Appendix J

R Package Documentation

J.1 Introduction and Installation

J.1.1 Introduction

Vectors and periodic phenomena (e.g., traffic concentration vs. time) have direction on a circular scale of 360º. Circular-spatial data consists of location and direction. Circular random fields generate circular-spatial data. A circular RF (CRF) is defined as a space containing circular random variables (CRV) with spatial correlation. In 2 dimensional space, the CRF is the set $\{\Theta(\boldsymbol{x}), \boldsymbol{x} \in \mathrm{R}^2\}$ with Θ the circular RV and $\boldsymbol{x}$ the location. A CRV takes random directions on a unit circle with the total probability of all possible directions distributed on the unit circle with support $[0, 2\pi)$ or $[-\pi, \pi)$. The starting point of the support is the same direction as the ending point. A CRV is expressed as either a scaler in units of radians or degrees (º), or as a unit vector. Spatial correlation increases as distance between measurement locations decreases, i.e., rotations from the mean direction tend to be more similar. In the form required by the circular kriging solution of chapter 4, spatial correlation is defined as the mean cosine of the angle between random components of directions (nonrandom component removed) vs. distance between measurement locations. An isotropic CRF is defined as a CRF in which spatial correlation is the same in all directions in space. A geometric anisotropic CRF is defined as a CRF in which spatial correlation varies with direction in space.

The main functions of package CircSpatial include:

- SimulateCRF: Simulate a circular random field (CRF). (Appendix J.2)

- CircResidual: Compute rotational residuals (deviations) from the mean direction in radians. (Appendix J.3)
- CosinePlots: Plot the empirical and fitted cosineograms of the spatial correlation of circular-spatial data. (Appendix J.4)
- KrigCRF: Krig and smooth the circular-spatial residuals. (Appendix J.5)
- InterpDirection: Interpolate the model of mean direction at each kriging location. (Appendix J.6)
- CircDataimage: Generate a graphical user interface (GUI) for interactive circular dataimages. (Appendix J.10)
- PlotVectors: Traditional plots of vector-spatial data. (Appendix J.11)

J.1.2 PC Windows Installation

CircSpatial depends on R packages CircStats, fields, geoR, RandomFields, sp, and spam. The R console menu item Packages item “Install package(s) ...” provides for loading the dependent packages, CircSpatial, and the CircSpatial documentation. To use CircSpatial:

1) Install the current version of Active State Tcl from http://downloads.activestate.com/ActiveTcl/Windows/ to the root directory C:\ (recommended). This installation is required to use the Tcl command img to add jpeg graphics to the CircSpatial function CircDataimage GUI.
2) Install CircSpatial

3) Before using the CircSpatial function CircDataimage, tell R where to find the special Tcl command img. Update the highlighted input following according to your user installation path of Active State TCL and enter the following commands from the R prompt:

```
require(tcltk, quietly=TRUE, warn.conflicts=TRUE)
Sys.setenv("TCL_LIBRARY"="C:/Tcl/lib/tcl8.5")
Sys.setenv("MY_TCLTK"="Yes")
addTclPath(path = "C:/Tcl/lib/teapot/package/win32-ix86/lib")
tclRequire("img::jpeg")
```

These commands were not embedded in the build of CircSpatial to make the build independent of the user installation path and version of Active State Tcl.

4) At the R prompt, load CircSpatial.

J.2 SimulateCRF

Generate an isotropic (depends on distance only) or geometric anisotropic (depends on distance and direction) CRF in a plane. An isotropic CRF is a space of circular random variables (CRV) with spatial correlation only dependent on distance. CRFs are implemented for the circular distributions uniform (U), triangular (Tri), cardioid (Card), von Mises (vM), or wrapped Cauchy (WrC), with mean resultant direction (mu) of zero (default), and specified mean resultant length (rho) and range (distance at which CRV are uncorrelated) of spatial correlation. Spatial correlation means that as the distance between measurement locations decreases, random rotations of direction from the mean direction tend to be more similar.

J.2.1 Principle

As shown in Figure J-1, the CRF is obtained by transforming a Gaussian random field (GRF), which is generated by package geoR or Random Fields (if N > 500), via the inverse cumulative distribution function (CDF) of the CRV operating on the CDF of the GRV, which in turn, operates on observations of the GRV in the GRF. Because spatially dependent random variables tend to be similar at short distances and the CDF is monotonic increasing, their cumulative probabilities also tend to be similar. Thus, spatial correlation is transformed from the GRF to the normal cumulative probabilities as shown in Figure J-2. In turn, similarity of cumulative probabilities transforms to similarity of CRV via the monotonic inverse CDF of the CRV. Details of the GRF are available in the Help files of packages geoR and RandomFields. The direction of the inverse circular CDF is obtained via interpolation of the circular CDF on a fine scale. For additional information, see Chapter 5, Simulation Of A Circular Random Field.

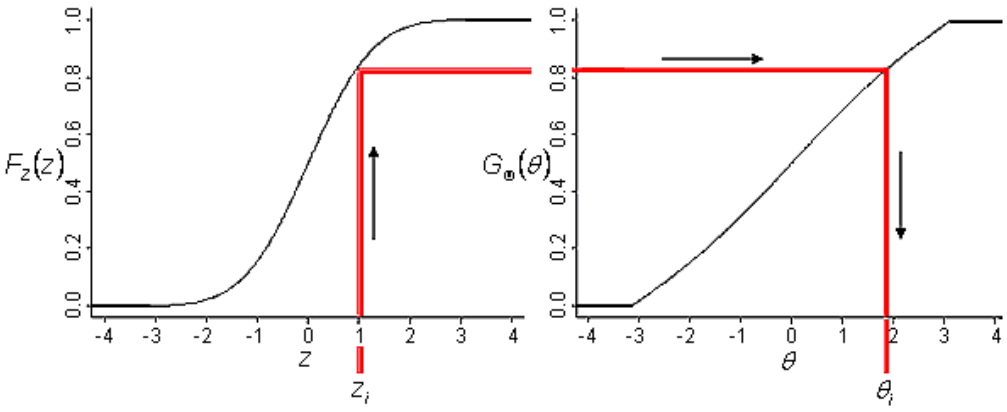


Figure J-1. Mapping a GRF to a CRF via CDFs. $\theta_i = G_\Theta^{-1}\{F_Z(z_i);\rho\}$. Direction of Θ is expressed in radian units.

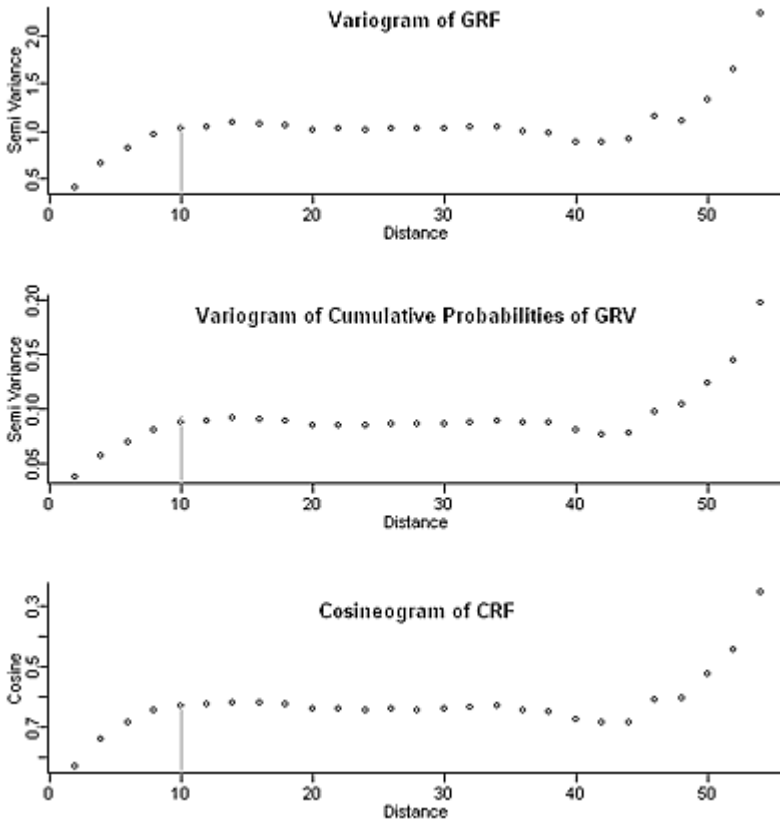


Figure J-2. Shapes of Variograms and Inverted Cosineogram Show Spatial Correlation

Transformed from the GRF with Spherical Covariance and Range $r = 10$.

J.2.2 Usage and Input Arguments

SimulateCRF(N=100, CircDistr, Rho, Mu=0, Range, Ext=1, CovModel, Grid=NULL, Anisotropy=NULL, OverFit=FALSE, Resolution=.01)

N: The number of observations to simulate.

CircDistr: Name of circular distribution “Card”, “Tri”, “U”, “vM”, or “WrC”. Circular random fields are implemented for the uniform, von Mises, cardioid, triangular, and wrapped Cauchy distributions.

Rho: Mean resultant length parameter ρ of the circular distribution: Wrapped Cauchy, $0 < \rho < 1$; von Mises, $0 < \rho < 1$; cardioid, $0 < \rho \le 0.5$; triangular, $0 < \rho \le 4/\pi^2$; uniform, $\rho = 0$ (all directions have equal density)

Mu: Mean resultant direction of circular distribution (rad). $|\mu| \le \pi$.

Range: Distance at which CRV are not correlated.

Ext: Ext >= 1. Range x Ext (Extension) is the horizontal and vertical width of square sample space. Simulation at distances beyond the range (distance at which CRV are no longer spatially dependent) reduces edge effect at the range resulting in a more accurate representation of the sill (mean cosine of independent CRV) near the range.

CovModel: Name of a spatial correlation function from package geoR function cov.spatial, e.g., “exponential”.

Grid: An n x 2 matrix of regular or irregular location coordinates of the simulated data. Grid overrides N and Ext.

Anisotropy: Vector of geometric anisotropy (angle, ratio). Angle in radians, ratio ≥ 1. See R Help for package geoR function grf.

OverFit: If TRUE, realizations of the GRV are standardized (centered to mean 0 and scaled to standard deviation 1) prior to the transformation to a CRF. Standardization stabilizes realizations of the GRV, enhancing the fit of the output CRF to the specified circular distribution. Standardization is suitable for demonstration with closer fit, visualization, and illustrations. Undesirable effects include loss of independence of the marginal GRVs, biased GRF covariance, and biased testing. If FALSE (default), the realizations are not standardized. Non standardization includes expected variation from transformation of variation in mean and standard deviation of the realization of the GRV of the GRF. OverFit=FALSE is recommended for the purposes of simulation, analysis, and testing.

Resolution: For nonclosed form inverse CDF, circular quantiles are interpolated at resolution Resolution. $0.001 \leq$ Resolution ≤ 0.01 recommended.

J.2.3 Output List

x: Vector of x coordinates of simulated observations.

y: Vector of y coordinates of simulated observations.

direction: Vector of direction of simulated observations in radians. To change mean direction, add a constant.

Z: Vector of simulated observations of the GRV of the GRF.

J.2.4 Example

```
## Compute isotropic vM CRF of 121 observations, Rho=sqrt(0.5) so sill about 0.5,
## from GRF (Range=4, spherical covariance) with OverFit=TRUE for demonstration.

require(CircSpatial)
x1<- 1:11; y1 <- 1:11; y1 <- rep(y1, 11); x1 <- rep(x1, each=11)
set.seed(666)
```

```
crf1<- SimulateCRF(CircDistr="vM", Rho=sqrt(0.5), Range=4, CovModel="spherical",
          Grid=cbind(x1, y1))
plot(crf1$x, crf1$y, type="n", xlab="", ylab="", asp=1)
arrow.plot(a1=crf1$x, a2=crf1$y, u=cos(crf1$direction), v=sin(crf1$direction), arrow.ex=0.1,
     xpd=TRUE, true.angle=TRUE, length=.1)
```

J.2.5 Additional Examples

```
## Compute isotropic Cardioid CRF of 200 observations, Rho=0.4 so sill about 0.16,
## from GRF(exponential covariance, Range=5)
crf2 <- SimulateCRF(N=200, CircDistr="Card", Rho=0.4, Range=5, Ext=3,
     CovModel="exponential")

## Compute isotropic uniform CRF of 100 observations, sill about 0,
## from GRF(Gaussian covariance, Range=8)
crf3 <- SimulateCRF(CircDistr="U", Range=8, Ext=3, CovModel="gaussian")

## Compute isotropic triangular CRF of 100 observations, sill about 0.04,
## from GRF(spherical covariance, Range=8)
crf4 <- SimulateCRF(CircDistr="Tri", Rho=0.5*4/pi^2, Range=8, Ext=3,
     CovModel="spherical")

## Compute isotropic wrapped Cauchy CRF of 100 observations, sill about 0.8,
## from GRF(exponential covariance, Range=8)
crf5 <- SimulateCRF(CircDistr="WrC", Rho=sqrt(0.8), Range=8, Ext=3,
     CovModel="exponential")

## Compute anisotropic wrapped Cauchy CRF of 400 observations, sill about 0.95,
## from GRF(spherical covariance, Range=8) with anisotropy angle pi/4 and ratio 3
crfaniso <- SimulateCRF(N=400, CircDistr="WrC", Rho=sqrt(0.95), Range=8, Ext=3,
     CovModel="spherical",  Anisotropy=c(pi/4, 3))
```

J.3 CircResidual

The first order trend, if any, must be removed from the data via an appropriate fitted model. Separately fit the cosine and sine components of direction to functions of the spatial coordinates to avoid the cross over problem (direction of 0º equals direction of 360º). Then, the fitted direction is obtained using R function atan2(fitted sines, fitted cosines). Spatial correlation is encoded in the residual rotations = the rotation in radians from the fitted model direction to the data direction. A positive residual rotation indicates that counterclockwise (CCW) rotation is required to rotate the fitted model direction to the data direction. A negative residual rotation indicates that CW rotation is required. CircResidual returns the residuals in radians, or plots data, model, and residuals with black, thick tan, and dashed red arrows, respectively. Figures J-3 (a) to (e) show the succession of model, CRF, sample, fitted model, and residual directions.

J.3.1 Usage and Input Arguments

CircResidual(X, Y, Raw, Trend, Plot=FALSE, AdjArrowLength=1, ...)

X: Vector of horizontal coordinates of observation and trend locations.

Y: Vector of vertical coordinates of observation and trend locations.

Raw: Vector of direction of observations in radians.

Trend: Vector of fitted direction in radians. NAs not allowed.

Plot: If FALSE return output list. If TRUE, plot data (black), model (tan), and residuals (dashed black) with asp=1.

AdjArrowLength: Multiplies length of arrows in plots.

. . . : Additional plot parameters.

J.3.2 Output List

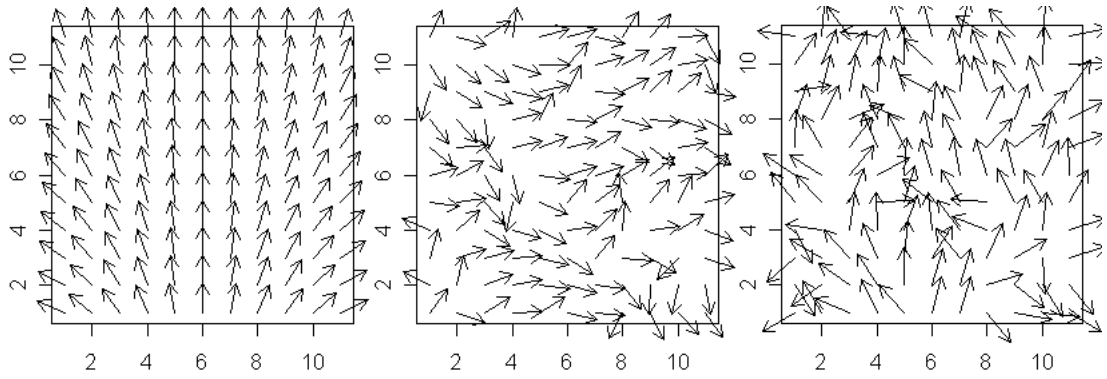
x: Vector of horizontal coordinates of residuals.

y: Vector of vertical coordinates of residuals.

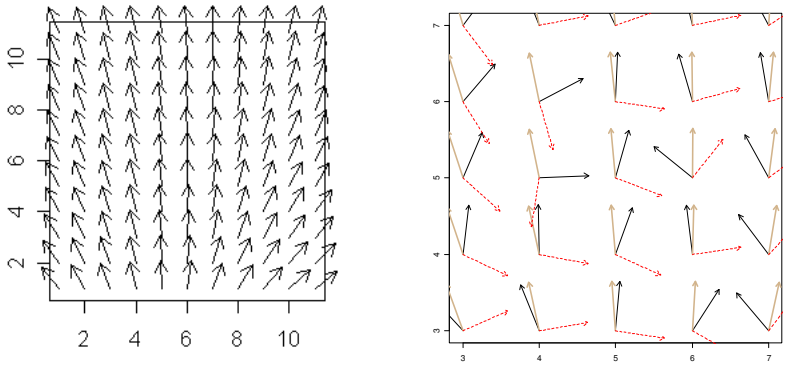
direction: Vector of direction residuals in radians.

J.3.3 Examples

```
require(CircSpatial)
## Construct Trend Model of 121 locations
x1<- 1:11; y1 <- 1:11; y1 <- rep(y1, 11); x1 <- rep(x1, each=11)
model.direction1 <- matrix(data=c(
   157, 141, 126, 113, 101, 90, 79, 67, 54, 40, 25, 152, 137, 123, 111, 100, 90, 80, 69, 57, 44, 30,
   147, 133, 120, 109,  99, 90, 81, 71, 60, 48, 35, 142, 129, 117, 107,  98, 90, 82, 73, 63, 52, 40,
   137, 125, 114, 105,  97, 90, 83, 75, 66, 56, 45, 132, 121, 111, 103,  96, 90, 84, 77, 69, 60, 50,
   127, 117, 108, 101,  95, 90, 85, 79, 72, 64, 55, 122, 113, 105,  99,  94, 90, 86, 81, 75, 68, 60,
   117, 109, 102,  97,  93, 90, 87, 83, 78, 72, 65, 112, 105,  99,  95,  92, 90, 88, 85, 81, 76, 70,
   107, 101,  96,  93,  91, 90, 89, 87, 84, 80, 75), ncol=11, byrow=TRUE)
model.direction1 <- as.vector(model.direction1)*pi/180

## Plot Trend Model, See Figure J-3 (a)
plot(x1, y1, type="n", xlab="", ylab="", asp=1)
arrow.plot(x1, y1, u=cos(model.direction1), v=sin(model.direction1), arrow.ex=0.1, xpd=TRUE,
          true.angle=TRUE,  length=.1)

## Compute vM CRF of 121 observations, Rho=sqrt(0.5) so sill about 0.5,
## from GRF (Range=4, spherical covariance).
set.seed(666)
crf1<- SimulateCRF(CircDistr="vM", Rho=sqrt(0.5), Range=4, CovModel="spherical",
          Grid=cbind(x1, y1), OverFit=TRUE)

## Plot CRF, See Figure J-3 (b)
par(mai=c(0.4, 0.35, .25, 0.25))
plot(crf1$x, crf1$y, type="n", xlab="", ylab="", asp=1)
arrow.plot(a1=crf1$x, a2=crf1$y, u=cos(crf1$direction), v=sin(crf1$direction), arrow.ex=0.1,
          xpd=TRUE, true.angle=TRUE, length=.1)

# Make sample
sample.direction1 <- model.direction1 + crf1$direction

## Plot Sample, See Figure J-3 (c)
sample.direction1 <- model.direction1 + crf1$direction
plot(x1, y1, type="n", asp=1)
arrow.plot(a1=x1, a2=y1, u=cos(sample.direction1), v=sin(sample.direction1), arrow.ex=0.125,
     xpd=TRUE, true.angle=TRUE, length=.1)

## Fit An Appropriate Model
## Code for median polish is contained in Appendix K, Section K.12
FitHoriz1 <- lm(cos(sample.direction1) ~ (x1 + y1))
FitVert1 <- lm(sin(sample.direction1)  ~ (x1 + y1))
fitted.direction1 <- atan2(FitVert1$fitted.values, FitHoriz1$fitted.values)
```



(a) True Model (b) Simulated CRF (c) Data



(d) Fitted Model (e) Data, Fitted Model, And Residual

Figure J-3. Plots of True Model, Simulated CRF, Data, Fitted Model, and Residuals. In (e), the data black, the fitted model labeled is tan, and residuals are dashed red. A negative residual rotation = data – model is a clockwise rotation from the model to the data.

```
## Plot Fitted Model, See Figure J-3 (d)
plot(x1, y1, type="n", asp=1, xlab="", ylab="")
arrow.plot(x1, y1, u=cos(fitted.direction1), v=sin(fitted.direction1), arrow.ex=0.1, xpd=TRUE,
     true.angle=TRUE,  length=.1)
## The estimated model in Figure J.3 (d) well approximates true model in Figure J.3 (a).

## Compute Residuals
resids1 <- CircResidual(X=x1, Y=y1, Raw=sample.direction1, Trend=fitted.direction1,
Plot=FALSE)

## Plot Sample, Fitted Model, and Residual Rotations, See Figure J-3 (e)
CircResidual(X=x1, Y=y1, Raw=sample.direction1, Trend=fitted.direction1, Plot=TRUE,
     xlim=c(3,7), ylim=c(3,7))
```

J.4 CosinePlots

The empirical omnidirectional cosineogram expresses the spatial correlation of an isotropic CRF (spatial correlation depends on distance not direction) as the mean cosine of the angular distances (Figure J-4) between random components of direction as a function of the linear distance between measurement locations.

J.4.1 Definitions

The nugget, range, sill, and cosine model, as shown in Figure J-5, are incorporated into the circular kriging solution. At the distance of zero, the mean cosine is 1. Close to 0 distance , the mean cosine is 1 minus the nugget. As distance increases spatial correlation decreases. This is reflected in a decreasing mean cosine. The range is a scale parameter. For the spherical covariance function, it is the distance at which rotations are no longer spatially dependent. The sill is the mean cosine at distances where CRV are not correlated. Theoretically, the sill is the square of the mean resultant length of the circular probability distribution (See Chapter 3, Section 3.3, for an extensive derivation of this result).

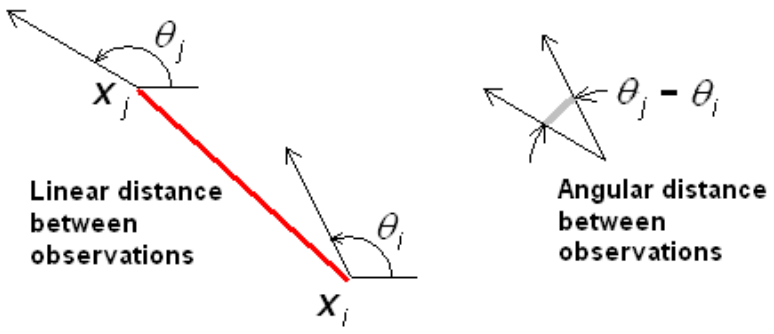


Figure J-4. Distance Between Locations (Red) vs. Angular Distance (Grey) Between Observations.

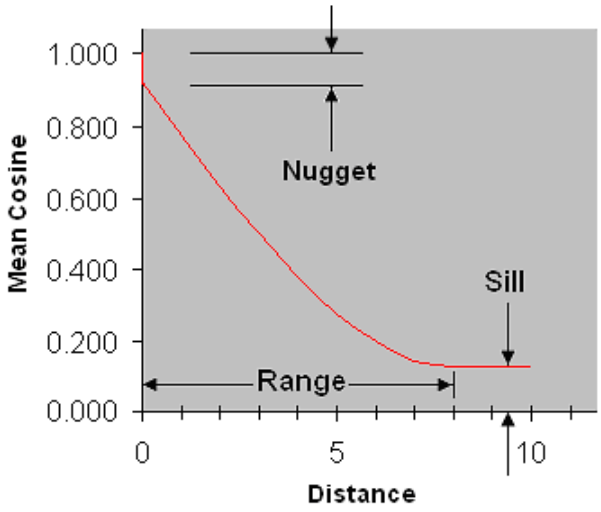


Figure J-5. Features of the Cosineogram Model. The cosineogram characterizes the correlation of random components of direction relative to distance between locations.

The cosineocloud plot, Figure J-6, which derives its name from the variocloud plot for linear kriging, may be useful to identify outliers that may be excluded from subsequent calculations. It shows all the cosine values computed from all pairs of directional observations.

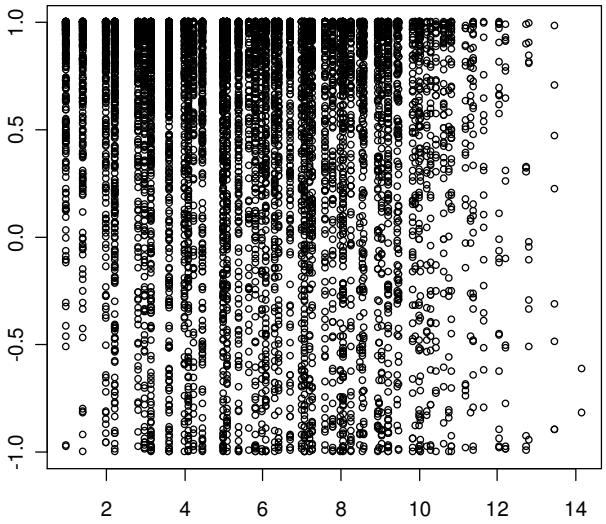


Figure J-6. Cosineocloud. The cosineocloud is a plot of cosines of pairs of random components of direction vs. distance between pairs. It is a diagnostic useful to identify outliers.

The empirical omnidirectional cosineogram captures spatial correlation of directional data or rotational residuals (when the first order trend is removed) as $\hat{\varsigma}(distance) = 1/n(distance)\sum_{i=1}^{n(distance)} \cos(\Delta_i)$ with Δ_i the circular distance between the i^{th} pair of residual rotations. Typically, range and sill are determined visually. The nugget may be determined by linear regression of initial points of the empirical cosineogram. Figure J-7 indicates a range of about 4 and a sill of about 0.5, which is consistent with the CRF (CircDistr="vM", Rho=sqrt(0.5), Range=4) with the sill $= \rho^2$. The number of lag points and bin width of the cosineogram are described in detail in Subsection J.4.5.

J.4.2 Cosine Models

The empirical cosineogram may be overplotted with exponential, Gaussian, and spherical cosine models to help fit a model. The cosine model is a monotonic decreasing function of increasing distance between measurement locations up to the range, and produces a positive definite matrix of cosines, which is required for circular kriging, when applied to the pairwise distances between measurement locations.

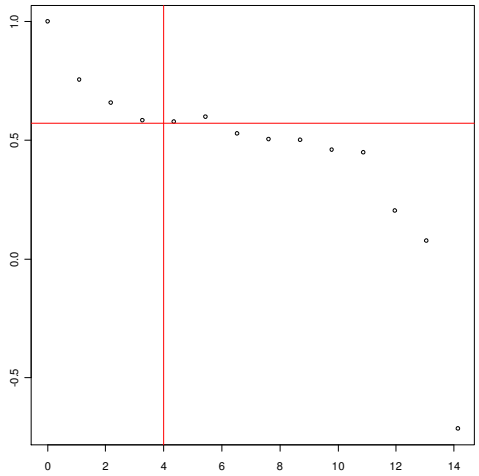


Figure J-7. Empirical Cosineogram. It reflects the circular-spatial correlation with range (distance at which cosine plateaus) of about 4 and sill (elevation of plateau) of about 0.5.

Cosineogram models, which were adapted from covariance models for linear kriging (Chapter 3, Section 3.6), have mean cosine = 1 at distance = 0, and sill $= \rho^2$. With $\varsigma(d) =$ mean cosine function of *d*, *d* = linear distance between measurement locations, ρ the mean resultant length of the circular probability distribution, n_g the nugget ($0 \le n_g \le 1-\rho^2$), and *r* = range, the implemented cosine models are:

Exponential (3.12):

$$\varsigma(d) = \begin{cases} 1, d = 0 \\ \rho^2 + (1 - n_g - \rho^2)\exp(-3\,d/r), d > 0 \end{cases} \quad \text{(J.1)}$$

Gaussian (3.13):

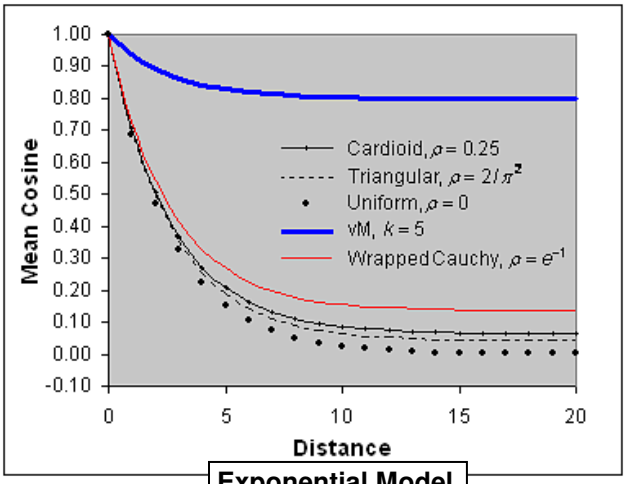
$$\varsigma(d) = \begin{cases} 1, d = 0 \\ \rho^2 + (1 - n_g - \rho^2)\exp\left(-3\,[d/r]^2\right), d > 0 \end{cases} \quad \text{(J.2)}$$

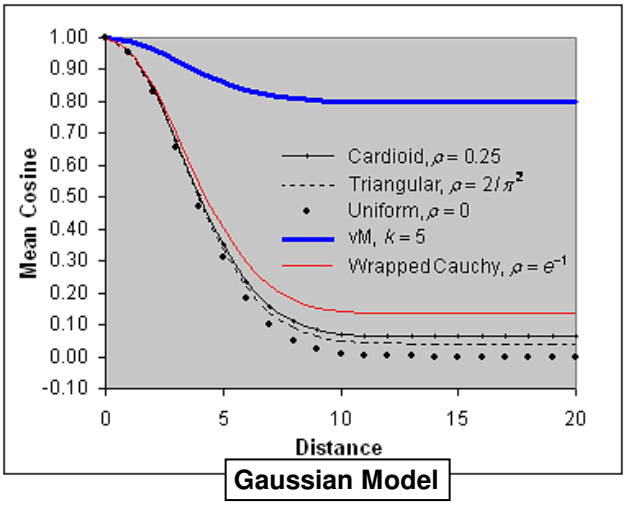
Spherical (3.14):

$$\varsigma(d) = \begin{cases} 1, d = 0 \\ 1 - n_g - (1 - n_g - \rho^2)\left(\frac{3}{2}[d/r] - \frac{1}{2}[d/r]^3\right), \quad 0 < d \le r \\ \rho^2, \quad d > r. \end{cases} \quad \text{(J.3)}$$

Plots of the implemented cosine models are shown in Figure J-8. Additional suitable functions are given in Appendix M, Section M.5.

J.4.3 Cosine Model Plots





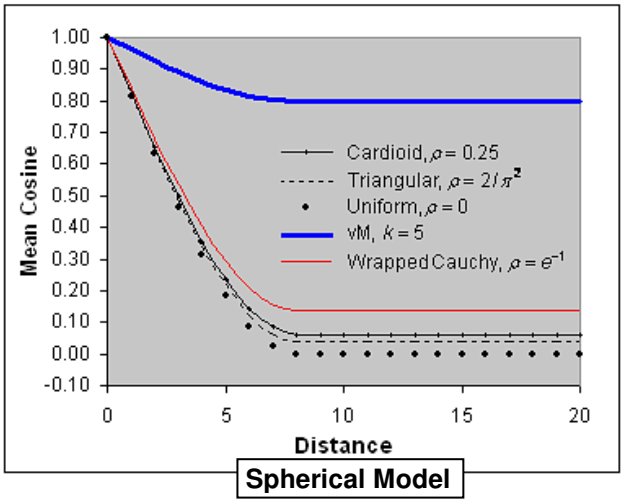


Figure J-8. Cosine Models for Circular-Spatial Data, Range $r = 8$.

J.4.4 Usage and Input Arguments

CosinePlots(x, y, directions, Lag=NULL, Lag.n.Adj=1, BinWAdj=1, Plot=TRUE, Cloud=FALSE, Model=FALSE, nugget=0, Range =NULL, sill =NULL, x.legend=0.6, y.legend=1.0, TrimMean=0.1, ...)

x: Vector of horizontal coordinates of observations.

y: Vector of vertical coordinates of observations.

directions: Vector of direction of observations or residual rotations in radians.

Lag: Vector of ascending distances, beginning with zero, where mean cosine is to be computed.

Lag.n.Adj: Multiplier (> 0) of the number of lag points. Value > 1 increases the number of points for more detail. Value < 1 decreases the number of points for less detail.

BinWAdj: Multiplier (≥ 1) of bin width. Value > 1 has a smoothing effect.

Plot: See Table J-1.

Cloud: See Table J-1.

Model: See Table J-1.

nugget: Model nugget or mean cosine near zero distance. $0 \leq \text{nugget} \leq 1$.

Range: Model range.

sill: Model sill.

x.legend: Model plot legend horizontal location as fraction of horizontal maximum coordinate.

y.legend: Model plot legend vertical coordinate.

TrimMean: Apply trimmed mean (0.0 to 0.5) in computing the mean cosine at a distance. See R Help for mean.

. . . : Additional plotting parameters.

J.4.5 Binning Details

The number of lag points (distances at which mean cosine is evaluated) and distance bin width are determined in a sequence:

1) Sturges rule determines nBins, the number of bins.

2) nBins and Lag.n.Adj determine Lag.n, the number of distances (lag points) to evaluate the mean cosine.

3) nBins is adjusted to Lag.n - 1 (To make bins narrower increase Lag.n.adj.).

4) nBins and BinWAdj determine bin width.

J.4.6 Output

Typically, range and sill are determined visually. When Plot=Model=TRUE, vary the range and sill parameters to fit. The nugget may be determined by linear regression of initial empirical cosineogram points.

Table J-1. Output of CosinePlots.

Plot	Cloud	Model	Value
FALSE	TRUE	FALSE	List of cosineocloud coordinates
FALSE	FALSE	FALSE	List of cosineogram coordinates
TRUE	TRUE	FALSE	Cosineocloud Plot
TRUE	FALSE	FALSE	Cosineogram Plot
TRUE	FALSE	TRUE	Cosineogram overplotted with cosine models

J.4.7 Examples

green highlighted code same as section J.3.3

```
require(CircSpatial)
## Construct Trend Model of 121 locations
x1<- 1:11; y1 <- 1:11; y1 <- rep(y1, 11); x1 <- rep(x1, each=11)
model.direction1 <- matrix(data=c(
   157, 141, 126, 113, 101, 90, 79, 67, 54, 40, 25, 152, 137, 123, 111, 100, 90, 80, 69, 57, 44, 30,
   147, 133, 120, 109,  99, 90, 81, 71, 60, 48, 35, 142, 129, 117, 107,  98, 90, 82, 73, 63, 52, 40,
   137, 125, 114, 105,  97, 90, 83, 75, 66, 56, 45, 132, 121, 111, 103,  96, 90, 84, 77, 69, 60, 50,
   127, 117, 108, 101,  95, 90, 85, 79, 72, 64, 55, 122, 113, 105,  99,  94, 90, 86, 81, 75, 68, 60,
   117, 109, 102,  97,  93, 90, 87, 83, 78, 72, 65, 112, 105,  99,  95,  92, 90, 88, 85, 81, 76, 70,
   107, 101,  96,  93,  91, 90, 89, 87, 84, 80, 75), ncol=11, byrow=TRUE)
model.direction1 <- as.vector(model.direction1)*pi/180

## Compute vM CRF of 121 observations, Rho=sqrt(0.5) so sill about 0.5,
## from GRF (Range=4, spherical covariance).
set.seed(666)
crf1<- SimulateCRF(CircDistr="vM", Rho=sqrt(0.5), Range=4, CovModel="spherical",
      Grid=cbind(x1, y1), OverFit=TRUE)

# Make sample
sample.direction1 <- model.direction1 + crf1$direction

## Fit An Appropriate Model
## Code for median polish is contained in Appendix K, Section K.12
FitHoriz1 <- lm(cos(sample.direction1) ~ (x1 + y1))
FitVert1 <- lm(sin(sample.direction1)  ~ (x1 + y1))
fitted.direction1 <- atan2(FitVert1$fitted.values, FitHoriz1$fitted.values)

## Compute Residuals
resids1 <- CircResidual(X=x1, Y=y1, Raw=sample.direction1, Trend=fitted.direction1,
      Plot=FALSE)

## Output list of cosineogram coordinates for fitting analytically
cosineogram.out <- CosinePlots(x=resids1$x, y=resids1$y, directions=resids1$direction,
      Lag.n.Adj=1, BinWAdj=1, Plot=FALSE, Cloud=FALSE,  Model=FALSE)

## Cosineocloud, Figure J-6
CosinePlots(x=resids1$x, y=resids1$y, directions=resids1$direction, Lag.n.Adj=1, BinWAdj=1,
      Plot=TRUE, Cloud=TRUE)

## Cosineogram, Figure J-7.
CosinePlots(x=resids1$x, y=resids1$y, directions=resids1$direction, Lag.n.Adj=1, BinWAdj=1,
      Plot=TRUE, Cloud=FALSE,  Model=FALSE)
abline(h=0.56, col=2); abline(v=4, col=2)

## Fit cosine Models, Figure J-9
CosinePlots(x=resids1$x, y=resids1$y, directions=resids1$direction, Lag.n.Adj=1, BinWAdj=1,
      Plot=TRUE, Cloud=FALSE, Model=TRUE,  nugget=0, Range=4.0,  sill=0.56, x.legend=.2,
      y.legend=0.3, xlim=c(0,8), ylim=c(0,1))
```

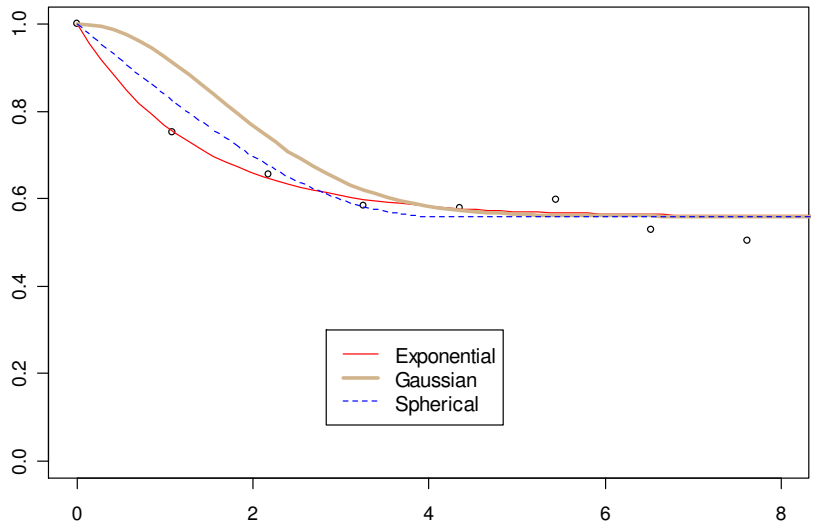


Figure J-9. Fitted Cosine Models. The exponential model with nugget 0.0, range 4.0 and a sill 0.56 has an adequate fit to the data at distances up to the range. The cosine may decrease or increase beyond the range.

J.5 KrigCRF

KrigCRF estimates circular-spatial data of an isotropic CRF with "exact interpolation" (estimate equals observed at measured locations). The solution is a linear combination of directional observations or residual rotations (when the first order trend is removed) that minimizes the squared length of the error vector. This is accomplished by incorporation of the spatial correlation (nugget, range, sill, and cosine model) as estimated by the cosineogram model fitted to the cosineogram. The circular kriging variance is an estimate of the mean squared length of the error vector. To avoid cross over, KrigCRF also separately applies image.smooth of R package fields to the horizontal and vertical components of kriged direction. The smoothed kriged direction is quadrant specific inverse tangent of the smoothed components. The circular kriging estimate can be combined with an interpolation of the fitted model. For additional information, see Chapter 4, Circular Kriging.

J.5.1 Solutions

With $\boldsymbol{U}$ the matrix of unit vector observations, row 1 the cosines, row 2 the sines, and one observation per column, $\boldsymbol{K}^{-1}$ the inverse of matrix of model cosines dependent on pairwise distances between measurement locations, and $\boldsymbol{c}$ the vector of model cosines dependent on distance between estimation location 0 and measurement locations 1, 2, 3, ..., n, the solutions are:

$\hat{\boldsymbol{u}}_0$: Estimated direction as a unit vector, $\hat{\boldsymbol{u}}_0 = \operatorname{atan2}\left(\left(\boldsymbol{U}\boldsymbol{K}^{-1}\boldsymbol{c}\right)[2], \left(\boldsymbol{U}\boldsymbol{K}^{-1}\boldsymbol{c}\right)[1]\right)$

$\hat{\sigma}^2_{CK}$: Circular kriging error variability, $\hat{\sigma}^2_{CK} = 2 - 2\sqrt{\boldsymbol{c}^T\boldsymbol{K}^{-1}\boldsymbol{c}}$.

J.5.2 Usage and Input Arguments

KrigCRF(krig.x, krig.y, resid.x, resid.y, resid.direction, Model, Nugget=0, Range, sill, Smooth=FALSE, bandwidth, Plot=FALSE, PlotVar=FALSE, Xlim=NULL, Ylim=NULL, ...)

krig.x: Vector of horizontal coordinates of kriging locations.

krig.y: Vector of vertical coordinates of kriging locations corresponding to krig.x.

resid.x: Vector of horizontal coordinates of rotational residuals or data.

resid.y: Vector of vertical coordinates of rotational residuals or data.

resid.direction: Vector of direction in radians of rotational residuals or data.

Model: Covariance model of R package RandomFields function CovarianceFct best fitting the empirical cosineogram.

Nugget: 1 - mean cosine at distance close to zero due to measurement error, micro scale variation or sampling.

Range: Distance at which spatially correlated CRV are not correlated for the spherical model, or scale factor of other models.

sill: Mean cosine at the Range.

Plot: See Table J-2. Xlim and Ylim are the plot limits.

PlotVar: Plot circular kriging variance. See Table J-2.

Smooth: See Table J-2.

bandwidth: Kernel smoothing bandwidth (>0).

... : Additional model parameters.

J.5.3 Output

Table J-2. Output of KrigCRF.

Plot	PlotVar	Smooth	Value
FALSE	TRUE/FALSE	FALSE	Vectors of x, y, and kriged direction (rad)
FALSE	TRUE/FALSE	TRUE	Vectors of x, y, and smoothed direction
TRUE	FALSE	FALSE	Arrow plot of kriged direction
TRUE	FALSE	TRUE	Arrow plot of smoothed kriged direction
TRUE	TRUE	TRUE/FALSE	Filled contour plot of circular kriging error

J.5.4 Examples

```
require(CircSpatial)
## Using the residuals resids1 from Subsection J.4.7
x2 <- seq(1,11, by=0.2); y2 <- x2 ## Kriging locations

## Krig to residuals using range and sill estimate from Figure J-9
krig2 <- KrigCRF(krig.x=x2, krig.y=y2, resid.x=resids1$x, resid.y=resids1$y,
     resid.direction=resids1$direction,  Model="exponential", Nugget=0.0,  Range=4, sill=0.56,
     Plot=FALSE)

## Plot Kriging, Residuals Overploted In Black, See Figure J-10
require(fields)
plot(krig2$x, krig2$y, ty="n", xlab="", ylab="", xlim=c(5, 8), ylim=c(5, 8), asp=1)
arrow.plot(krig2$x, krig2$y, u = cos(krig2$direction), v = sin(krig2$direction), arrow.ex = 0.06,
     xpd=FALSE, true.angle = TRUE, length=.05, col="tan")
arrow.plot(resids1$x, resids1$y, u = cos(resids1$direction), v = sin(resids1$direction), arrow.ex =
     0.09, xpd=FALSE, true.angle = TRUE, length=.05, col=1)
```

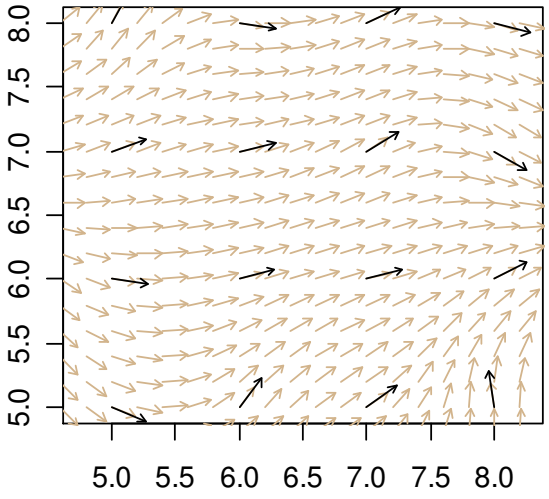


Figure J-10. Residual Rotations (Black) Overplotted on the Circular Kriging (Tan). The kriged direction appears to be equal to the residual direction at sample locations. This is called “exact interpolation”.

Figure J-11 shows that increasing the nugget smoothes the kriging component of estimated direction at unsampled locations. The kriging estimate at sample locations is “exact”. At the maximum nugget, the data are uncorrelated, the unsampled locations are uncorrelated to the sample locations, and the kriging component of estimated direction at unsampled locations is zero. The kriging component is added to the fitted model direction. At uncorrelated locations, the estimated direction is the fitted direction.

```
KrigCRF(krig.x = x2, krig.y = y2, resid.x= resids1$x, resid.y= resids1$y, resid.direction=
    resids1$direction, Model="exponential", Nugget=0.0,   Range=4, sill=0.56, Plot=TRUE,
    Xlim=c(7,10), Ylim=c(7,10))
# Repeat with Nugget = 0.15 and 0.3
KrigCRF(krig.x = x2, krig.y = y2, resid.x=resids1$x, resid.y=resids1$y, resid.direction=
    resids1$direction, Model="exponential", Nugget=0.44, Range=4, sill=0.56, Plot=TRUE,
    Xlim=c(7,10), Ylim=c(7,10))
arrow.plot(resids1$x, resids1$y, u =cos(resids1$direction), v = sin(resids1$direction), arrow.ex =
    0.09, xpd = F, true.angle = T,  length=.05, col=2)
```

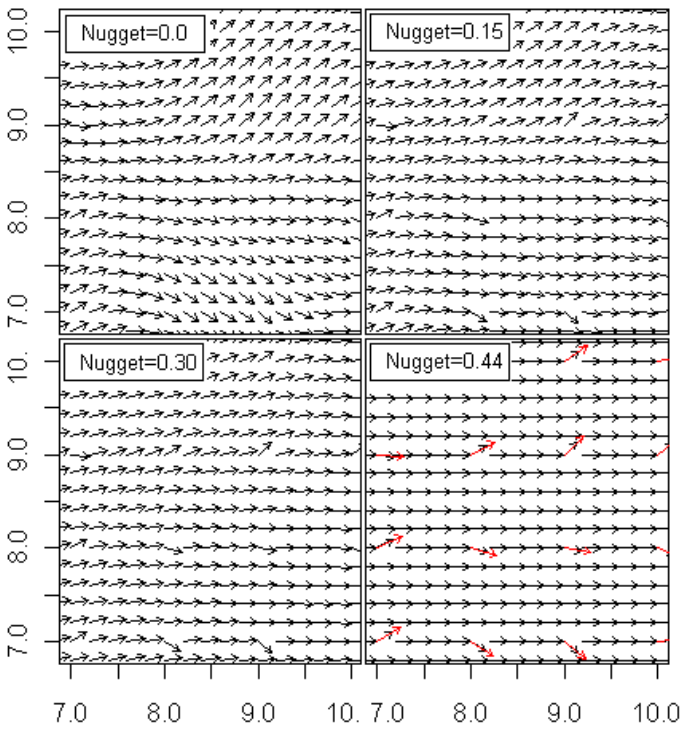


Figure J-11. Smoothing via the Nugget Not Effective at Data Locations. Red out of line arrows in the lower right panel are the residual rotations. This smoothing method does not affect estimates at data locations.

Smoothing applied to the kriging component of estimated direction smoothes direction over all points. See Figure J-12.

```
KrigCRF(krig.x = x2, krig.y = y2, resid.x= resids1$x, resid.y= resids1$y, resid.direction=
        resids1$direction, Model="exponential", Nugget=0.0,  Range=4, sill=0.56, Plot=TRUE,
        Xlim=c(7,10), Ylim=c(7,10), Smooth=TRUE, bandwidth=0.1)
KrigCRF(krig.x = x2, krig.y = y2, resid.x= resids1$x, resid.y= resids1$y, resid.direction=
     resids1$direction, Model="exponential", Nugget=0.0,  Range=4, sill=0.56, Plot=TRUE,
     Xlim=c(7,10), Ylim=c(7,10),  Smooth=TRUE, bandwidth=2)
KrigCRF(krig.x = x2, krig.y = y2, resid.x= resids1$x, resid.y= resids1$y, resid.direction=
     resids1$direction, Model="exponential", Nugget=0.0,  Range=4, sill=0.56, Plot=TRUE,
     Xlim=c(7,10), Ylim=c(7,10), Smooth=TRUE, bandwidth=4)
KrigCRF(krig.x = x2, krig.y = y2, resid.x=resids1$x, resid.y=resids1$y, resid.direction=
     resids1$direction, Model="exponential", Nugget=0.0,  Range=4, sill=0.56, Plot=TRUE,
     Xlim=c(7,10), Ylim=c(7,10), Smooth=TRUE, bandwidth=10)
```

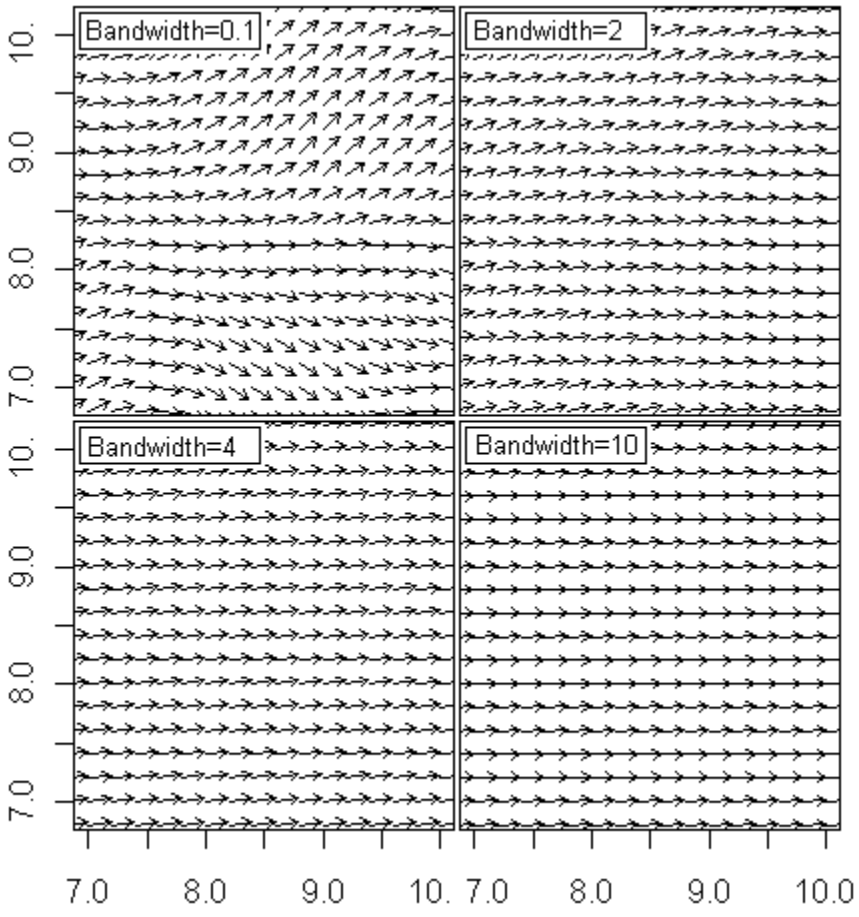


Figure J-12. Smoothing the Kriging Components is Effective at All Locations.

```
## Plot kriging estimate variability at sample locations on a regular grid, See Figure J-13
KrigCRF(krig.x = x2, krig.y = y2, resid.x= resids1$x, resid.y= resids1$y, resid.direction=
    resids1$direction, Model="exponential", Nugget=0.0,  Range=4, sill=0.56, Plot=TRUE,
    Smooth=FALSE, PlotVar=TRUE)

## Plot kriging estimate variability with random sample locations, See Figure J-14.
set.seed(13)
crf6 <- SimulateCRF(N=400, CircDistr="Card", Rho= 0.4, Range=4, Ext=3,
    CovModel="spherical")
## Best fit is spherical with range=2.85 and sill=0.15
CosinePlots(x=crf6$x, y=crf6$y, directions=crf6$direction, Lag.n.Adj=1.5, BinWAdj=1,
    Plot=TRUE, Cloud=FALSE, Model=TRUE, nugget=0, Range=2.85, sill=0.15, x.legend=.14,
    y.legend=0.75, xlim=c(0,6), ylim=c(0,1))
x6 <- seq(4,7, by=0.02); y6 <- x6
# This may take a long time
KrigCRF(krig.x =x6, krig.y =y6, resid.x=crf6$x, resid.y=crf6$y, resid.direction=crf6$direction,
    Model="spherical", Nugget=0.0,  Range=2.85, sill=0.15, Plot=TRUE, PlotVar=TRUE)
```

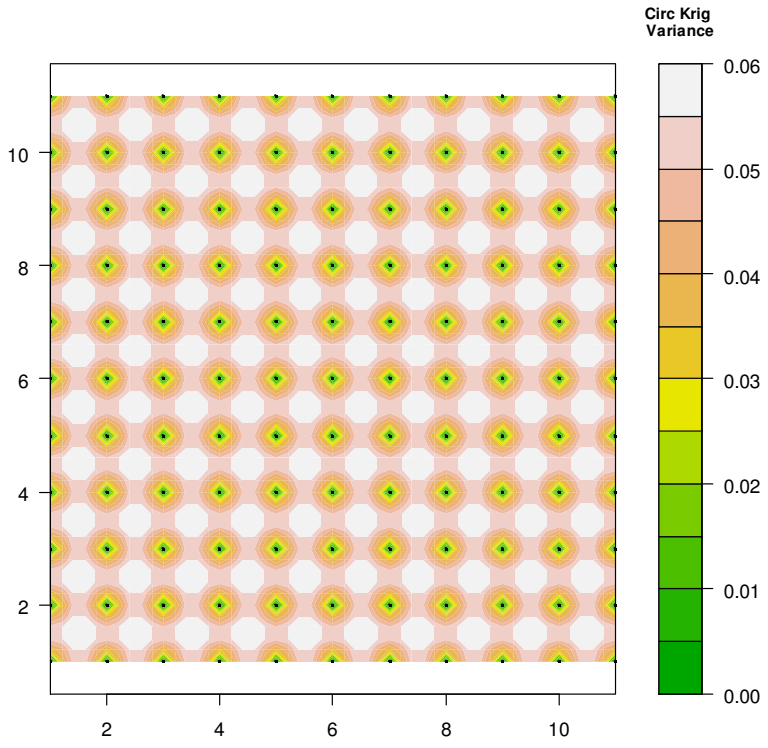


Figure J-13. Variability of the Circular Kriging Estimate with Locations on a Regular Grid. Measurement locations are indicated by black points. Estimate variability is zero at a sample location. $0 \le \sigma^2_{CK} < 4$ (Chapter 4, Section 4.6).

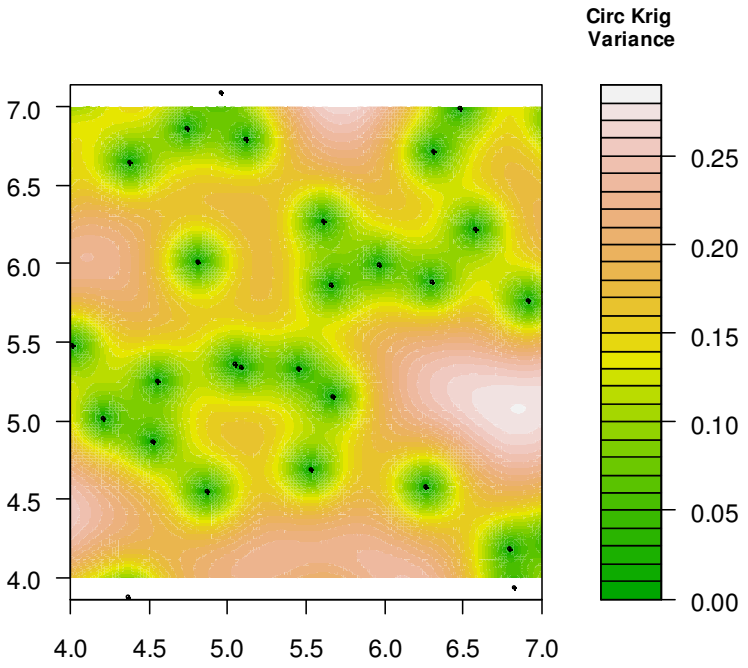


Figure J-14. Variability of the Circular Kriging Estimate with Random Locations. Measurement locations are indicated by black points. Estimate variability is zero at a sample location. $0 \le \sigma^2_{CK} < 4$ (Chapter 4, Section 4.6).

J.6 InterpDirection

The interpolated model direction is added to the kriged, or smoothed kriged residuals, to obtain the estimated direction. To avoid cross over, the cosine and the sine of the model direction are separately interpolated. The algorithm has 6 cases of interpolation location as indicated by the labeled red dots in Figure J-15. The corners of the grey rectangle are observation locations with observations indicated by the blue unit vectors. For example, assume the interpolation location falls in the lower triangle with label f. For the interpolation of the cosine, a plane is fitted to the three points $\{(x_1, y_1, \cos(\theta_1)), (x_2, y_2, \cos(\theta_2)), (x_3, y_3, \cos(\theta_3))\}$. The interpolated value is the elevation of the plane at the interpolation location. The inverse tangent is applied to the interpolated cosine and sine components. Figure J-16 shows the result of interpolating smoothed average wind direction. Interpolation outside the model gives an error.

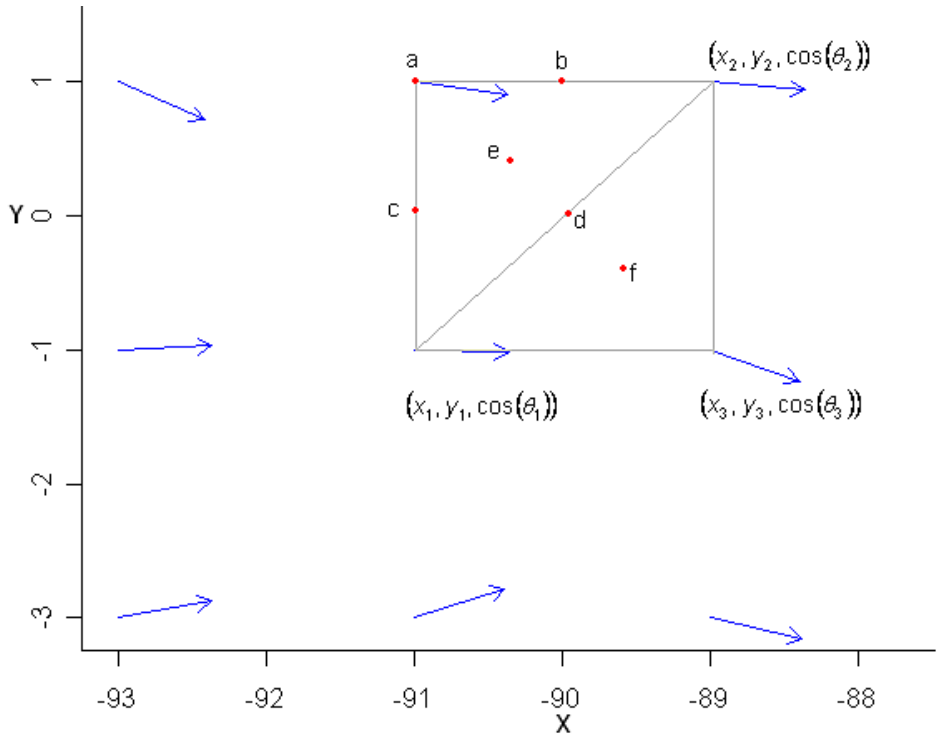


Figure J-15. Six Cases of Interpolation Location Indicated by Labeled Red Dots. The corners of the grey rectangle are observation locations. Planes are fitted to the triangular partitions. The interpolated component, cosine or sine, is the elevation of the plane at the location of interpolation.

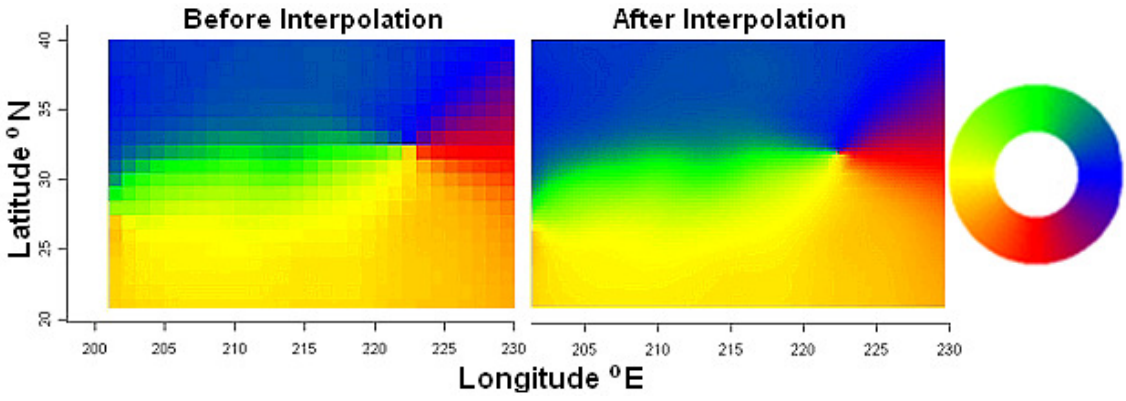


Figure J-16. Effect of Interpolation on Smoothed Average Wind Direction with BGYR Color Wheel.

J.6.1 Usage and Input Arguments

InterpDirection(in.x, in.y, in.direction, out.x, out.y)

in.x: Vector of input horizontal coordinates.

in.y: Vector of input vertical coordinates.

in.direction: Vector of input direction in radians.

out.x: Vector of output horizontal coordinates.

out.y: Vector of output vertical coordinates.

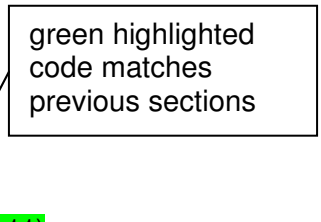
J.6.2 Output List

x: out.x.

y: out.y.

direction: Vector of interpolated direction in radians.

J.6.3 Examples




```
require(CircSpatial)
## Construct Trend Model of 121 locations
x1<- 1:11; y1 <- 1:11; y1 <- rep(y1, 11); x1 <- rep(x1, each=11)
model.direction1 <- matrix(data=c(
   157, 141, 126, 113, 101, 90, 79, 67, 54, 40, 25, 152, 137, 123, 111, 100, 90, 80, 69, 57, 44, 30,
   147, 133, 120, 109,  99, 90, 81, 71, 60, 48, 35, 142, 129, 117, 107,  98, 90, 82, 73, 63, 52, 40,
   137, 125, 114, 105,  97, 90, 83, 75, 66, 56, 45, 132, 121, 111, 103,  96, 90, 84, 77, 69, 60, 50,
   127, 117, 108, 101,  95, 90, 85, 79, 72, 64, 55, 122, 113, 105,  99,  94, 90, 86, 81, 75, 68, 60,
   117, 109, 102,  97,  93, 90, 87, 83, 78, 72, 65, 112, 105,  99,  95,  92, 90, 88, 85, 81, 76, 70,
   107, 101,  96,  93,  91, 90, 89, 87, 84, 80, 75), ncol=11, byrow=TRUE)
model.direction1 <- as.vector(model.direction1)*pi/180

## Compute vM CRF of 121 observations, Rho=sqrt(0.5) so sill about 0.5,
## from GRF (Range=4, spherical covariance).
set.seed(666)
crf1<- SimulateCRF(CircDistr="vM", Rho=sqrt(0.5), Range=4, CovModel="spherical",
      Grid=cbind(x1, y1), OverFit=TRUE)

# Make sample
sample.direction1 <- model.direction1 + crf1$direction

## Fit An Appropriate Model
## Code for median polish is contained in Appendix K, Section K.12
FitHoriz1 <- lm(cos(sample.direction1) ~ (x1 + y1))
FitVert1 <- lm(sin(sample.direction1)  ~ (x1 + y1))
```

```
fitted.direction1 <- atan2(FitVert1$fitted.values, FitHoriz1$fitted.values)

## Compute Residuals
resids1 <- CircResidual(X=x1, Y=y1, Raw=sample.direction1, Trend=fitted.direction1,
     Plot=FALSE)

## Fit cosine Models
CosinePlots(x=resids1$x, y=resids1$y, directions=resids1$direction, Lag.n.Adj=1, BinWAdj=1,
     Plot=TRUE, Cloud=FALSE, Model=TRUE,  nugget=0, Range=4.0,  sill=0.56, x.legend=0.2,
     y.legend=0.4, xlim=c(0,8), ylim=c(0,1))

## Krig to residuals using cosine Model (Figure J-9)
x2 <- seq(1,11, by=0.2); n <- length(x2); y2 <- x2; y2 <- rep(y2, n); x2 <- rep(x2, each=n); rm(n)
krig2 <- KrigCRF(krig.x=x2, krig.y=y2, resid.x=resids1$x, resid.y=resids1$y, resid.direction=
resids1$direction, Model="exponential", Nugget=0.0, Range=4, sill=0.56, Plot=FALSE)

## Interpolate Fitted Model
interp2 <- InterpDirection(in.x=x1, in.y=y1, in.direction=fitted.direction1, out.x=krig2$x,
out.y=krig2$y)

## Plot Interpolated Fitted Model and Overplot Fitted Model.  See Figure J-17.
plot(interp2$x, interp2$y, type="n", asp=1, xlim=c(5,8), ylim=c(5,8), xlab="", ylab="")
arrow.plot(interp2$x, interp2$y, u=cos(interp2$direction), v=sin(interp2$direction), arrow.ex=0.09,
     xpd=FALSE, true.angle=TRUE, length=.1, col="tan")
arrow.plot(x1, y1, u =cos(fitted.direction1), v=sin(fitted.direction1), arrow.ex=0.06, xpd=FALSE,
     true.angle=TRUE, length=.1, col=1)
```

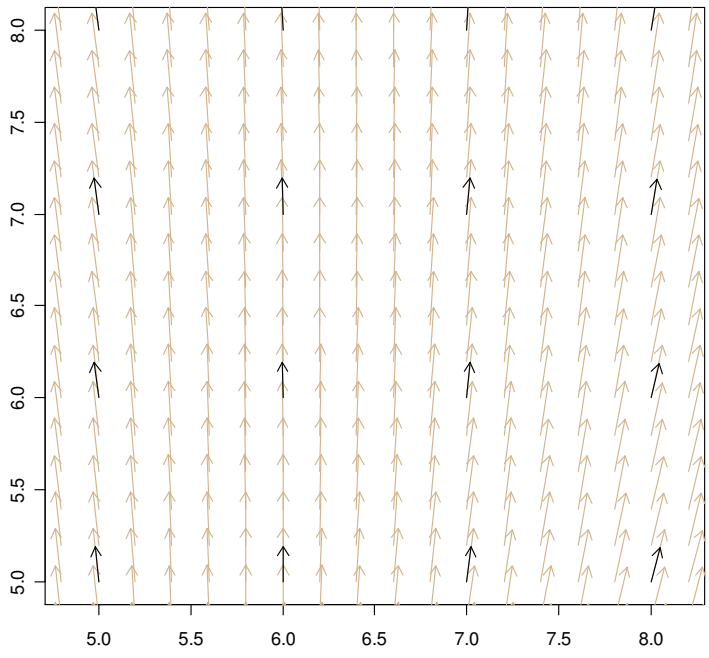


Figure J-17. Fitted Model (Black) Overplotted on the Fitted Model Interpolation (Tan). The fitted model is an enlargement of Figure J-3 (d). The interpolated direction is the same as the model direction at model locations, i.e. the interpolation is “exact”.

```
## Plot Estimate Of Direction And Overplot Sample.  See Figure J-18.
estimate2=interp2$direction + krig2$direction
plot(interp2$x, interp2$y, type="n", xlab="", ylab="", asp=1)
arrow.plot(interp2$x, interp2$y, u=cos(estimate2), v= sin(estimate2), arrow.ex=0.05, xpd=FALSE,
     true.angle=TRUE, length=.05, col="tan")
arrow.plot(x1, y1, u=cos(sample.direction1), v=sin(sample.direction1), arrow.ex=0.05,
     xpd=FALSE, true.angle=TRUE, length=.05, col=1)

## Zoom.  See Figure J-19.
plot(interp2$x, interp2$y, type="n", xlab="", ylab="", asp=1, xlim=c(3,6), ylim=c(3,6))
arrow.plot(interp2$x, interp2$y, u=cos(estimate2), v=sin(estimate2), arrow.ex=0.075,
          xpd=FALSE, true.angle=TRUE, length=.05, col="tan")
arrow.plot(x1, y1, u=cos(sample.direction1), v=sin(sample.direction1), arrow.ex=0.05,
          xpd=FALSE, true.angle=TRUE, length=.05, col=1)
```

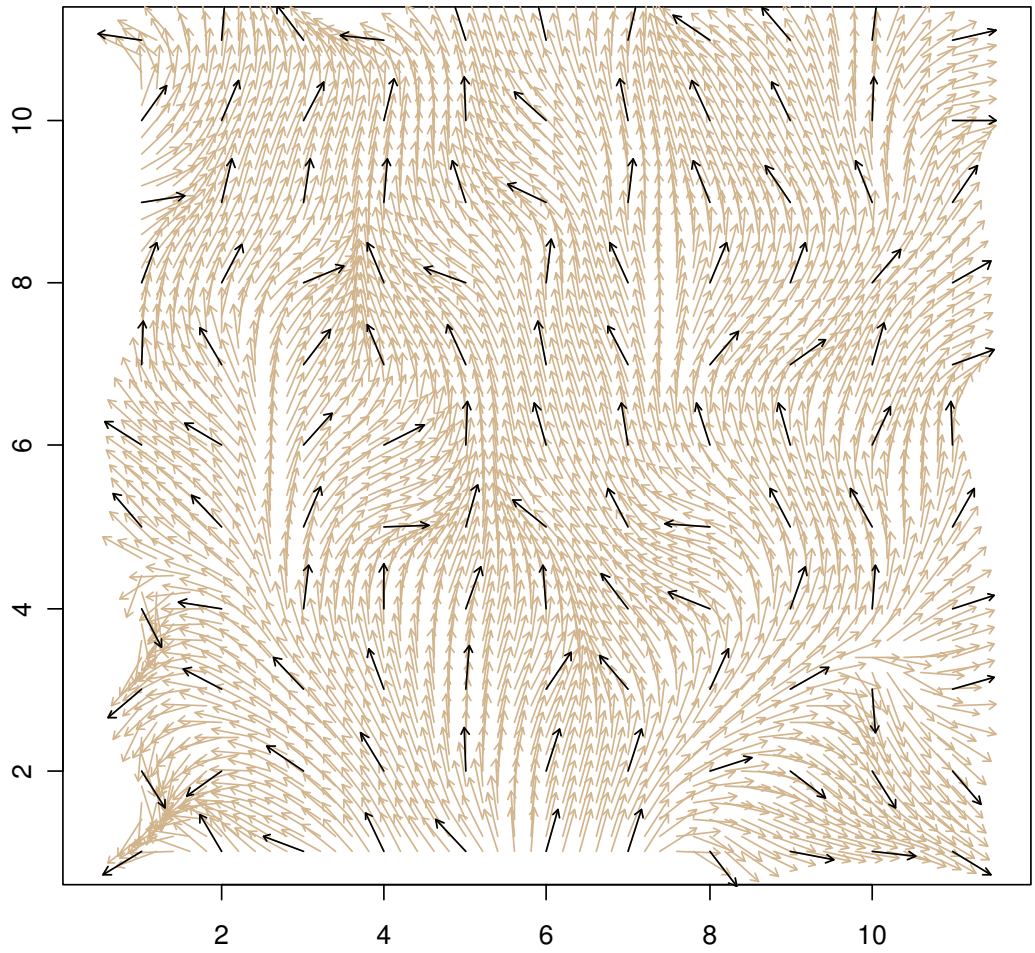


Figure J-18. Original Data (Black) Overplotted on the Estimates (Tan).

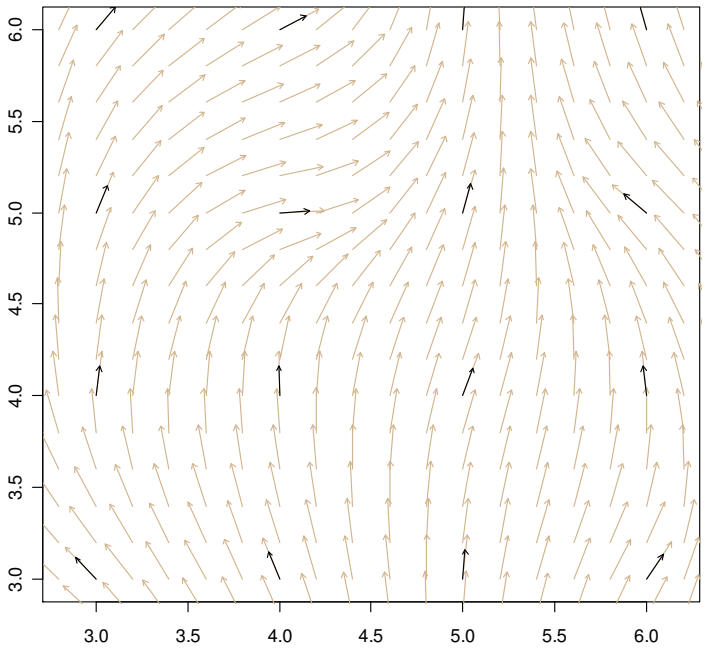


Figure J-19. Enlargement of Figure J-18. The estimate (tan) equals the sample direction (black) at sample locations. This is called “exact interpolation”.

J.7 TestPattern

The function TestPattern (Appendix K, Section K.1) makes an intuitive simple test pattern to explore the function CircDataimage, which produces a GUI for interactive imaging of circular-spatial data. TestPattern computes direction such that the direction at any point is the angle between the line from origin to point and the horizon.

J.7.1 Usage

testpattern <- TestPattern()

J.7.2 Output Dataframe

x: Vector of horizontal location coordinates.

y: Vector of vertical location coordinates.

u: Vector of horizontal component of cosines of direction.

v: Vector of vertical component of sines of direction.

J.8 OceanWind

OceanWind provides a large dataset (495,688 observations) to further explore the function CircDataimage. The OceanWind data was freely obtained from ICOADS at http://dss.ucar.edu/datasets/ds540.1/data/msga.form.html for the El Nino years 1972, 1976, 1982, 1987, 1991, 1994, and 1997, January through April, in 1º increments for the area of longitude 0.5º E to +359.5º E by latitude -59.5º N to +60.5º N. See Chapter 2, Subsection 2.2.1.

J.8.1 Usage

data(OceanWind)

J.8.2 Dataframe

year: Vector of time of observation = year + month/12, month in [1972.083, 1997.333]

x: Vector of longitude.

y: Vector of latitude.

u: Vector of east component of wind (0.01 meters/second).

v: Vector of north component of wind (0.01 meters/second).

J.9 WorldMask

WorldMask is used by function CircDataimage to restore land contours to the circular dataimage of smoothed OceanWind. WorldMask was derived from the R package fields dataset world.dat.

J.9.1 Usage

data(WorldMask)

J.9.2 Value

WorldMask is a matrix of 360 rows (0.5º to 359.5º longitude) x 121 columns (-59.5º to +60.5º latitude) suited to OceanWind, with elements NA where wind data is not missing and 1 where wind data is missing. Figure J-20 is an image plot of WorldMask.

J.10 CircDataimage

CircDataimage generates a GUI for interaction with circular dataimages to facilitate the discovery of structure in circular-spatial data.

J.10.1 Introduction

The circular dataimage is useful for visualization of random, model, and kriged circular-spatial data. The circular dataimage is constructed by displaying direction as the color in a color wheel at the same angle. This implementation of color provides for simultaneous recognition of fine circular-spatial structure on a small scale and large-scale circular-spatial structure on a global scale (Figure J-21 b, d, f). Depending on data density and smoothness, this structure is lost traditional arrow plots (Figure J-21 a, c).

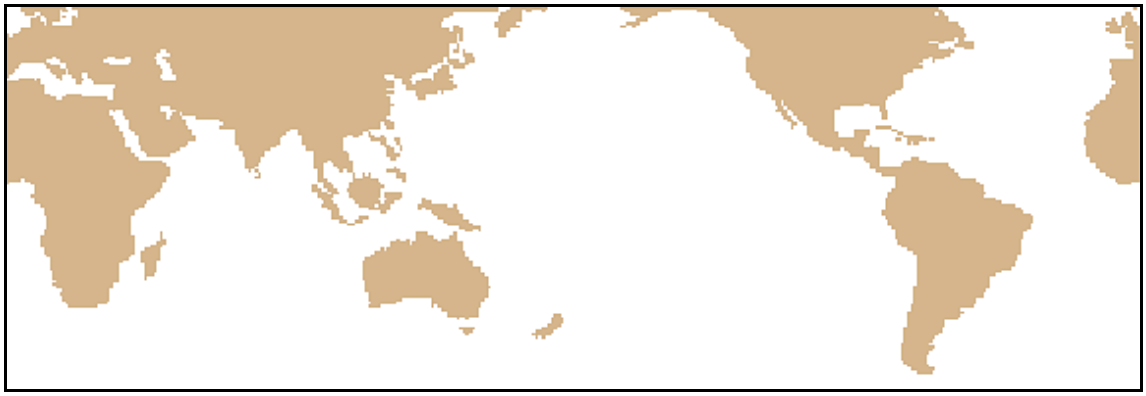


Figure J-20. Image Plot of WorldMask. Land masses are tan.

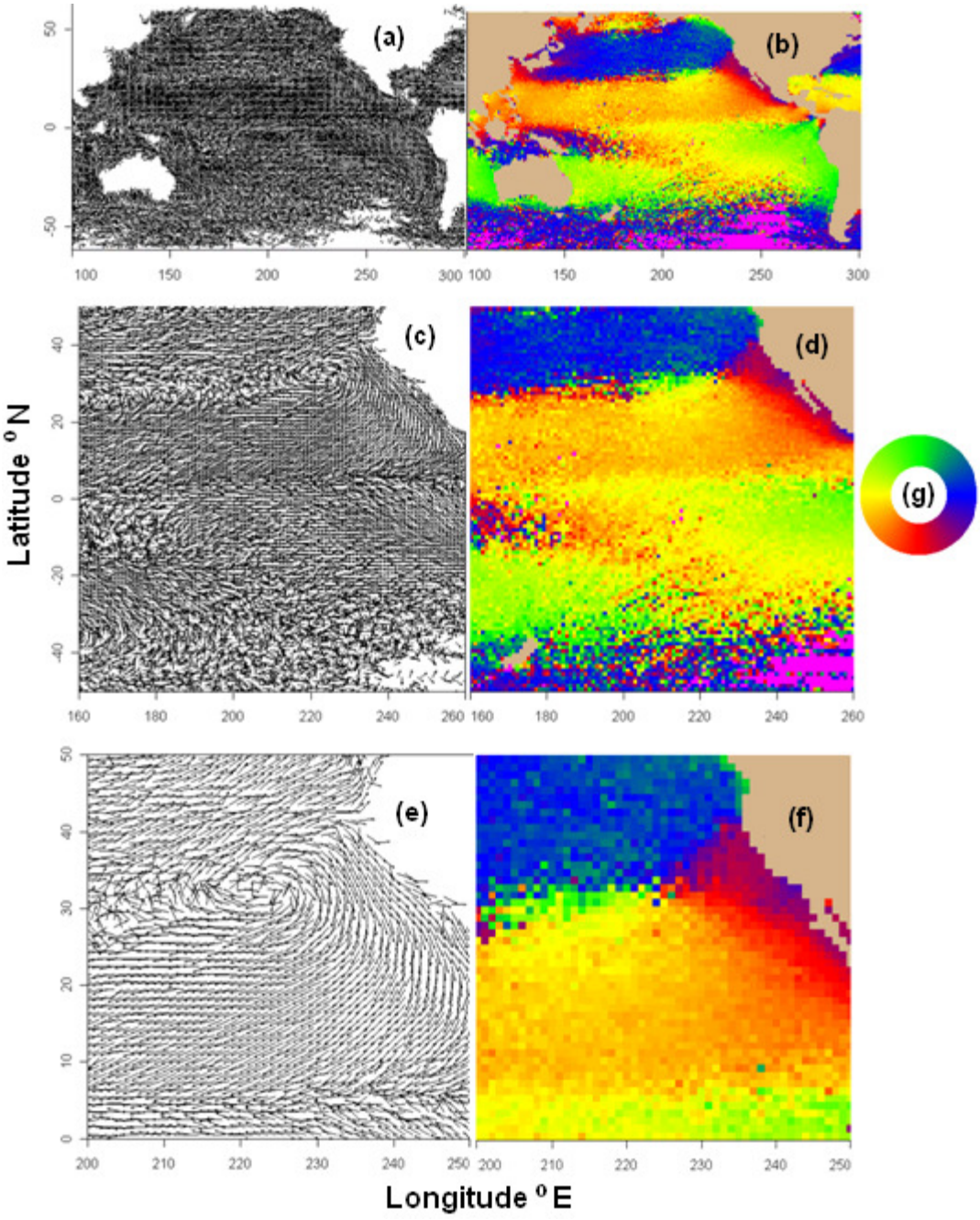


Figure J-21. Comparison of Arrow and Circular Dataimage Plots of Ocean Wind Average Direction. Plots (a), (b) cover 200º of longitude; (c), (d) cover 100º Of longitude; (e), (f) cover 50º of longitude; (g) BGYR Circular Color Wheel. Ability to recognize structure depends on plot type, smoothness and density of data and arrows, and distribution of missing data.

J.10.2 The Color Wheel

The color wheel evolved as shown in Figure J-22. In Figure J-22 (d) every point on the color wheel is color continuous, every direction is represented by a unique color, and the color change is linear. The resulting color wheel is named a Yellow-Red-Green-Blue (YRGB) Color Wheel. In general, the color wheel is composed of a sequence of color gradients with continuity between connecting color gradients such that the color coding the 0º direction is the same as the color coding the 360º direction. Thus, image discontinuity from using a single color gradient for visualization of circular-spatial data is eliminated. Package provides 7 continuous color wheels with 1 for red-green color impaired, 1 Hue Saturation Value (HSV), and 2 divergent, and 6 discrete color wheels with 3 divergent. For additional information, see Chapter 2, Circular Dataimage, A High Resolution Image Of Circular-Spatial Data.

J.10.3 Input Requirements

The input data.frame entered into the GUI Input Dataframe entry box contains:

x: Vector of measurement location horizontal coordinates on regular grid.

y: Vector of measurement location vertical coordinates on regular grid.

u: Vector of horizontal component of measured vector. NA is not allowed.

v: Vector of vertical component of measured vector. NA is not allowed.

Note: The input data are measured on a regular grid. The vertical and horizontal grid spacing may be different. Spacing is computed from the data as the second smallest location coordinate minus minimum location coordinate. Multiple observations at the same location will be replaced with the mean resultant = vector resultant/number of observations. Missing data are permitted, but a data.frame with rows of (x, y, NA, NA) are not allowed. u=v=0 OK.

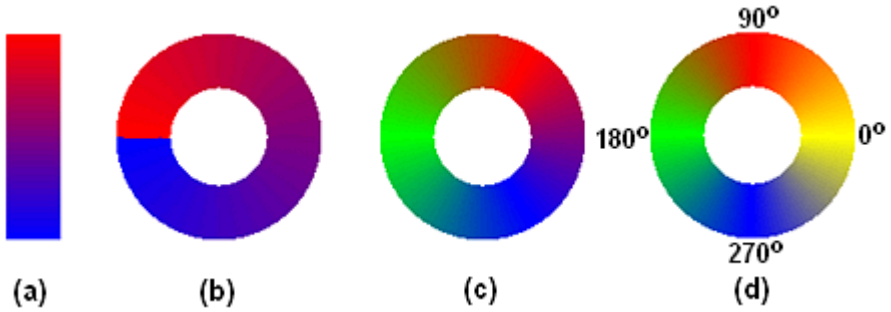


Figure J-22. Evolution of the YRGB Color Wheel. (a) Blue-red linear color scale, (b) Color scale (a) wrapped on circle, (c) red-green-blue linear color gradient inserted at 180º, and (d) blue-yellow-red gradient inserted at 0º. The YRGB color wheel (d) aligns the 4 main colors to the 4 main directions (0, 90, 180, 270º).

The mask matrix entered into the GUI Mask Matrix entry box is optional (See J.9 WorldMask). It must have rows and columns equal to the rows and columns of the measurement grid, respectively. Mask cell values are NA where pixel is not to be masked and 1 where a pixel is to be masked.

J.10.4 Output

The list object CircDataimageGlobals, which contains the GUI inputs and results of computations, is written into the R workspace.

J.10.5 Startup Example

```
## Consider setting the R GUI preference to SDI
require(CircSpatial)
data(OceanWind)
data(WorldMask)
wind.subset <- OceanWind[, -1] # Using all the data, about 500k records
CircDataimage()
```

J.10.6 GUI Demonstration Using Ocean Wind Data

The initial display appears as in Figure J-23. “unknown” or an empty Input Dataframe (bolding indicates a GUI element in the referenced figure) entry box or Mask Matrix entry box indicate the absence of input.

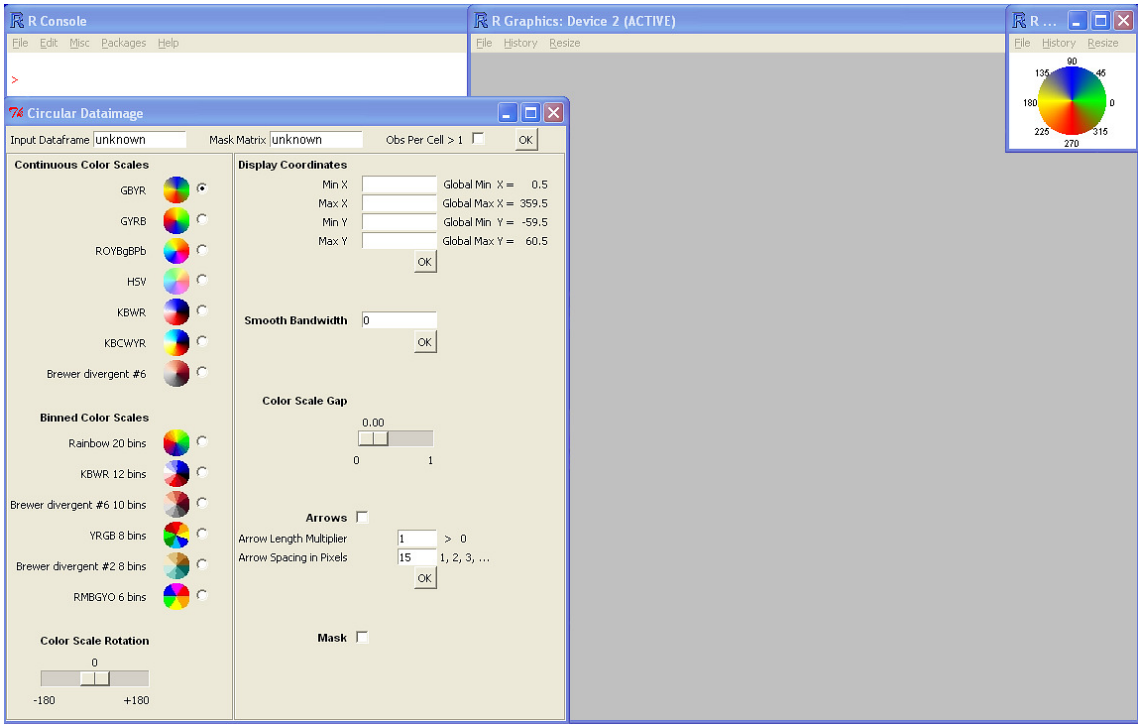


Figure J-23. Initial Display of the GUI, the Circular Dataimage Window (R Graphics Device 2), and the Color Wheel Window.

1 Enter the name of the data.frame wind.subset in the Input Dataframe entry box. Enter the example optional mask matrix WorldMask in the Mask Matrix entry box. N.B.: The input data.frame and optional mask must be input before other GUI controls will function correctly. After data is input, the controls may be operated in any order.

2 Check the Obs Per Cell > 1 checkbox because wind.subset has more than one observation per location. When the observations per cell <= 1, the looping structure is replaced with vector-matrix expression to reduce computation time.

3 To accept the input data and optional mask, click the OK button at the right of the mask entry box. The OK button will remain depressed until processing is complete. If the Obs Per Cell > 1 checkbox is checked, the message “The computations necessarily may take significant time” is sent to the R GUI. The computation of average wind direction over the 500k observations of wind.subset is relatively slow. However, testpattern (J.2) is imaged instantaneously (Obs Per Cell > 1 checkbox is unchecked).

4 The display now appears as in Figure J-24. The circular dataimage appears in R Graphics: Device 2 and the current color wheel in R Graphics: Device 3. The Display Coordinates and reference global (extreme) coordinates to the right are updated. The Smooth Bandwidth is set to 0, and the Arrows and Mask check buttons are unchecked. Arrange the windows as necessary.

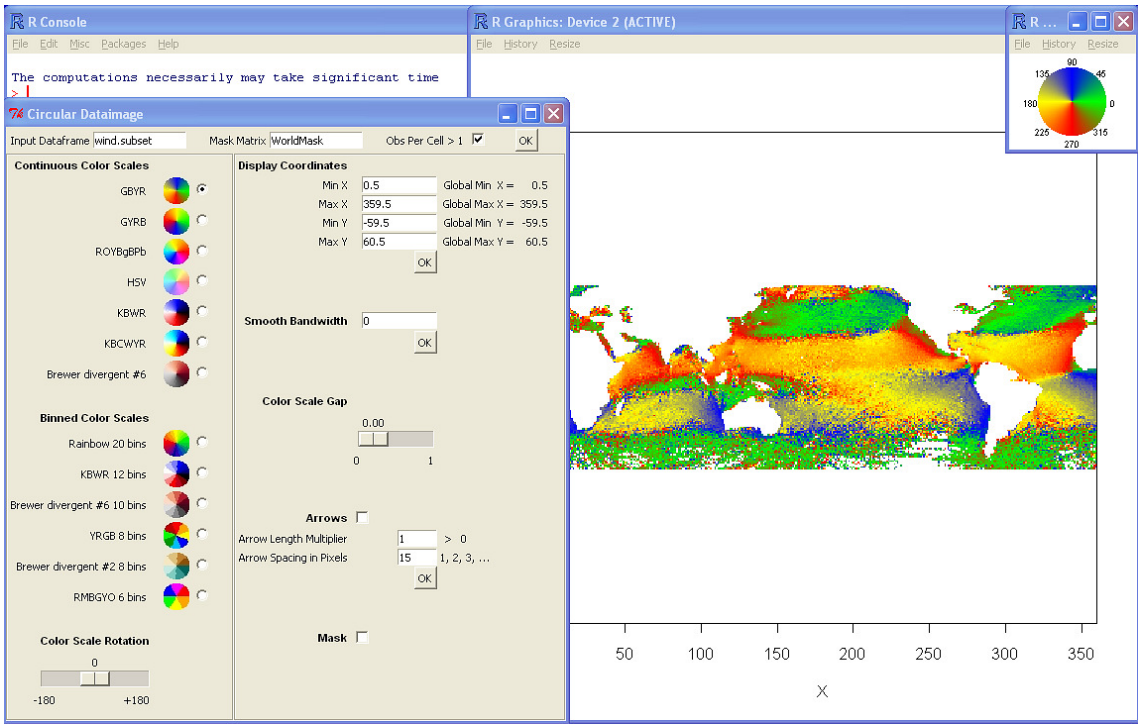


Figure J-24. Display with Circular Dataimage of Average Direction after Inputs Entered.

5 Select any color wheel scale by clicking the radio button next to the color wheel label and graphic, and the circular dataimage and color wheel (R Graphics: Device 3) are quickly updated.

6 Rotation is used to highlight structure in the displayed area of interest. The selected color wheel may be varied by moving the Color Scale Rotation slider. As the slider is moved, the label above the slider updates. When the mouse button (left) is released, the displayed color wheel rotation angle is processed with a positive value resulting in a counterclockwise rotation of the color wheel by the displayed value. The circular dataimage and color wheel are updated, but the data are unchanged.

7 Coordinates are entered to pan and zoom. Display coordinates reference the center of a pixel. Enter display coordinates into the Display Coordinates entry box. The user may enter any value, but entry box coordinates will snap to the coordinates of the nearest datum. Thus, coordinates outside the range of data are adjusted to the extremes of the data. A zoomed display may be reduced to as few as 2*2 pixels. 2*1, 1*1, and 1*2 pixel displays result in nonmatrix input to the image plot function and an image plot error message. Enter Min X=100 and Max X=275. Click OK to process the entered coordinates. The displayed coordinates in the entry boxes snap to Min X = 99.5 and Max X=274.5 because the input data.frame has longitude from 0.5º to 359.5º in 1º increments.

8 Smoothing is useful to reduce visual noise. Smoothing applies image.smooth of package fields separately to the horizontal and vertical components of the mean vector resultant to avoid “cross over”. The smoothed direction is obtained by applying the R function atan2 to the smoothed components. Enter a smooth bandwidth in the Smooth Bandwidth entry box and click OK. The OK button will remain depressed until processing is complete.

9 The result of selecting the Green-Yellow-Red-Blue (GYRB) color wheel with a +90º rotation (blue in the color wheel has rotated 90º CCW), and entering a smooth bandwidth=2.5 º longitude and latitude is shown in Figure J-25. Choice of bandwidth is determined by experimentation on the data.

10 Color scale gap increases the contrast at the boundaries of areas of similar direction (in the same color gradient component of the color wheel) by truncating the upper fraction of the gradient (stretching out the lower fraction or retarding the terminal color). For example, a 4 color-gradient, e.g., GBYR with Color Scale Gap=1 codes all directions in the range 0º to 90º as pure green. Move the slider of Color Scale Gap and the label above the slider updates. When the mouse button is released, the displayed color wheel gap is processed, and the color wheel and circular dataimage are updated.

11 Now select the HSV color wheel.

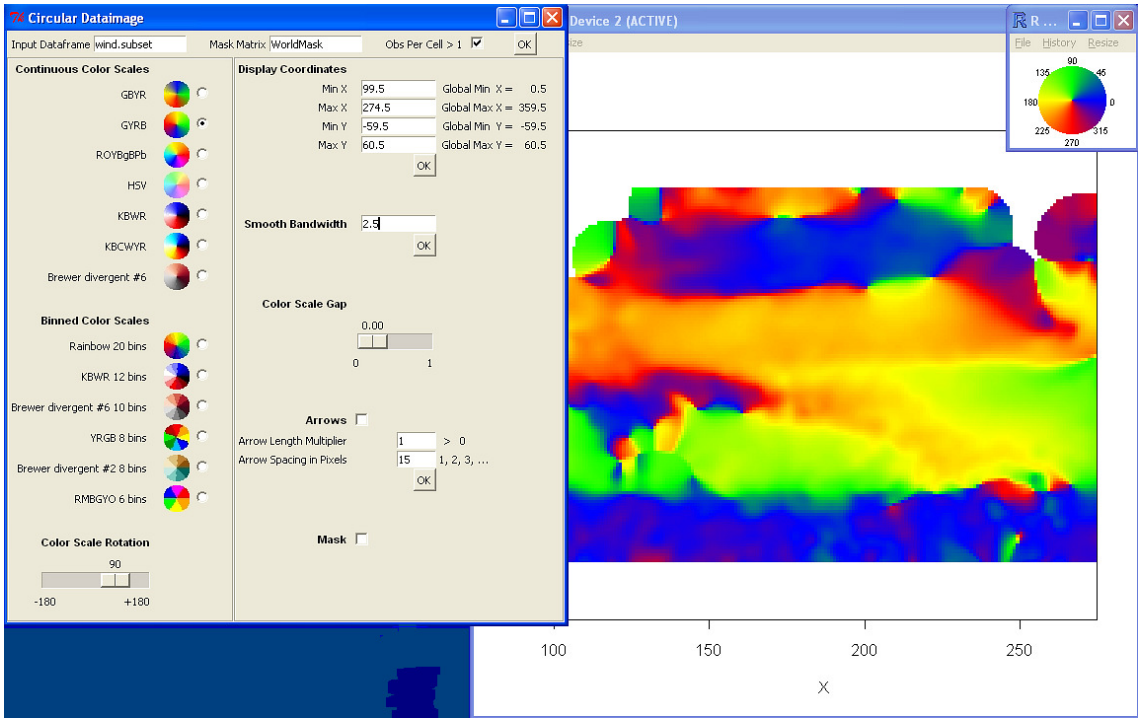


Figure J-25. GYRB Color Wheel Rotated 90º, Data Smoothed with Bandwidth 2.5, and Display Coordinates Changed (Zoomed).

12 To add arrows check the Arrows check box. An arrow is plotted if the value of direction at the arrow location is not NA, or the NA value is replaced by smoothing and the location is not a masked. Arrow Spacing in Pixels is the horizontal and vertical spacing between arrows. Valid values of Arrow Spacing in Pixels are indicated by the label to the right of the entry box, i.e., 1, 2, 3, ... The minimum value of Arrow Spacing in Pixels of 1 results in the maximum possible arrow density with one arrow per datum. At some "large" spacing and with missing data, no arrows may be displayed. If no arrows are displayed, the message "No arrows can be displayed at current spacing" is displayed in the R GUI. Increase arrow density by entering 5 in the Arrow Spacing in Pixels entry box and click OK. As arrow density is increased, arrow length can be decreased by decreasing the Arrow Length Multiplier to a value > 0. Enter 0.5 in the Arrow Length Multiplier entry box and click OK. The current result should appear as in Figure J-26.

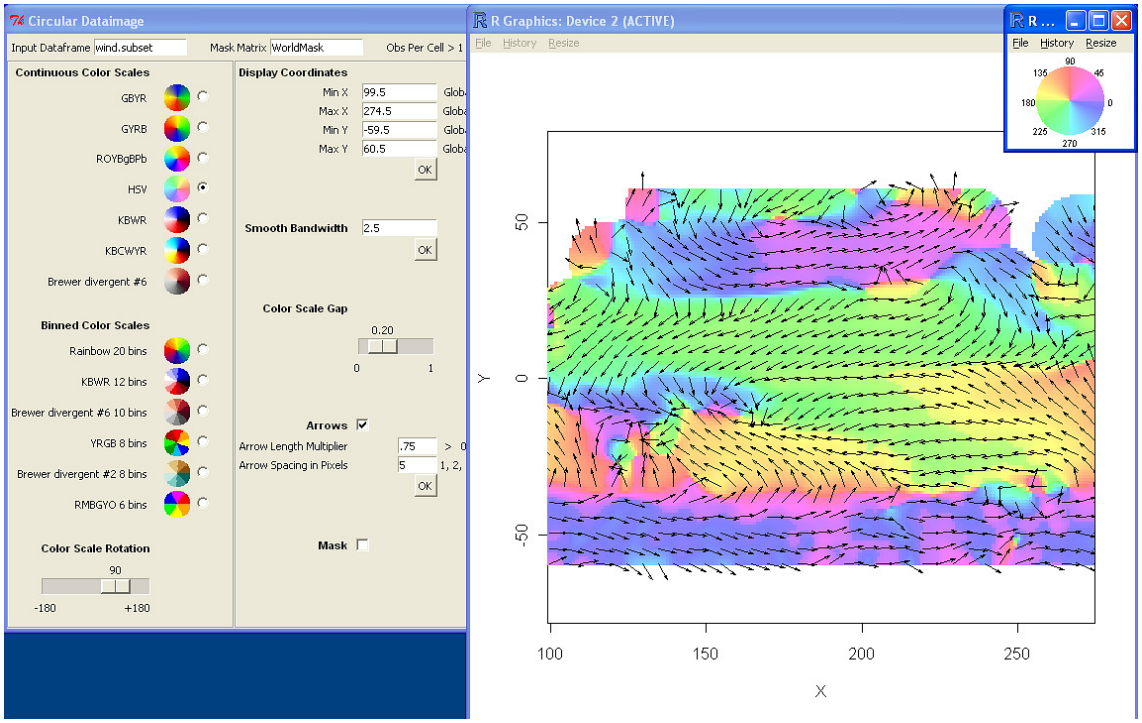


Figure J-26. HSV Color Wheel Rotated 90º, Data Smoothed with Bandwidth 2.5, Color Scale Gap 0.20, and Arrows on.

13 Smoothing replaces missing data (land in the wind.subset). The function of mask WorldMask is to restore the outlines of land masses. Pixels corresponding to matrix cells of 1 are overplotted in tan, and pixels corresponding to matrix cells of NA remain unchanged. If a mask matrix was not entered and the mask checkbox is clicked, nothing happens. Now check the Mask check box. The current result should appear as in Figure J-27.

14 Click the X in the upper right corners to exit.

J.11 PlotVectors

PlotVectors plots vector-spatial data as unit vectors, vectors, or triangle icons (Ware, 2004) with or without jittering. Triangle icons have area proportional to vector magnitude. Jittering can be help clarify structure when vectors overlap. However, excessive jitter can obscure directional structure. The underlying fields function arrows will not plot any arrow of length less than 0.001 inch.

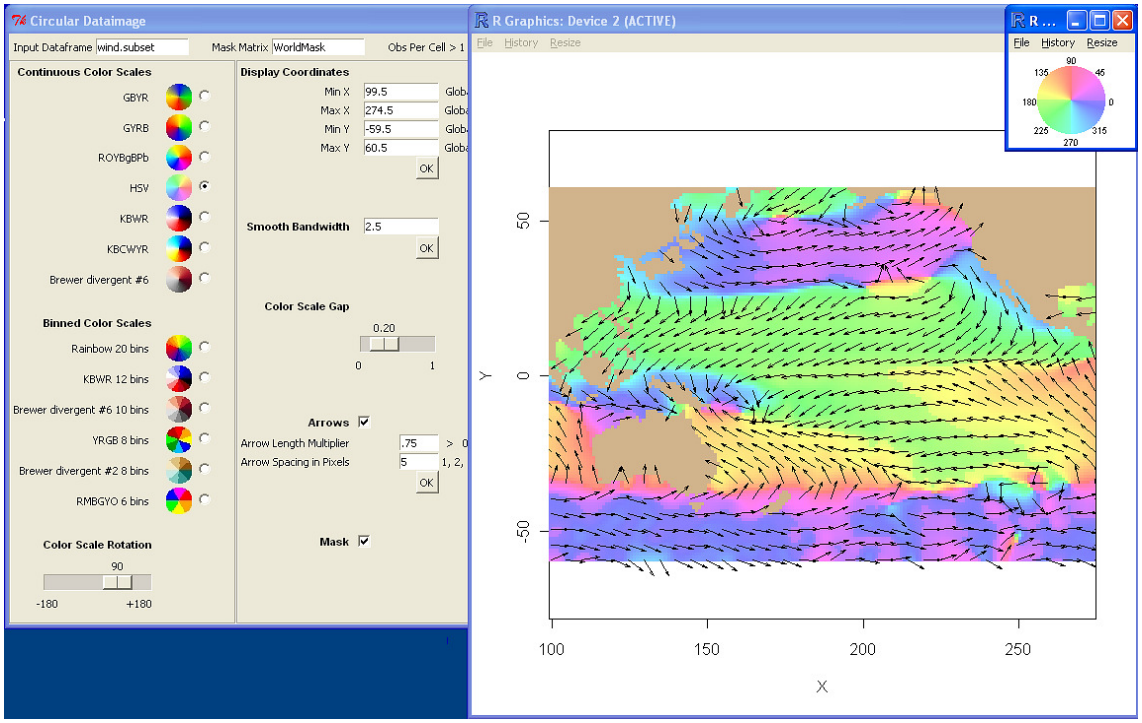


Figure J-27. Mask Restores Land Mass Shapes in Smoothed Data.

J.11.1 Usage and Input Arguments

PlotVectors(x, y, h, v, UnitVector=TRUE, TriIcon=FALSE, AdjArrowLength=1, AdjHeadLength=1, TriIconAdj=1, TriRatio=4, JitterPlot=FALSE, Jitter=1, \dots)

x: Vector of x coordinates.

y: Vector of y coordinates.

h: Vector of horizontal component. Missing values are permitted.

v: Vector of vertical component. Missing values are permitted.

UnitVector: TRUE or FALSE. See Table J-3.

TriIcon: TRUE or FALSE. See Table J-3.

AdjArrowLength: Arrow length multiplier.

AdjHeadLength: Arrow head length multiplier.

TriIconAdj: Multiplies size of icons.

TriRatio: Length to width ratio of triangle icon.

JitterPlot: If TRUE, add jitter to location coordinates.

Jitter: Amount of jitter = Jitter x runif value.

. . . : Additional plot parameters.

J.11.2 Output

Table J-3. Output of PlotVectors.

UnitVector	TriIcon	Plot Type
FALSE	FALSE	Direction and magnitude as variable length arrow
TRUE	FALSE	Direction as constant length arrow
FALSE	TRUE	Triangle icons with area proportional to magnitude
TRUE	TRUE	Direction as constant length arrow

J.11.3 Examples

```
require(CircSpatial)
data(OceanWind)
wind.1997.Jan <- OceanWind[OceanWind$year>1997 & OceanWind$year<1997.1,-1]


## Direction Only, Figure J-28
PlotVectors(x=wind.1997.Jan$x, y=wind.1997.Jan$y, h=wind.1997.Jan$u, v=wind.1997.Jan$v,
UnitVector=TRUE, AdjArrowLength=0.75, AdjHeadLength=0.75, xlim=c(320,350), ylim=c(0,30))

## Direction and Magnitude, Figure J-29
# fields function arrows omits arrowheads with a warning on
# any arrow of length less than 0.001 inch.
PlotVectors(x=wind.1997.Jan$x, y=wind.1997.Jan$y, h=wind.1997.Jan$u, v=wind.1997.Jan$v,
UnitVector=FALSE, TriIcon=FALSE, AdjArrowLength=3, AdjHeadLength=0.4, xlim=c(320,350),
ylim=c(0,30))

## Triangle Icons, Figure J-30
PlotVectors(x=wind.1997.Jan$x, y=wind.1997.Jan$y, h=wind.1997.Jan$u, v=wind.1997.Jan$v,
UnitVector=FALSE, TriIcon=TRUE, TriIconAdj=0.25, TriRatio=4, xlim=c(320,350),
        ylim=c(0,30))
```

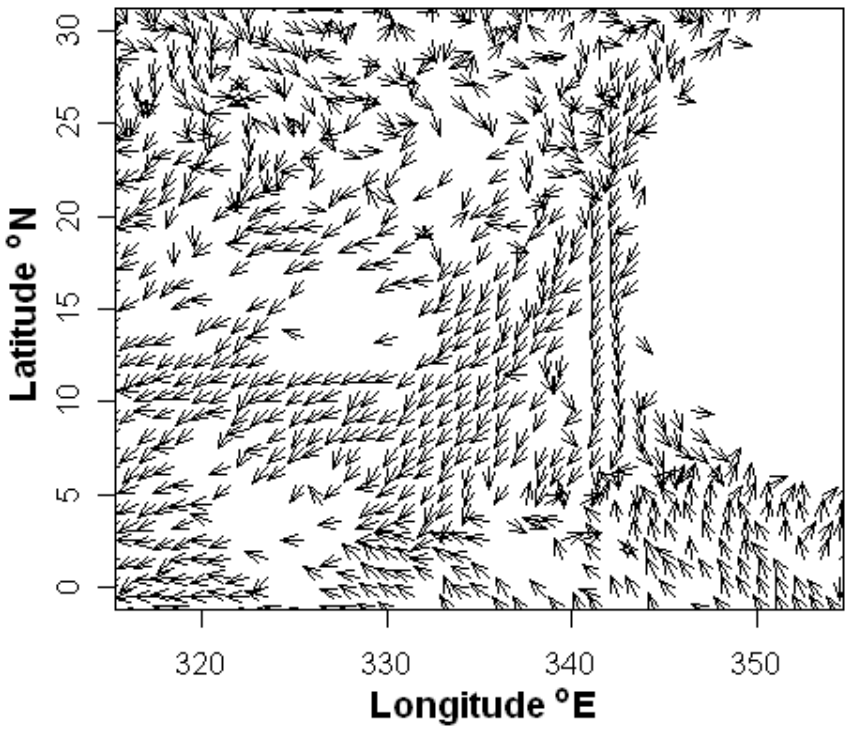


Figure J-28. Unit Vector Plot of Ocean Wind Data.

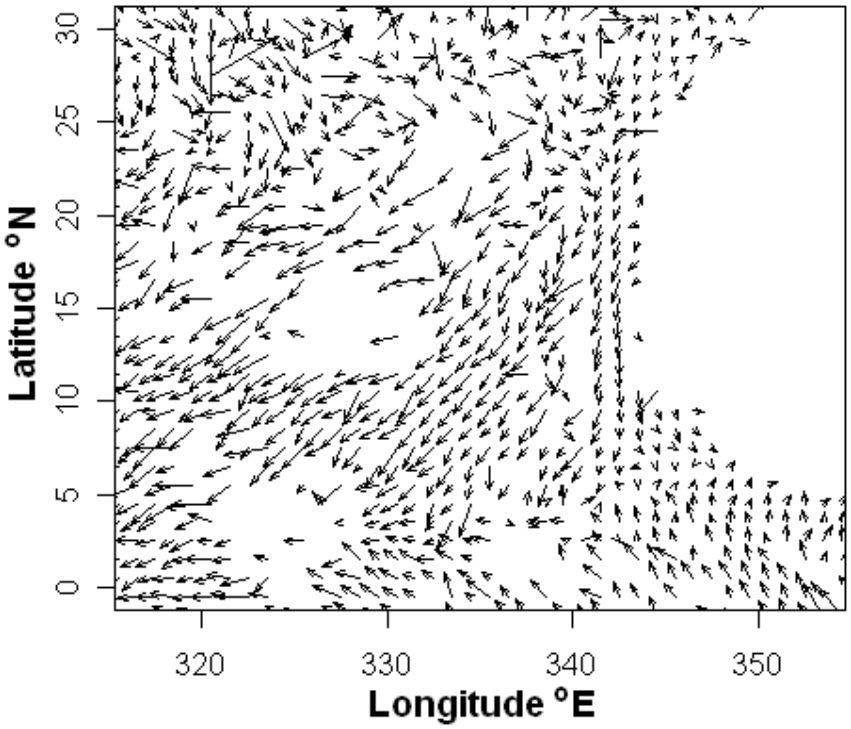


Figure J-29. Vector Plot of Ocean Wind Data.

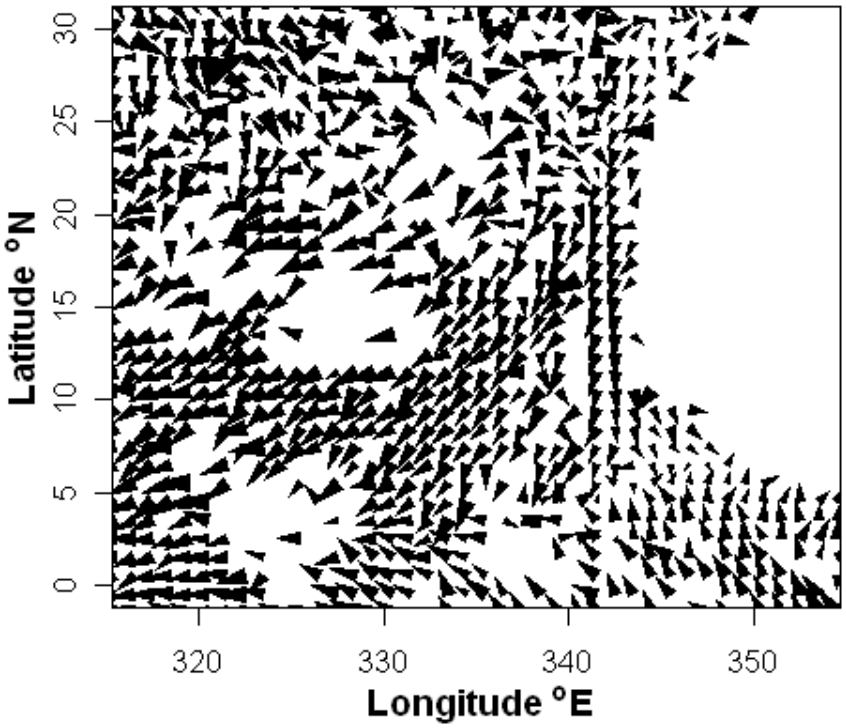


Figure J-30. Triangle Icon Plot of Ocean Wind Data.