Package 'npsp'

January 27, 2015

Type Package

Title Nonparametric spatial (geo)statistics

Version 0.3-6

Date 2014-10-10

Author Ruben Fernandez-Casal

Maintainer Ruben Fernandez-Casal <rubenfcasal@gmail.com>

Depends R (>= 2.14.0), graphics

Imports quadprog

Suggests sp, gstat, geoR, fields, DEoptim

Description Multidimensional nonparametric spatio-temporal geostatistics.
S3 classes and methods for multidimensional: gridded data, linear binning,
local polynomial kernel regression, density and variogram estimation.
Nonparametric methods for trend and/or variogram inferences.

License GPL (>= 2)

LazyData yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-10-21 01:58:50

R topics documented:

`npsp-package` *npsp: Nonparametric spatial (geo)statistics*

Description

This package implements nonparametric methods which may be useful in geostatistical practice.

Main functions

`locpol`, `np.den` and `np.svar` use local polynomial kernel methods to compute nonparametric estimates of a multidimensional regression function, a probability density function or a semivariogram (or their first derivatives), respectively. Estimates of these functions can be constructed for any dimension (the amount of available memory is the only limitation). To speed up computations, linear binning is used to discretize the data. A full bandwidth matrix and a multiplicative triweight kernel is used to compute the weights. Main calculations are performed in FORTRAN using the LAPACK library.

`np.svariso.corr` computes a bias-corrected nonparametric semivariogram estimate using an iterative algorithm similar to that described in Fernandez-Casal and Francisco-Fernandez (2014). This procedure tries to correct the bias due to the direct use of residuals, obtained from a nonparametric estimation of the trend function, in semivariogram estimation.

`fitsvar.sb.iso` fits a 'nonparametric' isotropic Shapiro-Botha variogram model by WLS. Currently, only isotropic semivariogram estimation is supported.

There are also functions for plotting data joint with a legend representing a continuous color scale. `splot` allows to combine a standard R plot with a legend. `spoints`, `simage` and `spersp` draw the

corresponding high-level plot with a legend strip for the color scale. These functions are based on `image.plot` of package **fields**.

Among the other functions intended for direct access by the user, the following could be emphasized: `binning`, `bin.den`, `svar.bin`, `h.cv` and `interp`. There are also some functions which can be used to interact with other packages. For instance, `as.variogram` (**geoR**) or `as.vgm` (**gstat**).

Kriging is not yet implemented in this package. Users are encouraged to use `krige` (or `krige.cv`) utilities in **gstat** package together with `as.vgm`.

Acknowledgments

Important suggestions and contributions to some techniques included here were made by Tomas Cotos-Ya\~nez (Dep. Statistics, University of Vigo, Spain).

Author(s)

Ruben Fernandez-Casal (Dep. Mathematics, University of A Coru\~na, Spain). Please send comments, error reports or suggestions to `rubenfcasal@gmail.com`.

References

Fernandez-Casal R. and Francisco-Fernandez M. (2014) Nonparametric bias-corrected variogram estimation under non-constant trend, *Stoch. Environ. Res. Ris. Assess*, **28**, 1247-1259.

Fernandez-Casal R., Gonzalez-Manteiga W. and Febrero-Bande M. (2003) Flexible Spatio-Temporal Stationary Variogram Models, *Statistics and Computing*, **13**, 127-136.

Rupert D. and Wand M.P. (1994) Multivariate locally weighted least squares regression. *The Annals of Statistics*, **22**, 1346-1370.

Shapiro A. and Botha J.D. (1991) Variogram fitting with a general class of conditionally non-negative definite functions. *Computational Statistics and Data Analysis*, **11**, 87-96.

Wand M.P. (1994) Fast Computation of Multivariate Kernel Estimators. *Journal of Computational and Graphical Statistics*, **3**, 433-445.

Wand M.P. and Jones M.C. (1995) *Kernel Smoothing*. Chapman and Hall, London.

---

aquifer *Wolfcamp aquifer data*

Description

The Deaf Smith County (Texas, bordering New Mexico) was selected as an alternate site for a possible nuclear waste disposal repository in the 1980s. This site was later dropped on grounds of contamination of the aquifer, the source of much of the water supply for west Texas. In a study conducted by the U.S. Department of Energy, piezometric-head data were obtained at 85 locations (irregularly scattered over the Texas panhandle) by drilling a narrow pipe through the aquifer.

This data set has been used in numerous papers. For instance, Cressie (1989) lists the data and uses it to illustrate kriging, and Cressie (1993, section 4.1) gives a detailed description of the data and results of different geostatistical analyses.

Format

A data frame with 85 observations on the following 3 variables:

lon relative longitude position (miles).

lat relative latitude position (miles).

head piezometric-head levels (feet above sea level).

Source

Harper, W.V. and Furr, J.M. (1986) Geostatistical analysis of potentiometric data in the Wolfcamp Aquifer of the Palo Duro Basin, Texas. *Technical Report BMI/ONWI-587*, Bettelle Memorial Institute, Columbus, OH.

References

Cressie, N. (1989) Geostatistics. *The American Statistician*, **43**, 197-202.

Cressie, N. (1993) *Statistics for Spatial Data*. New York. Wiley.

Examples

```
str(aquifer)
summary(aquifer)
with(aquifer, spoints(lon, lat, head, main = "Wolfcamp aquifer"))
```

---

bin.den *Linear binning for density estimation*

Description

Creates a bin.den-class (gridded binned density) object with linear binning counts.

Usage

```
bin.den(x, nbin = NULL)

as.bin.den(object, ...)

## S3 method for class 'bin.data'
as.bin.den(object, ...)
```

Arguments

x	vector or matrix of covariates (e.g. spatial coordinates). Columns correspond with dimensions and rows with observations.
nbin	vector with the number of bins on each dimension.
object	(gridded data) used to select a method.
...	further arguments passed to or from other methods.

Details

If parameter nbin is not specified is set to rep(25, ncol(x)).

Value

Returns an S3 object of class bin.den (extends data.grid). A list with the following 3 components:

binw	vector or array (dimension nbin) with the bin counts (weights).
grid	a grid.par-class object with the grid parameters.
data	a list with a component $x with argument x.

See Also

np.den, bin.data, bin.data, locpol.

---

binning *Linear binning*

Description

Discretizes the data into a regular grid (computes a binned approximation) using the multivariate linear binning technique described in Wand (1994).

Usage

```
binning(x, y = NULL, nbin = NULL, set.NA = FALSE)

as.bin.data(object, ...)

## S3 method for class 'data.grid'
as.bin.data(object, data.ind = 1,
  weights.ind = NULL, ...)
```

Arguments

x	vector or matrix of covariates (e.g. spatial coordinates). Columns correspond with covariates (coordinate dimension) and rows with data.
y	vector of data (response variable).
nbin	vector with the number of bins on each dimension.
set.NA	logical. If TRUE, sets binning cells without data to missing.
object	(gridded data) used to select a method.
...	further arguments passed to or from other methods.
data.ind	integer or character with the index or name of the component containing the bin averages.
weights.ind	integer or character with the index or name of the component containing the bin counts/weights (if not specified, they are set to as.numeric(is.finite(object[[data.ind]]))).

Details

If parameter nbin is not specified is set to rep(25, ncol(x)).

Setting set.NA = TRUE (equivalent to biny[binw == 0] <- NA) may be useful for plotting the binned averages $biny (the hat matrix should be handled with care when using locpol).

Value

If y != NULL, an S3 object of class bin.data (gridded binned data; extends bin.den) is returned. A data.grid object with the following 4 components:

biny	vector or array (dimension nbin) with the bin averages.
binw	vector or array (dimension nbin) with the bin counts (weights).
grid	a grid.par-class object with the grid parameters.
data	a list with 3 components: • x argument x. • y argument y. • med (weighted) mean of the (binned) data.

If y == NULL, bin.den is called and a bin.den-class object is returned.

References

Wand M.P. (1994) Fast Computation of Multivariate Kernel Estimators. *Journal of Computational and Graphical Statistics*, **3**, 433-445.

See Also

data.grid, locpol, bin.den, h.cv.

Examples

```
with(earthquakes, spoints(lon, lat, mag, main = "Earthquake data"))

bin <- binning(earthquakes[, c("lon", "lat")], earthquakes$mag, nbin = c(30,30), set.NA = TRUE)

simage(bin, main = "Binning averages")
with(earthquakes, points(lon, lat, pch = 20))
```

covar *Covariance values*

Description

Computes covariance values (or pseudo-covariances) given a variogram model or covariance estimates given a semivariogram estimate.

Usage

```
covar(x, h, ...)

## S3 method for class 'svarmod'
covar(x, h, sill = x$sill, ...)

## S3 method for class 'np.svar'
covar(x, h, sill = NULL, ...)
```

Arguments

x	variogram model (svarmod object) or semivariogram estimate.
h	vector (isotropic case) or matrix of lag values.
...	further arguments passed to or from other methods.
sill	(theoretical or estimated) variance $C(0) = \sigma^2$ or pseudo-sill (unbounded variograms).

Value

A vector of (pseudo) covariance values $C(h_i) = \sigma^2 - \gamma(h_i)$ or covariance estimates.

See Also

sv, varcov.

cpu.time *Total and partial CPU time used*

Description

Returns and (optionally) prints the total and/or partial (since the last call to this function) real and CPU times.

Usage

```
cpu.time(..., reset = FALSE, total = TRUE, last = TRUE,
  flush = FALSE)
```

Arguments

...	objects (describing the last operation) to be printed (using cat), if last == TRUE.
reset	logical; if TRUE, time counters are initialized.
total	logical; if TRUE, the total time used is printed.
last	logical; if TRUE, the partial time used is printed.
flush	logical; if TRUE, flush.console is called.

Value

Invisibly returns a list with the following 3 components (objects of class "proc_time"):

time	user, system, and total elapsed times for the currently running R process (result of a call to proc.time).
last, total	differences between the corresponding proc.time calls.

See Also

proc.time, system.time, flush.console.

Examples

```
## Not run:

 cpu.time(reset=TRUE)
 res <- median(runif(100000))
 cpu.time('\nSample median of', 100000, 'values =', res)
 res <- median(runif(1000))
 cpu.time('\nSample median of', 1000, 'values =', res)

## End(Not run)
```

---

data.grid *Gridded data (S3 class "data.grid")*

Description

Defines data on a full regular (spatial) grid. Constructor function of the data.grid-class.

Usage

```
data.grid(..., grid = NULL)
```

Arguments

...	vectors or arrays of data with length equal to prod(grid$n).
grid	a grid.par-class object (optional).

Details

If parameter `grid.par` is not specified it is set from first argument.

S3 "version" of the `SpatialGridDataFrame-class` of the **sp** package.

Value

Returns an object of `class` `data.grid`, a list with the arguments as components.

See Also

`grid.par`, `binning`, `locpol`.

`disc.sb` *Discretization nodes of a Shapiro-Botha variogram model*

Description

Computes the discretization nodes of a 'nonparametric' extended Shapiro-Botha variogram model, following Gorsich and Genton (2004), as the scaled roots of Bessel functions.

Usage

```
disc.sb(nx, dk = 0, rmax = 1)
```

Arguments

`nx`	number of discretization nodes.
`dk`	dimension of the kappa function.
`rmax`	maximum lag considered.

Details

If `dk >= 1`, the nodes are computed as:

$$x_i = q_i/rmax; i = 1, \ldots, nx,$$

where q_i are the first n roots of $J_{(d-2)/2}$, J_p is the Bessel function of order p and $rmax$ is the maximum lag considered. The computation of the zeros of the Bessel function is done using the efficient algorithm developed by Ball (2000).

If `dk == 0` (corresponding to a model valid in any spatial dimension), the nodes are computed so the gaussian variogram models involved have practical ranges:

$$r_i = (1 + (i-1))rmax/nx; i = 1, \ldots, nx.$$

References

Ball, J.S. (2000) Automatic computation of zeros of Bessel functions and other special functions. *SIAM Journal on Scientific Computing*, **21**, 1458-1464.

Gorsich, D.J. and Genton, M.G. (2004) On the discretization of nonparametric covariogram estimators. *Statistics and Computing*, **14**, 99-108.

See Also

kappasb, fitsvar.sb.iso.

Examples

```
disc.sb( 12, 1, 1.0)

nx <- 1
dk <- 0
x <- disc.sb(nx, dk, 1.0)
h <- seq(0, 1, length = 100)
plot(h, kappasb(x * h, 0), type="l", ylim = c(0, 1))
abline(h = 0.05, lty = 2)
```

---

earthquakes *Earthquake data*

Description

The data set consists of 1859 earthquakes (with magnitude above or equal to 2.0 in Richter's scale), which occurred from 25 November 1944 to 16 October 2013 in the northwest (NW) part of the Iberian Peninsula. The area considered is limited by the coordinates 41N-44N and 6W-10W, which contains the autonomic region of Galicia (Spain) and northern Portugal.

Format

A data frame with 1859 observations on the following 6 variables:

date Date and time (POSIXct format).

time Time (years since first event).

lon Longitude.

lat Latitude.

depth Depth (km).

mag Magnitude (Richter's scale).

Source

National Geographic Institute (IGN) of Spain:
http://www.ign.es/ign/layout/sismologiaObtencionDatosSismiscos.do.

References

Francisco-Fernandez M., Quintela-del-Rio A. and Fernandez-Casal R. (2012) Nonparametric methods for spatial regression. An application to seismic events, *Environmetrics*, **23**, 85-93.

Examples

```
str(earthquakes)
summary(earthquakes)
with(earthquakes, spoints(lon, lat, mag, main = "Earthquake data"))
```

`fitsvar.sb.iso` *Fit an isotropic Shapiro-Botha variogram model*

Description

Fits a 'nonparametric' isotropic Shapiro-Botha variogram model by WLS through quadratic programming. Following Gorsich and Genton (2004), the nodes are selected as the scaled roots of Bessel functions (see `disc.sb`).

Usage

```
fitsvar.sb.iso(esv, dk = ncol(esv$data$x), nx = NULL,
  rmax = esv$grid$max, min.contrib = 10,
  method = c("cressie", "equal", "npairs", "linear"),
  iter = 10, tol = sqrt(.Machine$double.eps))
```

Arguments

`esv`	pilot semivariogram estimate, a `np.svar-class` (or `svar.bin`) object. Typically an output of the function `np.svariso`.
`dk`	dimension of the kappa function (`dk == 0` corresponds to a model valid in any dimension; if `dk > 0`, it should be greater than or equal to the dimension of the spatial process `ncol(esv$data$x)`).
`nx`	number of discretization nodes. Defaults to `min(nesv - 1, 50)`, where `nesv` is the number of semivariogram estimates.
`rmax`	maximum lag considered in the discretization (range of the fitted variogram on output).
`min.contrib`	minimum number of (equivalent) contributing pairs (pilot estimates with a lower number are ignored, with a warning).
`method`	string indicating the WLS fitting method to be used (e.g. `method = "cressie"`). See "Details" below.
`iter`	maximum number of interations of the WLS algorithm (used only if `method == "cressie"`).
`tol`	absolute convergence tolerance (used only if `method == "cressie"`).

Details

The fit is done using a (possibly iterated) weighted least squares criterion, minimizing:

$$WLS(\theta) = \sum_i w_i[(\hat{\gamma}(h_i)) - \gamma(\theta; h_i)]^2.$$

The different options for the argument `method` define the WLS algorithm used:

`"cressie"` The default method. The procedure is iterative, with $w_i = 1$ (OLS) used for the first step and with the weights recalculated at each iteration, following Cressie (1985), until convergence:

$$w_i = N(h_i)/\gamma(\hat{\theta}; h_i)^2,$$

where $N(h_i)$ is the (equivalent) number of contributing pairs in the estimation at lag h_i.

`"equal"` Ordinary least squares: $w_i = 1$.

`"npairs"` $w_i = N(h_i)$.

`"linear"` $w_i = N(h_i)/h_i^2$ (default fitting method in **gstat** package).

Function `solve.QP` of **quadprog** package is used to solve the quadratic programming problem. If `nx` and/or `dim(esv)` are large, this function may fail with error message "matrix D in quadratic function is not positive definite!".

Value

Returns the fitted variogram model, an object of `class` `fitsvar` with an additional component `fit` containing:

`u`	vector of lags/distances.
`sv`	vector of pilot semivariogram estimates.
`fitted.sv`	vector of fitted semivariances.
`wls`	value of the WLS objective function.
`method`	string indicating the WLS fitting method used.
`iter`	number of WLS iterations (if `method == "cressie"`).

References

Ball, J.S. (2000) Automatic computation of zeros of Bessel functions and other special functions. *SIAM Journal on Scientific Computing*, **21**, 1458-1464.

Cressie, N. (1985) Fitting variogram models by weighted least squares. *Mathematical Geology*, **17**, 563-586.

Cressie, N. (1993) *Statistics for Spatial Data*. New York. Wiley.

Fernandez Casal R., Gonzalez Manteiga W. and Febrero Bande M. (2003) Flexible Spatio-Temporal Stationary Variogram Models, *Statistics and Computing*, **13**, 127-136.

Gorsich, D.J. and Genton, M.G. (2004) On the discretization of nonparametric covariogram estimators. *Statistics and Computing*, **14**, 99-108.

Shapiro, A. and Botha, J.D. (1991) Variogram fitting with a general class of conditionally non-negative definite functions. *Computational Statistics and Data Analysis*, **11**, 87-96.

See Also

svarmod.sb.iso, disc.sb, plot.fitsvar.

---

grid.par *Grid parameters (S3 class "grid.par")*

Description

Defines a full regular (spatial) grid. Constructor function of the grid.par-class.

Usage

```
grid.par(n, min, max = min + (n - 1) * lag,
  lag = (max - min)/(n - 1), dimnames = names(min))
```

Arguments

n	integer vector; number of nodes in each dimension.
min	vector; minimun coordinates values.
max	vector; maximum coordinates values (optional).
lag	vector; lag in each dimension (optional).
dimnames	character vector; names used to label the dimensions.

Details

All parameters must have the same length. Only one of the arguments max or lag must be specified.

S3 'version' of the GridTopology-class of the **sp** package.

Value

Returns an object of class grid.par, a list with the arguments as components and an additional component $nd = length(n).

See Also

data.grid.

Examples

```
grid.par(n = c(100, 100), min = c(-10, 42), max = c(-7.5, 44))
grid.par(n = c(100, 100), min = c(-10, 42), lag = c(0.03, 0.02))
```

h.cv *Cross-validation methods for bandwidth selection*

Description

Selects the bandwidth of a local polynomial kernel (regression, density or variogram) estimator using (standart or modified) CV, GCV or MASE criteria.

Usage

```
 h.cv(bin, ...)

 ## S3 method for class 'bin.data'
h.cv(bin,
   objective = c("CV", "GCV", "MASE"), h.start = NULL,
   h.lower = NULL, h.upper = NULL, degree = 1,
   ncv = ifelse(objective == "GCV", 0, 1), cov.bin = NULL,
   DEalgorithm = FALSE, warn = FALSE, ...)

 ## S3 method for class 'bin.den'
h.cv(bin, h.start = NULL,
   h.lower = NULL, h.upper = NULL, degree = 1, ncv = 1,
   DEalgorithm = FALSE, warn = FALSE, ...)

 hcv.data(bin, objective = c("CV", "GCV", "MASE"),
   h.start = NULL, h.lower = NULL, h.upper = NULL,
   degree = 1, ncv = ifelse(objective == "GCV", 0, 1),
   cov = NULL, DEalgorithm = FALSE, warn = FALSE, ...)
```

Arguments

bin	object used to select a method (binned data, binned density or binned semivariogram).
...	further arguments passed to or from other methods (e.g. parameters of the optimization routine).
objective	character; optimal criterion to be used ("CV", "GCV" or "MASE").
h.start	vector; initial values for the parameters (diagonal elements) to be optimized over. If `DEalgorithm == FALSE` (otherwise not used), defaults to `(3 + ncv) * lag`, where `lag = bin$grid$lag`.
h.lower	vector; lower bounds on each parameter (diagonal elements) to be optimized. Defaults to `(1.5 + ncv) * bin$grid$lag`.
h.upper	vector; upper bounds on each parameter (diagonal elements) to be optimized. Defaults to `1.5 * dim(bin) * bin$grid$lag`.
DEalgorithm	logical; if `TRUE`, the differential evolution optimization algorithm in package **DEoptim** is used.

ncv	integer; determines the number of cells leaved out in each dimension. (0 to GCV considering all the data, > 0 to traditional or modified cross-validation). See "Details" bellow.
cov.bin	covariance matrix of the binned data. Defaults to identity.
warn	logical; sets the handling of warning messages (normally due to the lack of data in some neighborhoods). If FALSE (the default) all warnings are ignored.
cov	covariance matrix of the data. Defaults to identity (uncorrelated data).
degree	degree of the local polynomial used. Defaults to 1 (local linear estimation).

Details

Currently, only diagonal windows are supported.

h.cv methods use binning approximations to the objective function values. If ncv > 0, estimates are computed by leaving out binning cells with indexes within the intervals $[x_i - ncv + 1, x_i + ncv - 1]$, at each dimension i, where x denotes the index of the estimation position. $ncv = 1$ corresponds with traditional cross-validation and $ncv > 1$ with modified CV (see e.g. Chu and Marron, 1991, for the one dimensional case). For standard GCV, set ncv = 0 (the full data is used). For theoretical MASE, set y = trend.teor, cov = cov.teor and ncv = 0.

If DEalgorithm == FALSE, the "L-BFGS-B" method in optim is used.

hcv.data evaluates the objective functions at the original data (combining a binning approximation to the nonparametric estimates with a linear interpolation). If ncv > 1 (modified CV), a similar algorithm to that in h.cv.bin.data is used, estimates are computed by leaving out binning cells with indexes within the intervals $[x_i - ncv + 1, x_i + ncv - 1]$.

Value

Returns a list containing the following 3 components:

h	the best (diagonal) bandwidth matrix found.
value	the value of the objective function corresponding to h.
objective	the criterion used.

References

Chu, C.K. and Marron, J.S. (1991) Comparison of Two Bandwidth Selectors with Dependent Errors. *The Annals of Statistics*, **19**, 1906-1918.

Francisco-Fernandez M. and Opsomer J.D. (2005) Smoothing parameter selection methods for non-parametric regression with spatially correlated errors. *Canadian Journal of Statistics*, **33**, 539-558.

See Also

locpol, locpolhcv, binning, np.svar.

Examples

```
bin <- binning(earthquakes[, c("lon", "lat")], earthquakes$mag, nbin = c(30,30))
hcv <- h.cv(bin, ncv = 2)
lp <- locpol(bin, h = hcv$h)
## Alternatively:
## lp <- locpolhcv(earthquakes[, c("lon", "lat")], earthquakes$mag, nbin = c(30,30), ncv = 2)

simage(lp, main = 'Smoothed magnitude')
contour(lp, add = TRUE)
with(earthquakes, points(lon, lat, pch = 20))

## Density estimation
hden <- h.cv(as.bin.den(bin))
den <- np.den(bin, h = hden$h)

plot(den, main = 'Estimated log(density)')
```

`interp` *Fast linear interpolation of a regular grid*

Description

Computes a linear interpolation of multidimensional regularly gridded data.

Usage

```
 interp(object, ...)

 ## S3 method for class 'grid.par'
interp(object, data, newx, ...)

 ## S3 method for class 'data.grid'
interp(object, data.ind = 1, newx,
   ...)

 ## S3 method for class 'locpol.bin'
predict(object, newx = NULL,
   hat.data = FALSE, ...)
```

Arguments

`object`	(gridded data) object used to select a method.
`...`	further arguments passed to or from other methods.
`data`	vector or array of data values.
`newx`	vector or matrix with the (irregular) locations to interpolate. Columns correspond with dimensions and rows with data.
`data.ind`	integer or character with the index or name of the data component.

hat.data	logical; if TRUE (and possible), the hat matrix corresponding to the (original) data is returned.

Details

interp methods are interfaces to the fortran routine interp_data_grid (in grid_module.f90).

predict.locpol.bin is an interface to the fortran routine predict_lp (in lp_module.f90).

Value

A list with two components:

x	interpolation locations.
y	interpolated values.

If newx == NULL, predict.locpol.bin returns the estimates (and optionally the hat matrix) corresponding to the data (otherwise interp.data.grid is called).

Note

Linear extrapolation is performed from the end nodes of the grid.

WARNING: May fail with missing values (especially if object$locpol$ncv > 0).

See Also

interp.surface.

---

kappasb *Coefficients of an extended Shapiro-Botha variogram model*

Description

Computes the coefficients of an extended Shapiro-Botha variogram model.

Usage

```
kappasb(x, dk = 0)
```

Arguments

x	numeric vector (on which the kappa function will be evaluated).
dk	dimension of the kappa function.

Details

If dk >= 1, the coefficients are computed as:

$$\kappa_d(x) = (2/x)^{(d-2)/2}\Gamma(d/2)J_{(d-2)/2}(x)$$

where J_p is the Bessel function of order p.
If dk == 0, the coefficients are computed as:

$$\kappa_\infty(x) = e^{-x^2}$$

(corresponding to a model valid in any spatial dimension).
NOTE: some authors denote these functions as Ω_d.

Value

A vector with the coefficients of an extended Shapiro-Botha variogram model.

References

Shapiro, A. and Botha, J.D. (1991) Variogram fitting with a general class of conditionally non-negative definite functions. *Computational Statistics and Data Analysis*, **11**, 87-96.

See Also

svarmod.sb.iso, besselJ.

Examples

```
kappasb(seq(0, 6*pi, len = 10), 2)

curve(kappasb(x/5, 0), xlim = c(0, 6*pi), ylim = c(-1, 1), lty = 2)
for (i in 1:10) curve(kappasb(x, i), col = gray((i-1)/10), add = TRUE)
abline(h = 0, lty = 3)
```

---

locpol *Local polynomial estimation*

Description

Estimates a multidimensional regression function (and its first derivatives) using local polynomial kernel smoothing of linearly binned data.

Usage

```
locpol(x, ...)

## Default S3 method:
locpol(x, y, h = NULL, nbin = NULL,
  degree = 1 + as.numeric(drv), drv = FALSE,
  hat.bin = FALSE, ncv = 0, set.NA = FALSE, ...)

## S3 method for class 'bin.data'
locpol(x, h = NULL,
  degree = 1 + as.numeric(drv), drv = FALSE,
  hat.bin = FALSE, ncv = 0, ...)

## S3 method for class 'svar.bin'
locpol(x, h = NULL, degree = 1,
  drv = FALSE, hat.bin = TRUE, ncv = 0, ...)

## S3 method for class 'bin.den'
locpol(x, h = NULL,
  degree = 1 + as.numeric(drv), drv = FALSE, ncv = 0,
  ...)

locpolhcv(x, y, nbin = NULL,
  objective = c("CV", "GCV", "MASE"),
  degree = 1 + as.numeric(drv), drv = FALSE,
  hat.bin = FALSE, set.NA = FALSE,
  ncv = ifelse(objective == "GCV", 0, 1), cov = NULL,
  ...)
```

Arguments

x	a (data) object used to select a method.
...	further arguments passed to or from other methods (e.g. to hcv.data).
y	vector of data (response variable).
h	(full) bandwidth matrix (controls the degree of smoothing).
nbin	vector with the number of bins on each dimension.
degree	degree of the local polynomial used. Defaults to 1 (local linear estimation).
drv	logical; if TRUE, the matrix of estimated first derivatives is returned.
hat.bin	logical; if TRUE, the hat matrix of the binned data is returned.
ncv	integer; determines the number of cells leaved out in each dimension. Defaults to 0 (the full data is used) and it is not normally changed by the user in this setting. See "Details" below.
set.NA	logical. If TRUE, sets binning cells without data to missing.
objective	character; optimal criterion to be used ("CV", "GCV" or "MASE").
cov	covariance matrix of the data. Defaults to identity (uncorrelated data).

Details

Standard generic function with a default method (interface to the fortran routine `lp_raw`), in which argument `x` is a vector or matrix of covariates (e.g. spatial coordinates).

If parameter `nbin` is not specified is set to `rep(25, ncol(x))`.

A multiplicative triweight kernel is used to compute the weights.

If `ncv > 0`, estimates are computed by leaving out cells with indexes within the intervals $[x_i - ncv + 1, x_i + ncv - 1]$, at each dimension i, where x denotes the index of the estimation position. $ncv = 1$ corresponds with traditional cross-validation and $ncv > 1$ with modified CV (see e.g. Chu and Marron, 1991, for the one dimensional case).

Setting `set.NA = TRUE` (equivalent to `biny[binw == 0] <- NA`) may be useful for plotting the binned averages `$biny` (the hat matrix should be handled with care).

`locpolhcv` calls `hcv.data` to obtain an "optimal" bandwith (additional arguments `...` are passed to this function). Argument `ncv` is only used here at the bandwith selection stage (estimation is done with all the data).

Value

Returns an S3 object of class `locpol.bin` (locpol + bin data + grid par.). A `bin.data` object with the additional (some optional) 3 components:

`est` vector or array (dimension `nbin`) with the local polynomial estimates.

`locpol` a list with 7 components:

- `degree` degree of the polinomial.
- `h` bandwidth matrix.
- `rm` residual mean.
- `rss` sum of squared residuals.
- `ncv` number of cells ignored in each direction.
- `hat` (if requested) hat matrix of the binned data.
- `nrl0` (if appropriate) number of cells with data (`binw > 0`) and missing estimate (`est == NA`).

`deriv` (if requested) matrix of first derivatives.

`locpol.svar.bin` returns an S3 object of class `np.svar` (locpol semivar + bin semivar + grid par.).

`locpol.bin.den` returns an S3 object of class `np.den` (locpol den + bin den + grid par.).

References

Chu, C.K. and Marron, J.S. (1991) Comparison of Two Bandwidth Selectors with Dependent Errors. *The Annals of Statistics*, **19**, 1906-1918.

Rupert D. and Wand M.P. (1994) Multivariate locally weighted least squares regression. *The Annals of Statistics*, **22**, 1346-1370.

See Also

`binning`, `data.grid`, `np.svariso`, `svar.bin`, `np.den`, `bin.den`, `hcv.data`.

Examples

```
lp <- locpol(earthquakes[, c("lon", "lat")], earthquakes$mag, h = diag(2, 2), nbin = c(41,41))
simage(lp, main = "Smoothed magnitude")
contour(lp, add = TRUE)

bin <- binning(earthquakes[, c("lon", "lat")], earthquakes$mag, nbin = c(41,41))
lp2 <- locpol(bin, h = diag(2, 2))
all.equal(lp, lp2)

## Alternatively:
## lp <- locpolhcv(earthquakes[, c("lon", "lat")], earthquakes$mag, ncv = 4)

den <- locpol(as.bin.den(bin), h = diag(1, 2))
plot(den, log = FALSE, main = 'Estimated density')
```

---

np.den *local polynomial density estimation*

Description

Estimates a multidimensional probability density function (and its first derivatives) using local polynomial kernel smoothing of linearly binned data.

Usage

```
np.den(x, ...)

## Default S3 method:
np.den(x, nbin = NULL, h = NULL,
   degree = 1 + as.numeric(drv), drv = FALSE, ncv = 0,
   ...)

## S3 method for class 'bin.den'
np.den(x, h = NULL,
   degree = 1 + as.numeric(drv), drv = FALSE, ncv = 0,
   ...)

## S3 method for class 'bin.data'
np.den(x, h = NULL,
   degree = 1 + as.numeric(drv), drv = FALSE, ncv = 0,
   ...)

## S3 method for class 'svar.bin'
np.den(x, h = NULL,
   degree = 1 + as.numeric(drv), drv = FALSE, ncv = 0,
   ...)
```

Arguments

x	a (data) object used to select a method.
...	further arguments passed to or from other methods.
nbin	vector with the number of bins on each dimension.
h	(full) bandwidth matrix (controls the degree of smoothing).
degree	degree of the local polynomial used. Defaults to 1 (local linear estimation).
drv	logical; if TRUE, the matrix of estimated first derivatives is returned.
ncv	integer; determines the number of cells leaved out in each dimension. Defaults to 0 (the full data is used) and it is not normally changed by the user in this setting. See "Details" below.

Details

Standard generic function with a default method (interface to the fortran routine lp_data_grid), in which argument x is a vector or matrix of covariates (e.g. spatial coordinates). In this case, the data are binned (calls bin.den) and the local fitting procedure is applied to the scaled bin counts (calls np.den.bin.den).

If parameter nbim is not specified is set to rep(25, ncol(x)).

A multiplicative triweight kernel is used to compute the weights.

If ncv > 1, estimates are computed by leaving out cells with indexes within the intervals $[x_i - ncv + 1, x_i + ncv - 1]$, at each dimension i, where x denotes the index of the estimation position.

Value

Returns an S3 object of class np.den (locpol den + bin den + grid par.). A bin.den object with the additional (some optional) 3 components:

est	vector or array (dimension nbin) with the local polynomial density estimates.
locpol	a list with 6 components: • degree degree of the polinomial. • h bandwidth matrix. • rm residual mean (of the escaled bin counts). • rss sum of squared residuals (of the escaled bin counts). • ncv number of cells ignored (in each dimension).
deriv	(if requested) matrix of first derivatives.

References

Wand, M.P. and Jones, M.C. (1995) *Kernel Smoothing*. Chapman and Hall, London.

See Also

bin.den, binning, data.grid.

Examples

```
bin <- binning(earthquakes[, c("lon", "lat")], earthquakes$mag, nbin = c(30,30))
hden <- h.cv(as.bin.den(bin))
den <- np.den(bin, h = hden$h)
## Equivalent to:
## den <- np.den(earthquakes[, c("lon", "lat")], h = hden$h, nbin = c(30,30))

plot(den, main = 'Estimated log(density)')
```

---

np.svar *Local polynomial estimation of the semivariogram*

Description

Estimates a multidimensional semivariogram (and its first derivatives) using local polynomial kernel smoothing of linearly binned semivariances.

Usage

```
np.svar(x, ...)

## Default S3 method:
np.svar(x, y, h = NULL, maxlag = NULL,
   nlags = NULL, minlag = maxlag/nlags, degree = 1,
   drv = FALSE, hat.bin = TRUE, ncv = 0, ...)

## S3 method for class 'svar.bin'
np.svar(x, h = NULL, degree = 1,
   drv = FALSE, hat.bin = TRUE, ncv = 0, ...)

np.svariso(x, y, h = NULL, maxlag = NULL, nlags = NULL,
  minlag = maxlag/nlags, degree = 1, drv = FALSE,
  hat.bin = TRUE, ncv = 0, ...)

np.svariso.hcv(x, y, maxlag = NULL, nlags = NULL,
  minlag = maxlag/nlags, degree = 1, drv = FALSE,
  hat.bin = TRUE, objective = c("CV", "GCV", "MASE"),
  ncv = ifelse(objective == "GCV", 0, 1), cov.bin = NULL,
  ...)

np.svariso.corr(lp, x = lp$data$x, h = NULL,
  maxlag = NULL, nlags = NULL, minlag = maxlag/nlags,
  degree = 1, drv = FALSE, hat.bin = TRUE, tol = 0.05,
  max.iter = 10, plot = FALSE,
  ylim = c(0, 2 * max(svar$biny, na.rm = TRUE)))
```

Arguments

x	object used to select a method. Usually a matrix with the coordinates of the data locations (columns correspond with dimensions and rows with data).
...	further arguments passed to or from other methods.
y	vector of data (response variable).
maxlag	maximum lag. Defaults to 55% of largest lag.
nlags	number of lags. Defaults to 101.
minlag	minimun lag.
hat.bin	logical; if TRUE, the hat matrix of the binned semivariances is returned.
cov.bin	covariance matrix of the binned semivariances. Defaults to identity.
lp	local polynomial estimate of the trend function (object of class locpol.bin).
tol	convergence tolerance. The algorithm stops if the average of the relative squared diferences is less than tol. Defaults to 0.04.
max.iter	maximum number of iterations. Defaults to 10.
plot	logical; if TRUE, the estimates obtained at each iteration are plotted.
ylim	y-limits of the plot (if plot == TRUE).
h	(full) bandwidth matrix (controls the degree of smoothing).
degree	degree of the local polynomial used. Defaults to 1 (local linear estimation).
drv	logical; if TRUE, the matrix of estimated first derivatives is returned.
ncv	integer; determines the number of cells leaved out in each dimension. Defaults to 0 (the full data is used) and it is not normally changed by the user in this setting. See "Details" below.
objective	character; optimal criterion to be used ("CV", "GCV" or "MASE").

Details

Currently, only isotropic semivariogram estimation is supported.

If parameter nlags is not specified is set to 101.

The computation of the hat matrix of the binned semivariances (hat.bin = TRUE) allows for the computation of approximated estimation variances (e.g. in fitsvar.sb.iso).

A multiplicative triweight kernel is used to compute the weights.

np.svariso.hcv calls h.cv to obtain an "optimal" bandwith (additional arguments ... are passed to this function). Argument ncv is only used here at the bandwith selection stage (estimation is done with all the data).

np.svariso.corr computes a bias-corrected nonparametric semivariogram estimate using an iterative algorithm similar to that described in Fernandez-Casal and Francisco-Fernandez (2014). This procedure tries to correct the bias due to the direct use of residuals (obtained in this case from a nonparametric estimation of the trend function) in semivariogram estimation.

Value

Returns an S3 object of class `np.svar` (locpol svar + binned svar + grid par.), extends `svar.bin`, with the additional (some optional) 3 components:

`est`	vector or array with the local polynomial semivariogram estimates.
`locpol`	a list of 6 components: • `degree` degree of the local polinomial used. • `h` smoothing matrix. • `rm` mean of residual semivariances. • `rss` sum of squared residual semivariances. • `ncv` number of cells ignored in each direction. • `hat` (if requested) hat matrix of the binned semivariances. • `nrl0` (if appropriate) number of cells with `binw > 0` and `est == NA`.
`deriv`	(if requested) matrix of estimated first semivariogram derivatives.

References

Fernandez Casal R., Gonzalez Manteiga W. and Febrero Bande M. (2003) Space-time dependency modeling using general classes of flexible stationary variogram models, *J. Geophys. Res.*, **108**, 8779, doi:10.1029/2002JD002909.

Garcia-Soidan P.H., Gonzalez-Manteiga W. and Febrero-Bande M. (2003) Local linear regression estimation of the variogram, *Stat. Prob. Lett.*, **64**, 169-179.

Fernandez-Casal R. and Francisco-Fernandez M. (2014) Nonparametric bias-corrected variogram estimation under non-constant trend, *Stoch. Environ. Res. Ris. Assess*, **28**, 1247-1259.

See Also

`svar.bin`, `data.grid`, `locpol`.

`npsp-geoR` *Interface to package "geoR"*

Description

Utilities to interact with the **geoR** package.

Usage

```
 as.variogram(x, ...)

 ## S3 method for class 'svar.bin'
as.variogram(x, ...)

 ## S3 method for class 'np.svar'
as.variogram(x, ...)
```

```
as.variomodel(m, ...)

## S3 method for class 'svarmod'
as.variomodel(m, ...)
```

Arguments

x	semivariogram estimate (e.g. svar.bin or np.svar object).
m	variogram model (e.g. svarmod object).
...	further arguments passed to or from other methods.

Details

as.variogram tries to convert a semivariogram estimate $\hat{\gamma}(h_i)$ to an object of the (not fully documented) **geoR**-class variogram (see e.g. variog).

as.variomodel tries to convert a semivariogram model $\gamma(pars; h)$ to an object of the **geoR**-class variomodel (see e.g. variofit).

See Also

variog, variofit, variomodel, svar.bin, np.svar.

---

npsp-gstat *Interface to package "gstat"*

Description

Utilities to interact with the **gstat** package.

Usage

```
as.vgm(x, ...)

## S3 method for class 'variomodel'
as.vgm(x, ...)

## S3 method for class 'svarmod'
as.vgm(x, ...)

vgm.tab.svarmod(x, h = seq(0, x$range, length = 1000),
  sill = x$sill, ...)

## S3 method for class 'sb.iso'
as.vgm(x,
  h = seq(0, x$range, length = 1000), sill = x$sill, ...)
```

Arguments

x	variogram model object (used to select a method).
...	further arguments passed to or from other methods.
h	vector of lags at which the covariogram is evaluated.
sill	sill of the covariogram (or pseudo-sill).

Details

Tries to convert a variogram object to vgm (variogramModel-class of **gstat** package). S3 generic function.

as.vgm.variomodel tries to convert an object of class variomodel defined in **geoR** (interface to as.vgm.variomodel defined in **gstat**).

vgm.tab.svarmod converts a svarmod object to a variogramModel-class object of type "Tab" (one-dimensional covariance table).

as.vgm.sb.iso is an alias of vgm.tab.svarmod.

See Also

vgm, svarmod.

---

rgraphics *R Graphics for gridded data*

Description

Draw an image, perspective, contour or filled contour plot for data on a bidimensional regular grid (S3 methods for class "codedata.grid").

Usage

```
 ## S3 method for class 'data.grid'
image(x, data.ind = 1, xlab = NULL,
   ylab = NULL, ...)

 ## S3 method for class 'data.grid'
persp(x, data.ind = 1, xlab = NULL,
   ylab = NULL, zlab = NULL, ...)

 ## S3 method for class 'data.grid'
contour(x, data.ind = 1,
   filled = FALSE, xlab = NULL, ylab = NULL, ...)
```

Arguments

x	a "codedata.grid"-class object.
data.ind	integer or character with the index or name of the component containing the values to be used for coloring the rectangles.
xlab	label for the x axis, defaults to dimnames(x)[1].
ylab	label for the y axis, defaults to dimnames(x)[2].
zlab	label for the z axis, defaults to names(x)[data.ind].
...	additional graphical parameters (to be passed to main plot function).
filled	logical; if FALSE (default), function contour is called, otherwise filled.contour.

See Also

image, persp, contour, filled.contour, data.grid.

Examples

```
# Regularly spaced 2D data
grid <- grid.par(n = c(50, 50), min = c(-1, -1), max = c(1, 1))

f2d <- function(x) x[1]^2 - x[2]^2
trend <- apply(coords(grid), 1, f2d)
set.seed(1)
y <- trend + rnorm(prod(dim(grid)), 0, 0.1)
gdata <- data.grid(trend = trend, y = y, grid = grid)

# perspective plot
persp(gdata, main = 'Trend', theta = 40, phi = 20, ticktype = "detailed")

# filled contour plot
contour(gdata, main = 'Trend', filled = TRUE, color.palette = jet.colors)

# Multiple plots with a common legend:
scale.range <- c(-1.2, 1.2)
scale.color <- jet.colors(64)
# 1x2 plot with some room for the legend...
old.par <- par(mfrow = c(1,2), omd = c(0.05, 0.85, 0.05, 0.95))
image(gdata, zlim = scale.range, main = 'Trend', col = scale.color)
contour(gdata, add = TRUE)
image(gdata, 'y', zlim = scale.range, main = 'Data', col = scale.color)
contour(gdata, 'y', add = TRUE)
par(old.par)
# the legend can be added to any plot...
splot(slim = scale.range, col = scale.color, add = TRUE)
```

simage *Image plot with a color scale*

Description

simage (generic function) draws an image (a grid of colored rectangles) and (optionally) adds a legend strip with the color scale (calls splot and image).

plot.np.den calls simage.data.grid (contour and points also by default).

Usage

```
 simage(x, ...)

 ## Default S3 method:
simage(x = seq(0, 1, len = nrow(s)),
   y = seq(0, 1, len = ncol(s)), s,
   slim = range(s, finite = TRUE), col = jet.colors(128),
   breaks = NULL, legend = TRUE, horizontal = FALSE,
   legend.shrink = 0.8, legend.width = 1.2,
   legend.mar = ifelse(horizontal, 3.1, 5.1),
   legend.lab = NULL, bigplot = NULL, smallplot = NULL,
   lab.breaks = NULL, axis.args = NULL,
   legend.args = NULL, graphics.reset = FALSE,
   xlab = NULL, ylab = NULL, ...)

 ## S3 method for class 'data.grid'
simage(x, data.ind = 1, xlab = NULL,
   ylab = NULL, ...)

 ## S3 method for class 'np.den'
plot(x, y = NULL, log = TRUE,
   contour = TRUE, points = TRUE, col = hot.colors(128),
   ...)
```

Arguments

x	grid values for x coordinate. If x is a list, its components x$x and x$y are used for x and y, respectively. For compatibility with image, if the list has component z this is used for s.
y	grid values for y coordinate.
s	matrix containing the values to be used for coloring the rectangles (NAs are allowed). Note that x can be used instead of s for convenience.
legend	logical; if TRUE (default), the plotting region is splitted into two parts, drawing the image plot in one and the legend with the color scale in the other. If FALSE only the image plot is drawn and the arguments related to the legend are ignored (splot is not called).

...	additional graphical parameters (to be passed to image or simage.default; e.g. xlim, ylim, ...). NOTE: graphical arguments passed here will only have impact on the main plot. To change the graphical defaults for the legend use the par function beforehand (e.g. par(cex.lab = 2) to increase colorbar labels).
data.ind	integer or character with the index or name of the component containing the values to be used for coloring the rectangles.
log	logical; if TRUE (default), log(x$est) is ploted.
contour	logical; if TRUE (default), contour lines are added.
points	logical; if TRUE (default), points at x$data$x are drawn.
slim	limits used to set up the color scale.
col	color table used to set up the color scale (see image for details).
breaks	(optional) numeric vector with the breakpoints for the color scale: must have one more breakpoint than col and be in increasing order.
horizontal	logical; if FALSE (default) legend will be a vertical strip on the right side. If TRUE the legend strip will be along the bottom.
legend.shrink	amount to shrink the size of legend relative to the full height or width of the plot.
legend.width	width in characters of the legend strip. Default is 1.2, a little bigger that the width of a character.
legend.mar	width in characters of legend margin that has the axis. Default is 5.1 for a vertical legend and 3.1 for a horizontal legend.
legend.lab	label for the axis of the color legend. Default is no label as this is usual evident from the plot title.
bigplot	plot coordinates for main plot. If not passed these will be determined within the function.
smallplot	plot coordinates for legend strip. If not passed these will be determined within the function.
lab.breaks	if breaks are supplied these are text string labels to put at each break value. This is intended to label axis on a transformed scale such as logs.
axis.args	additional arguments for the axis function used to create the legend axis (see image.plot for details).
legend.args	arguments for a complete specification of the legend label. This is in the form of list and is just passed to the mtext function. Usually this will not be needed (see image.plot for details).
graphics.reset	logical; if FALSE (default) the plotting region (par("plt")) will not be reset to make it possible to add more features to the plot (e.g. using functions such as points or lines). If TRUE will reset plot parameters to the values before entering the function.
xlab	label for the x axis, defaults to a description of x.
ylab	label for the y axis, defaults to a description of y.

Value

Invisibly returns a list with the following 3 components:

`bigplot`	plot coordinates of the main plot. These values may be useful for drawing a plot without the legend that is the same size as the plots with legends.
`smallplot`	plot coordinates of the secondary plot (legend strip).
`old.par`	previous graphical parameters (`par(old.par)` will reset plot parameters to the values before entering the function).

Side Effects

After exiting, the plotting region may be changed (`par("plt")`) to make it possible to add more features to the plot (set `graphics.reset = FALSE` to avoid this).

Author(s)

Based on `image.plot` function from package **fields**: fields, Tools for spatial data. Copyright 2004-2013, Institute for Mathematics Applied Geosciences. University Corporation for Atmospheric Research.

Modified by Ruben Fernandez-Casal <rubenfcasal@gmail.com>.

See Also

`splot`, `spoints`, `spersp`, `image`, `image.plot`, `data.grid`.

Examples

```
#
# Regularly spaced 2D data
nx <- c(40, 40) # ndata =  prod(nx)
x1 <- seq(-1, 1, length.out = nx[1])
x2 <- seq(-1, 1, length.out = nx[2])
trend <- outer(x1, x2, function(x,y) x^2 - y^2)
simage( x1, x2, trend, main = 'Trend')

#
# Multiple plots
set.seed(1)
y <- trend + rnorm(prod(nx), 0, 0.1)
x <- as.matrix(expand.grid(x1 = x1, x2 = x2)) # two-dimensional grid
# local polynomial kernel regression
lp <- locpol(x, y, nbin = nx, h =  diag(c(0.3, 0.3)))
# 1x2 plot
old.par <- par(mfrow = c(1,2))
simage( x1, x2, y, main = 'Data')
simage(lp, main = 'Estimated trend')
par(old.par)
```

spersp *Perspective plot with a color scale*

Description

spersp (generic function) draws a perspective plot of a surface over the x-y plane with the facets being filled with different colors and (optionally) adds a legend strip with the color scale (calls splot and persp).

Usage

```
spersp(x, ...)

## Default S3 method:
spersp(x = seq(0, 1, len = nrow(z)),
   y = seq(0, 1, len = ncol(z)), z, s = z,
   slim = range(s, finite = TRUE), col = jet.colors(128),
   breaks = NULL, legend = TRUE, horizontal = FALSE,
   legend.shrink = 0.8, legend.width = 1.2,
   legend.mar = ifelse(horizontal, 3.1, 5.1),
   legend.lab = NULL, bigplot = NULL, smallplot = NULL,
   lab.breaks = NULL, axis.args = NULL,
   legend.args = NULL, graphics.reset = FALSE,
   xlab = NULL, ylab = NULL, zlab = NULL, theta = 40,
   phi = 20, ticktype = "detailed", cex.axis = 0.75, ...)

## S3 method for class 'data.grid'
spersp(x, data.ind = 1,
   s = x[[data.ind]], xlab = NULL, ylab = NULL,
   zlab = NULL, ...)
```

Arguments

x	grid values for x coordinate. If x is a list, its components x$x and x$y are used for x and y, respectively. If the list has component z this is used for z.
y	grid values for y coordinate.
z	matrix containing the values to be plotted (NAs are allowed). Note that x can be used instead of z for convenience.
s	matrix containing the values used for coloring the facets.
legend	logical; if TRUE (default), the plotting region is splitted into two parts, drawing the perspective plot in one and the legend with the color scale in the other. If FALSE only the (coloured) perspective plot is drawn and the arguments related to the legend are ignored (splot is not called).
zlab	label for the z axis, defaults to a description of z.
theta	x-y rotation angle for perspective (azimuthal direction).

phi	z-angle for perspective (colatitude).
ticktype	character; "simple" draws just an arrow parallel to the axis to indicate direction of increase; "detailed" draws normal ticks as per 2D plots.
cex.axis	magnification to be used for axis annotation (relative to the current setting of par("cex")).
...	additional graphical parameters (to be passed to persp or spersp.default; e.g. xlim, ylim, zlim, ...). NOTE: graphical arguments passed here will only have impact on the main plot. To change the graphical defaults for the legend use the par function beforehand (e.g. par(cex.lab = 2) to increase colorbar labels).
data.ind	integer or character with the index or name of the component containing the z values to be plotted.
slim	limits used to set up the color scale.
col	color table used to set up the color scale (see image for details).
breaks	(optional) numeric vector with the breakpoints for the color scale: must have one more breakpoint than col and be in increasing order.
horizontal	logical; if FALSE (default) legend will be a vertical strip on the right side. If TRUE the legend strip will be along the bottom.
legend.shrink	amount to shrink the size of legend relative to the full height or width of the plot.
legend.width	width in characters of the legend strip. Default is 1.2, a little bigger that the width of a character.
legend.mar	width in characters of legend margin that has the axis. Default is 5.1 for a vertical legend and 3.1 for a horizontal legend.
legend.lab	label for the axis of the color legend. Default is no label as this is usual evident from the plot title.
bigplot	plot coordinates for main plot. If not passed these will be determined within the function.
smallplot	plot coordinates for legend strip. If not passed these will be determined within the function.
lab.breaks	if breaks are supplied these are text string labels to put at each break value. This is intended to label axis on a transformed scale such as logs.
axis.args	additional arguments for the axis function used to create the legend axis (see image.plot for details).
legend.args	arguments for a complete specification of the legend label. This is in the form of list and is just passed to the mtext function. Usually this will not be needed (see image.plot for details).
graphics.reset	logical; if FALSE (default) the plotting region (par("plt")) will not be reset to make it possible to add more features to the plot (e.g. using functions such as points or lines). If TRUE will reset plot parameters to the values before entering the function.
xlab	label for the x axis, defaults to a description of x.
ylab	label for the y axis, defaults to a description of y.

Value

Invisibly returns a list with the following 4 components:

pm	the viewing transformation matrix (see persp for details), a 4 x 4 matrix that can be used to superimpose additional graphical elements using the function trans3d.
bigplot	plot coordinates of the main plot. These values may be useful for drawing a plot without the legend that is the same size as the plots with legends.
smallplot	plot coordinates of the secondary plot (legend strip).
old.par	previous graphical parameters (par(old.par) will reset plot parameters to the values before entering the function).

Side Effects

After exiting, the plotting region may be changed (par("plt")) to make it possible to add more features to the plot (set graphics.reset = FALSE to avoid this).

Author(s)

Based on image.plot function from package **fields**: fields, Tools for spatial data. Copyright 2004-2013, Institute for Mathematics Applied Geosciences. University Corporation for Atmospheric Research.

Modified by Ruben Fernandez-Casal <rubenfcasal@gmail.com>.

See Also

splot, spoints, simage, image, image.plot, data.grid, persp.

Examples

```
#
# Regularly spaced 2D data
nx <- c(40, 40) # ndata =  prod(nx)
x1 <- seq(-1, 1, length.out = nx[1])
x2 <- seq(-1, 1, length.out = nx[2])
trend <- outer(x1, x2, function(x,y) x^2 - y^2)
spersp( x1, x2, trend, main = 'Trend', zlab = 'y')

#
# Multiple plots
set.seed(1)
y <- trend + rnorm(prod(nx), 0, 0.1)
x <- as.matrix(expand.grid(x1 = x1, x2 = x2)) # two-dimensional grid
# local polynomial kernel regression
lp <- locpol(x, y, nbin = nx, h =  diag(c(0.3, 0.3)))
# 1x2 plot
old.par <- par(mfrow = c(1,2))
spersp( x1, x2, y, main = 'Data')
spersp(lp, main = 'Estimated trend', zlab = 'y')
par(old.par)
```

splot *Utilities for plotting with a color scale*

Description

splot is designed to combine a standard R plot with a legend representing a (continuous) color scale. This is done by splitting the plotting region into two parts. Keeping one for the main chart and putting the legend in the other.

sxxxx functions (spoints, simage and spersp) draw the corresponding high-level plot (xxxx) with a legend strip for the color scale.

These functions are based on function image.plot of package **fields**, see its documentation for additional information.

jet.colors and hot.colors create a color table useful for contiguous color scales and scolor assigns colors to a numerical vector.

Usage

```
splot(slim = c(0, 1), col = jet.colors(128),
  breaks = NULL, horizontal = FALSE, legend.shrink = 0.9,
  legend.width = 1.2,
  legend.mar = ifelse(horizontal, 3.1, 5.1),
  legend.lab = NULL, bigplot = NULL, smallplot = NULL,
  lab.breaks = NULL, axis.args = NULL,
  legend.args = NULL, add = FALSE)

scolor(s, col = jet.colors(128),
  slim = range(s, finite = TRUE))

jet.colors(n)

hot.colors(n, rev = TRUE)
```

Arguments

slim	limits used to set up the color scale.
col	color table used to set up the color scale (see image for details).
breaks	(optional) numeric vector with the breakpoints for the color scale: must have one more breakpoint than col and be in increasing order.
horizontal	logical; if FALSE (default) legend will be a vertical strip on the right side. If TRUE the legend strip will be along the bottom.
legend.shrink	amount to shrink the size of legend relative to the full height or width of the plot.
legend.width	width in characters of the legend strip. Default is 1.2, a little bigger that the width of a character.

legend.mar	width in characters of legend margin that has the axis. Default is 5.1 for a vertical legend and 3.1 for a horizontal legend.
legend.lab	label for the axis of the color legend. Default is no label as this is usual evident from the plot title.
bigplot	plot coordinates for main plot. If not passed these will be determined within the function.
smallplot	plot coordinates for legend strip. If not passed these will be determined within the function.
lab.breaks	if breaks are supplied these are text string labels to put at each break value. This is intended to label axis on a transformed scale such as logs.
axis.args	additional arguments for the axis function used to create the legend axis (see image.plot for details).
legend.args	arguments for a complete specification of the legend label. This is in the form of list and is just passed to the mtext function. Usually this will not be needed (see image.plot for details).
add	logical; if TRUE the legend strip is just added to the existing plot.
s	values to be converted to the color scale.
n	number of colors (>= 1) to be in the palette.
rev	logical; if TRUE, the palette is reversed (decreasing overall luminosity).

Details

scolor converts a real valued vector to a color scale. The range slim is divided into length(col) + 1 pieces of equal length. Values which fall outside the range of the scale are coded as NA.

jet.colors generates a rainbow style color table similar to the MATLAB (TM) jet color scheme. It may be appropriate to distinguish between values above and below a central value (e.g. between positive and negative values).

hot.colors generates a color table similar to the MATLAB (TM) hot color scheme (reversed by default). It may be appropriate to represent values ranging from 0 to some maximum level (e.g. density estimation). The default value rev = TRUE may be adecuate to grayscale convertion.

Value

splot invisibly returns a list with the following 3 components:

bigplot	plot coordinates of the main plot. These values may be useful for drawing a plot without the legend that is the same size as the plots with legends.
smallplot	plot coordinates of the secondary plot (legend strip).
old.par	previous graphical parameters (par(old.par) will reset plot parameters to the values before entering the function).

jet.colors and hot.colors return a character vector of colors (similar to heat.colors or terrain.colors; see rgb).

Side Effects

After exiting, the plotting region may be changed (par("plt")) to make it possible to add more features to the plot.

Author(s)

Based on image.plot function from package **fields**: fields, Tools for spatial data. Copyright 2004-2013, Institute for Mathematics Applied Geosciences. University Corporation for Atmospheric Research.

Modified by Ruben Fernandez-Casal <rubenfcasal@gmail.com>.

See Also

spoints, simage, spersp, image, image.plot.

Examples

```
scale.range <- range(aquifer$head)
splot(slim = scale.range)
with( aquifer, plot(lon, lat, col = scolor(head, slim = scale.range),
       pch = 16, cex = 1.5, main = "Wolfcamp aquifer data"))
# equivalent to:
# with( aquifer, spoints(lon, lat, head, main = "Wolfcamp aquifer data"))

#
# Multiple plots with a common legend:
#
# regularly spaced 2D data...
set.seed(1)
nx <- c(40, 40) # ndata =  prod(nx)
x1 <- seq(-1, 1, length.out = nx[1])
x2 <- seq(-1, 1, length.out = nx[2])
trend <- outer(x1, x2, function(x,y) x^2 - y^2)
y <- trend + rnorm(prod(nx), 0, 0.1)
scale.range <- c(-1.2, 1.2)
scale.color <- heat.colors(64)
# 1x2 plot with some room for the legend...
old.par <- par(mfrow = c(1,2), omd = c(0.05, 0.85, 0.05, 0.95))
image( x1, x2, trend, zlim = scale.range, main = 'Trend', col = scale.color)
image( x1, x2, y, zlim = scale.range, main = 'Data', col = scale.color)
par(old.par)
# the legend can be added to any plot...
splot(slim = scale.range, col = scale.color, add = TRUE)
## note that argument 'zlim' in 'image' corresponds with 'slim' in 'sxxxx' functions.
```

spoints *Scatter plot with a color scale*

Description

spoints (generic function) draws an scatterplot with points filled with different colors and (optionally) adds a legend strip with the color scale (calls splot and plot.default).

Usage

```
spoints(x, ...)

## Default S3 method:
spoints(x, y = NULL, s,
   slim = range(s, finite = TRUE), col = jet.colors(128),
   breaks = NULL, legend = TRUE, horizontal = FALSE,
   legend.shrink = 0.9, legend.width = 1.2,
   legend.mar = ifelse(horizontal, 3.1, 5.1),
   legend.lab = NULL, bigplot = NULL, smallplot = NULL,
   lab.breaks = NULL, axis.args = NULL,
   legend.args = NULL, add = FALSE, graphics.reset = add,
   pch = 16, cex = 1.5, xlab = NULL, ylab = NULL, ...)

## S3 method for class 'data.grid'
spoints(x, s = x[[1]], xlab = NULL,
   ylab = NULL, ...)
```

Arguments

x	x coordinates. Any reasonable way of defining the coordinates is acceptable. See the function xy.coords for details.
y	y coordinates. Alternatively, a single argument x can be provided.
s	numerical vector containing the values used for coloring the points.
legend	logical; if TRUE (default), the plotting region is splitted into two parts, drawing the main plot in one and the legend with the color scale in the other. If FALSE only the (coloured) main plot is drawn and the arguments related to the legend are ignored (splot is not called).
bigplot	plot coordinates for main plot. If not passed, and legend is TRUE, these will bedetermined within the function.
smallplot	plot coordinates for legend strip. If not passed, and legend is TRUE, these will be determined within the function.
add	logical; if TRUE the scatter plot is just added to the existing plot.
graphics.reset	logical; if FALSE (default) the plotting region (par("plt")) will not be reset to make it possible to add more features to the plot (e.g. using functions such as points or lines). If TRUE will reset plot parameters to the values before entering the function.

pch	vector of plotting characters or symbols: see points.
cex	numerical vector giving the amount by which plotting characters and symbols should be scaled relative to the default. This works as a multiple of par("cex").
xlab	label for the x axis, defaults to a description of x.
ylab	label for the y axis, defaults to a description of y.
...	additional graphical parameters (to be passed to the main plot function or sxxxx.default; e.g. xlim, ylim, ...). NOTE: graphical arguments passed here will only have impact on the main plot. To change the graphical defaults for the legend use the par function beforehand (e.g. par(cex.lab = 2) to increase colorbar labels).
slim	limits used to set up the color scale.
col	color table used to set up the color scale (see image for details).
breaks	(optional) numeric vector with the breakpoints for the color scale: must have one more breakpoint than col and be in increasing order.
horizontal	logical; if FALSE (default) legend will be a vertical strip on the right side. If TRUE the legend strip will be along the bottom.
legend.shrink	amount to shrink the size of legend relative to the full height or width of the plot.
legend.width	width in characters of the legend strip. Default is 1.2, a little bigger that the width of a character.
legend.mar	width in characters of legend margin that has the axis. Default is 5.1 for a vertical legend and 3.1 for a horizontal legend.
legend.lab	label for the axis of the color legend. Default is no label as this is usual evident from the plot title.
lab.breaks	if breaks are supplied these are text string labels to put at each break value. This is intended to label axis on a transformed scale such as logs.
axis.args	additional arguments for the axis function used to create the legend axis (see image.plot for details).
legend.args	arguments for a complete specification of the legend label. This is in the form of list and is just passed to the mtext function. Usually this will not be needed (see image.plot for details).

Value

Invisibly returns a list with the following 3 components:

bigplot	plot coordinates of the main plot. These values may be useful for drawing a plot without the legend that is the same size as the plots with legends.
smallplot	plot coordinates of the secondary plot (legend strip).
old.par	previous graphical parameters (par(old.par) will reset plot parameters to the values before entering the function).

Side Effects

After exiting, the plotting region may be changed (par("plt")) to make it possible to add more features to the plot (set graphics.reset = FALSE to avoid this).

Author(s)

Based on image.plot function from package **fields**: fields, Tools for spatial data. Copyright 2004-2013, Institute for Mathematics Applied Geosciences. University Corporation for Atmospheric Research.

Modified by Ruben Fernandez-Casal <rubenfcasal@gmail.com>.

See Also

splot, simage, spersp, image, image.plot, data.grid, plot.default.

Examples

```
#
with( aquifer, spoints(lon, lat, head, main = "Wolfcamp aquifer data"))
```

---

sv *Evaluate a semivariogram model*

Description

Evaluates an svarmod object x at lags h (S3 generic function).

Usage

```
sv(x, h, ...)

## Default S3 method:
sv(x, h, ...)

## S3 method for class 'svarmod'
sv(x, h, ...)

## S3 method for class 'sb.iso'
sv(x, h, ...)
```

Arguments

x	variogram model (svarmod object).
h	vector (isotropic case) or matrix of lag values.
...	further arguments passed to or from other methods.

Value

A vector of semivariance values $\gamma(h_i)$.

See Also

covar

svar.bin *Linear binning of semivariances*

Description

Creates a svar.bin (binned semivar. + grid parameters) object with linearly binned semivariances.

Usage

```
svar.bin(x, ...)

## Default S3 method:
svar.bin(x, y, maxlag = NULL,
  nlags = NULL, minlag = maxlag/nlags,
  estimator = c("classical", "modulus"), ...)

svariso(x, y, maxlag = NULL, nlags = NULL,
  minlag = maxlag/nlags,
  estimator = c("classical", "modulus"), ...)
```

Arguments

x	object used to select a method. Usually a matrix with the coordinates of the data locations (columns correspond with dimensions and rows with data).
...	further arguments passed to or from other methods.
estimator	character, estimator name (e.g. "classical"). See "Details" below.
y	vector of data (response variable).
maxlag	maximum lag. Defaults to 55% of largest lag.
nlags	number of lags. Defaults to 101.
minlag	minimun lag.

Details

Currently, only isotropic semivariogram estimation is supported.

If parameter nlags is not specified is set to 101.

Value

Returns an S3 object of class svar.bin (extends bin.data), a data.grid object with the following 4 components:

biny	array (dimension nlags) with the binned semivariances.
binw	array (dimension nlags) with the bin counts (weights).
grid	a grid.par-class object with the grid parameters.
data	a list with 3 components:

- x argument x.
- y argument y.
- med (weighted) mean of the (binned) semivariances.

svar — a list of 2 components:

- type character, type of estimation (e.g. "isotropic").
- estimator character, estimator name (e.g. "classical").

See Also

np.svariso, np.svar, data.grid, binning, locpol.

---

svar.plot — *Plot a semivariogram object*

Description

Utilities for plotting pilot semivariograms or fitted models.

plot.fitsvar plots a fitted variogram model.

plot.svar.bin plots the binned semivariances.

plot.np.svar plots a local polynomial estimate of the semivariogram.

Usage

```
 ## S3 method for class 'fitsvar'
plot(x, y = NULL, legend = TRUE,
   xlab = "distance", ylab = "semivariance",
   ylim = c(0, 1.25 * max(x$fit$sv, na.rm = TRUE)),
   lwd = c(1, 2), add = FALSE, ...)

 ## S3 method for class 'svar.bin'
plot(x, y = NULL, xlab = "distance",
   ylab = "semivariance",
   ylim = c(0, max(x$biny, na.rm = TRUE)), add = FALSE,
   ...)

 ## S3 method for class 'np.svar'
plot(x, y = NULL, xlab = "distance",
   ylab = "semivariance",
   ylim = c(0, max(x$biny, na.rm = TRUE)), add = FALSE,
   ...)
```

Arguments

x	a variogram object. Typically an output of functions np.svariso or fitsvar.sb.iso.
y	ignored argument.
legend	logical; if TRUE (default), a legend is added to the plot.
xlab	label for the x axis (defaults to "distance").
ylab	label for the y axis (defaults to "semivariance").
ylim	y-limits.
lwd	line widths for points (estimates) and lines (fitted model) respectively.
...	additional graphical parameters (see par).
add	logical; if TRUE the semivariogram plot is just added to the existing plot.

See Also

svariso, np.svariso, fitsvar.sb.iso.

---

svarmod *Define a (semi)variogram model*

Description

Defines a variogram model specifying the parameter values. Constructor function of the svarmod-class.

Usage

```
svarmod(model, type = "isotropic", par = NA,
  nugget = NULL, sill = NULL, range = NULL)

svarmod.sb.iso(dk, x, z, nu, range, sill = nu)

svarmodels(type = "isotropic")
```

Arguments

model	string indicating the variogram family (see Details below).
type	string indicating the type of variogram, e.g. "isotropic".
par	vector of variogram parameters.
nugget	nugget value c_0.
sill	variance σ^2 or sill of the variogram (NA for unbounded variograms).
range	range (practical range or scale parameter) of the variogram (NA for unbounded variograms; maybe a vector for anisotropic variograms).
dk	dimension of the kappa function.
x	discretization nodes.
z	jumps (of the spectral distibution) at the discretization nodes.
nu	parameter ν_0 (can be thought of as the sill).

Value

svarmod returns an svarmod-class object, a list with function arguments as components.

svarmod.sb.iso returns an S3 object of class sb.iso (extends svarmod) corresponding to a 'non-parametric' isotropic Shapiro-Botha model.

svarmodels returns a named character vector with the available models of the corresponding type (when appropriate, component values could be used as cov.model argument in **geoR** routines and component names as model argument in **gstat** routines).

Note

svarmod does not check the consistency of the parameter values.

References

Shapiro, A. and Botha, J.D. (1991) Variogram fitting with a general class of conditionally non-negative definite functions. *Computational Statistics and Data Analysis*, **11**, 87-96.

See Also

sv, covar.

---

varcov *Covariance matrix*

Description

Computes the covariance matrix a corresponding to a set of spatial locations given a variogram model or a semivariogram estimate.

Usage

```
varcov(x, coords, ...)

## S3 method for class 'isotropic'
varcov(x, coords, sill = x$sill,
  range.taper, ...)

## S3 method for class 'np.svar'
varcov(x, coords, sill = max(x$est),
  range.taper = x$grid$max, ...)
```

Arguments

x	variogram model (svarmod object) or semivariogram estimate.
coords	matrix of coordinates (columns correspond with dimensions and rows with data).
...	further arguments passed to or from other methods.
sill	(theoretical or estimated) variance $C(0) = \sigma^2$ or pseudo-sill (unbounded variograms).
range.taper	(optional) if provided, covariances corresponding to distances larger than this value are set to 0.

Value

The covariance matrix of the data.

See Also

sv, covar.

Index