaining choices in conformal prediction using two sets of example data. We also present the theoretical and technical background, and demonstrate these tools with R codes.

For data on the torus, there are a few previous works for mixture modeling and clustering. Mardia et al. (2007) proposed a mixture of bivariate von Mises distributions for data on $\mathbb{T}^2$, with an application to modeling protein backbone chain angles. Mardia et al. (2012) proposed a density estimation on the torus, based on a mixture of approximated von Mises sine distributions, for higher dimensional cases, but the proposed EM algorithm tends to be unstable when sample sizes are limited. The R package **BAMBI** (Chakraborty and Wong, 2019, 2020) provides routines to fit such von Mises mixture models using MCMC, but is only applicable to bivariate (and univariate) angles in $\mathbb{T}^2$. We have implemented EM algorithms (for $p = 2$) and the elliptical k-means algorithm (for any p), originally proposed for vector-valued data (Sung and Poggio, 1998; Bishop, 2006; Shin et al., 2019), for fitting mixture models on the torus. To the best of authors' knowledge, **ClusTorus** is the first implementation of methods for

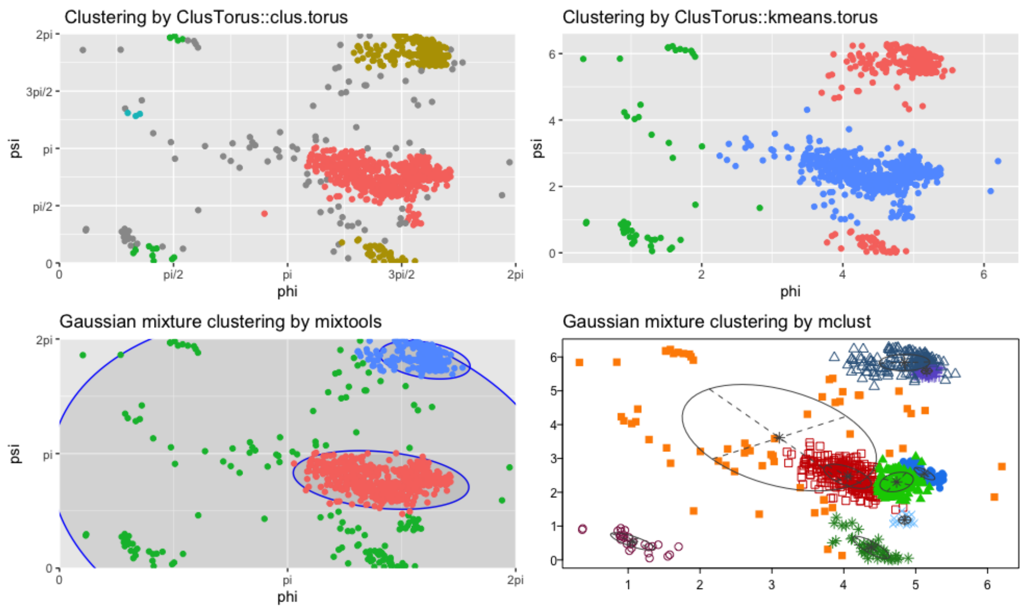

Figure 1: Several clustering results on Ramachandran plot for SARS-CoV-2 by using `clus.torus` (top left) and `kmeans.torus` (top right), both implemented in **ClusTorus**, `mixtools::mvnormalmixEM` (bottom left), in which the number of components 3 is prespecified, and `mclust::Mclust` (bottom right), in which the number of components is chosen by BIC. Gray points in the top-left panel are "outliers", automatically assigned by `clus.torus`.

fitting mixture models on multidimensional tori of any dimension.

Algorithmic clustering for data on the torus has also been proposed. For example, Gao et al. (2018) used an extrinsic k-means algorithm for clustering protein angles. While this algorithm is not always satisfactory, it is implemented in **ClusTorus** for a quick-and-dirty analysis. The top right panel of Figure 1 depicts the result of applying this algorithm with $k = 3$. Note that the popular R packages **mixtools** (Benaglia et al., 2009) and **mclust** (Scrucca et al., 2016) provide misleading clustering results, when applied to data on the torus. As we illustrate in Figure 1, these tools do not take into account the cyclic nature of the angular data.

The main contribution of **ClusTorus** is an implementation of the predictive clustering approaches of Jung et al. (2021) and Shin et al. (2019). For this, the conformal prediction framework of Vovk et al. (2005) is extended for multivariate angular data. The conformal prediction is a distribution-free method of constructing prediction sets, and our implementation uses kernel density estimates and mixture models, both based on the multivariate von Mises distribution (Mardia et al., 2012). Furthermore, by using Gaussian-like approximations of the von Mises distributions and a graph-theoretic approach, flexible clusters, composed of unions of ellipsoids on $\mathbb{T}^p$, can be identified. The proposed predictive clustering can be obtained by simply using `clus.torus` as follows.

```
library(ClusTorus)
set.seed(2021)
ex <- clus.torus(SARS_CoV_2)
plot(ex)
```

The result of the predictive clustering is visualized in the top left panel of Figure 1, which is generated by `plot(ex)`. The dataset `SARS_CoV_2`, included in **ClusTorus**, collects the dihedral angles ϕ, ψ in the backbone chain B of SARS-CoV-2 spike glycoprotein. The raw coronavirus protein data are available at Protein Data Back with id 6VXX (Walls et al., 2020), and can be retrieved by using R package **bio3d**. The function `clus.torus` performs three core procedures—conformal prediction, hyperparameter selection and cluster assignment—for predictive clustering.

The rest of this article focuses on introducing the three core procedures: (i) the conformal prediction framework, including our choices of the conformity scores, (ii) hyperparameter selection and (iii) cluster assignment. After demonstrating how the package **ClusTorus** can be used for clustering of $\mathbb{T}^2$- and $\mathbb{T}^4$-valued data, we describe how the main function `clus.torus` and other clustering algorithms such as k-means and hierarchical clustering can be used to analyze data on the torus. In the Appendix, we provide technical details and options in fitting mixture models on the torus, and a list of S3 classes defined in **ClusTorus**.

2 Conformal prediction

The conformal prediction framework (Vovk et al., 2005) is one of the main ingredients of our development. Based on the work of Vovk et al. (2005) and Lei et al. (2013, 2015), we briefly introduce the basic concepts and properties of conformal prediction. Suppose that we observe a sample of size n, $X_i \sim F$ where $X_i \in \mathbb{T}^p$ for each i and that the sequence $\mathbb{X}_n = \{X_1, \cdots, X_n\}$ is *exchangeable*. Then, for a new $X_{n+1} \sim F$, the prediction set $C_n = C_n(\mathbb{X}_n)$ is said to be valid at level $1-\alpha$ if:

$$P(X_{n+1} \in C_n) \geq 1 - \alpha, \quad \alpha \in (0,1), \tag{1}$$

where P is the corresponding probability measure for $\mathbb{X}_{n+1} = \mathbb{X}_n \cup \{X_{n+1}\}$.

For a given $x \in \mathbb{T}^p$, write $\mathbb{X}_n(x) = \mathbb{X}_n \cup \{x\}$. Consider the null hypothesis $H_0 : X_{n+1} = x$, where $X_{n+1} \sim F$. To test the hypothesis, the conformal prediction framework uses *conformity scores* σ_i defined as follows:

$$\sigma_i(x) := g(X_i, \mathbb{X}_n(x)), \ \forall i = 1, \cdots, n+1,$$
$$\sigma(x) := g(x, \mathbb{X}_n(x)) = \sigma_{n+1}(x),$$

for some real valued function g, which measures the conformity or similarity of a point to the given set. If $X_{(1)}, \cdots, X_{(n+1)}$ are ordered to satisfy $\sigma_{(1)} \leq \cdots \leq \sigma_{(n+1)}$ for $\sigma_{(i)} = g\left(X_{(i)}, \mathbb{X}_{n+1}\right)$, then we may say that $X_{(n+1)}$ is the most similar point to $\mathbb{X}_{n+1}$.

Consider the following quantity:

$$\pi(x) = \frac{1}{n+1}\sum_{i=1}^{n+1} I(\sigma_i(x) \leq \sigma_{n+1}(x)), \quad I(A) = \begin{cases} 1, & A \text{ is true,} \\ 0, & \text{otherwise,} \end{cases}$$

which can be understood as a p-value for the null hypothesis H_0. The *conformal prediction set* of level $1-\alpha$ is constructed as

$$C_n^\alpha = \{x : \pi(x) > \alpha\}. \tag{2}$$

Because the sequence $\mathbb{X}_n(x)$ is exchangeable under H_0, $\pi(x)$ is uniformly distributed on $\left\{\frac{1}{n+1}, \cdots, 1\right\}$. With this property, it can be shown that the conformal prediction set is valid for finite samples, i.e., (1) holds with C_n replaced by C_n^α for any F, that is, the prediction set is distribution-free (Lei et al., 2013). The performance of the conformal prediction highly depends on the choice of conformity score σ. In some previous works on conformal prediction (Lei et al., 2013, 2015; Shin et al., 2019; Jung et al., 2021), the quality of prediction sets using density based conformity scores has been satisfactory.

We demonstrate a construction of the conformal prediction set with a kernel density estimate-based conformity score, defined later in (3), for the data shown in Figure 1. With the conformity score given by (3), cp.torus.kde computes the conformal prediction set C_n^α at a given level $1-\alpha$ ($\alpha = 0.1$ below), by performing the kernel density estimation. The function tests the inclusion in C_n^α of each point (ϕ, ψ) over a fine grid of $\mathbb{T}^2$, and the result of the testing is shown as the boolean indices in the column Cn of the output below. The columns Lminus and Lplus provide approximated prediction sets, defined in Jung et al. (2021).

```
cp.kde <- cp.torus.kde(SARS_CoV_2)
cp.kde

Conformal prediction sets (Lminus, Cn, Lplus) based on kde with concentration 25

Testing inclusion to the conformal prediction set with level = 0.1 :
-------------
          phi psi Lminus    Cn Lplus level
1  0.00000000   0  FALSE FALSE FALSE   0.1
2  0.06346652   0  FALSE FALSE FALSE   0.1
3  0.12693304   0  FALSE FALSE FALSE   0.1
4  0.19039955   0  FALSE FALSE FALSE   0.1
5  0.25386607   0  FALSE FALSE FALSE   0.1
6  0.31733259   0  FALSE FALSE FALSE   0.1
7  0.38079911   0  FALSE FALSE FALSE   0.1
8  0.44426563   0  FALSE FALSE FALSE   0.1
9  0.50773215   0  FALSE FALSE FALSE   0.1
10 0.57119866   0  FALSE FALSE FALSE   0.1
```

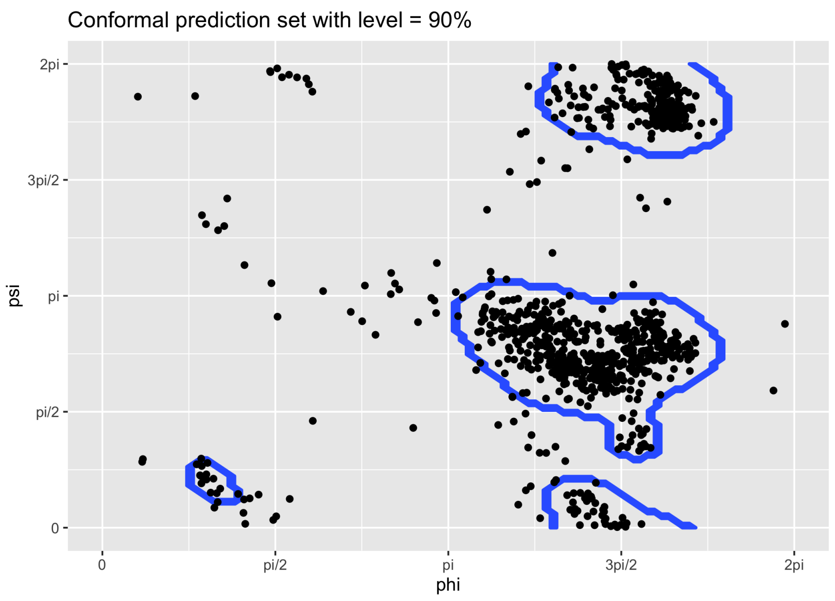

Figure 2: The Ramachandran plot for SARS-CoV-2, with boundaries of the conformal prediction set, whose conformity score is (3) with $\kappa = 25$ for level $1 - \alpha = 0.9$.

```
 9990 rows are omitted.
```

The concentration parameter κ of the kernel density estimation and the level(s) of the prediction set can be designated by providing arguments `concentration` and `level`. By default, these values are set as `concentration = 25` and `level = 0.1`.

The output `cp.kde` is an S3 object with class `cp.torus.kde`, for which a generic method `plot` is available. The conformal prediction set for `SARS_CoV_2` data can be displayed on the Ramachandran plot, as follows. The result is shown in Figure 2.

```
plot(cp.kde)
```

Inductive conformal prediction

If the sample size n and the number N of grid points over $\mathbb{T}^p$ are large, evaluating $n + N$ conformity scores may take a long time. That is, constructing the conformal prediction set suffers from high computational costs. A workaround for this inefficiency is *inductive conformal prediction*, which enjoys significantly lower computational cost. The inductive conformal prediction framework is based on splitting the data into two sets. The algorithm for inductive conformal prediction is given in Algorithm 1.

Algorithm 1 Inductive Conformal Prediction

```
1: procedure INDUCTIVE CONFORMAL PREDICTION($\{X_1, \cdots, X_n\}, \alpha, n_1 < n$)
2:     Split the data randomly into $\mathbb{X}_1 = \{X_1, \cdots, X_{n_1}\}$, $\mathbb{X}_2 = \{X_{n_1+1}, \cdots, X_n\}$.
3:     Construct $\sigma$ with $\sigma(x) = g(x, \mathbb{X}_1)$ for some function $g$.
4:     Put $\sigma_i = g(X_{n_1+i}, \mathbb{X}_1)$ and order as $\sigma_{(1)} \le \cdots \le \sigma_{(n_2)}$, where $n_2 = n - n_1$.
5:     Construct $\hat{C}_n^\alpha = \left\{x : \sigma(x) \ge \sigma_{(i_{n_2,\alpha})}\right\}$ where $i_{n,\alpha} = \lfloor (n+1)\alpha \rfloor$.
6: end procedure
```

While the sizes n_1 and n_2 of two split data sets can be of any size, they are typically set as equal sizes. It is well-known that the output $\hat{C}_n^\alpha$ of the algorithm also satisfies the distribution-free finite-sample validity (Vovk et al., 2005; Lei et al., 2015). For fast computation, the inductive conformal prediction is primarily used in constructing prediction sets and clustering, in our implementation of **ClusTorus**. Specifically, `icp.torus` implements Algorithm 1 for several prespecified conformity scores. As already mentioned, we need to choose the conformity score σ carefully for better clustering performances.

Before we discuss our choices of the conformity scores, we first illustrate how the functions in **ClusTorus** are used to produce inductive conformal prediction sets. The following codes show a calculation of the inductive conformal prediction set for the data `SARS_CoV_2`. The conformal prediction set with the conformity score given by kernel density estimates (3) can be constructed by `icp.torus` and `icp.torus.eval`. The function `icp.torus` computes σ_i's in line 4 of Algorithm 1 and `icp.torus.eval` tests whether pre-specified evaluation points are included in $\hat{C}_n^\alpha$. If these evaluation points are not supplied, then `icp.torus.eval` creates a grid of size 100×100 (for $p = 2$).

```
set.seed(2021)
icp.torus.kde <- icp.torus(SARS_CoV_2, model = "kde", concentration = 25)
icp.kde <- icp.torus.eval(icp.torus.kde, level = 0.1)
icp.kde

Conformal prediction set (Chat_kde)

Testing inclusion to the conformal prediction set with level = 0.1:
-------------
          X1 X2 inclusion
1  0.00000000  0     FALSE
2  0.06346652  0     FALSE
3  0.12693304  0     FALSE
4  0.19039955  0     FALSE
5  0.25386607  0     FALSE
6  0.31733259  0     FALSE
7  0.38079911  0     FALSE
8  0.44426563  0     FALSE
9  0.50773215  0     FALSE
10 0.57119866  0     FALSE

 9990 rows are omitted.
```

In the codes above, the data splitting for `icp.torus` is done internally, and can be inspected by `icp.torus.kde$split.id`.

We now introduce our choices for the conformity score σ in the next two subsections.

Conformity score from kernel density estimates

For the 2-dimensional case, Jung et al. (2021) proposed to use the kernel density estimate based on the von Mises kernel (Marzio et al., 2011) for the conformity score. A natural extension to the p-dimensional tori, for $p \geq 2$, is

$$g\left(u, \mathbb{X}_n\left(x\right)\right) = \frac{1}{n+1}\sum_{i=1}^{n+1} K_\kappa\left(u - X_i\right), \quad K_\kappa\left(v\right) = \prod_{i=1}^{p} \frac{e^{\kappa\cos\left(v_i\right)}}{2\pi I_0\left(\kappa\right)}, \quad v = \left(v_1, \cdots, v_p\right)^T \in \mathbb{T}^p \tag{3}$$

where I_0 is the modified Bessel function of the first kind of order 0, and κ is a prespecified concentration parameter. The function `kde.torus` provides the multivariate von Mises kernel density estimation. For conformal prediction, we take $\sigma\left(x_i\right) = g\left(x_i, \mathbb{X}_n\left(x\right)\right)$, and for inductive conformal prediction, we take $\sigma\left(x\right) = g\left(x, \mathbb{X}_1\right)$.

Conformity scores from mixtures of multivariate von Mises

Our next choices of conformity scores are based on mixture models. Since the multivariate normal distributions are not defined on $\mathbb{T}^p$, we instead use the multivariate von Mises distribution (Mardia et al., 2008), whose density on $\mathbb{T}^p$ is

$$f\left(y; \mu, \kappa, \Lambda\right) = C\left(\kappa, \Lambda\right) \exp\left\{-\frac{1}{2}\left[\kappa^T\left(2 - 2c\left(y, \mu\right)\right) + s\left(y, \mu\right)^T \Lambda s\left(y, \mu\right)\right]\right\} \tag{4}$$

where $y = (y_1, \cdots, y_p)^T \in \mathbb{T}^p$, $\mu = (\mu_1, \cdots, \mu_p)^T \in \mathbb{T}^p$, $\kappa = (\kappa_1, \ldots, \kappa_p)^T \in (0, \infty)^p$, $\Lambda = (\lambda_{j,l})$ for $1 \le j, l \le p$, $-\infty < \lambda_{jl} < \infty$,

$$c(y, \mu) = (\cos(y_1 - \mu_1), \cdots, \cos(y_p - \mu_p))^T,$$
$$s(y, \mu) = (\sin(y_1 - \mu_1), \cdots, \sin(y_p - \mu_p))^T,$$
$$(\Lambda)_{jl} = \lambda_{jl} = \lambda_{lj},\ j \neq l, \quad (\Lambda)_{jj} = \lambda_{jj} = 0,$$

and for some normalizing constant $C(\kappa, \Lambda) > 0$. We write $f(y; \theta) = f(y; \mu, \kappa, \Lambda)$ for $\theta = (\mu, \kappa, \Lambda)$.

For any positive integer J and a mixing probability $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_J)$, consider a J-mixture model:

$$p(u; \boldsymbol{\pi}, \boldsymbol{\theta}) = \sum_{j=1}^{J} \pi_j f(u; \theta_j) \tag{5}$$

where $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_J)$, $\theta_j = (\mu_j, \kappa_j, \Lambda_j)$ for $j = 1, \cdots, J$. Let $(\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\theta}})$ be appropriate estimators of $(\boldsymbol{\pi}, \boldsymbol{\theta})$ based on $\mathbb{X}_1$. The plug-in density estimate based on (5) is then

$$p(\cdot; \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\theta}}) = \sum_{j=1}^{J} \hat{\pi}_j f(\cdot; \hat{\theta}_j), \tag{6}$$

which can be used as a conformity score by setting $g(\cdot, \mathbb{X}_1) = \hat{p}(\cdot)$. Assuming high concentrations, an alternative conformity score can be set as $g(\cdot, \mathbb{X}_1) = p^{max}(\cdot, \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\theta}})$ where

$$p^{max}(u; \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\theta}}) := \max_{j=1,\cdots,J} \left(\hat{\pi}_j f(u; \hat{\theta}_j)\right) \approx p(u; \hat{\boldsymbol{\pi}}, \hat{\boldsymbol{\theta}}). \tag{7}$$

On the other hand, Mardia et al. (2012) introduced an approximated density function f^* for the p-variate von Mises sine distribution (4) for sufficiently high concentrations and when $\Sigma \succ 0$:

$$f^*(y;, \mu, \Sigma) = (2\pi)^{-p/2} |\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}\left[\kappa^T (2 - 2c(y, \mu)) + s(y, \mu)^T \Lambda s(y, \mu)\right]\right\}$$

where $(\Sigma^{-1})_{jl} = \lambda_{jl}$, $(\Sigma^{-1})_{jj} = \kappa_j$, $j \neq l$. By further approximating via $\theta \approx \sin\theta$, $1 - \frac{\theta^2}{2} \approx \cos\theta$, we write

$$f^*(y;, \mu, \Sigma) \approx (2\pi)^{-p/2} |\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}\left[(y \ominus \mu)^T \Sigma^{-1} (y \ominus \mu)\right]\right\}, \tag{8}$$

where the angular subtraction $\ominus$ stands for

$$X \ominus Y := \left(\arg\left(e^{i(\phi_{x1} - \phi_{y1})}\right), \cdots, \arg\left(e^{i(\phi_{xp} - \phi_{yp})}\right)\right)^T,$$

for $X = (\phi_{x1}, \cdots, \phi_{xp})^T \in \mathbb{T}^p$ and $Y = (\phi_{y1}, \cdots, \phi_{yp})^T \in \mathbb{T}^p$ as defined in Jung et al. (2021) for $p = 2$. By replacing the von Mises density f in (7) with the approximate normal density (8), $\log(p^{max}(\cdot; \boldsymbol{\pi}, \boldsymbol{\theta}))$ is approximated by

$$\log(p^{max}(u; \boldsymbol{\pi}, \boldsymbol{\theta})) \approx \frac{1}{2} \max_j e(u; \pi_j, \theta_j) + c,$$
$$e(u; \pi_j, \theta_j) = -(u \ominus \mu_j)^T \Sigma_j^{-1} (u \ominus \mu_j) + 2\log\pi_j - \log|\Sigma_j| \tag{9}$$

where $\theta_j = (\mu_j, \Sigma_j)$, $\mu_j = (\mu_{1j}, \cdots, \mu_{pj})^T \in \mathbb{T}^p$, $\Sigma_j \in \mathbb{R}^{p \times p}$ and a constant $c \in \mathbb{R}$. Our last choice of the conformity score is

$$g(\cdot, \mathbb{X}_1) = \max_j e(\cdot, \hat{\pi}_j, \hat{\theta}_j). \tag{10}$$

Note that with this choice of conformity score, the conformal prediction set can be expressed as the union of ellipsoids on the torus. That is, the following equalities are satisfied (Shin et al., 2019; Jung et al., 2021): Let C_n^e be the level $1-\alpha$ prediction set using (10). Then

$$\begin{aligned} C_n^e &:= \left\{ x \in \mathbb{T}^p : g\left(x, \mathbb{X}_1\right) \geq g\left(X_{(i_{n_2,\alpha})}, \mathbb{X}_1\right) \right\} \\ &= \bigcup_{j=1}^{J} \hat{E}_j\left(\sigma_{(i_{n_2,\alpha})}\right) \end{aligned} \tag{11}$$

where $\hat{E}_j(t) = \left\{ x \in \mathbb{T}^p : \left(x \ominus \hat{\mu}_j\right)^T \hat{\Sigma}_j^{-1} \left(x \ominus \hat{\mu}_j\right) \leq 2\log \hat{\pi}_j - \log\left|\hat{\Sigma}_j\right| - t \right\}$ for $t \in \mathbb{R}$. Note that $\hat{E}_j(t)$ is automatically vanished if $t \geq 2\log\hat{\pi}_j - \log\left|\hat{\Sigma}_j\right|$.

Implementation

We have implemented four conformity scores, described in the previous section. These are based on

1. kernel density estimate (3),
2. mixture model (6),
3. max-mixture model (7), and
4. ellipsoids obtained by approximating the max-mixture (10).

The function `icp.torus` in **ClusTorus** computes these conformity scores using the inductive conformal prediction framework, and returns `icp.torus` object(s). Table 1 illustrates several important arguments of the function `icp.torus`.

Arguments	Descriptions
`data`	$n \times d$ matrix of toroidal data on $[0, 2\pi)^d$ or $[-\pi, \pi)^d$
`model`	A string. One of "kde", "mixture", and "kmeans" which determines the model or estimation methods. If "kde", the model is based on the kernel density estimates. It supports the kde-based conformity score only. If "mixture", the model is based on the von Mises mixture, fitted with an EM algorithm. It supports the von Mises mixture and its variants based conformity scores. If "kmeans", the model is also based on the von Mises mixture, but the parameter estimation is implemented with the elliptical k-means algorithm illustrated in Appendix. It supports the log-max-mixture based conformity score only. If the dimension of data space is greater than 2, only "kmeans" is supported. Default is `model = "kmeans"`.
`J`	A scalar or numeric vector for the number(s) of components for `model = c("mixture", "kmeans")`. Default is `J = 4`.
`concentration`	A scalar or numeric vector for the concentration parameter(s) for `model = "kde"`. Default is `concentration = 25`.

Table 1: Key arguments and descriptions for the function `icp.torus`

The argument `model` of the function `icp.torus` indicates which conformity score is used. By setting `model = "kde"`, the kde-based conformity score (3) is used. By setting `model = "mixture"` the mixture model (6) is estimated by an EM algorithm, and conformity scores based on (6), (7), (10) are all provided. Setting `model = "kmeans"` provides a mixture model fit by the elliptical *k*-means algorithm and conformity score based on (10).

The arguments `J` and `concentration` specify the model fitting hyperparameters. To compute conformity scores based on kernel density estimate (3), one needs to specify the concentration parameter κ. Likewise, the number of mixtures, J, needs to be specified in order to fit the mixture model (6) and the variants (7) and (10). The function `icp.torus` takes either a single value (e.g., `J = 4` is the default), or a vector (e.g., `J = 4:30` or `concentration = c(25,50)`) for arguments `J` and `concentration`. If `J` (or `concentration`) is a scalar, then `icp.torus` returns an `icp.torus` object.

On the other hand, if `J` (or `concentration`) is a numeric vector containing at least two values, then `icp.torus` returns a list of `icp.torus` objects, one for each value in `J` (or `concentration`, respectively). Typically, the hyperparameter J (or κ) is not predetermined, and one needs to choose among a set of candidates. A list of `icp.torus` objects, evaluated for each candidate in vector-valued `J` (or `concentration`) is required for our hyperparameter selection procedure, discussed in a later section.

Let us present an R code example for creating an `icp.torus` object, fitted with `model = "kmeans"` (the default value for argument `model`) and `J = 12`.

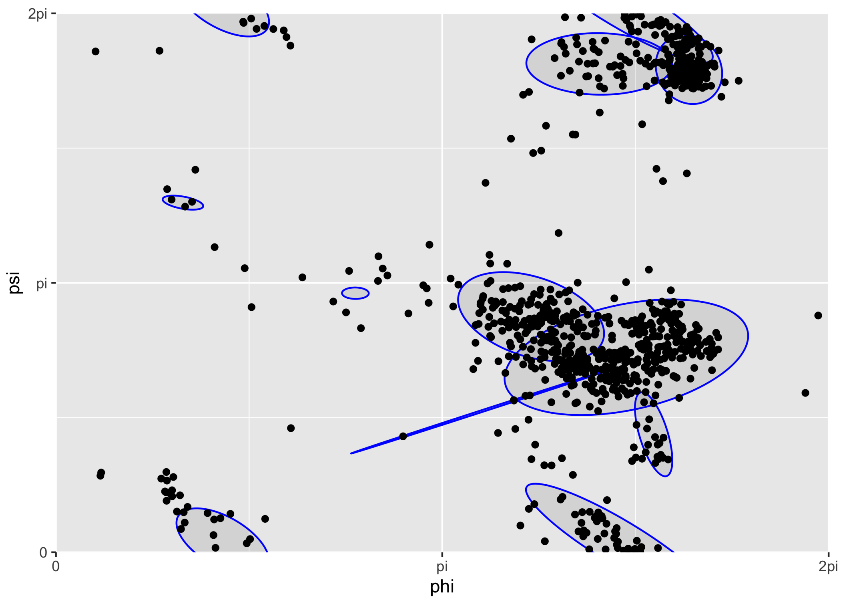

Figure 3: The Ramachandran plot for SARS-CoV-2, with conformal prediction set whose conformity score is (10) with $J = 12$ for level $\alpha = 0.1111$. The plot demonstrates the union of ellipses as (11).

```
set.seed(2021)
icp.torus.12 <- icp.torus(SARS_CoV_2, J = 12)
plot(icp.torus.12, level = 0.1111)
```

The `icp.torus` object has an S3 method `plot`, and the R code `plot(icp.torus.12,level = 0.1111)` plots the ellipses in (11) with α specified by argument `level = 0.1111`. The union of these ellipses is in fact the inductive conformal prediction set of level $1 - \alpha$. The boundaries of the inductive conformal prediction set can be displayed by specifying `ellipse = FALSE`, as follows.

```
plot(icp.torus.12, ellipse = FALSE)
```

The resulting graphic is omitted.

Conformity scores based on mixture model and its variants need appropriate estimators of the parameters, $\boldsymbol{\pi}$ and $\boldsymbol{\theta}$. If the parameters are poorly estimated, the conformal prediction sets will be constructed trivially and thus become useless. We have implemented two methods of estimation: EM algorithms and the elliptical k-means algorithm, also known as the generalized Lloyd's algorithm (Sung and Poggio, 1998; Bishop, 2006; Shin et al., 2019). EM algorithms for the mixture model (6) are described in Jung et al. (2021), for the 2-dimensional case. Since the EM estimates require long computation time and large sample sizes, extensions to higher-dimensional tori do not seem to apt. The EM estimates of the mixture model parameters can be naturally used for the case of max-mixture (7) and ellipsoids (10) as well. The argument `model = "mixture"` of `icp.torus` works only for the 2-dimensional case. On the other hand, the elliptical k-means algorithm converges much faster even for moderately high-dimensional tori. The elliptical k-means algorithm is used for estimating parameters in the approximated normal density (8), and for computation of the conformity score of ellipsoids (10). The elliptical k-means algorithms for data on the torus are further discussed in the Appendix.

Table 2 summarizes the four choices of conformity scores in terms of model-fitting methods, dimensionality of the data space, and whether clustering is available. Our predictive clustering is implemented only based on the "ellipsoids" conformity score (10). The rational for this choice is due to the relatively simple form of prediction sets (a union of ellipsoids (11)).

3 Clustering by conformal prediction

We now describe our clustering strategies using the conformal prediction sets. Suppose for now that the level α and the hyperparameter J of the prediction set are given. The basic idea of clustering is to take each connected component of the prediction set as a cluster. For this, we need an algorithm identifying connected components from any prediction set. Since the prediction sets are in general of irregular shapes, such an identification is a quite difficult task. However, as shown in Jung et al. (2021), if the conformal prediction set is of the form (11), clusters are identified by testing the intersection

Conformity Scores	EM	k-means	dim = 2	dim > 2	Clustering
Kernel density (3)			✓		
Mixture (6)	✓		✓		
Max-mixture (7)	✓		✓		
Ellipsoids (10)	✓	✓	✓	✓	✓

Table 2: Conformity scores against available fitting methods, dimensions of the torus, and whether cluster assignment is available.

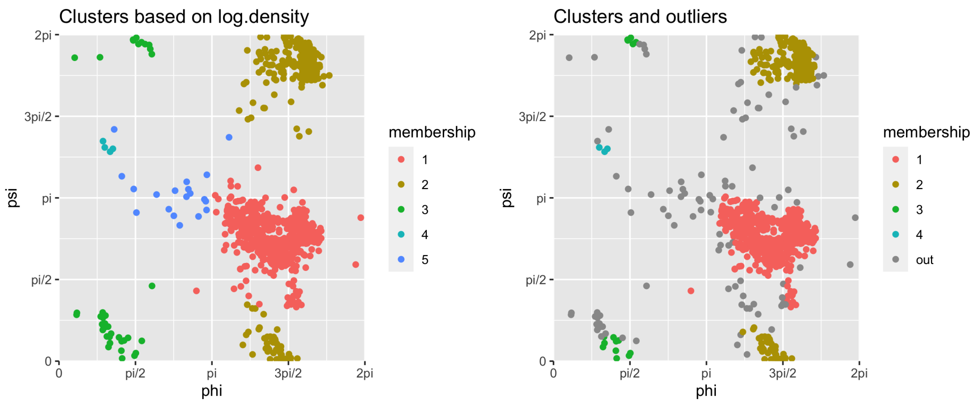

Figure 4: 1

of ellipsoids. Suppose $C_n^e = \cup_{j=1}^J \hat{E}_j$ where each $\hat{E}_j$ is an ellipsoid. Let the (i,j)th entry of a square matrix A be 0 if $\hat{E}_i \cap \hat{E}_j = \emptyset$, 1 otherwise. Then, A is the adjacent matrix of a graph whose nodes and edges represent the ellipsoids and intersections, respectively. The adjacent matrix A gives a partition $I_1, \cdots, I_K \subseteq \{1, \cdots, J\}$ satisfying

$$\hat{E}_{i_k} \cap \hat{E}_{i_{k'}} = \emptyset, \quad k \neq k'$$

where $1 \leq k, k' \leq K, i_k \in I_k, i_{k'} \in I_{k'}$. This implies that the union of ellipsoids, $U_k = \cup_{i \in I_k} \hat{E}_i$, whose indices are in a connected component I_k for some k, can be regarded as a cluster. That is, $U_1, \cdots, U_K$ are the disjoint clusters. With this, the conformal prediction set naturally generates K clusters. Note that testing the intersection of ellipsoids can be done efficiently (which is a univariate root finding problem (Gilitschenski and Hanebeck, 2012)), while testing the intersection of arbitrarily shaped sets is not feasible in general. This is the reason why we only use the conformity score of the form (10), the prediction set from which is exactly the union of ellipsoids.

We now describe how the cluster labels are assigned to data points. Each data point included in the prediction set is automatically assigned to the cluster which contains the point. For the data points which are not included in the conformal prediction set, we have implemented two different types for cluster assignment, as defined in Jung et al. (2021). The first is to assign the *closest* cluster label. The notion of closest cluster can be defined either by the Mahalanobis distance $(x \ominus \hat{\mu}_j)^T \hat{\Sigma}_j^{-1} (x \ominus \hat{\mu}_j)$, the approximate log-density (9), or the largest posterior probability $\hat{P}(Y = k | X = x)$. For example, for $x \notin C_n^e$, let E_i be the set with the largest approximate log-density $\hat{e}_i(x)$. If $i \in I_k$, then x is assigned to the cluster k. These provide three choices of cluster assignment, depending on the definition of "closeness." The last choice is to regard the excluded points as outliers. That is, if $x \notin C_n^e$, then the point x is labeled as "outlier." This outlier-disposing clustering may be more appropriate for the cases where some of data points are truely outliers. Figure 4 compares the two different types of clustering assignment.

The function `cluster.assign.torus`, which takes as input an `icp.torus` object and `level` α, generates the clusters as we described above. The output of the function is an S3 object with class `cluter.obj`, and includes the cluster memberships of all data points, for each and every cluster assignment method we discussed above. The output of `cluster.assign.torus` includes the number of clusters detected, the cluster assignment results for the first 10 observations, and cluster sizes, as shown in the code example below.

```
c <- cluster.assign.torus(icp.torus.12, level = 0.1111)
```

```
c

Number of clusters: 5
-------------
Clustering results by log density:
 [1] 1 1 1 1 2 4 2 1 3 1
cluster sizes: 538 372 39 4 19

Clustering results by posterior:
 [1] 5 5 5 5 2 4 3 5 3 5
cluster sizes: 6 310 104 4 548

Clustering results by representing outliers:
 [1] 1 1 1 1 2 6 2 1 6 1
cluster sizes: 508 343 15 3 0 103

 Note: cluster id = 6 represents outliers.

Clustering results by Mahalanobis distance:
 [1] 1 1 1 1 2 4 2 1 3 1
cluster sizes: 533 372 39 4 24

962 clustering results are omitted.
```

The clustering results contained in the object c can be visualized as follows.

```
plot(c, assignment = "log.density")
plot(c, assignment = "outlier")
```

The results are displayed in Figure 4. When the argument `assignment` is not specified, the outlier disposing assignment is chosen by default.

4 Hyperparameter selection

Poor choices of conformity score result in too wide prediction sets. Thus, we need to choose the hyperparameters elaborately for a better conformal prediction set and for a better clustering performance. The hyperparameters are the concentration parameter κ (for the case (3)) or the number of mixture components J (for the cases (6), (7), (10)), as well as the level α for all cases. There have been some efforts to select the optimal hyperparameters by introducing adequate criteria. Lei et al. (2013) and Jung et al. (2021) each proposed criteria based on the minimum volume of the conformal prediction set. However, as we shall see, these approaches become computationally infeasible for higher dimensions.

We briefly review the criterion used in Jung et al. (2021). Assume for now that mixture models are used; that is, (J, α) are the hyperparameters of interest. For a set $C \subseteq \mathbb{T}^p$, let $\mu(C)$ be the volume of C. Without loss of generality, we can assume that $\mu\left(\mathbb{T}^p\right) = 1$. For a given level α, the optimal choice of hyperparameter J minimizes $\mu\left(C_n(\alpha, J)\right)$ of conformal prediction set $C_n\left(\alpha, J\right)$. To choose α and J altogether, Jung et al. (2021) proposed to use the following criterion:

$$\left(\hat{\alpha}, \hat{J}\right) = \arg\min_{\alpha, J} \alpha + \mu\left(C_n\left(\alpha, J\right)\right). \tag{12}$$

Note that if (κ, α) are the hyperparameters, then J and $\hat{J}$ in (12) are replaced by κ and $\hat{\kappa}$.

To evaluate (12), one needs to have a set of candidates for J (or κ), and conformal prediction sets corresponding to each choice of J (or κ, respectively). For this purpose, the function `icp.torus` is designed to take as input a set of hyperparameter candidates. As an example, the following code evaluates the inductive conformal prediction sets for data `SARS_CoV_2`, fitted by mixture models with the number of components given by each $J = 3, 4, \ldots, 35$.

```
set.seed(2021)
icp.torus.objects <- icp.torus(SARS_CoV_2, J = 3:35)
```

The result, `icp.torus.objects`, is a list of 33 `icp.torus` objects. Evaluating Jung et al. (2021)'s criterion (12) is implemented in the function `hyperparam.torus`. There, the criterion (12) is termed "elbow", since the minimizer $(\hat{\alpha}, \hat{J})$ is typically found at an elbow of the graph of the objective function.

```
hyperparam.out <- hyperparam.torus(icp.torus.objects)
hyperparam.out

Type of conformity score: kmeans general
Optimizing method: elbow
-------------
Optimally chosen parameters. Number of components =  12 , alpha =  0.1111111
Results based on criterion elbow :
      J     alpha     mu criterion
2241 12 0.1111111 0.1215 0.2326111
2244 12 0.1172840 0.1169 0.2341840
2242 12 0.1131687 0.1211 0.2342687
2243 12 0.1152263 0.1198 0.2350263
2001 11 0.1172840 0.1179 0.2351840
2240 12 0.1090535 0.1265 0.2355535
2245 12 0.1193416 0.1169 0.2362416
2002 11 0.1193416 0.1175 0.2368416
2494 13 0.1316872 0.1053 0.2369872
2004 11 0.1234568 0.1136 0.2370568
2246 12 0.1213992 0.1161 0.2374992
2003 11 0.1213992 0.1163 0.2376992
2005 11 0.1255144 0.1123 0.2378144
1999 11 0.1131687 0.1248 0.2379687
2239 12 0.1069959 0.1310 0.2379959

8004 rows are omitted.

Available components:
[1] "model"     "option"    "results"   "icp.torus" "Jhat"      "alphahat"
```

It can be checked that the choice of $J = 12$ and $\alpha = 0.1111$ in the previous examples was indeed given by the option "elbow".

In computing the criterion (12), the volume $\mu\left(C_n\left(\alpha, J\right)\right)$ is numerically approximated. This is feasible for data on $\mathbb{T}^2 = [0, 2\pi)^2$ by inspecting the inclusion of each point of a fine grid. However, for high dimensional cases, for example $\mathbb{T}^4$, evaluating the volume becomes computationally infeasible. In fact, as the dimension increases, the number of required inspections grows exponentially. Furthermore, the function $(\alpha, J) \to \alpha + \mu\left(C_n(\alpha, J)\right)$ is typically not a convex function and has multiple local minima. Thus, the choice of $\left(\hat{\alpha}, \hat{J}\right)$ by (12) tends to be unstable, resulting in high variability of the clustering results. Therefore, evaluating (12) is not practical for high-dimensional data.

To this end, we have developed and implemented a computationally more efficient procedure for hyperparameter selection, which also provides more stable clustering results. This procedure is a two-step procedure, first choosing the model parameter J, then choosing the level α. The two-step procedure is implemented for choosing J and α, but not for κ and α. Our approach is in contrast to the approaches in Lei et al. (2013) and Shin et al. (2019) in which they only choose the model parameter for a prespecified level α.

The first step of the procedure is to choose J, without making any reference to the level α. Choosing J can be regarded as selecting an appropriate mixture model. The model selection is based on either the (prediction) risk, Akaike information criterion (Akaike, 1974), or Bayesian information criterion (Schwarz, 1978). Since the mixture model-based conformity scores (6), (7) and (10) are actually the density or the approximated log-density of the mixture model, we use the conformity scores in place of the likelihood. For example, the sum of the conformity scores (10) over the given data is exactly the fitted log-likelihood. Specifically, let $\mathbb{X}_1, \mathbb{X}_2$ be the splitted datasets given by Algorithm 1 and $\mathbb{X} = \mathbb{X}_1 \cup \mathbb{X}_2$. Let $\sigma(\cdot) = \log g\left(\cdot; \mathbb{X}_1\right)$ if g is given by (6) and (7) or $\sigma(\cdot) = g\left(\cdot; \mathbb{X}_1\right)$ if g is given by (10). Recall that g is the conformity score, and it depends on the estimated model $\hat{p}$. Then, the function σ we defined above also depends on the model $\hat{p}$, and the criterion R can be defined as follows:

$$R\left(\mathbb{X}, \hat{p}\right) = \begin{cases} -2\sum_{x \in \mathbb{X}_2} \sigma(x) & \text{if the criterion is the risk,} \\ -2\sum_{x \in \mathbb{X}_1} \sigma(x) + 2k & \text{if the criterion is AIC,} \\ -2\sum_{x \in \mathbb{X}_1} \sigma(x) + k \log n_1 & \text{if the criterion is BIC,} \end{cases}$$

where k is the number of model parameters and n_1 is the cardinality of $\mathbb{X}_1$. The function `hyperparam.J` computes the minimizer $\hat{J}$ of the criterion, as summarized in Algorithm 2.

The fitted models $\hat{p}_{j_1}, \cdots, \hat{p}_{j_n}$ of Algorithm 2 are exactly the outputs of `icp.torus` for various

Algorithm 2 hyperparam.J

```
1: procedure HYPERPARAM.J($\mathbb{X} \subset \mathbb{T}^p$, fitted models $\hat{p}_{j_1}, \cdots, \hat{p}_{j_n}$, criterion $R$)
2:     Evaluate $R_j = R\left(\mathbb{X}, \hat{p}_j\right)$ for $j = j_1, \cdots, j_n$.
3:     Evaluate $\hat{J} = \arg\min_{j \in \{j_1, \cdots, j_n\}} R_j$.
4:     Output $\hat{J}, \hat{p}_{\hat{J}}$.
5: end procedure
```

$J = j_1, \cdots, j_n$. Which criterion to use is specified by setting the argument `option` of `hyperparam.J`. The argument `option = "risk"`, `"AIC"`, or `"BIC"` is for the risk, AIC, or BIC, respectively. By choosing $\hat{J}$, we also fix the model $\hat{p}_{\hat{J}}$ for the next step.

The second step is to choose the level $\alpha \in (0,1)$ for the chosen $\hat{J}$ and $\hat{p}_{\hat{J}}$, so that the clustering result is stable over perturbations of α. If the number of clusters does not change by varying the level $\alpha \in I$ for some interval I, we regard that the clustering result is stable on I. If I is sufficiently wide, it is reasonable to choose an $\alpha \in I$. Thus, our strategy is to find the most wide interval $I = [a,b] \subseteq (0,1)$ whose elements construct the same number of clusters, and to set $\hat{\alpha}$ as the midpoint of the interval, i.e. $\hat{\alpha} = (a+b)/2$. However, choosing α large, e.g. $\alpha > 0.5$, results in a too small coverage $1-\alpha$ of the prediction set. Thus, we restrict the searching area as $[0, M]$ for $M \in (0,1)$ which is close to 0, and find the desirable I in the restricted area $[0, M]$ rather than the whole interval $[0,1]$. This strategy is implemented in `hyperparam.alpha`, and the algorithm is described in Algorithm 3.

Algorithm 3 hyperparam.alpha

```
1: procedure HYPERPARAM.ALPHA(fitted model $\hat{p}$, $n_2 := |\mathbb{X}_2|$, $M \in [0,1]$)
2:     Evaluate the number of clusters $c_{\alpha_j}$ for $\alpha_j = j/n_2$, $j = 1, \cdots, \lfloor n_2 M \rfloor$.
3:     Set $A = \{j : c_{\alpha_{j-1}} \neq c_{\alpha_j}, \quad j = 2, \cdots, \lfloor n_2 M \rfloor\}$.
4:     For $A = \{\alpha_{j_1}, \cdots, \alpha_{j_N}\}$ find $i = \arg\max_{k \in \{1, \cdots, N-1\}} \alpha_{j_{k+1}} - \alpha_{j_k}$.
5:     Output $\hat{\alpha} = \left(\alpha_{j_{i+1}} + \alpha_{j_i}\right)/2$
6: end procedure
```

Note that we could alternatively input an array of levels, for the argument `alphavec` of `hyperparam.alpha`, if there is a prespecified searching area. In our experience, setting $M = 0.15$ gives generally satisfying results. By setting $M = 0.15$, at most 15% of the data points are not included in the prediction set, and at most 15% of the data can be regarded as the outliers. The default value for argument `alpha.lim` of `hyperparam.alpha`, which is M in Algorithm 3, is 0.15. We may interpret this level selecting procedure as finding the representative modes for the given mixture model; the chosen level is the cutoff value for which the most stable modes are not vanished.

In summary, we first choose the number of model components J in view of model selection, and then find the most stable level $\hat{\alpha}$ in the sense of invariability of the number of clusters. The function `hyperparam.torus` combines and implements Algorithms 2 and 3 sequentially and thus chooses J and α. This two-step hyperparameter selection procedure is used when mixture models are used to produce the conformal prediction sets, and can be invoked when the argument `option` of `hyperparam.torus` is set as `option = "risk"`, `"AIC"`, or `"BIC"`. If `option = "elbow"` (the default value, if the dimension of data is $p = 2$), then the "elbow" criterion (12) is used to choose either (J, α) or (κ, α). The function `hyperparam.torus` returns the chosen hyperparameters $(\hat{J}, \hat{\alpha})$ (or $(\hat{\kappa}, \hat{\alpha})$), as well as the corresponding model as an `icp.torus` object.

As an example, the following code applies the two-step procedure with `option = "risk"` to `icp.torus.objects` we evaluated earlier.

```
hyperparam.risk.out <- hyperparam.torus(icp.torus.objects, option = "risk")
hyperparam.risk.out

Type of conformity score: kmeans general
Optimizing method: risk
-------------
Optimally chosen parameters. Number of components =  12 , alpha =  0.132716
Results based on criterion risk :
    J criterion
1   3  2016.575
2   4  1990.566
```

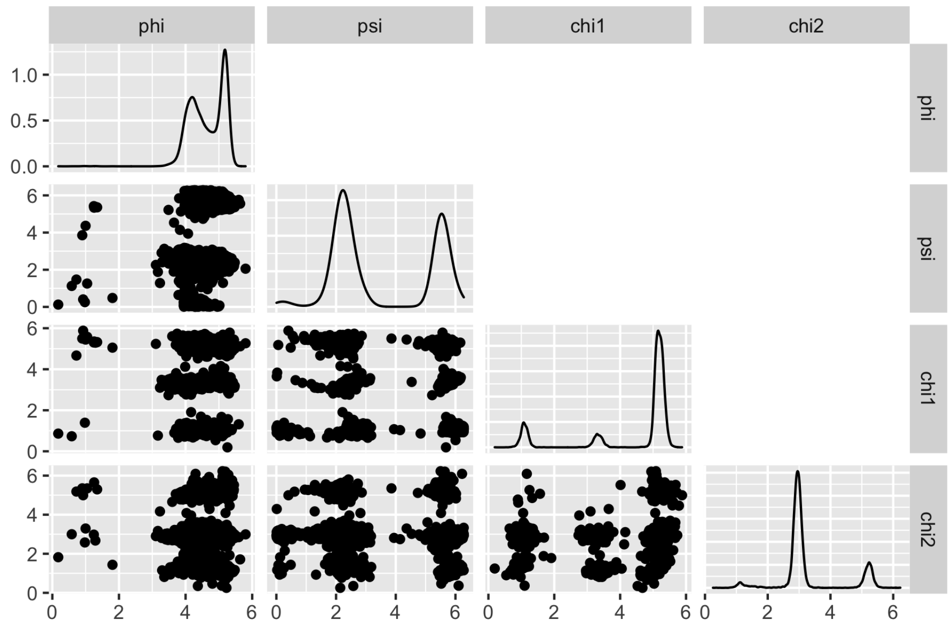

Figure 5: The pairwise scatter plots of `ILE` data, in which there are four variables (angles) ϕ, ψ, χ_1 and χ_2. Each diagonal entry of the plot shows a marginal kernel density estimate for the corresponding angle. Each off-diagonal panel is a scatter plot for a pair of variables.

```
3   5  1907.887
4   6  1922.430
5   7  1924.768
... (omitted)
```

With the option "risk," $(\hat{J}, \hat{\alpha}) = (12, 0.0.1327)$. Recall that with option "elbow", we have chosen $(\hat{J}, \hat{\alpha}) = (12, 0.1111)$. The hyperparameter selection procedures can be visualized by `plot(hyperparam.out)` and `plot(hyperparam.risk.out)`. (The resulting graphic is omitted.)

In the next section, the two-step procedures for hyperparameter selection are used in a cluster analysis of data on $\mathbb{T}^4$.

5 Clustering data on $\mathbb{T}^4$

In this section, we give an example of clustering `ILE` data in $\mathbb{T}^4$. `ILE` is a dataset included in **ClusTorus**, which represents the structure of the isoleucine. This dataset is obtained by collecting several different '.pdb' files in the Protein Data Bank (Berman et al., 2003). We used PISCES (Wang and Dunbrack, 2003) to select high-quality protein data, by using several benchmarks—resolution is 1.6Å or better, R-factor is 0.22 or better, sequence percentage identity is equal to or less than 25—as described in Harder et al. (2010) and Mardia et al. (2012). The `ILE` data consist of $n = 8080$ instances of four angles $(\phi, \psi, \chi_1, \chi_2) \in \mathbb{T}^4$, and is displayed in Figure 5.

For predictive clustering of `ILE` data, the conformal prediction sets and scores are built from mixture models, fitted with the elliptical k-means algorithm (i.e., `model = "kmeans"`). Other choices of models such as `"kde"` and `"mixture"` are not applicable for this data set with $p > 2$. The number J of components in the mixture model needs to be tuned, and we set the candidates for J as $\{10, \ldots, 40\}$. In the code example below, conformal prediction sets from mixture models are constructed by the function `icp.torus`, with `J = 10:40` indicating the candidates of J.

```
set.seed(2021)
icp.torus.objects <- icp.torus(ILE, J = 10:40)
```

Next step is to select the hyperparameter J, and the level α of the prediction set, using the function `hyperparam.torus`. As discussed in the previous section, for this data set with $p = 4$, evaluating the "elbow" criterion is computationally infeasible, and is not supported in `hyperparam.torus`, if $p > 2$. For $p > 2$, `hyperparam.torus` uses the two-step procedure, discussed in the previous section, with `option = "risk"` as the default choice for the criterion. In the code example below, we use the two-step procedure, but apply all three available criteria (`option = "risk"`, `"AIC"`, and `"BIC"`) in choosing $\hat{J}$.

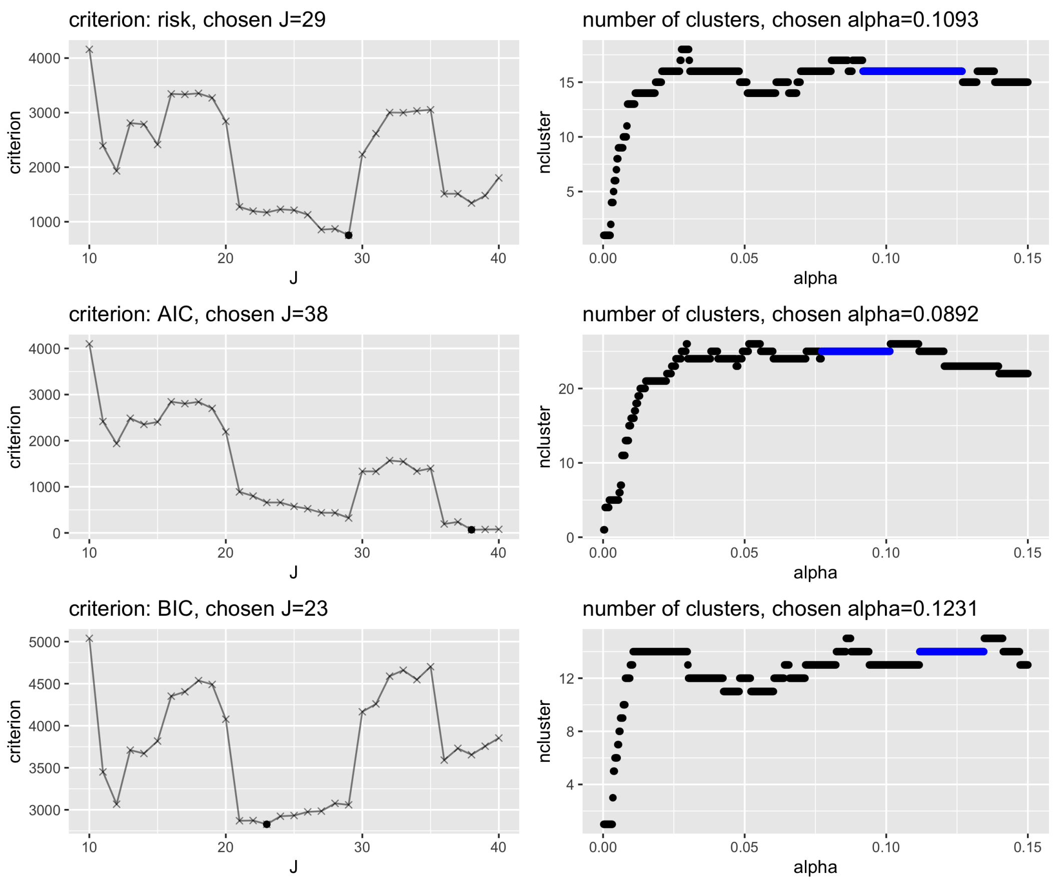

Figure 6: Hyperparameter selection for `ILE` data, generated from the outputs of `hyperparam.torus`. Rows correspond to different choices of criteria "risk", "AIC" and "BIC". In each row, the left panel shows the values of criterion over J, with the optimal $\hat{J}$ indicated by a thicker dot; the right panel shows the number of clusters over varying α, in which the longest streak is highlighted. The optimal $\hat{\alpha}$ is the midpoint of the longest streak.

```
output_list <- sapply( c("risk", "AIC", "BIC"), function(opt) {
     hyperparam.torus(icp.torus.objects, option = opt)},
     simplify = FALSE,
     USE.NAMES = TRUE)
```

The result `output_list` is a list of length 3, consisting of outputs of the function `hyperparam.torus`. The details of hyperparameter selection can be visualized, and are shown in Figure 6. The first row of the figure is created by `plot(output_list$risk)`, and shows that the evaluated prediction risk is the smallest at $\hat{J} = 29$. On the right panel, it can be seen that the longest streak of the number of clusters over varying level α occurs at 16, which is given by a range of levels around $\hat{\alpha} = 0.1093$. The second and third rows are similarly generated, and they show the results of AIC- and BIC-based hyperparameter selection. While the results of hyperparameter selection from the three criteria do not always agree with each other, we observe that using BIC tends to choose parsimonious models than others, for this and many other data sets we tested.

The number of clusters, given by the conformal prediction set $C_n(\hat{\alpha}, \hat{J})$, can be seen in the right panels of Figure 6. For example, in the top right panel, with $\hat{J} = 29$ and $\hat{\alpha} = 0.1093$, the number of clusters is 16 (the vertical position of the blue-colored longest streak). For the subsequent analysis, we use the risk criterion, thus choosing $(\hat{J}, \hat{\alpha}) = (29, 0.1093)$.

```
hyperparam.risk.out <- output_list$risk
```

Finally, the function `cluster.assign.torus` is used for cluster membership assignment for each data point in `ILE`. In the code below, the function `cluster.assign.torus` takes as input `hyperparam.risk.out`, an output of `hyperparam.torus`, and we have not specified any level. Since the object `hyperparam.risk.out` contains the chosen level $\hat{\alpha}$ (in its value `alphahat`), the level of the conformal prediction set is, by default, set as `hyperparam.risk.out$alphahat`.

```
cluster.out <- cluster.assign.torus(hyperparam.risk.out)
```

The output `cluster.out` contains the membership assignment results as well as the number of clusters, which can be retrieved by `cluster.out$ncluster` or by simply printing the output `cluster.out`. The assigned cluster memberships can be displayed on the pairwise scatter plots of the four angles. We demonstrate the outlier-disposing membership assignment (the default behavior for S3 method `plot`), as well as the membership assignment based on the maximum of log-densities. Figure 7 displays the scatter plots generated by the codes:

```
plot(cluster.out, assignment = "outlier")     # Top panel of Figure 7
plot(cluster.out, assignment = "log.density") # Bottom panel of Figure 7
```

Note that these cluster assignments are based on the conformal prediction set $C_n(\hat{\alpha}, \hat{J})$. The information to construct $C_n(\alpha, \hat{J})$ (for any $\alpha \in (0,1)$) is contained in the object `hyperparam.risk.out` as value `icp.torus`. Since the conformal prediction set is a union of 4-dimensional toroidal ellipsoids, projections of such ellipsoids onto coordinate planes are plotted by the following code, and is shown in Figure 8.

```
set.seed(2021)
plot(hyperparam.risk.out$icp.torus,
     data = ILE[sample(1:nrow(ILE),500),],
     level = hyperparam.risk.out$alphahat)
```

Scatter plots of $n = 8080$ observations are typically too busy, especially when other information (such as the ellipses) is overlaid. In the code example above, we use the argument `data = ILE[sample(1:nrow(ILE),500),]` to plot randomly selected observations.

6 All-in-one function: `clus.torus`

The predictive clustering for data on the torus is obtained by sequentially applying functions `icp.torus`, `hyperparam.torus` and `cluster.assign.torus`, as demonstrated for `ILE` data in the previous section. The function `clus.torus` is a user-friendly all-in-one function, which performs the predictive clustering by sequentially calling the three core functions.

Using `clus.torus` can be as simple as `clus.torus(data)`, as shown in the first code example, resulting in Figure 1, in Introduction. In this case, the three functions are called sequentially with default choices for their arguments. On the other hand, users can specify which models and fitting methods are used, whether hyperparameter tuning is required, and, if so, which criterion is used for `hyperparam.torus`, and so on. Key arguments of `clus.torus` are summarized in Table 3. The argument `model` only takes `"kmeans"` and `"mixture"` as input, which is passed to `icp.torus` inside the function. Since the function concerns clustering, conformal prediction sets consisting of ellipsoids (10) are needed, and such prediction sets are given by both `model = "kmeans"` and `"mixture"`. Next, the values of the arguments `J` and `level` determine whether tuning is needed for hyperparameters J and α. If both are not specified, i.e., `J = NULL` and `level = NULL`, then `hyperparam.torus` is used to select both parameters, with argument `option` (see Table 3). If either `J` or `level` is specified as a scalar, then the function simply uses the given value for constructing the conformal prediction sets and for clustering. Other arguments available for `icp.torus` and `hyperparam.torus` can be specified, and the function passes those arguments to corresponding functions, if applicable.

The output of the function is a list of three objects, with S3 class `clus.torus`. The three objects in the output are

1. a `cluster.obj` object, containing the results of cluster membership assignments,
2. an `icp.torus` object, corresponding to the model with $\hat{J}$ (or the specified `J`), and
3. if applicable, a `hyperparam.torus`, `hyperparam.J` or `hyperparam.alpha` object.

Each of these objects can be plotted via `plot`, defined for S3 class `clus.torus`. For example, recall that `ex` is a `clus.torus` object we created in Introduction. By setting the argument `panel` of the method `plot` as `panel = 1`, the `cluster.obj` object is plotted.

```
plot(ex, panel = 1) # equivalent to plot(ex)
```

The result is shown in Figure 1 (top left). If the data dimension is $p > 2$, then figures similar to Figure 7 will be created. If `panel = 2`, the `icp.torus` object is plotted, similar to Figures 3 and 8. Finally, if `panel = 3`, the graphics relevant to hyperparameter selection are created, similar to Figure 6.

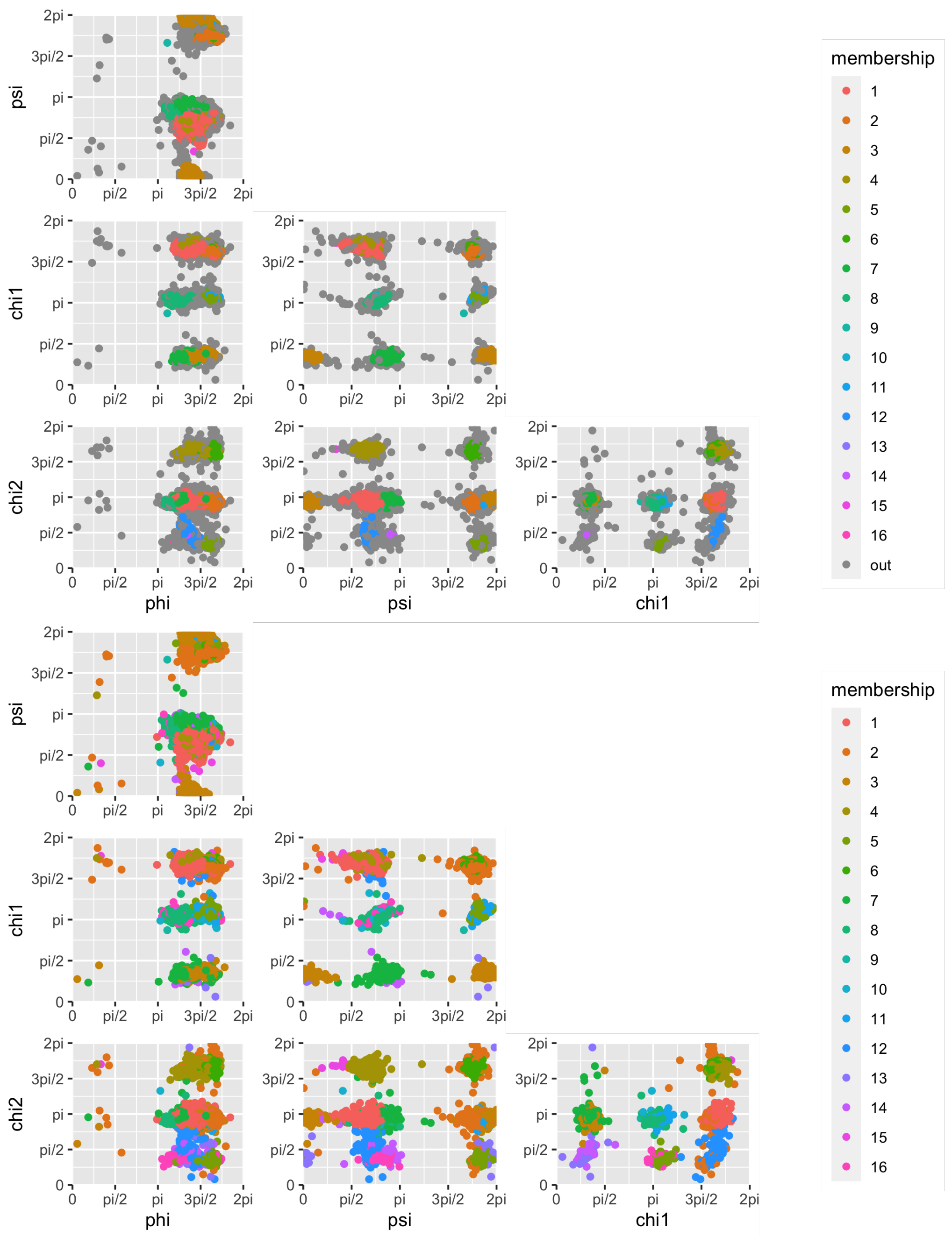

Figure 7: The pairwise scatter plots of `ILE` data with cluster assignments. (Top) `assignment = "outlier"`. (Bottom) `assignment = "log.density"`.

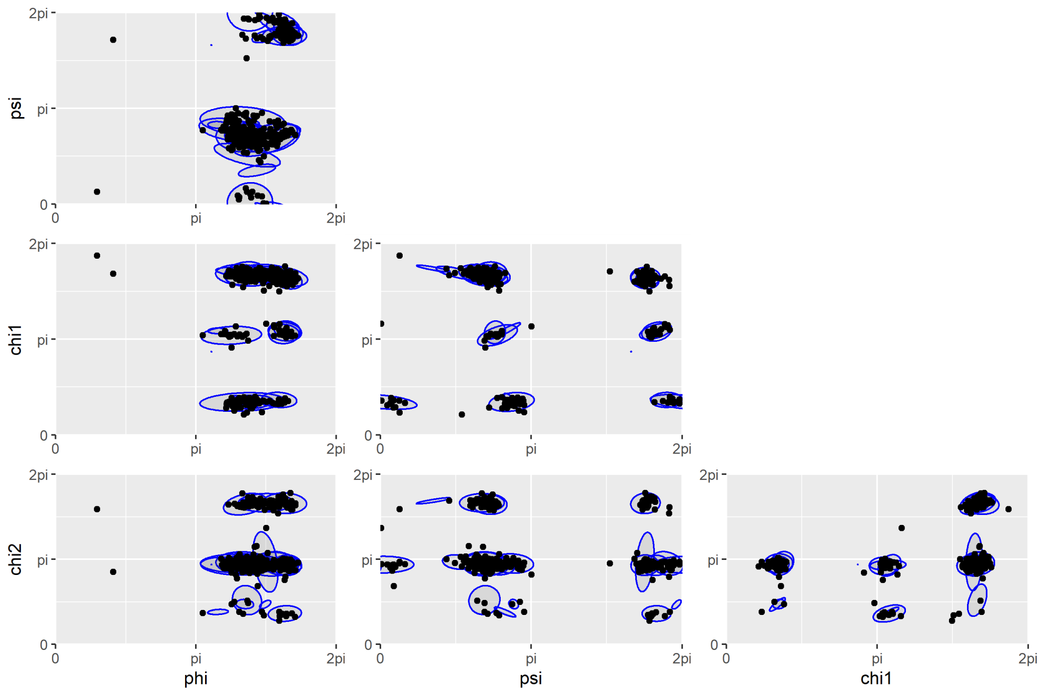

Figure 8: The pairwise scatter plots of `ILE` data, overlaid with the (projected) ellipsoids that constitute the conformal prediction set $C_n(\hat{\alpha}, \hat{J})$.

Arguments	Descriptions
`data`	$n \times d$ matrix of toroidal data on $[0, 2\pi)^d$ or $[-\pi, \pi)^d$
`model`	A string. One of `"kmeans"` and `"mixture"` which determines the model or estimation methods. If `"mixture"`, the model is the von Mises mixture, fitted with an EM algorithm. If `"kmeans"`, the model is also the von Mises mixture, fitted by the elliptical k-means algorithm. If the dimension of data space is greater than 2, only `"kmeans"` is supported. Default is `model = "kmeans"`.
`J`	A scalar or numeric vector. If `J` is scalar, the number of components J is set as `J`. If `J` is a vector, then `hyperparam.torus` or `hyperparam.J` is used to select $\hat{J}$. Default is `J = NULL`, in which case `J = 4:30` is used.
`level`	A scalar in $[0, 1]$. The level of the conformal prediction set used for clustering. Default is `level = NULL`, in which case `hyperparam.alpha` is used to choose optimal `level` $\hat{\alpha}$.
`option`	A string. One of `"elbow"`, `"risk"`, `"AIC"`, or `"BIC"`, determining the criterion used for `hyperparam.torus` andr `hyperparam.J`. Default is `option = "elbow"` if $d = 2$, and `option = "risk"` if $d > 2$.

Table 3: Key arguments and descriptions of the function `clus.torus`

7 Other methods of clustering on the torus

Gao et al. (2018) and Jung et al. (2021) used the *extrinsic k-means*, which uses Euclidean embedding and enjoys fast computation of the vanilla k-means algorithm. That is, consider the mapping $f : \mathbb{T}^p \to \mathbb{R}^{2p}$ as

$$f(\phi_1, \cdots, \phi_p) = (\cos\phi_1, \cdots, \cos\phi_p, \sin\phi_1, \cdots, \sin\phi_p)$$

which is the simple Euclidean embedding and is injective. Since $\mathbb{R}^{2p}$ is a Euclidean space, the k-means clustering for vector-valued data can be used. The function `kmeans.torus` implements the extrinsic k-means clustering. In the simple code example below, the number of cluster is set to $k = 3$, and the result shows the membership assignment by the extrinsic k-means algorithm.

```
set.seed(2021)
exkmeans <- kmeans.torus(SARS_CoV_2, centers = 3, nstart = 30)
head(exkmeans$membership)
```

```
  27.B.ALA   28.B.TYR   29.B.THR   30.B.ASN   31.B.SER   32.B.PHE
```

```
     1          1          1          1          2          3
```

Distance-based clustering methods, such as hierarchical clustering, only requires a pairwise distances of the data points. The function `ang.pdist` generates the distance matrix for the input data in which the angular distance between the two points on $\mathbb{T}^p$ is measured. Combined with `hclust`, the pairwise angular distances are used to provide a hierarchical clustering using, e.g., the complete linkage, as done in the following example.

```
distmat <- ang.pdist(SARS_CoV_2)
hc <- hclust(distmat, method = "complete")
hc.result <- cutree(hc, k = 3)
head(hc.result)
```

```
[1] 1 1 1 1 2 3
```

Figure 9 shows the results for the two clustering algorithms, discussed above. The left panel shows that the Euclidean embedding reflects the rotational nature of angular data. The right panel shows that the distance-based clustering methods is well-applied with `ang.pdist`. Note that both the extrinsic k-means and the hierarchical clustering results are invariant to different representations of angles. That is, the cluster assignments do not change if $\mathbf{X}_n$ is replaced by $\mathbf{X}_n - \pi$. However, these methods are inadequate when true clusters are irregularly shaped and when there are outliers (Jung et al., 2021). In addition, the number of clusters needs to be predetermined for both methods. In contrast, these weaknesses are mostly resolved by using the predictive clustering.

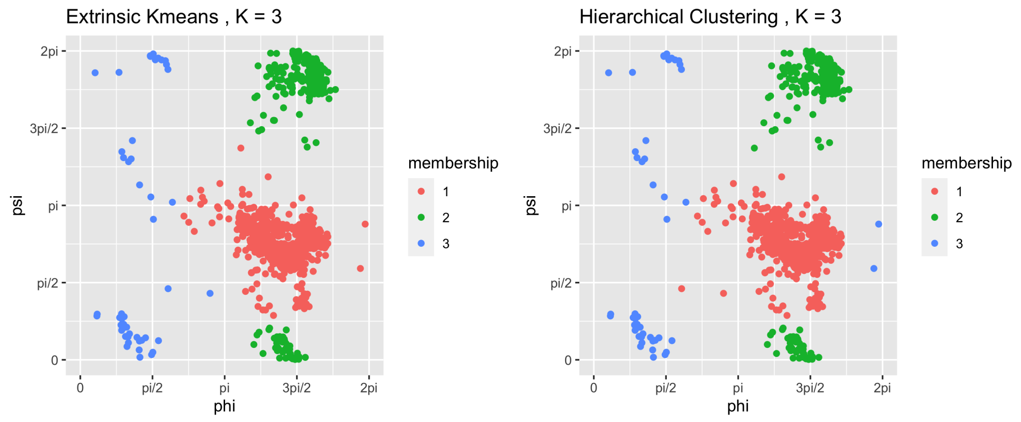

Figure 9: The clustering results for SARS-CoV-2 by using extrinsic k-means and hierarchical clustering under the 3 clusters assumption. The left panel shows the result for extrinsic k-means, and the right panel shows the result for hierarchical clustering.

8 Summary and discussion

In this paper, we introduced the package **ClusTorus** which contains various tools and routines for multivariate angular data, including kernel density estimates and mixture model estimates. **ClusTorus** performs clustering based on conformal prediction sets. We demonstrated our implementation with data on $\mathbb{T}^4$. The clustering by **ClusTorus** can result in cluster assignment either with or without an outlier class. A reviewer pointed out that the package **MoEClust** (Murphy and Murphy, 2020, 2021) can also dispose some points as outliers. However, **MoEClust** only works on Euclidean space, not on $\mathbb{T}^p$.

There are some possible future developments for **ClusTorus**. First, EM algorithms for von Mises mixture models on high dimensional tori (e.g., $\mathbb{T}^4$) can be implemented assuming independence of angles in each component. Using closed-form approximations of maximum likelihood estimators for univariate von Mises-Fisher distributions (Banerjee et al., 2005; Hornik and Bettina, 2014), fitting mixtures of product components can be done efficiently (Grim, 2017). Another direction is obtained by viewing clustering based on (11) by varying α as surveying birth and death of connected components. This can be dealt with a persistence diagram, a concept of topological data analysis. Hence, instead of using Algorithm 3, one may choose desirable α using persistence diagram.

9 Appendix

Elliptical *k*-means algorithm

In this appendix, we outline the elliptical *k*-means algorithm for the data on the torus, implemented in the function `ellip.kmeans.torus`. The algorithm is used to estimate the parameters of the mixture model (5), approximated as in (8). Note that the EM algorithm can be used for parameter estimation for mixture models in low dimensions. The EM algorithms of Jung et al. (2021) is implemented in the function `EMsinvMmix`, but works for $p = 2$ only. For $p > 3$, EM algorithms suffer from high computational costs (Mardia et al., 2012). To circumvent this problem, we estimate the parameters by modifying the generalized Lloyd's algorithm (Shin et al., 2019), also known as the elliptical *k*-means algorithm (Sung and Poggio, 1998; Bishop, 2006). For vector-valued data, Shin et al. (2019) showed that the elliptical *k*-means algorithm estimates the parameters sufficiently well for the max-mixture density case as (7).

Suppose $y_1, \cdots, y_n \in \mathbb{T}^p$ are an independent and identically distributed sample. Using the approximated density (8), the approximated likelihood, L', is

$$L'(\mu, \Sigma) = (2\pi)^{-np/2} |\Sigma|^{-n/2} \exp\left[-\frac{n}{2} tr\left(S\Sigma^{-1}\right)\right] \tag{13}$$

where $S = \frac{1}{n}\sum_{i=1}^n (y_i \ominus \mu)(y_i \ominus \mu)^T$. Thus, if μ is known, $\hat{\Sigma} = S$ maximizes L'. Following Mardia et al. (2012), the mean μ is estimated as follows. Let $\bar{U}_j = \sum_{i=1}^n \cos\left(y_{ij}\right)/n$ and $\bar{V}_j = \sum_{i=1}^n \sin\left(y_{ij}\right)/n$ for $j = 1, \cdots, p$. Then, $\hat{\mu} = \left(\hat{\mu}_1, \cdots, \hat{\mu}_p\right)^T$,

$$\hat{\mu}_j = \arctan\frac{\bar{V}_j}{\bar{U}_j}, \quad j = 1, \cdots, p \tag{14}$$

which is the maximum likelihood estimator of mean direction of von Mises-Fisher distribution (Mardia and Jupp, 1999).

With these approximated maximum likelihood estimators, the elliptical *k*-means algorithm, described in Algorithm 4, maximizes the likelihood corresponding to the max-mixture model (7). The algorithm is implemented in the function `ellip.kmeans.torus`.

Algorithm 4 Elliptical *k*-means algorithm for the torus

```
1: procedure ELLIPTICAL K-MEANS($\{X_1, \cdots, X_n\}, J$)
2:     Initialize $\pi_j, \theta_j = \left(\mu_j, \Sigma_j\right), j = 1, \cdots, J$
3:     set
           $w_{i,j} = \begin{cases} 1, & \text{if } j = \arg\max_l \left[-\left(X_i \ominus \mu_l\right)^T \Sigma_l^{-1}\left(X_i \ominus \mu_l\right) - \log|\Sigma_l| + 2\log\pi_l\right] \\ 0, & \text{otherwise}\end{cases}$
           $I_j = \left\{i \in \{1, \cdots, n\} \,|\, w_{i,j} = 1\right\}$
4:     Update $\mu_j$ as (14) with $\{X_i\}_{i \in I_j}$ for $j = 1, \cdots, J$
5:     Update $\Sigma_j = \frac{1}{\sum_{i=1}^n w_{i,j}} \sum_{i=1}^n w_{i,j}\left(X_i \ominus \mu_j\right)\left(X_i \ominus \mu_j\right)^T$ for $j = 1, \cdots, J$
6:     Update $\pi_j = \frac{1}{n}\sum_{i=1}^n w_{i,j}$ for $j = 1, \cdots, J$
7:     Repeat step 3-6 until converge
8: end procedure
```

Note that the initial values require an initial clustering. For this, we use other clustering algorithms such as the extrinsic *k*-means or the hierarchical clustering algorithms, and can be specified by argument `init` of `ellip.kmeans.torus` and `icp.torus`. One may specify arguments for either `hlcust` or `kmeans` in `icp.torus`. For example, one may specify the choice of initial values as follows.

```
icp.torus(data = SARS_CoV_2, J = 4, init = "kmeans", nstart = 30)
icp.torus(data = SARS_CoV_2, J = 4, init = "hierarchical", method = "complete")
```

By default, the hierarchical clustering with complete linkage is used. Data analysis in this article using `icp.torus` or `clus.torus` was performed with the default initialization.

Constraints for mixture models

The protein structure data we aim to analyze typically consist of hundreds of angles (observations). Fitting the mixture with a large number of components may give inefficient estimators. Thus, we have implemented options for reducing the number of model parameters, by constraining the shape of the ellipsoids, or the covariance matrices. Applying the constraints lead much faster convergence for estimating parameters (Grim, 2017). We list three types of constraints for covariance matrices Σ_j. These constraints are specified by setting the arguments `mixturefitmethod` and `kmeansfitmethod` (for `icp.torus`) and `type` (for `EMsinvMmix` and `ellip.kmeans.torus`). We explain in terms of the arguments for the function `icp.torus`.

- $\Sigma_j = \sigma_j^2 I_p$ for some $\sigma_j^2 > 0$ for all j, and the prediction set will be the union of spheres. `mixturefitmethod = "circular"` and `kmeansfitmethod = "heterogeneous-circular"` represents this constraint. Furthermore, if $\sigma_1^2 = \cdots = \sigma_J^2$ and $\pi_j = 1/J$ for all j, then all the spheres have the same radii and this constraint can be designated with `kmeansfitmethod = "homogeneous-circular"`.
- $\Sigma_j = diag\left(\sigma_{jk}^2\right)_{k=1,\cdots,p}$ for $\sigma_{jk}^2 > 0$, and the fitted ellipsoids $\hat{E}_j$ $(j = 1, \cdots, J)$ are the axis-aligned ellipsoids. `mixturefitmethod = "axis-aligned"` represents this constraint.
- No constraint for Σ_j, and $\hat{E}_j$ $(j = 1, \cdots, J)$ are any ellipsoids. This option can be designated by `mixturefitmethod = "general"` and `kmeansfitmethod = "general"`.

The default values for `icp.torus` are `kmeansfitmethod = "general"` and `mixturefitmethod = "axis-aligned"`.

List of S3 classes defined in ClusTorus

Several S3 classes are defined in the packages **ClusTorus**. A list of the S3 classes is given in Table 4.

S3 class	functions	methods
`cp.torus.kde`	`cp.torus.kde`	`print`, `plot`
`icp.torus`	`icp.torus`	`print`, `plot`, `LogLik`, `predict`
`icp.torus.eval`	`icp.torus.eval`	`print`
`cluster.obj`	`cluster.assign.torus`	`print`, `plot`
`kmeans.torus`	`kmeans.torus`	`print`, `predict`
`hyperparam.torus`	`hyperparam.torus`	`print`, `plot`
`hyperparam.J`	`hyperparam.J`	`print`, `plot`
`hyperparam.alpha`	`hyperparam.alpha`	`print`, `plot`
`clus.torus`	`clus.torus`	`print`, `plot`

Table 4: List of S3 classes, the functions returning a list with corresponding S3 class, and available methods for the S3 class.

10 Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2C2002256).

Bibliography

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. URL `https://doi.org/10.1109/TAC.1974.1100705`. [p197]

A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, 6(46):1345–1382, 2005. URL `http://jmlr.org/papers/v6/banerjee05a.html`. [p204]

T. Benaglia, D. Chauveau, D. R. Hunter, and D. Young. mixtools: An R package for analyzing finite mixture models. *Journal of Statistical Software*, 32(6):1–29, 2009. URL `http://www.jstatsoft.org/v32/i06/`. [p188]

H. Berman, K. Henrick, and H. Nakamura. Announcing the worldwide protein data bank. *Nature Structural and Molecular Biology*, 10:980, 2003. URL https://doi.org/10.1038/nsb1203-980. [p187, 199]

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science+Business Media, Inc., 33 Spring Street, New York, NY 10013, USA, 2006. [p187, 194, 205]

C. Branden and J. Tooze. *Introduction to Protein Structure*. Garland Publishing, Inc, 19 Union Square West, New York, 1999. [p187]

S. Chakraborty and S. W. Wong. *BAMBI: Bivariate Angular Mixture Models*, 2020. URL https://CRAN.R-project.org/package=BAMBI. R package version 2.3.0. [p187]

S. Chakraborty and S. W. K. Wong. BAMBI: An R package for Fitting Bivariate Angular Mixture Models, 2019. URL https://arxiv.org/abs/1708.07804. [p187]

Coronaviridae Study Group of the International Committee on Taxonomy of Viruses., A. e. Gorbalenya, S. C. baker, R. S. baric, R. J. de Groot, C. Drosten, A. A. Gulyaeva, bart l. Haagmans, C. lauber, A. M. leontovich, benjamin W. Neuman, D. Penzar, S. Perlman, leo l. M. Poon11, D. V. Samborskiy, I. A. Sidorov, I. Sola, and J. Ziebuhr. The species severe acute respiratory syndrome- related coronavirus: classifying 2019-ncov and naming it sars-cov-2. *Nature Microbiology*, 5:536—-544, 2020. URL https://doi.org/10.1038/s41564-020-0695-z. [p187]

Y. Gao, S. Wang, and M. Deng. Raptorx-angle: real-value prediction of protein backbone dihedral angles through a hybrid method of clustering and deep learning. *BMC Biometrics*, 19(100), 2018. URL https://doi.org/10.1186/s12859-018-2065-x. [p188, 203]

I. Gilitschenski and U. D. Hanebeck. A robust computational test for overlap of two arbitrary-dimensional ellipsoids in fault-detection of kalman filters. In *2012 15th International Conference on Information Fusion*, pages 396–401, 2012. URL https://ieeexplore.ieee.org/document/6289830. [p195]

B. Grant, X.-Q. Yao, L. Skjaerven, and J. Ide. *bio3d: Biological Structure Analysis*, 2021. URL https://CRAN.R-project.org/package=bio3d. R package version 2.4-2. [p187]

B. J. Grant, A. P. C. Rodrigues, K. M. ElSawy, J. A. McCammon, and L. S. D. Caves. Bio3d: an R package for the comparative analysis of protein structures. *Bioinformatics*, 22(21):2695–2696, 08 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl461. URL https://doi.org/10.1093/bioinformatics/btl461. [p187]

J. Grim. 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, 2017. [p204, 206]

T. Harder, W. Boomsma, M. Paluszewski, J. Frellsen, K. E. Johansson, and T. Hamelryck. Beyond rotamers: a generative, probabilistic model of side chains in proteins. *BMC Bioinformatics*, 11:306, 2010. URL https://doi.org/10.1186/1471-2105-11-306. [p199]

K. Hornik and G. Bettina. On maximum likelihood estimation of the concentration parameter of von mises–fisher distributions. *Computational Statistics*, 29:945—-957, 2014. URL https://doi.org/10.1007/s00180-013-0471-0. [p204]

S. Jung and S. Hong. *ClusTorus: Prediction and Clustering on the Torus by Conformal Prediction*, 2021. URL https://CRAN.R-project.org/package=ClusTorus. R package version 0.2.1. [p187]

S. Jung, K. Park, and B. Kim. Clustering on the torus by conformal prediction. *Annals of Applied Statistics*, 15(4):1583–1603, 2021. URL https://doi.org/10.1214/21-AOAS1459. [p188, 189, 191, 192, 194, 195, 196, 203, 204, 205]

J. Lei, J. Robins, and L. Wasserman. Distribution-free prediction sets. *Journal of the American Statistical Association*, 108(501):278–287, 2013. doi: 10.1080/01621459.2012.751873. URL https://doi.org/10.1080/01621459.2012.751873. [p189, 196, 197]

J. Lei, A. Rinaldo, and L. Wasserman. A conformal prediction approach to explore functional data. *Ann Math Artif Intell*, 74:29–43, 2015. URL https://doi.org/10.1007/s10472-013-9366-63. [p189, 190]

S. C. Lovell, I. W. Davis, W. B. Arendall III, P. I. De Bakker, J. M. Word, M. G. Prisant, J. S. Richardson, and D. C. Richardson. Structure validation by cα geometry: ϕ, ψ and cβ deviation. *Proteins: Structure, Function, and Bioinformatics*, 50(3):437–450, 2003. [p187]

K. V. Mardia and P. E. Jupp. *Directional Statistics*. Wiley, New York, 1999. [p205]

K. V. Mardia, C. C. Taylor, and G. K. Subramaniam. Protein bioinformatics and mixtures of bivariate von mises distributions for angular data. *Biometrics*, 63(2):505—-512, 2007. URL https://doi.org/10.1111/j.1541-0420.2006.00682.x. [p187]

K. V. Mardia, G. Hughes, C. C. Taylor, and H. Singh. A multivariate von mises distribution with applications to bioinformatics. *The Canadian Journal of Statistics / La Revue Canadienne de Statistique*, 36(1):99–109, 2008. URL http://www.jstor.org/stable/20445295. [p191]

K. V. Mardia, J. T. Kent, Z. Zhang, and C. C. T. . T. Hamelryck. Mixtures of concentrated multivariate sine distributions with applications to bioinformatics. *Journal of Applied Statistics*, 39(11):2475–2492, 2012. URL https://doi.org/10.1080/02664763.2012.719221. [p187, 188, 192, 199, 205]

M. D. Marzio, A. Panzera, and C. C. Taylor. Kernel density estimation on the torus. *Journal of Statistical Planning and Inference*, 141(6):2156–2173, 2011. ISSN 0378-3758. doi: https://doi.org/10.1016/j.jspi.2011.01.002. URL https://www.sciencedirect.com/science/article/pii/S037837581100019X. [p191]

K. Murphy and T. B. Murphy. Gaussian parsimonious clustering models with covariates and a noise component. *Advances in Data Analysis and Classification*, 14(2):293–325, 2020. URL https://doi.org/10.1007/s11634-019-00373-8. [p204]

K. Murphy and T. B. Murphy. *MoEClust: Gaussian Parsimonious Clustering Models with Covariates and a Noise Component*, 2021. URL https://cran.r-project.org/package=MoEClust. R package version 1.4.2. [p204]

K. Oberholser. Proteopedia entry: Ramachandran plots. *Biochemistry and Molecular Biology Education*, 38(6):430–430, 2010. [p187]

G. Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461 – 464, 1978. URL https://doi.org/10.1214/aos/1176344136. [p197]

L. Scrucca, M. Fop, T. B. Murphy, and A. E. Raftery. mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1):289–317, 2016. URL https://doi.org/10.32614/RJ-2016-021. [p188]

J. Shin, A. Rinaldo, and L. Wasserman. Predictive clustering, 2019. URL https://arxiv.org/abs/1903.08125. [p187, 188, 189, 192, 194, 197, 205]

K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998. URL https://doi.org/10.1109/34.655648. [p187, 194, 205]

V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic Learning in a Random World*. Springer Science+Business Media, Inc., 33 Spring Street, New York, NY 10013, USA, 2005. [p187, 188, 189, 190]

A. C. Walls, Y.-J. Park, M. A. Tortorici, A. Wall, A. T. McGuire, and D. Veesler. Structure, function, and antigenicity of the sars- cov-2 spike glycoprotein. *Cell*, 181, 2020. URL https://doi.org/10.1016/j.cell.2020.02.058. [p188]

G. Wang and J. Dunbrack, Roland L. PISCES: a protein sequence culling server. *Bioinformatics*, 19(12):1589–1591, 08 2003. ISSN 1367-4803. URL https://doi.org/10.1093/bioinformatics/btg224. [p199]

Seungki Hong
Department of Statistics, Seoul National University
1, Gwanak-ro, Gwanak-gu, Seoul
Republic of Korea
skgaboja@snu.ac.kr

Sungkyu Jung
Department of Statistics, Seoul National University
1, Gwanak-ro, Gwanak-gu, Seoul
Republic of Korea
sungkyu@snu.ac.kr