Package: cope (via r-universe)

September 6, 2024

Type Package

Title Coverage Probability Excursion (CoPE) Sets

Version 0.2.3

Date 2017-02-13

Imports abind (>= 1.4-3), fields (>= 7.1), MASS (>= 7.3-34), Matrix (>= 1.2-3), mvtnorm (>= 1.0-0), nlme (>= 3.1-122)

Depends maps (>= 2.3-7)

Description Provides functions to compute and plot Coverage Probability Excursion (CoPE) sets for real valued functions on a 2-dimensional domain. CoPE sets are obtained from repeated noisy observations of the function on the entire domain. They are designed to bound the excursion set of the target function at a given level from above and below with a predefined probability. The target function can be a parameter in spatially-indexed linear regression. Support by NIH grant R01 CA157528 is gratefully acknowledged.

License GPL-2

RoxygenNote 5.0.1

NeedsCompilation no

Author Max Sommerfeld [aut, cre], Joshua French [aut], Armin Schwartzman [aut]

Maintainer Max Sommerfeld

`<max.sommerfeld@mathematik.uni-goettingen.de>`

Repository CRAN

Date/Publication 2017-02-13 12:57:47

Contents

`ARCoeffMap` *Generate the AR coefficient map.*

Description

Generate the AR coefficient map.

Usage

```
ARCoeffMap(Ns = 64)
```

Arguments

`Ns`	Number of pixels of the result in one direction. The resulting picture will have Ns x Ns pixels.

Value

A list containing x and y, the coordinates of the grid and z, a matrix of dimensions Ns x Ns giving the AR coefficients map.

ComputeCope	*Compute CoPE sets.*

Description

Computes CoPE sets for the data Y using the algorithm from Sommerfeld, Sain and Schwartzman (2015).

Usage

```
ComputeCope(Z, level, X = NULL, w = NULL, correlation = NULL,
  corpar = NULL, groups = NULL, V = NULL, alpha = 0.1, N = 1000,
  mu = NULL, mask = NULL)
```

Arguments

`Z`	A list with components "x", "y" and "z". Here, x and y are the coordinates of the grid on which the data is observed and z is an array with dimensions c(length(x),length(y),n), containing the data. n is the number of observations.
`level`	The level of interest.
`X`	The design matrix of the linear model. If NULL, it is set to matrix(rep(1,dim(Y)[3]),ncol=1) corresponding to i.i.d. data.
`w`	A vector of length nrow(X) indicating the desired linear combination of coefficients to be used in inference, i.e., t(w) NULL, the default is c(1, rep(0, ncol(X) - 1)).
`correlation`	Type of correlation assumed for the spatially indexed indexed linear models. This is a string that is passed to the function gls from the nlme package. Defaults to NULL which corresponds to i.i.d. errors.
`corpar`	A list of parameters passed to the correlation function.
`groups`	A factor vector describing groups that are used in the `correlation` function. Should have the same length as `X`.
`V`	A 4-dimensional array containing the covariance matrix associated with each element of `Z$z`. See Details.
`alpha`	The significance level. Inclusion for CoPE sets holds with probability 1-alpha.
`N`	The number of bootstrap realizations to generate for determining the threshold.
`mu`	The true parameter function. Use the default NULL if unknown.
`mask`	Pixels outside the mask (i.e. where mask is == NA) are ignored.

Details

The `V` argument is a 4-dimensional array containing the covariance matrices associated with `Z$z`. Specifically, `V[i,j,,]` is the covariance matrix of the data in `Z$z[i,j,]`. If `V` is specified, then the covariance matrix in each element of the array is used to transform `X` and the appropriate element of `Z$z` before fitting the linear model. This is used in place of estimating the covariance matrix withing the `nlme::gls` function.

Value

An object of class cope. A list containing the following

- x, y: The grid coordinates from the input.
- mu, level, tau, X, N, alpha, mask: The corresponding values from the input.
- mu_hat, norm_est: The estimatot for mu and its normalized version.
- a_MB, a_MB_true, a_Tay, a_Tay_true: Thresholds for the CoPE sets determined using the multiplier bootstrap or Taylor's method and the estimated or the true contour, respectively.
- incl_MB, incl_MB_true, incl_Tay, incl_Tay_true: Booleans indicating whether inclusion of the excursion set in the upper CoPE set and inclusion of the lower CoPE set in the excursion set holds, when CoPE sets are determined by a_MB, a_MB_true, a_Tay or a_Tay_true, respectively. Only available if mu is given.

References

M. Sommerfeld, S. Sain and A. Schwartzman. Confidence regions for excursion sets in asymptotically Gaussian random fields, with an application to climate. Preprint, 2015.

Examples

```
# An example using the ToyNoise and ToySignal of this package.
## Not run:
n = 30
Data = ToyNoise1(n = n)
Data$z = Data$z + rep(ToySignal()$z, n)
CopeSet = ComputeCope(Data,level=4/3, mu=ToySignal()$z)
PlotCope(CopeSet)
## End(Not run)
```

---

cope *Coverage Probability Excursion (CoPE) Sets*

Description

The package cope computes and plots CoPE sets defined in Sommerfeld, Sain and Schwartzman (2015) for 2D functions. CoPE sets for a real-valued target function $\mu(s)$ on a two-dimensional domain are designed to bound the excursion set $\mu(s) >= c$ from above and below with a predefined probability. The target function can be a parameter in spatially indexed linear regression $Y(s) = X * b(s) + \epsilon(s)$, where s is the spatial location, X is a known n by p design matrix, $\epsilon(s)$ is an error field and $Y(s)$ is the observed data.

Major functions

- ComputeCope Computes CoPE sets for 2D data.
- PlotCope Plots CoPE sets.

Toy example functions

- ToySignal Generates a toy signal.
- ToyNoise1, ToyNoise2, ToyNoise3 Generates realizations of toy noise fields.

References

M. Sommerfeld, S. Sain and A. Schwartzman. Confidence regions for excursion sets in asymptotically Gaussian random fields, with an application to climate. Preprint, 2015.

Examples

```
# An example using the ToySignal and the Toy Noise1 of this package.

# Set sample size.
n = 30
# Generate n realizations of the toy noise field.
Data = ToyNoise1(n = n)
# Add the toy signal to the noise.
Data$z = Data$z + rep(ToySignal()$z, n)
# Compute the CoPE sets.
CopeSet = ComputeCope(Data,level=4/3, mu=ToySignal()$z)
# Plot the result.
PlotCope(CopeSet)
```

DrawContour *Draws the contour f=c.*

Description

Draws the contour f=c.

Usage

```
DrawContour(..., level, col, lty = 1)
```

Arguments

`...`	An image. Either as a list with components x,y and z or as vectors x and y and a matrix z of dimensions c(length(x),length(y)).
`level`	The level of the contour.
`col`	Color of the contour.
`lty`	Line type for the contour.

`ImageMap` *Plots an image on a map of the world.*

Description

Plots an image on a map of the world.

Usage

```
ImageMap(lon, lat, img, mask = NULL, xlab = "Longitude",
  ylab = "Latitude", ...)
```

Arguments

`lon`	Longitude.
`lat`	Latitude.
`img`	The image to plot as a matrix of dimensions c(length(lon),length(lat)).
`mask`	Matrix of dimensions c(length(lon),length(lat)) defining a mask to cut out of the picture.
`xlab`	Label for the x-axis passed to fields::image.plot. Default is "Longitude".
`ylab`	Label for the y-axis passed to fields::image.plot. Default is "Latitude".
`...`	Additional graphical parameters passed to fields::image.plot.

`MBContour` *Computes Multiplier Bootstrap realizations of the supremum of a Gaussian field on a contour.*

Description

Computes Multiplier Bootstrap realizations of the supremum of a Gaussian field on a contour.

Usage

```
MBContour(x, y, R, cont, N = 1000)
```

Arguments

`x`	x-Coordinates of the grid on which the data is observed.
`y`	y-Coordinates of the grid on which the data is observed.
`R`	An array of dimension c(length(x),length(y),n) containing the realizations of the field.
`cont`	The contours of f at value level
`N`	The number of Bootstrap realizations to produce. Default is 1000.

Value

A vector of length N containing the Bootstrap realizations of the supremum.

`plot.cope` *Plots CoPE sets.*

Description

Plots CoPE sets.

Usage

```
## S3 method for class 'cope'
plot(x, ..., taylor = FALSE, use.true.function = FALSE,
  colc = "purple", lwdc = 3, ltyc = 1, colp = "darkred", lwdp = 3,
  ltyp = 1, colm = "darkgreen", lwdm = 3, ltym = 1,
  conlist = list(drawlabels = FALSE))
```

Arguments

`x`	An object of class cope to be plotted.
`...`	Additional graphical parameters passed to fields::image.plot.
`taylor`	Boolean indicating whether the CoPE sets with the threshold obtained by Taylor's method should be plotted. Default is FALSE.
`use.true.function`	Boolean indicating whether the threshold obtained from the true function should be used. Default is FALSE.
`colc`	Color of contour line for A_c.
`lwdc`	Width of contour line for A_c.
`ltyc`	Type of contour line for A_c.
`colp`	Color of contour line for $\hat{A}_c^+$.
`lwdp`	Width of contour line for $\hat{A}_c^+$.
`ltyp`	Type of contour line for $\hat{A}_c^+$.
`colm`	Color of contour line for $\hat{A}_c^-$.
`lwdm`	Width of contour line for $\hat{A}_c^-$.
`ltym`	Type of contour line for $\hat{A}_c^-$.
`conlist`	A list of additional arguments to pass to the `contour` function. By default, the contour labels are not shown.

References

M. Sommerfeld, S. Sain and A. Schwartzman. Confidence regions for excursion sets in asymptotically Gaussian random fields, with an application to climate. Preprint, 2015.

Examples

```
# An example using the ToyNoise and ToySignal of this package.
## Not run:
n = 30
Data = ToyNoise1(n = n)
Data$z = Data$z + rep(ToySignal()$z, n)
CopeSet = ComputeCope(Data, level=4/3, mu=ToySignal()$z)
plot(CopeSet)
## End(Not run)
```

PlotCope *Plots CoPE sets.*

Description

Plots CoPE sets.

Usage

```
PlotCope(cope, plot.taylor = FALSE, use.true.function = FALSE,
  map = FALSE, ...)
```

Arguments

`cope`	An object of class cope to be plotted.
`plot.taylor`	Boolean indicating whether the CoPE sets with the threshold obtained by Taylor's method should be plotted. Default is FALSE.
`use.true.function`	Boolean indicating whether the threshold obtained from the true function should be used. Default is FALSE.
`map`	If TRUE plot the cope set on a map of the world. The coordinates in this case are interpreted as longitude and latitude.
`...`	Additional graphical parameters passed to fields::image.plot.

TaylorContour *Computes tail probabilities of a Gaussian field on a contour with Taylor's method.*

Description

Computes tail probabilities of a Gaussian field on a contour with Taylor's method.

Usage

```
TaylorContour(x, y, cont, R)
```

Arguments

`x`	x-Coordinates of the grid on which the data is observed.
`y`	y-Coordinates of the grid on which the data is observed.
`cont`	The contour of f at value level
`R`	An array of dimension c(length(x),length(y),n) containing the realizations of the field.

Value

A function g that computes for u>0 the probility that the supremum of the field exceeds u.

`ToyNoise1` *Generate a realization of the Toy Noise 1.*

Description

Generate a realization of the Toy Noise 1.

Usage

```
ToyNoise1(n = 1, Ns = 64, model = list(), theta = 0.1, l1 = 1,
  l2 = 4, tau = 12)
```

Arguments

`n`	The number of realizations to produce.
`Ns`	Number of pixels of the result in one direction. The resulting picture will have Ns x Ns pixels.
`model`	The correlation structure of the noise, as used by arima.sim. Default is list() which gives i.i.d. noise.
`theta`	Bandwidth of kernel used to smooth the noise.
`l1, l2`	Pixel size of the noise blocks in either side of the domain. See main reference for details.
`tau`	Scaling factor with which noise is multiplied after generation.

Value

A list containing x and y, the coordinates of the grid and z and array of dimensions c(64,64,n) giving n reallizations of the Toy Noise 1.

ToyNoise1Presmooth *Generate a realization of the Toy Noise 1 before smoothing.*

Description

Generate a realization of the Toy Noise 1 before smoothing.

Usage

```
ToyNoise1Presmooth(n = 1, Ns = 64, model = list(), theta = 0.1,
  l1 = 1, l2 = 4, tau = 12)
```

Arguments

`n`	The number of realizations to produce.
`Ns`	Number of pixels of the result in one direction. The resulting picture will have Ns x Ns pixels.
`model`	The correlation structure of the noise, as used by arima.sim. Default is list() which gives i.i.d. noise.
`theta`	Bandwidth of kernel used to smooth the noise.
`l1, l2`	Pixel size of the noise blocks in either side of the domain. See main reference for details.
`tau`	Scaling factor with which noise is multiplied after generation.

Value

A list containing x and y, the coordinates of the grid and z and array of dimensions c(64,64,n) giving n reallizations of the Toy Noise 1 before smoothing.

ToyNoise2 *Generate a realization of the Toy Noise 2.*

Description

Generate a realization of the Toy Noise 2.

Usage

```
ToyNoise2(n = 1, Ns = 64, model = list(), theta = 0.1, l1 = 1,
  l2 = 4, tau = 40)
```

Arguments

`n`	The number of realizations to produce.
`Ns`	Number of pixels of the result in one direction. The resulting picture will have Ns x Ns pixels.
`model`	The correlation structure of the noise, as used by arima.sim. Default is list() which gives i.i.d. noise.
`theta`	Bandwidth of kernel used to smooth the noise.
`l1, l2`	Pixel size of the noise blocks in either side of the domain. See main reference for details.
`tau`	Scaling factor with which noise is multiplied after generation.

Value

A list containing x and y, the coordinates of the grid and z and array of dimensions c(64,64,n) giving n reallizations of the Toy Noise 2.

`ToyNoise3` *Generate a realization of the Toy Noise 3.*

Description

Generate a realization of the Toy Noise 3.

Usage

```
ToyNoise3(n = 1, Ns = 64, model = list(), theta = 0.05, l1 = 1,
  l2 = 4, tau = 10)
```

Arguments

`n`	The number of realizations to produce.
`Ns`	Number of pixels of the result in one direction. The resulting picture will have Ns x Ns pixels.
`model`	The correlation structure of the noise, as used by arima.sim. Default is list() which gives i.i.d. noise.
`theta`	Bandwidth of kernel used to smooth the noise.
`l1, l2`	Pixel size of the noise blocks in either side of the domain. See main reference for details.
`tau`	Scaling factor with which noise is multiplied after generation.

Value

A list containing x and y, the coordinates of the grid and z and array of dimensions c(64,64,n) giving n reallizations of the Toy Noise 3.

`ToyNoiseMap` *Generate an AR sequence with the ToyFUN as base noise and the AR coefficients given by the ARCoeffMap.*

Description

Generate an AR sequence with the ToyFUN as base noise and the AR coefficients given by the ARCoeffMap.

Usage

```
ToyNoiseMap(n = 1, ToyFUN, ...)
```

Arguments

`n`	Length of sequence.
`ToyFUN`	Base noise to build the sequence from.
`...`	Additional parameters passed to ToyFUN.

Value

A list containing x and y, the coordinates of the grid and z and array of dimensions c(64,64,n) giving n realizations of the Toy Noise.

`ToySignal` *Return the Toy Signal.*

Description

Return the Toy Signal.

Usage

```
ToySignal(ImRange = c(0, 1), NPixel = 64)
```

Arguments

`ImRange`	A vector with two components giving the range of the region on which the Toy Signal is to be computed.
`NPixel`	Number of pixels of the result in one direction. The resulting picture will have NPixel x NPixel pixels.

Value

A list with components "x", "y" and "z". Here, x and y are the coordinates of the grid and z is matrix of dimensions c(NPixel,NPixel) giving the Toy Signal.

`ToySignalDC` *Return the Toy Signal with discontinuities.*

Description

Return the Toy Signal with discontinuities.

Usage

```
ToySignalDC(ImRange = c(0, 1), NPixel = 64)
```

Arguments

`ImRange`	A vector with two components giving the range of the region on which the Toy Signal is to be computed.
`NPixel`	Number of pixels of the result in one direction. The resulting picture will have NPixel x NPixel pixels.

Value

A list with components "x", "y" and "z". Here, x and y are the coordinates of the grid and z is matrix of dimensions c(NPixel,NPixel) giving the Toy Signal.

`ToySlope` *The toy slope.*

Description

The toy slope.

Usage

```
ToySlope(ImRange = c(0, 1), NPixel = 64)
```

Arguments

`ImRange`	A vector with two components giving the range of the region on which the Toy Slope is to be computed.
`NPixel`	Number of pixels of the result in one direction. The resulting picture will have NPixel x NPixel pixels.

Value

A list with components "x", "y" and "z". Here, x and y are the coordinates of the grid and z is matrix of dimensions c(NPixel,NPixel) giving the Toy Signal.

Index