Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

CircSpaceTime: an R package for spatial and spatio-temporal modeling of Circular data

Giovanna Jona Lasinio
Sapienza University of Rome

Mario Santoro
IAC CNR

Gianluca Mastrantonio
Polytechnic of Turin, ITALY

Abstract

CircSpaceTime is an R package that implements Bayesian models, recently developed, for spatial and spatio-temporal interpolation of circular data. Such data are often found in applications where, among the many, wind directions, animal movement directions, and wave directions are involved. To analyze such data we need models for observations at locations $\mathbf{s}$ and times t, so-called geostatistical models, providing structured dependence which is assumed to decay in distance and time. For example, for wave directions in a body of water, we conceptualize a wave direction at every location and every time and introduce structured dependence into these angular data. The approach we take begins with Gaussian processes defined for linear variables over space and time. Then, we use either wrapping or projection to obtain processes for circular data. The models are cast as hierarchical, with fitting and inference within a Bayesian framework. Altogether, this package implements work developed by a series of papers; the most relevant being Jona Lasinio, Gelfand, and Jona Lasinio (2012); Wang and Gelfand (2014); Mastrantonio, Jona Lasinio, and Gelfand (2016b). All procedures are written using Rcpp. Estimates are obtained by MCMC allowing parallelized multiple chains runs. As running example, for the spatial setting, we use a wave direction dataset while simulated data are used to illustrate the spatio-temporal models.

Keywords: Directional data, spatial model, spatio-temporal model, Rcpp.

1. Introduction

In the last ten years the interest in circular data has increased, with new theoretical results and models (for an extended review of both theory and applications see Ley and Verdebout 2017, 2019). There exist several R packages dealing with circular data. The best known are **circular** (Agostinelli and Lund 2017) and **CircStats** (Lund and Agostinelli 2018), both implementing inference for univariate data as described in Jammalamadaka and SenGupta (2001).

Another recent set of functions specifically devoted to wrapped distributions is **Wrapped** (Nadarajah and Zhang 2017). The package computes the probability density function, cumulative distribution function, quantile function and many more features for several (about fifty) univariate wrapped distributions. A very interesting set of functions is implemented in **CircSizer** (Oliveira, Crujeiras, and Rodríguez-Casal 2014) where a non-parametric approach is adopted. Based on scale-space ideas, **CircSiZer** presents a graphical device to assess which observed features are statistically significant, both for density and regression analysis. Also a book on circular data in R has been published (Pewsey, Neuh auser, and Ruxton 2013) with many nice examples and a narrative of the topic that makes easy to learn how to run inferences on univariate data. In 2013 the first version of the package **isocirc** was presented (Barragán, Fernández, Rueda, and Das Peddada 2013), making available functions to perform constrained inference using isotonic regression for circular data (Rueda, Fernández, and Peddada 2009; Fernàndez, Rueda, and Peddada 2011). The **CircOutlier** (Ghazanfarihesari and Mashhad 2016) collects functions to detect outliers in circular-circular regression as proposed in Abuzaid, Hussin, and Mohamed (2013). Bayesian estimation for univariate regression models is implemented in **bpnreg** (Cremers 2018) that presents models developed in Nuñez-Antonio and Gutiérrez-Peña (2014) and Cremers, Mulder, and Klugkist (2017). Again in the Bayesian framework the work in Mulder and Klugkist (2017) is implemented in **circglmbayes**. More recent is the **Directional** (Tsagris, Athineou, Sajib, Amson, and Waldstein 2018) package, mostly linked to the textbook by Mardia and Jupp (1999). A series of wrapper functions to implement the 10 maximum likelihood models of animal orientation, described by Schnute and Groot (1992), are included in the **CircMLE**. The proposals in Zimmermann and Wright (2017) are presented in **circumplex**.

Dependent and multivariate circular data are often found in applications (see Ley and Verdebout 2019, for recent developments). To handle them in a likelihood framework we can refer to the package **CircNNTSR**, that implements functions to plot, fit by maximum likelihood, and simulate models based on non-negative trigonometric sums for circular, multivariate circular, and spherical data.

None of the above packages deal with spatial or spatio-temporal interpolation of circular data, that is the main objective of **CircSpaceTime**, the package we are proposing. In what follows we are going to present models that have been developed, starting from 2012 (Jona Lasinio *et al.* 2012); a summary of these models can also be found in Jona Lasinio, Mastrantonio, and Gelfand (2019). **CircSpaceTime** is currently submitted to CRAN and available at `https://github.com/santoroma/CircSpaceTime`.

There are different approaches to specify valid circular distributions, see for example Jammalamadaka and SenGupta (2001), here we focus on the two methods that allow to built a circular distribution starting from a linear one, namely the wrapping, and the projection. Both revealed to be useful in the definition of spatial and spatio-temporal data modeling, see for example Mastrantonio, Gelfand, and Jona Lasinio (2016a) and Wang and Gelfand (2014). Under both methods, the resulting distribution has a complex functional form but introducing suitable latent variables, the joint distribution of observed and latent variables, in a fully Bayesian framework, are really easy to handle.

2. From Linear to Circular models

In this section we are going to introduce, in both univariate and multivariate settings, the two approaches that we exploit to built distributions/processes for circular variables, starting from probability distributions with support the real line; we start with *linear random variables* to obtain *circular random variables*. The two methods that we are going to illustrate are totally general and can be applied to all linear variables with any distribution. We focus on the Gaussian case, since the resulting models are highly flexible and they are those implemented in **CircSpaceTime**.

2.1. The wrapping approach

With the wrapping approach, the idea is to "wrap" the density of a linear variable around the unit circle, obtaining than a circular density. In more details, let $Y \in \mathbb{R}$ be a linear random variable with pdf $f_Y(\cdot|\boldsymbol{\psi})$, where $\boldsymbol{\psi}$ is a vector of parameters. We obtain a circular random variable using the following transformation:

$$\Theta = Y \bmod 2\pi \in [0, 2\pi).$$

The pdf of Θ is then

$$f_\Theta(\theta|\boldsymbol{\psi}) = \sum_{k=-\infty}^{\infty} f_Y(\theta + 2\pi k|\boldsymbol{\psi}). \tag{1}$$

It is easy to find the relation between Y and Θ that is the following: $Y = \Theta + 2\pi K$, where K is the so-called *winding number*. Equation (1) wraps the density $f_Y(\cdot|\boldsymbol{\psi})$ around the unit circle and then Θ is called the *wrapped* version of $\mathbf{Y}$ of period 2π. If Y is normally distributed, then Θ follows a *wrapped normal* (WN) distribution.

The evaluation of (1) is not easy since it involves an infinite sum, possibly, making inference based on the wrapped distribution a really complex task. Following Coles and Casson (1998), this problem can be easily overcome if we consider K as a (latent) random variable. We have then $f_{\Theta,K}(\theta, k|\boldsymbol{\psi}) = f_Y(\theta + 2\pi k|\boldsymbol{\psi})$, i.e. the joint density of (Θ, K) is $f_Y(\theta + 2\pi k|\boldsymbol{\psi})$, that is the density Y, seen as function of θ and k. Notice that (1) can be seen as a marginalization over K of the joint density of (Θ, K). The conditional distribution of K, that is needed for the implementation of the MCMC, is easy to handle since it is proportional to $f_Y(\theta+2\pi k|\boldsymbol{\psi})$. This result shows that it is much easier to work with the joint density of (Θ, K), with respect to the marginal of Θ.

The wrapping approach can be easily extended to a multivariate setting, see for example Jona Lasinio *et al.* (2012). In details, let $\mathbf{Y} = (Y_1, \dots, Y_n)'$ be a n-variate vector with pdf $f_{\mathbf{Y}}(\cdot|\boldsymbol{\psi})$. We have then that $\boldsymbol{\Theta} = (\Theta_1, \dots, \Theta_n)'$, with

$$\Theta_i = Y_i \bmod 2\pi, i = 1, \dots, n, \tag{2}$$

is a vector of circular variables with density

$$f_{\boldsymbol{\Theta}}(\boldsymbol{\theta}|\boldsymbol{\psi}) = \sum_{k_1=-\infty}^{\infty} \cdots \sum_{k_n=-\infty}^{\infty} f_{\mathbf{Y}}(\boldsymbol{\theta} + 2\pi\mathbf{k}|\boldsymbol{\psi}).$$

Extending the reasoning applied to the univariate setting, we can easily find that the full conditional of $\mathbf{K}$ is proportional to $f_{\mathbf{Y}}(\boldsymbol{\theta} + 2\pi\mathbf{k}|\boldsymbol{\psi})$ and the joint density of $(\boldsymbol{\Theta}, \mathbf{K})$ is $f_{\mathbf{Y}}(\boldsymbol{\theta} +$

$2\pi\mathbf{k}|\boldsymbol{\psi})$. If we assume that $\mathbf{Y}$ is normally distributed, this means that the joint density of $(\boldsymbol{\Theta}, \mathbf{K})$ is

$$\phi_n(\boldsymbol{\theta} + 2\pi\mathbf{k}|\boldsymbol{\mu}, \boldsymbol{\Lambda}), \tag{3}$$

where $\phi_n(\cdot|\boldsymbol{\mu}, \boldsymbol{\Lambda})$ is the $n-$variate normal pdf with mean vector and covariance matrix given by $(\boldsymbol{\mu}, \boldsymbol{\Lambda})$. Here again it is easy to see that any type of model implementation is facilitated if $\mathbf{K}$ is introduced as a latent variable.

2.2. The projection approach

With the wrapping approach, from one linear variable we obtain a circular one. The projection approach is quite different in this regard since it requires two linear variables to obtain one circular. The basic idea is to take two random variables, that can be seen as coordinates in a Euclidean space, and express them as polar coordinates. The associated angle is itself a random variables and it is circular.

Let $\mathbf{Y} = (Y_1, Y_2)$ be a bivariate vector of linear variables with pdf $f_{\mathbf{Y}}(\cdot|\boldsymbol{\psi})$. The relation between the angle Θ and $\mathbf{Y}$ is expressed by the following:

$$\tan(\Theta) = \frac{Y_2}{Y_1}. \tag{4}$$

The circular variable can be then obtained using the inverse tangent function but some care must be taken. The inverse has domain of length π while we want our random variable to have domain $[0, 2\pi)$. This is generally achieved by using the atan* inverse (see Jammalamadaka and SenGupta 2001, pag.13 for a detailed definition), that takes into account the sign of the components of $\mathbf{Y}$, to identify to which of the 4 quadrants the circular variable belongs. Between Θ and $\mathbf{Y}$ the following relation exists $\mathbf{Y} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = R \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} = R\mathbf{U}$, with $R = ||\mathbf{Y}||$, where the joint vector $(\Theta, R)'$ is the representation in polar coordinates of $\mathbf{Y}$. From equation (4) is easy to see that distributions based on the projection are not identifiable, since $c\mathbf{Y}$ and $\mathbf{Y}$, with $c > 0$, gives rise to the same circular variables; an identification constraint is then needed (Wang and Gelfand 2013).

The pdf of Θ is

$$f_\Theta(\theta|\boldsymbol{\psi}) = \int_{\mathbb{R}^+} r f_{\mathbf{Y}}((r\cos(\theta), r\sin(\theta))'|\boldsymbol{\psi}) dr, \tag{5}$$

that is obtained by finding first the joint density of (Θ, R), which is

$$r f_{\mathbf{Y}}((r\cos(\theta), r\sin(\theta))'|\boldsymbol{\psi}),$$

and then, a marginalization over R, gives the density of Θ. The integral in equation (5) is not easy to solve and, even when a closed form exists, the resulting pdf has a complicated functional structure. For example, if we assume $\mathbf{Y} \sim N_2(\boldsymbol{\alpha}, \boldsymbol{\Xi})$, with $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)'$ and

$$\boldsymbol{\Xi} = \begin{pmatrix} \sigma^2 & \tau\sigma \\ \tau\sigma & 1 \end{pmatrix}, \tag{6}$$

where $(\boldsymbol{\Xi})_{2,2} = 1$ is needed for identifiability purpose, as explained above, the density of Θ, computed using (5), is

$$f_\Theta(\theta|\boldsymbol{\psi}) = \frac{\phi_2(\boldsymbol{\alpha}|\mathbf{0}_2, \boldsymbol{\Xi}) + aD(\theta)\Phi(D(\theta)|\mathbf{0}, \mathbf{I}_2)\phi(aC(\theta)^{-\frac{1}{2}}(\alpha_1\sin(\theta)) - \alpha_2\cos(\theta))}{C(\theta)}, \tag{7}$$

where $\Phi_n(\cdot|\cdot,\cdot)$ is the $n-$variate cumulative density function, with

$$\begin{aligned}
&a = \left(\sigma\sqrt{1-\tau^2}\right)^{-1},\\
&C(\theta) = a^2\left(\cos^2(\theta) + \sigma^2\sin^2(\theta) - \tau\sigma\sin(2\theta)\right),\\
&D(\theta) = a^2 C(\theta)^{-\frac{1}{2}}\left(\alpha_1(\cos(\theta) - \tau\sigma\sin(\theta)) + \alpha_2\sigma(\sigma\sin(\theta) - \rho\cos(\theta))\right).
\end{aligned}$$

Under the normal assumption, we have that the joint density of (Θ, R) is

$$r\phi_2((r\cos(\theta), r\sin(\theta)|\boldsymbol{\alpha}, \boldsymbol{\Xi}). \tag{8}$$

Equation (8) is less complex than (7) but, as for the wrapping approach, to work with it we need to introduce a latent variable, that is R.

The extension of the projection approach to multivariate variable is straightforward. We take $\mathbf{Y}$ as a $2n-$variate linear variable, and we build the $n-$variate vector of (projected) circular variables through the following transformation:

$$\Theta_i = \text{atan}^*\left(\frac{Y_{2i}}{Y_{2i-1}}\right), i = 1, \dots, n, \tag{9}$$

i.e., the $2n$ elements of $\mathbf{Y}$ are grouped in n sets, each of them containing only two values, and where one element of $\mathbf{Y}$ can only belong to one of the n sets. Each set gives rise to a circular variable. The density of the vector $\boldsymbol{\Theta} = (\Theta_1, \dots, \Theta_p)'$ is then

$$f_{\boldsymbol{\Theta}}(\boldsymbol{\theta}|\boldsymbol{\psi}) = \int_{\mathbb{R}^+} \cdots \int_{\mathbb{R}^+} \prod_{i=1}^{n} r_i f_{\mathbf{Y}}(\mathbf{y}(\boldsymbol{\theta}, \mathbf{r})|\boldsymbol{\psi}) dr_1 \dots dr_n, \tag{10}$$

where $r_i = ||(y_{2i-1}, y_{2i})'||$ and $\mathbf{y}(\boldsymbol{\theta}, \mathbf{r})$ is used to indicated that $\mathbf{y}$ must be seen as function of $\boldsymbol{\theta}$ and $\mathbf{r} = (r_1, \dots, r_n)$. Even assuming a normal density for $\mathbf{Y}$, it is not possible to write in closed form (10). As for the univariate case, it is much easier to work with the joint density of $(\boldsymbol{\Theta}, \mathbf{R})$ since, if for example a normal distribution of $\mathbf{Y}$ is assumed, it is then

$$\prod_{i=1}^{n} r_i \phi_{2n}(\mathbf{y}(\boldsymbol{\theta}, \mathbf{r})|\boldsymbol{\mu}, \boldsymbol{\Lambda}), \tag{11}$$

where, as in equation (3), $(\boldsymbol{\mu}, \boldsymbol{\Lambda})$ are the mean and covariance matrix.

Differences between the two approaches

There are considerable differences between the two approaches discussed above. The wrapping creates circular distributions that are in general similar to their counter-parts on the real line. For example, the wrapped normal is still unimodal and symmetric. Moreover, the circular mean and variance (see Jammalamadaka and SenGupta 2001, for the definitions and details), are simple functions of the mean and variance of the linear variable.

On the other hand, even under the normal assumption, the projection approach creates a distribution that has often properties very different from its linear partners. Again in the Gaussian setting, the projected normal can be bimodal, asymmetric and antipodal. Moreover, only in few special cases, exists a closed form expression of the mean and variance of the projected distribution.

Name	Function	Parameters
Matèrn	$\frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}\frac{h_{\mathrm{sp}}}{\rho}\right)^{\nu} K_{\nu}\left(\sqrt{2\nu}\frac{h_{\mathrm{sp}}}{\rho}\right)$	ν, ρ
Exponential	$\exp\left(-\rho h_{\mathrm{sp}}\right)$	ρ
Gaussian	$\exp\left(-\rho^2 h_{\mathrm{sp}}\right)$	ρ
Gneiting (spatio-temporal)	$\frac{1}{\rho_t h_{\mathrm{t}}^2+1}\exp\left(-\frac{\rho_{sp} h_{\mathrm{sp}}}{(\rho_t h_{\mathrm{t}}^2+1)^{\frac{\eta}{2}}}\right)$	ρ_t, ρ_{sp}, η

Table 1: Correlation functions implemented. h_{sp} and h_{t} are, respectively, spatial and temporal distance.

The main reasons to propose these two approaches is that, in both, it is easy to introduce dependence (spatial, temporal or both). The wrapping gives results that are really easy to interpret in terms of phenomena behaviour, while the projection is very useful when interpretation is not central, and a highly flexible model is required (Mastrantonio *et al.* 2016b).

2.3. Spatio-temporal processes for circular variables

A stochastic process can be defined through its finite dimensional distribution, i.e. the distribution of an $n-$dimensional realization, that has a multivariate pdf (Gelfand, Diggle, Fuentes, and Guttorp 2010). Since in the previous sections we have already given multivariate distributions for circular variables, it is then easy to define circular processes. More precisely, starting from a distribution for linear variables, we can use the wrapping or the projection approach to obtain a multivariate circular distribution and then, as a consequence, from an $n-$dimensional realization of a linear process we can obtain the associated $n-$dimensional realization of a circular one.

Let $Y(\mathbf{s}) = \{Y_i(\mathbf{s})\}_{i=1}^p \in \mathbb{R}^p$ be a $p-$variate Gaussian process (GP) defined over a $s-$dimensional domain, i.e., $\mathbf{s} \in \mathcal{S} \subset \mathbb{R}^d$, and let $\mathbf{y} \equiv (\mathbf{y}_1, \dots, \mathbf{y}_n)' \in \mathbb{R}^p \times \mathbb{R}^n$ be the set of n realizations. The vector $\mathbf{y}$ is then normally distributed. Given $\mathbf{Y}(\mathbf{s})$ we can easily built the wrapped and the projected GP using (2) and (9), respectively. More precisely, the former is obtained as

$$\Theta(\mathbf{b}) = \mathbf{Y}(\mathbf{s}) \bmod 2\pi.$$

while the latter is given by

$$\Theta(\mathbf{s}) = \text{atan}^*\left(\frac{Y_2(\mathbf{s})}{Y_1(\mathbf{s})}\right).$$

The same transfromation, applied to $\mathbf{y}$, give the realization of the wrapped and projected GP.

For both, the spatial or space-time representations are obtained by considering a spatially or spatio-temporally structured covariance matrix.

3. The implemented models

The basic linear model we adopt is the following:

$$\begin{aligned} \mathbf{Y}(\mathbf{s}) &= \boldsymbol{\alpha} + \boldsymbol{\omega}(\mathbf{s}), \\ \boldsymbol{\omega}(\mathbf{s}) &\sim GP_p(\mathbf{0}_p, C(\mathbf{h}; \boldsymbol{\varphi}) \otimes \boldsymbol{\Xi}), \end{aligned} \tag{12}$$

Parameter	Wrapped Normal	Projected Normal
Spatial and temporal decay $(\rho, \rho_{sp}, \rho_t)$	uniform	uniform
σ^2	inverse gamma	inverse gamma
Separability parameter ($\eta =$ `sep_par`)	Beta	Beta
mean parameters (α)	Wrapped Gaussian	Gaussian
correlation between components (τ)		uniform

Table 2: Available prior distributions

where $\boldsymbol{\alpha} \in \mathbb{R}^p$ is a mean vector, $\boldsymbol{\omega}(\mathbf{s})$ is a zero mean $p-$variate GP and $\otimes$ is the usual Kronecker product. The correlation function of the GP is $C(\mathbf{h}; \boldsymbol{\varphi})$ and it depends on parameter $\boldsymbol{\varphi}$ and the vector of distances $\mathbf{h}$, that contains the spatial and temporal distances h_{sp} and h_{t} if $d = 3$, and only h_{sp} if $d = 2$. Under $p = 1$ the model (12) is used to build a wrapped GP while $p = 2$ is used for the projected. Table 1 shows the correlation functions and Table 2 the choice of prior distributions available in the package **CircSpaceTime**. The parameter $\boldsymbol{\Xi}$ is defined as in equation (6) if $p = 2$, and σ^2 if $p = 1$.

3.1. Model implementation

As we stated in Section 2, the multivariate wrapped and projected normal have densities too complex to be used for model inference but, if we introduce as latent observations the vector $\mathbf{k}$, for the wrapped normal, and $\mathbf{r}$ for the projected, inference through MCMC algorithm is straightforward.

The key element here is that, once the latent variables are obtained, the full conditionals of the model parameters are exactly the same as if we were working with observed linear variables. This is due to the the functional form of the likelihood of circular and latent variables, see (3) and (11), which gives the same contribution to the full conditionals of the parameters, as if the data has a multivariate normal density.

Due to the priors available in the package, see Table 2, the update of $\boldsymbol{\alpha}$, i.e. the mean parameter, is done with a Gibbs sample since its full conditional is multivariate normal under the projected model and wrapped normal under the wrapped model. The parameters of the covariance structure are updated all together, within a Metropolis step. To speed up convergence and to have the possibility to choose the final acceptance rate, for the multivariate proposal distribution we implemented the algorithm shown in Andrieu and Thoms (2008), algorithm 4.

The latent discrete variable $\mathbf{k}$ is updated component-wise with Metropolis steps. In each one of them, given the current value k_i, the proposal is selected from the set $(k_i - 1, k_i, k_i + 1)$, where each element has probability 1/3 to be selected. The latent variable $\mathbf{r}$ is also updated component-wise with Metropolis steps, but in this case we use the algorithm proposed by Rosenthal (2007).

In fitting the wrapped Gaussian models we center data (and all related quantities) around π. Outputs for the mean and the interpolated values are given on the original scale, while the winding numbers are those obtained centering around π.

To improve performances we implemented everything in *C++* (Wickham 2015) and using **Rcpp** package we simplify the integration between C++ and R codes (Eddelbuettel and Francois 2011). In particular we used the **RcppArmadillo** package (Eddelbuettel and Sanderson 2014), that implements the Armadillo matrix library, for its simplicity and elegance (Eddelbuettel 2017), although the **RcppEigen** is a bit faster (Bates and Eddelbuettel 2013). For a fast multiple chain estimations we use **doParallel** package (MicrosoftCorporation and Weston 2017).

4. A package's tutorial

4.1. Functions structure

Estimation of posterior distributions are obtained via the following functions:

- `WrapSp`, `ProjSp` - Spatial models;
- `WrapSpTi`, `ProjSpTi` - Spatio-temporal models.

All functions require, as input data, a list of starting values, a list of priors parameters and information for the adaptive MCMC algorithm and the MCMC run (number of iterations, burnin, thinning and the number of chains). All functions return outputs organized with as many lists as the number of chains and each list has as many elements as the number of model parameters.

Prediction in space and/or space and time is performed using:

- `WrapKrigSp`, `ProjKrigSp` - Spatial models;
- `WrapKrigSpTi`, `ProjKrigSpTi` - Spatio-temporal models.

As input, the interpolation functions require the results from the posterior distribution estimation and the locations where we want to predict the process. Outputs are organized as lists with 3 elements: posterior predictive mean and variance (M_out, V_out), and predicted values (Pred_out).

4.2. Data for the spatial examples

We consider an example based on wave directions (and heights). These data are obtained as outputs from a deterministic computer model implemented by Istituto Superiore per la Protezione e la Ricerca Ambientale (ISPRA) (Inghilesi, Orasi, and Catini 2016). The computer model starts from a wind forecast model predicting the surface wind over the entire Mediterranean. The hourly evolution of sea wave spectra is obtained by solving energy transport equations using the wind forecast as input. Wave spectra are locally modified using a source function describing the wind energy, the energy redistribution due to nonlinear wave interactions, and energy dissipation due to wave fracture. The model produces estimates every hour on a grid with 10×10 km cells (Inghilesi *et al.* 2016). The ISPRA dataset has forecasts for a total of 4941 grid points over the Italian Mediterranean. Over the Adriatic sea area, there are 1494 points.

We load a list containing 4 data frames each of 35856 rows and 7 columns

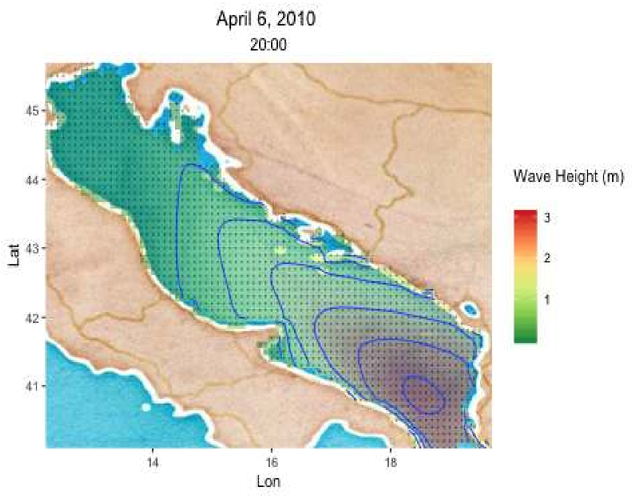

Figure 1: Adriatic sea: waves directions on April 6, 2010 at 8pm, the color scale follows the state of the sea from green to red (state of the sea according to wave height <1m, calm, between 1 and 2 m, transition, above 2m, storm)

- Date: Date, format: yyyy-mm-dd;
- hour: Factor w/ 24 levels corresponding to the 24h, format: 00:00;
- Lon, Lat: Decimal longitude and latitude;
- Hm0: Significant wave heights in meters;
- Dm: Direction of waves in degrees (North=0);
- state: Factor w/ 3 levels "calm","transition", "storm", the categories are built on the basis of the wave height (Hm0): $Hm0 \leq 1m$ calm, $1 < Hm0 < 2$ transition, $Hm0 \geq 2$ storm.

We select the month of April 2010 at 20:00:

```
R> require(CircSpaceTime)
R> data(april)
R> storm1 <- april$apr6.2010[april$apr6.2010$hour == "20:00",]
```

We convert the directions into radians

```
R>  storm1$Dmr<-storm1$Dm*pi/180
```

We plot the samples in terms of the sea state

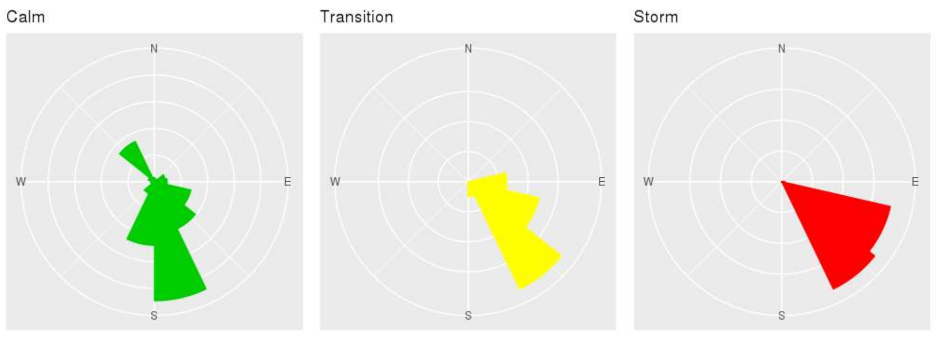

Figure 2: Adriatic sea: rose diagrams of waves directions according to the state of the sea (wave height <1m, calm, between 1 and 2 m, transition, above 2m, storm)

```
R> require(gridExtra)
R>
R> r1 <- rose_diag(storm1$Dmr[storm1$state == "calm"],
R>              bins = 15, color = 3, template = "wind_rose") +
R>              ggtitle("Calm")
R> r2 <- rose_diag(storm1$Dmr[storm1$state == "transition"],
R>              bins = 15, color = "yellow", template = "wind_rose") +
R>              ggtitle("Transition")
R> r3 <- rose_diag(storm1$Dmr[storm1$state == "storm"],
R>              bins = 15, col = 2, template = "wind_rose") +
R>              ggtitle("Storm")
R> grid.arrange(r1, r2, r3, ncol = 3)
```

From Figure 2 it is very clear that the variability decreases from *calm* to *storm*. This finding is very relevant in terms of model choices. When the variability is too high, i.e. the distribution is close to the circular uniform, the variance parameter of the wrapped normal is not identifiable (see Jona Lasinio *et al.* 2012)). Then, When modeling highly variable data, the projected normal may be a better choice than the wrapped normal. When dealing with concentrated data, as for the storm regime, both models may be tested as we'll do in what follows. We'll choose the model that "best" fit the data according to prediction error and a proper score (see below for details).

First we select the area where the storm is still in full force.

```
R> storm2 <- storm1[(storm1$state == "storm" & storm1$Lat<=41),]
```

Further, it is better to convert the coordinates into UTM as the estimation algorithm uses Euclidean distance. We also hold out 20% of the locations for validation purposes, defining a training and testing sets. In Figure 3 we show the two datasets

```
R> nval                      <- round(nrow(storm2)*0.2)
R> sample.val                <- sort(sample(c(1:nrow(storm2)),nval))
R> train                     <- storm2[-sample.val,]
```

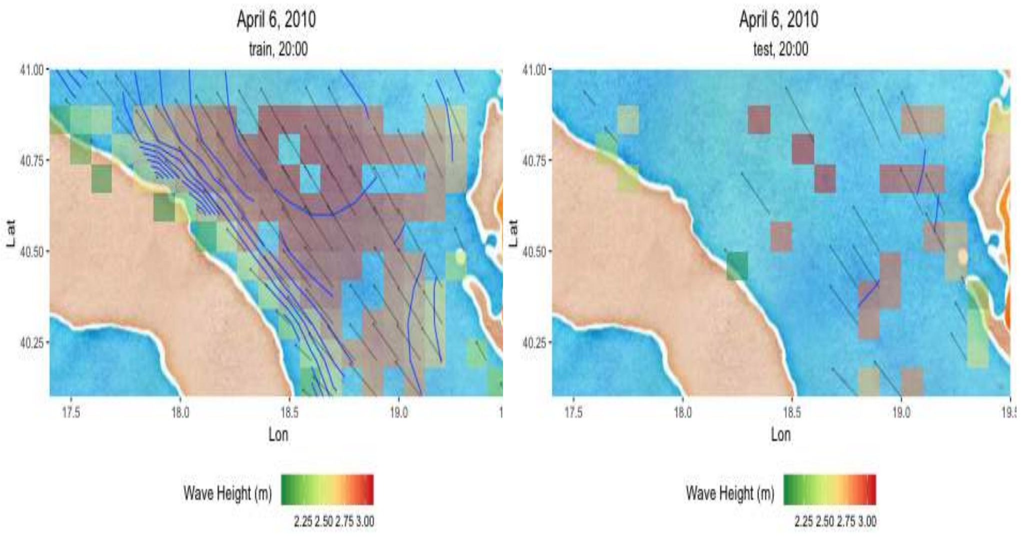

Figure 3: Adriatic sea: test and training sets from the storm on April 6, 2010 at 8pm, the color scale follows the state of the sea from green to red (state of the sea according to wave height <1m, calm, between 1 and 2 m, transition, above 2m, storm)

```
R> test                          <- storm2[sample.val,]
R> coords                        <- storm2[,3:4]
R> colnames(coords)              <- c("X","Y")
R> attr(coords,"projection")     <- "LL"
R> attr(coords,"zone")           <- 32
R> coords_2                      <- PBSmapping::convUL(coords,km = TRUE)
R> coords.train                  <- coords_2[-sample.val,]
R> coords.test                   <- coords_2[sample.val,]
R> distance_matrix               <- dist(coords_2)
```

The elements for the definition of prior distributions are organized in a list that is passed to the functions `WrapSp` or `ProjSp`.

From the distance matrix we compute the empirical practical range interval, that is used in the definition of the uniform prior distribution adopted for the spatial correlation decay parameter.

```
R> rho_max <- 3./min(distance_matrix[which(distance_matrix > 0)])
R> rho_min <- 3./max(distance_matrix[which(distance_matrix > 0)])
```

4.3. Spatial Wrapped Gaussian model

The first example we illustrate involves the WN model and starts with the estimation of the posterior distribution.

The function `WrapSp` produces samples from the wrapped normal spatial model.

In our example we choose an exponential correlation and we run 2 chains in parallel. The available prior distributions for the wrapped and projected models are described in Table 2. Notice that we choose a vector of zeros as starting value for k. This is a reasonable value when circular data are centered around π.

```
R> start1 <- list(
R> "alpha"  = c(2*pi,3.14),
R> "rho"    = c(.5*(rho_min + rho_max),.1*(rho_min + rho_max)),
R> "sigma2" = c(0.1,1),
R> "k"      = c(rep(0, nrow(train)),rep(0, nrow(train)))
R> )
R> # Running WrapSp may take some time
R> storm <- WrapSp(
R>   x            = train$Dmr,
R>   coords       = coords.train,
R>   start        = start1,
R>   priors       = list("alpha"  = c(pi,10),
R>                       "rho"    = c(rho_min, rho_max),
R>                       "sigma2" = c(3,0.5)),
R>   sd_prop      = list("sigma2" = 1,
R>                       "rho"    = 0.3),
R>   iter         = 30000,
R>   BurninThin   = c(burnin = 15000,
R>                    thin   = 10),
R>   accept_ratio = 0.5,
R>   adapt_param  = c(start = 100,
R>                    end   = 10000,
R>                    exp   = 0.8),
R>   corr_fun     = "exponential",
R>   n_chains     = 2,
R>   parallel     = TRUE,
R>   n_cores      = 2
R>   )
```

We can control the convergence of the MCMC using the function `ConvCheck`, that returns an mcmc.list (mcmc), to be used with the `coda` package, and the Potential scale reduction factors (Rhat) of the model parameters, computed using the `gelman.diag` function in the coda package.

```
R>  check <- ConvCheck(storm)
R>  check$Rhat
```

```
|         | Point est.| Upper C.I. |
```

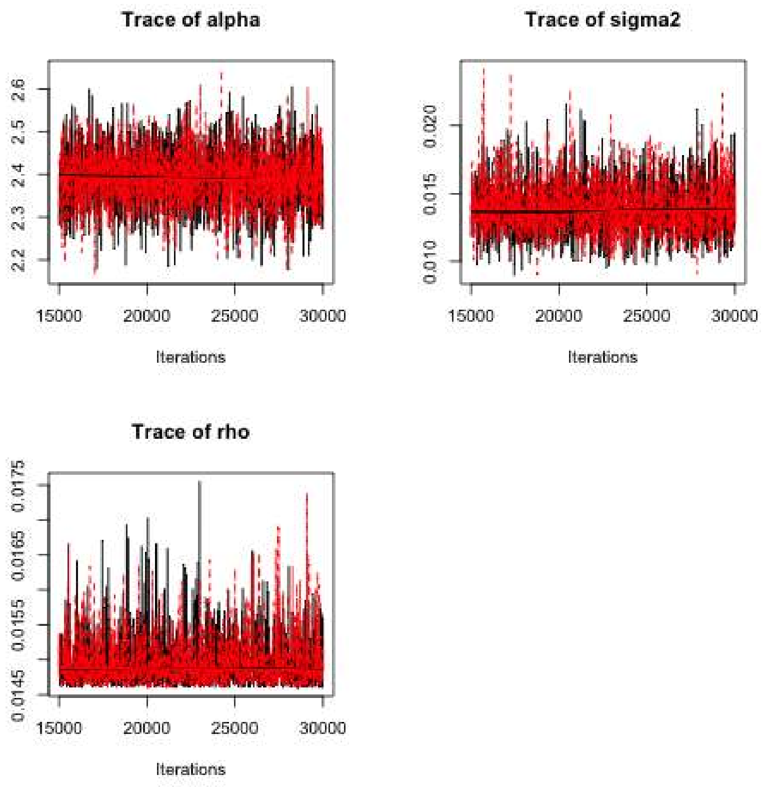

Figure 4: Traces from the mcmc run for the wrapped Gaussian spatial example - April 6 2010, storm data.

```
|---------|:---------:|-----------:|
| alpha   | 1.00      | 1.00       |
| rho     | 1.00      | 1.01       |
| sigma2  | 1.00      | 1.00       |

Multivariate psrf

1
```

We draw the traceplots always remebering that `alpha` is a circular variable

```
R>  library(coda)
R>  plot(check$mcmc, trace = TRUE, density = FALSE)
```

Convergence is clearly achieved and we can proceed to estimate the values on the test sites, using the posterior samples we just obtained.

```
R> Pred.storm <- WrapKrigSp(WrapSp_out = storm,
R>                          coords_obs = coords.train,
R>                          coords_nobs = coords.test,
R>                            x_obs = train$Dmr
R>                            )
```

Once the estimates are obtained, we compute the Average Prediction Error (APE) (see Jona Lasinio *et al.* 2012) and the circular continuous ranked probability score (CRPS) (Grimit, Gneiting, Berrocal, and Johnson 2006), computed using the functions `APEcirc` and `CRPScirc`, available in the package. They both return the average index value, computed over the test sites, and a vector of scores computed on each site.

```
R> Ape.storm.wrap  <- APEcirc(test$Dmr,Pred.storm$Prev_out)
R> crps.storm.wrap <- CRPScirc(test$Dmr,Pred.storm$Prev_out)
R> Ape.storm.wrap$Ape
R> [1] 0.001149621
R> crps.storm.wrap$CRPS
R> [1] 0.0002420352
```

4.4. Spatial Projected Normal model

In this section we are going to run the predictions, on the same storm data used for the wrapped normal example, adopting a projected normal (PN) spatial model (Mastrantonio *et al.* 2016b). The function `ProjSp` produces samples from the PN. In our example we choose again an exponential correlation and we run 2 chains in parallel.

```
R> set.seed(12345)
R> start_PN <- list(
R>  "alpha"  = c(2*pi, pi/4, pi*2/3, pi*2/3),
R>  "tau"    = c(-0.9,-0.7),
R>  "rho"    = c(0.015,0.02),
R>  "sigma2" = c(1,0.1),
R>  "r"      = abs(rnorm(nrow(train)))
R>  )
R>
R> storm_PN <- ProjSp(
R>  x            = train$Dmr,
R>  coords       = coords.train,
R>  start        = start_PN ,
R>  priors       = list("rho"         = c(rho_min,rho_max/2),
R>                      "tau"         = c(-1,1),
R>                      "sigma2"      = c(1,1),
R>                      "alpha_mu"    = c(0, 0),
R>                      "alpha_sigma" = diag(20,2)),
R>  sd_prop      = list("sigma2" = .1,
R>                      "tau"    = .1,
R>                      "rho"    = .1,
R>                      "sdr"    = rep(.01,nrow(train))),
R>  iter         = 50000,
R>  BurninThin   = c(burnin = 25000, thin = 10),
R>  accept_ratio = 0.234,
R>  adapt_param  = c(start = 100, end = 10000, exp = 0.5),
R>  corr_fun     = "exponential",
R>  n_chains     = 2 ,
R>  parallel     = TRUE ,
R>  n_cores      = 2
R>  )

R> check_PN <- ConvCheck(storm_PN)
R> check_PN$Rhat
R> Potential scale reduction factors:
R>
R>       Point est. Upper C.I.
R> sigma2       1.21       1.71
R> rho          1.01       1.06
R> tau          1.23       1.79
R> alpha1       1.00       1.02
R> alpha2       1.00       1.01
R>
R> Multivariate psrf
R>
R> 1.24
```

From the convergence check and the traceplots (not reported) we see that the chains need to be updated. This can be easily done by setting as starting values the last values of all chains, and removing the adaptive step by setting a starting iteration for the adaptation larger than the number of iterations.

```
R> n           <- length(storm_PN[[1]]$sigma2)
R> start_PN.up <-list("alpha"  = c(storm_PN[[1]]$alpha[1,n],
R>                                   storm_PN[[1]]$alpha[2,n],
R>                                   storm_PN[[2]]$alpha[1,n],
R>                                   storm_PN[[2]]$alpha[2,n]),
R>                   "tau"    = c(storm_PN[[1]]$tau[n],
R>                                   storm_PN[[2]]$tau[n]),
R>                   "rho"    = c(storm_PN[[1]]$rho[n],
R>                                   storm_PN[[2]]$rho[n]),
R>                  "sigma2"  = c(storm_PN[[1]]$sigma2[n],
R>                                   storm_PN[[2]]$sigma2[n]),
R>                  "r"       = abs(rnorm(nrow(train)))
R>                  )
R>
R> storm_PN.up <- ProjSp(
R>  x            = train$Dmr,
R>  coords       = coords.train,
R>  start        = start_PN.up,
R>  priors        = list("rho"          = c(rho_min,rho_max/2),
R>                       "tau"         = c(-1,1),
R>                       "sigma2"      = c(1,1),
R>                       "alpha_mu"    = c(0, 0),
R>                       "alpha_sigma" = diag(20,2)),
R>  sd_prop       = list("sigma2" = .1,
R>                       "tau"    = .1,
R>                       "rho"    = .1,
R>                       "sdr"    = rep(.01,nrow(train))),
R>  iter          = 50000,
R>  BurninThin    = c(burnin = 25000,
R>                    thin   = 10),
R>  accept_ratio = 0.234,
R>  adapt_param   = c(start = 70001, #no adaptive mcmc
R>                    end   = 70001,
R>                    exp   = 0.5),
R>  corr_fun     = "exponential",
R>  n_chains     = 2 ,
R>  parallel     = TRUE ,
R>  n_cores      = 2
R>  )
```

We run the update 4 times and eventually we reach convergence:

```
R> check.PNstorm1<-ConvCheck(storm_PN.up)
R> check.PNstorm1$Rhat
R>
R> Potential scale reduction factors:
R>
R>        Point est. Upper C.I.
R> sigma2       1.09       1.22
R> rho          1.01       1.03
R> tau          1.03       1.08
R> alpha1       1.00       1.02
R> alpha2       1.00       1.00
R>
R> Multivariate psrf
R>
R> 1.07
```

We can then move to the prediction step.

```
R> Pred.krig_PN <- ProjKrigSp(storm_PN.up,
R>                            coords_obs  = coords.train,
R>                            coords_nobs = coords.test,
R>                            x_obs       = train$Dmr)
```

We can compare the predictions of the two models using the APE and the CRPS.

```
R> Ape.storm.PN <- APEcirc(real = test$Dmr,
R>                         sim  = Pred.krig_PN$Prev_out
R>                         )
R>
R> crps.storm.PN <- CRPScirc(real = test$Dmr,
R>                           sim  = Pred.krig_PN$Prev_out
R>                           )
```

And we can compare results with the wrapped spatial model.

```
R> > Ape.storm.PN$Ape
R> [1] 0.1161423
R> > crps.storm.PN$CRPS
R> [1] 0.01660417
R> > Ape.storm.wrap$Ape
R> [1] 0.001149621
R> > crps.storm.wrap$CRPS
R> [1] 0.0002420352
```

According to these diagnostic, the wrapped normal, in this example, performs better than the projected. However both models are highly accurate as both APE are smaller than one radians.

In the following sections we drop real data and we move to two simple simulated examples, to illustrate the implementation of both circular space-time models, wrapped and projected.

4.5. Spatio-temporal circular wrapped normal model

We simulate 100 spatial coordinates from a uniform distribution on (0,100) and 100 time coordinates from the same interval. Then we generate 100 values from a GP with Gneiting correlation function (Gneiting 2002) (see Table 1) with the following setting:

- constant mean `alpha=0.5` on each location;
- spatial decay `rho=0.05`;
- time decay `rhoT=0.01`;
- process variance `sigma2=0.3`;
- separability parameter `sep_par=0.5`.

Then we wrap the obtained results around the circle component-wise, using (2), generating the data reported in Figure 5.

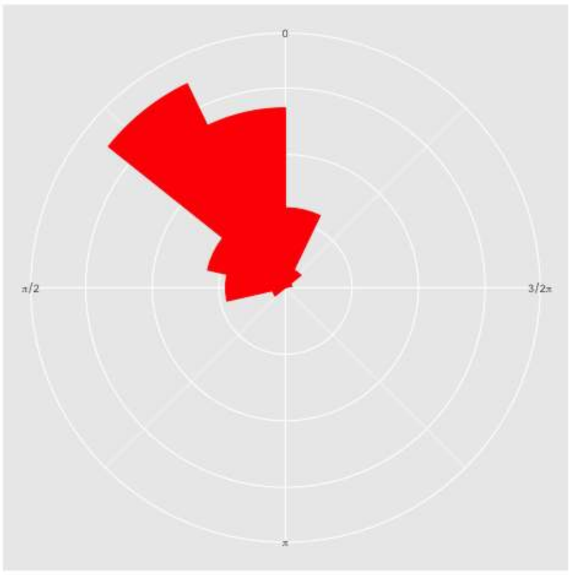

Figure 5: Simulated values for the wrapped normal space-time example

```
R> rmnorm=function(n = 1, mean = rep(0, d), varcov)
R> {
R>  d <- if (is.matrix(varcov))
R>            ncol(varcov)
R>       else 1
R>  z <- matrix(rnorm(n * d), n, d) %*% chol(varcov)
R>  y <- t(mean + t(z))
R>  return(y)
R> }
R>
R> ######################################
R> ## Simulation                      ##
R> ######################################
R> set.seed(1)
R> n = 100
R> ### simulate coordinates from a uniform distribution
R> coords  = cbind(runif(n,0,100), runif(n,0,100)) #spatial coordinates
R> coordsT = sort(runif(n,0,100)) #time coordinates (ordered)
R> Dist    = as.matrix(dist(coords))
R> DistT   = as.matrix(dist(coordsT))
R>
R> rho     = 0.05 #spatial decay
R> rhoT    = 0.01 #temporal decay
R> sep_par = 0.5 #separability parameter
R> sigma2  = 0.3 # variance of the process
R> alpha   = c(0.5)
R> #Gneiting covariance
```

```
R> SIGMA    = sigma2*(rhoT*DistT^2+1)^(-1)*
R>            exp(-rho*Dist/(rhoT*DistT^2+1)^(sep_par/2))
R>
R> Y        = rmnorm(1,rep(alpha,times=n), SIGMA) #generate the linear variable
R> theta    = c()
R> ## wrapping step
R> for(i in 1:n)
R> {
R>  theta[i] = Y[i]%%(2*pi)
R> }
```

As in the spatial case we set values for the spatial and temporal range priors, we build training and testing sets and we run the `WrapSpTi` function with a large number of iterations[1]

```
R> ### use this values as references for the
R> ### definition of initial values and priors
R> rho_sp.min <- 3/max(Dist)
R> rho_sp.max <- rho_sp.min+0.5
R> rho_t.min  <- 3/max(DistT)
R> rho_t.max  <- rho_t.min+0.5
R> val        <- sample(1:n,round(n*0.2)) #validation set
R>
R> set.seed(100)
R> mod <- WrapSpTi(
R>   x            = theta[-val],
R>   coords       = coords[-val,],
R>   times        = coordsT[-val],
R>   start        = list("alpha"   = c(1, 0.1),
R>                       "rho_sp"  = c(runif(1,0.01,rho_sp.max),
R>                                     runif(1,0.001,rho_sp.max)),
R>                       "rho_t"   = c(runif(1,0.01,rho_t.max),
R>                                     runif(1,0.001,rho_t.max)),
R>                       "sigma2"  = c(0.1, 1),
R>                       "sep_par" = c(0.4, 0.01),
R>                       "k"       = rep(0,length(theta))),
R>   priors       = list("rho_sp"  = c(0.01,3/4),
R>                       "rho_t"   = c(0.01,3/4),
R>                       "sep_par" = c(1,1),
R>                       "sigma2"  = c(5,5),
R>                       "alpha"   =  c(0,20)
R>   )  ,
R>   sd_prop      = list("sigma2"  = 0.1,
R>                       "rho_sp"  = 0.1,
R>                       "rho_t"   = 0.1,
R>                       "sep_par" =0.1),
```

[1]it is reasonably fast with this data size

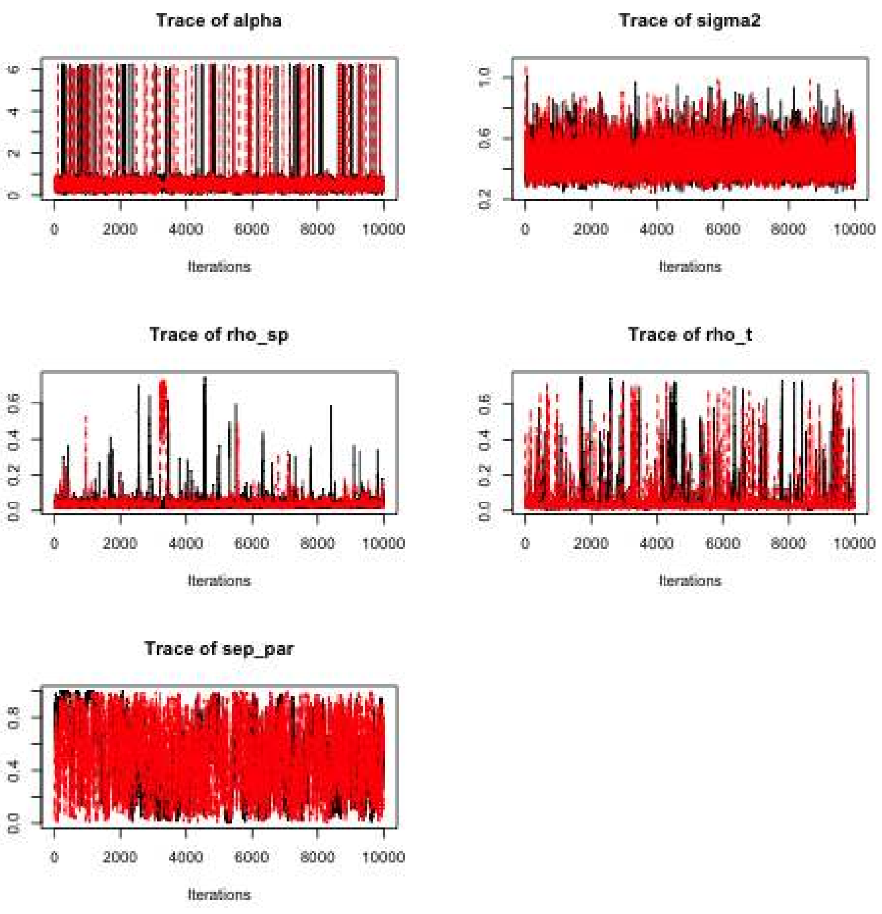

Figure 6: Trace plots for the wrapped normal space-time example

```
R>    iter          = 150000,
R>    BurninThin    = c(burnin = 50000,
R>                      thin   = 10),
R>    accept_ratio = 0.234,
R>    adapt_param  = c(start = 1,
R>                      end   = 1000,
R>                      exp   = 0.5),
R>    n_chains      = 2 ,
R>    parallel      = TRUE ,
R>    n_cores       = 2
R>    )
```

The convergence check returns a Multivariate psrf of 1.01 and all parameters chains seem to have reached convergence, as it can be seen in Figure 6

Again we can move to the prediction on the test sites.

```
R> Krig <- WrapKrigSpTi(
R>    WrapSpTi_out = mod,
R>   coords_obs   = coords[-val,],
R>   coords_nobs  = coords[val,],
R>   times_obs    = coordsT[-val],
R>   times_nobs   = coordsT[val],
R>   x_obs        = theta[-val]
R>   )
R> ### checking the prediction
R> Wrap_Ape  <- APEcirc(theta[val], Krig$Prev_out)
R> Wrap_CRPS <- CRPScirc(theta[val], Krig$Prev_out)
```

The Average prediction error on the 20 test locations is 0.25 radians, ranging from 0.123 to 0.899, while the average CRPS is 0.113, with point values ranging from 0.010 to 0.760.

4.6. Spatio-temporal projected normal model

We simulate data using the same scheme of the wrapped spatio-temporal example, but the simulated GP is here bivariate. The parameters are:

- constant mean `alpha=(0.5,0.5)'` on each location;
- spatial decay `rho=0.05`;
- time decay `rhoT=0.01`;
- process variance `sigma2=0.3`;
- `tau=0.2`;
- separability parameter `sep_par=0.1`.

After the GP is simulated, we obtain the projected normal realization using (9).

```
R> set.seed(1)
R> n       <- 100
R> coords  <- cbind(runif(n,0,100), runif(n,0,100))
R> coordsT <- cbind(runif(n,0,100))
R> Dist    <- as.matrix(dist(coords))
R> DistT   <- as.matrix(dist(coordsT))
R>
R> rho     <- 0.05
R> rhoT    <- 0.01
R> sep_par <- 0.1
R> sigma2  <- 0.3
R> alpha   <- c(0.5)
R> SIGMA   <- sigma2*(rhoT*DistT^2+1)^(-1)*
R>            exp(-rho*Dist/(rhoT*DistT^2+1)^(sep_par/2))
R> tau     <- 0.2
R>
```

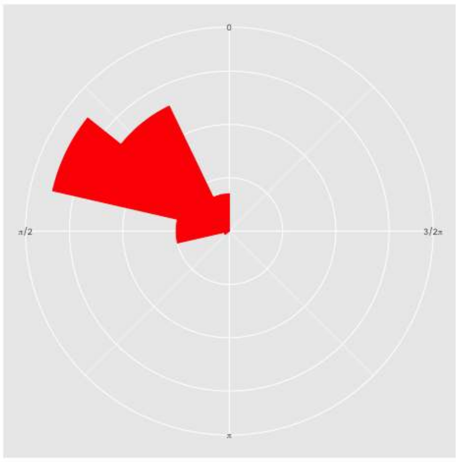

Figure 7: Data generated from a projected normal spatio-temporal process

```
R> Y        <- rmnorm(1,rep(alpha,times=n),
R>                    kronecker(SIGMA,matrix(c(sigma2,sqrt(sigma2)*
R>                              tau,sqrt(sigma2)*tau,1 ),nrow=2)))
R> theta    <- c()
R> for(i in 1:n)
R> {
R>  theta[i] <- atan2(Y[(i-1)*2+2],Y[(i-1)*2+1])
R> }
R> rose_diag(theta)
```

Data are plotted in Figure 7

Again we compute minimimum and maximum practical ranges, we define training and testing sets, and we run the estimation of the projected normal spatio-temporal process.

```
R> set.seed(100)
R> mod = ProjSpTi(
R>   x             = theta[-val],
R>   coords        = coords[-val,],
R>   times         = coordsT[-val],
R>   start         = list("alpha"   = c(1,1,pi/4,pi/4),
R>                        "rho_sp"  = c(0.1, rho_sp.max),
R>                        "rho_t"   = c(0.1, rho_t.max),
R>                        "sep_par" = c(0.4, 0.01),
R>                        "tau"     = c(0.1, 0.5),
R>                        "sigma2"  = c(0.1, 1),
R>                        "r"       = abs(rnorm(  length(theta))  )),
R>   priors        = list("rho_sp"      = c(0.001,3/4),
R>                        "rho_t"       = c(0.001,3/4),
R>                        "sep_par"     = c(1,1),
R>                        "tau"         = c(-1,1),
R>                          "sigma2"      = c(5,5),
R>                          "alpha_mu"    = c(0, 0),
R>                          "alpha_sigma" = diag(10,2)),
R>   sd_prop      = list("sep_par"= 0.1,
R>                          "sigma2" = 0.1,
R>                          "tau"    = 0.1,
R>                          "rho_sp" = 0.1,
R>                          "rho_t"  = 0.1,
R>                          "sdr"    = rep(.05,length(theta))),
R>   iter         = 150000,
R>   BurninThin   = c(burnin = 50000,
R>                       thin   = 10),
R>   accept_ratio = 0.234,
R>   adapt_param  = c(start = 1,
R>                       end   = 1000,
R>                       exp   = 0.5),
R>   n_chains     = 2,
R>   parallel     = TRUE,
R>   n_cores      = 2
R>   )
```

Convergence is achieved in a reasonable time, in both runs (Figure 8)
We move again to interpolation on the validation set

```
R> Pred.krig_PNSpt <- ProjKrigSpTi(
R>   ProjSpTi_out = mod,
R>   coords_obs   =  coords[-val,],
R>   coords_nobs  =  coords[val,],
R>   times_obs    =  coordsT[-val],
R>   times_nobs   =  coordsT[val],
R>   x_obs        = theta[-val]
R>   )
```

And again we can check results using APE (average = 0.38, range= (0.057,0.813)) and CRPS (average = 0.21, range= (0.007, 0.66)).

5. Concluding Remarks

In this paper we presented the **CircSpaceTime** package, that collects functions implementing spatial and spatio-temporal Bayesian models allowing interpolation of dependent circular data. The implemented models have been introduced in a series of papers that are briefly summarized here (see Jona Lasinio *et al.* 2012; Wang and Gelfand 2013, 2014; Mastrantonio *et al.* 2016b, for details). The current version of the package do not include models with nugget and allows for constant mean only. Future versions will include nugget and regression-type mean. We also plan to include more models for dependent circular data spreading from the works Bulla, Lagona, Maruotti, and Picone (2012), Lagona and Picone (2012), Bulla,

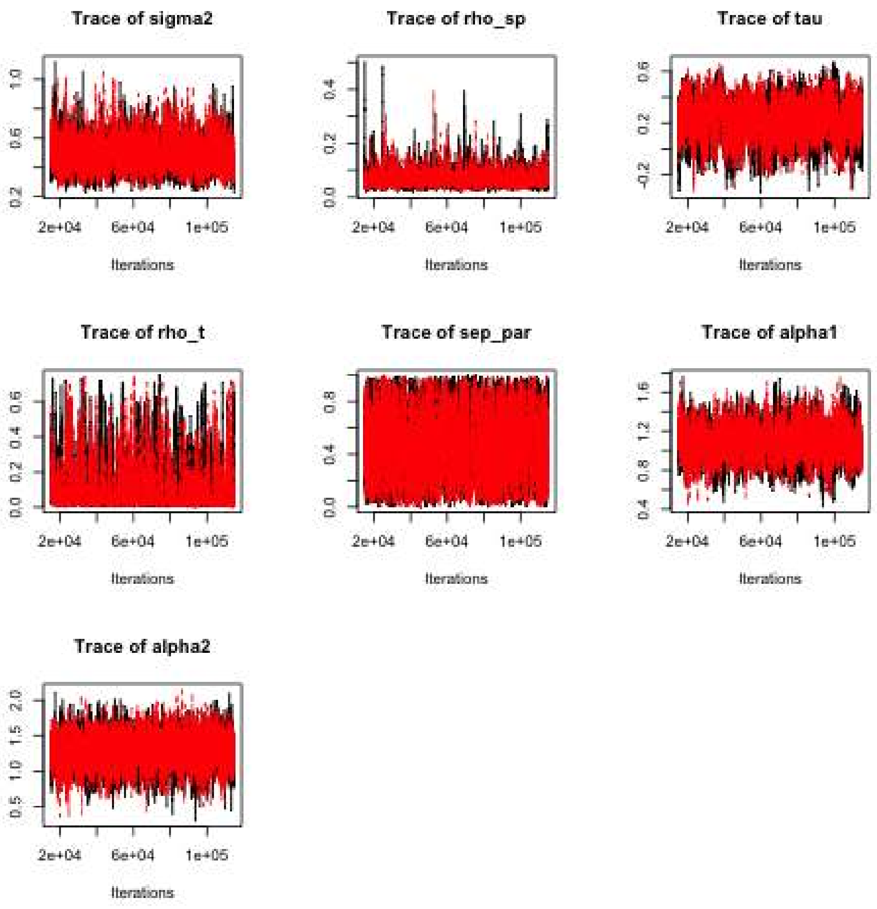

Figure 8: Trace plots of spatio temporal PN example

Lagona, Maruotti, and Picone (2015), Lagona, Picone, Maruotti, and Cosoli (2015) and Maruotti, Punzo, Mastrantonio, and Lagona. (2016), that develop likelihood based estimation for complex modeling of circular and cilindrical data.

Acknowledgments

This work has been partially developed under the PRIN2015 supported-project *Environmental processes and human activities: capturing their interactions via statistical methods* (EPHA-Stat) funded by MIUR (Italian Ministry of Education, University and Scientific Research) (20154X8K23-SH3). Gianluca Mastrantonio research has been partially supported by MIUR grant Dipartimenti di Eccellenza 2018 - 2022 (E11G18000350001).

Affiliation:

Giovanna Jona Lasinio
Sapienza University of Rome
P.le Aldo Moro 5, 00185 Rome, ITALY
E-mail: `giovanna.jonalasinio@uniroma1.it`

Mario Santoro
IAC CNR
Via dei Taurini, 19, 00185 Rome, ITALY
E-mail: `m.santoro@iac.cnr.it`

Gianluca Mastrantonio
Polytechnic of Turin, ITALY
Corso Duca degli Abruzzi, 24, 10129 Turin, ITALY
E-mail: `gianluca.mastrantonio@polito.it`

Journal of Statistical Software `http://www.jstatsoft.org/`
published by the Foundation for Open Access Statistics `http://www.foastat.org/`
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
`doi:10.18637/jss.v000.i00` *Accepted:* yyyy-mm-dd