# Package 'npsp'

February 19, 2024

**Type** Package

**Title** Nonparametric Spatial Statistics

**Version** 0.7-13

**Date** 2024-02-17

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

**Depends** R (>= 2.14.0), graphics

**Imports** sp, methods, quadprog, spam

**Suggests** gstat, geoR, fields, DEoptim, knitr

**Description** Multidimensional nonparametric spatial (spatio-temporal) geostatistics.
S3 classes and methods for multidimensional: linear binning,
local polynomial kernel regression (spatial trend estimation), density and variogram estimation.
Nonparametric methods for simultaneous inference on both spatial trend
and variogram functions (for spatial processes).
Nonparametric residual kriging (spatial prediction).
For details on these methods see, for example, Fernandez-Casal and Francisco-Fernandez (2014)
<doi:10.1007/s00477-013-0817-8> or Castillo-
Paez et al. (2019) <doi:10.1016/j.csda.2019.01.017>.

**License** GPL (>= 2)

**URL** https://rubenfcasal.github.io/npsp/,
https://github.com/rubenfcasal/npsp/

**BugReports** https://github.com/rubenfcasal/npsp/issues/

**LazyData** yes

**Encoding** UTF-8

**VignetteBuilder** knitr

**RoxygenNote** 7.2.3

**NeedsCompilation** yes

**Author** Ruben Fernandez-Casal [aut, cre]
(<https://orcid.org/0000-0002-5785-3739>)

**Repository** CRAN

**Date/Publication** 2024-02-19 17:20:02 UTC

# R topics documented:

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

## Description

This package implements nonparametric methods for inference on multidimensional spatial (or spatio-temporal) processes, which may be (especially) useful in (automatic) geostatistical modeling and interpolation.

## Main functions

**Nonparametric methods for inference on both spatial trend and variogram functions**:

`np.fitgeo` (automatically) fits an isotropic nonparametric geostatistical model by estimating the trend and the variogram (using a bias-corrected estimator) iteratively (by calling `h.cv`, `locpol`, `np.svariso.corr` and `fitsvar.sb.iso` at each iteration).

`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.

**Nonparametric residual kriging** (sometimes called external drift kriging):

`np.kriging` computes residual kriging predictions (and the corresponding simple kriging standard errors).

`kriging.simple` computes simple kriging predictions, standard errors

Currently, only global simple kriging is implemented in this package. Users are encouraged to use `krige` (or `krige.cv`) utilities in **gstat** package together with `as.vgm` for local kriging.

## Other functions

Among the other functions intended for direct access by the user, the following (methods for multidimensional linear binning, local polynomial kernel regression, density or variogram estimation) could be emphasized: `binning`, `bin.den`, `svar.bin`, `h.cv` and `interp`.

There are 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**.

There are also some functions which can be used to interact with other packages. For instance, `as.variogram` (**geoR**) or `as.vgm` (**gstat**).

## Acknowledgments

Important suggestions and contributions to some techniques included here were made by Sergio Castillo-Paez (Universidad de las Fuerzas Armadas ESPE, Ecuador) and Tomas Cotos-Yañez (Dep. Statistics, University of Vigo, Spain).

## Author(s)

Ruben Fernandez-Casal (Dep. Mathematics, University of A Coruña, Spain). Please send comments, error reports or suggestions to <rubenfcasal@gmail.com>.

## References

Castillo-Páez S., Fernández-Casal R. and García-Soidán P. (2019) A nonparametric bootstrap method for spatial data, **137**, *Comput. Stat. Data Anal.*, 1-15, doi:10.1016/j.csda.2019.01.017.

Fernandez-Casal R., Castillo-Paez S. and Francisco-Fernandez M. (2018) Nonparametric geostatistical risk mapping, *Stoch. Environ. Res. Ris. Assess.*, **32**, 675-684, doi:10.1007/s004770171407y.

Fernandez-Casal R., Castillo-Paez S. and Garcia-Soidan P. (2017) Nonparametric estimation of the small-scale variability of heteroscedastic spatial processes, *Spa. Sta.*, **22**, 358-370, doi:10.1016/j.spasta.2017.04.001.

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, doi:10.1007/s0047701308178.

Fernandez-Casal R., Gonzalez-Manteiga W. and Febrero-Bande M. (2003) Flexible Spatio-Temporal Stationary Variogram Models, *Statistics and Computing*, **13**, 127-136, doi:10.1023/A:1023204525046.

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

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

---

`as.data.grid` *data.grid-class methods*

---

## Description

S3 class `data.grid` methods.

## Usage

```
as.data.grid(object, ...)

## S3 method for class 'SpatialGridDataFrame'
as.data.grid(object, data.ind = NULL, ...)

## S3 method for class 'data.grid'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  data.ind = NULL,
  coords = FALSE,
  sp = FALSE,
  check.names = coords,
  ...
)
```

## Arguments

| | |
|---|---|
| object | (gridded data) used to select a method. |
| ... | further arguments passed to data.grid. |
| data.ind | integer or character vector with the indexes or names of the components. |
| x | a data.grid object. |
| row.names | NULL, column to be used as row names, or vector giving the row names for the data frame. |
| optional | logical; Not currently used (see as.data.frame). |
| coords | logical; if TRUE, the (spatial) coordinates of the object are added. |
| sp | logical; if TRUE, the second dimension of the data is reversed (as it is stored in **sp** package). |
| check.names | logical; if TRUE, the names of the variables in the data frame are checked and adjusted if necessary. |

## Value

as.data.grid returns a data.grid object.

as.data.frame returns a data frame.

## See Also

data.grid.

## as.sp *Convert npsp object to sp object*

### Description

Converts a npsp object to a sp object.

### Usage

```
as.sp(obj, ...)

## S3 method for class 'grid.par'
as.sp(obj, ...)

## S3 method for class 'data.grid'
as.sp(obj, data.ind = NULL, proj4string = CRS(as.character(NA)), ...)
```

### Arguments

| | |
|---|---|
| obj | a npsp object. |
| ... | further arguments passed to or from other methods. |
| data.ind | integer or character; vector with indexes or names of the data components. |
| proj4string | a CRS-class object. |

### Value

as.sp.grid.par returns a GridTopology-class object.

as.sp.data.grid returns a SpatialGridDataFrame-class object.

### See Also

as.data.frame.data.grid

## 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 'data.grid'
as.bin.den(object, weights.ind = 1, ...)

## S3 method for class 'bin.den'
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. |
| weights.ind | integer or character with the index or name of the component containing the bin counts/weights. |

## Details

If parameter nbin is not specified is set to pmax(25, rule.binning(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, h.cv, bin.data, locpol, rule.binning.

## Examples

```
binden <- bin.den(earthquakes[, c("lon", "lat")], nbin = c(30,30))
bindat <- binning(earthquakes[, c("lon", "lat")], earthquakes$mag, nbin = c(30,30))
all.equal(binden, as.bin.den(bindat))
```

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, window = NULL, ...)

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

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

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

## S3 method for class 'SpatialGridDataFrame'
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 the bin averages corresponding to cells without data to NA. |
| window | spatial window (values outside this window will be masked), currently an sp-object of class extending SpatialPolygons. |
| ... | further arguments passed to mask.bin.data(). |
| object | (gridded data) used to select a method. |
| 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 pmax(25, rule.binning(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:<br>• x argument x.<br>• y argument y.<br>• 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))
```

---

# coords *(spatial) coordinates*

---

## Description

Retrieves the (spatial) coordinates of the object.

## Usage

```
coords(x, ...)

## S3 method for class 'grid.par'
coords(x, ...)

## S3 method for class 'data.grid'
coords(x, masked = FALSE, ...)
```

### Arguments

| | |
|---|---|
| x | a (spatial) object used to select a method. |
| ... | further arguments passed to or from other methods. |
| masked | logical; If TRUE, only the coordinates corresponding to unmasked cells are returned (see mask). |

### Value

A matrix of coordinates (columns correspond with dimensions and rows with data).

### See Also

coordvalues.

---

## coordvalues *Coordinate values*

---

### Description

Returns the coordinate values in each dimension.

### Usage

```
coordvalues(x, ...)

## S3 method for class 'grid.par'
coordvalues(x, ...)

## S3 method for class 'data.grid'
coordvalues(x, ...)
```

### Arguments

| | |
|---|---|
| x | a (spatial) object used to select a method. |
| ... | further arguments passed to or from other methods. |

### Value

A list with the (unique) coordinates along each axis.

### See Also

coords.

## 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, discretize = FALSE, ...)

## 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). |
| discretize | logical. If TRUE the variogram is previously discretized. |

### 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

```
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)
```

---

## 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,
  window = NULL,
  mask = NULL,
  set.NA = FALSE,
  warn = FALSE
)
```

### Arguments

| | |
|---|---|
| ... | vectors or arrays of data with length equal to prod(grid$n). |
| grid | a grid.par-class object (optional). |
| window | spatial window (values outside this window will be masked), currently an sp-object of class extending SpatialPolygons. |
| mask | logical; vector (or array) indicating the selected values (not masked). |
| set.NA | logical; If TRUE, the values corresponding to masked cells are set to NA. |
| warn | logical; If TRUE a warning message is generated when original data is masked. |

### 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

as.data.grid, grid.par, mask, binning, locpol.

### Examples

```
# Grid parameters
grid <- grid.par(n = c(15,15), min = c(x = -1, y = -1), max = c(1, 1))
coordinates <- coords(grid)
plot(coordinates)
coordvs <- coordvalues(grid)
abline(v = coordvs[[1]], lty = 3)
abline(h = coordvs[[2]], lty = 3)
# Gridded data
y <- apply(coordinates, 1, function(x) x[1]^2 - x[2]^2 )
datgrid <- data.grid(y = y, grid = grid)
spersp(datgrid, main = 'f(x,y) = x^2 - y^2')
dim(datgrid)
all.equal(coordinates, coords(datgrid))
```

---

## 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 (dk >= 1, see Details below). |
| 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:

$$r_i = 2(1 + (i-1))rmax/nx; i = 1, \ldots, nx.$$

## Value

A vector with the discretization nodes.

## 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:
https://www.ign.es/web/ign/portal/sis-area-sismicidad.

## 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 = 4 * 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 a strictly convex quadratic program. To avoid problems, the Cholesky decomposition of the matrix corresponding to the original problem is computed using `chol` with `pivot = TRUE`. If this matrix is only positive semi-definite (non-strictly convex QP), the number of discretization nodes will be less than `nx`.

## Value

Returns the fitted variogram model, an object of `class fitsvar`. A `svarmod` object with additional components `esv` (pilot semivariogram estimate) and `fit` containing:

| | |
|---|---|
| `u` | vector of lags/distances. |
| `sv` | vector of pilot semivariogram estimates. |
| `fitted.sv` | vector of fitted semivariances. |
| `w` | vector of (least squares) weights. |
| `wls` | value of the 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.

## Examples

```
# Trend estimation
lp <- locpol(aquifer[,1:2], aquifer$head, nbin = c(41,41),
             h = diag(100, 2), hat.bin = TRUE)
                              # 'np.svariso.corr()' requires a 'lp$locpol$hat' component

# Variogram estimation
esvar <- np.svariso.corr(lp, maxlag = 150, nlags = 60, h = 60, plot = FALSE)

# Variogram fitting
svm2 <- fitsvar.sb.iso(esvar)  # dk = 2
svm3 <- fitsvar.sb.iso(esvar, dk = 0) # To avoid negative covariances...
svm4 <- fitsvar.sb.iso(esvar, dk = 10) # To improve fit...

plot(svm4, main = "Nonparametric bias-corrected semivariogram and fitted models", legend = FALSE)
plot(svm3, add = TRUE)
plot(svm2, add = TRUE, lty = 3)
legend("bottomright", legend = c("NP estimates", "fitted model (dk = 10)", "dk = 0", "dk = 2"),
            lty = c(NA, 1, 1, 3), pch = c(1, NA, NA, NA), lwd = c(1, 2, 1, 1))
```

---

# 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; minimum values of the coordinates. |
| max | vector; maximum values of the coordinates (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 (standard 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 == "CV", 2, 0),
  cov.bin = NULL,
  DEalgorithm = FALSE,
  warn = TRUE,
  tol.mask = npsp.tolerance(2),
  ...
)

## S3 method for class 'bin.den'
```

```
h.cv(
  bin,
  h.start = NULL,
  h.lower = NULL,
  h.upper = NULL,
  degree = 1,
  ncv = 2,
  DEalgorithm = FALSE,
  ...
)

## S3 method for class 'svar.bin'
h.cv(
  bin,
  loss = c("MRSE", "MRAE", "MSE", "MAE"),
  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 == "CV", 1, 0),
  cov.dat = NULL,
  DEalgorithm = FALSE,
  warn = TRUE,
  ...
)
```

## 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. |
| degree | degree of the local polynomial used. Defaults to 1 (local linear estimation). |
| 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 | (optional) covariance matrix of the binned data or semivariogram model (svarmod-class) of the (unbinned) data. Defaults to the identity matrix. |
| DEalgorithm | logical; if TRUE, the differential evolution optimization algorithm in package **DEoptim** is used. |
| warn | logical; sets the handling of warning messages (normally due to the lack of data in some neighborhoods). If FALSE all warnings are ignored. |
| tol.mask | tolerance used in the aproximations. Defaults to npsp.tolerance(2). |
| loss | character; CV error. See "Details" bellow. |
| cov.dat | covariance matrix of the data or semivariogram model (of class extending svarmod). Defaults to the identity matrix (uncorrelated data). |

## Details

Currently, only diagonal bandwidths are supported.

h.cv methods use binning approximations to the objective function values (in almost all cases, an averaged squared error). 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 location. $ncv = 1$ corresponds with traditional cross-validation and $ncv > 1$ with modified CV (it may be appropriate for dependent data; see e.g. Chu and Marron, 1991, for the one dimensional case). Setting ncv >= 2 would be recommended for sparse data (as linear binning is used). For standard GCV, set ncv = 0 (the whole data would be used). For theoretical MASE, set bin = binning(x, y = trend.teor), cov = cov.teor and ncv = 0.

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

The different options for the argument loss in h.cv.svar.bin() define the CV error considered in semivariogram estimation:

"MSE" Mean squared error

"MRSE" Mean relative squared error

"MAE" Mean absolute error

"MRAE" Mean relative absolute error

hcv.data evaluates the objective function at the original data (combining a binning approximation to the nonparametric estimates with a linear interpolation), this can be very slow (and memory demanding; consider using h.cv instead). If ncv > 1 (modified CV), a similar algorithm to that in h.cv 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.den, np.svar.

## Examples

```
# Trend estimation
bin <- binning(earthquakes[, c("lon", "lat")], earthquakes$mag)
hcv <- h.cv(bin, ncv = 2)
lp <- locpol(bin, h = hcv$h)
# Alternatively, `locpolhcv()` could be called instead of the previous code.

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, ...)

## S3 method for class 'np.den'
predict(object, newx = NULL, ...)
```

## 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 $\Omega_d$.

## 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 (and linear binning).

## 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 == "CV", 2, 0),
  cov.dat = 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; only the upper triangular part of h is used). |
| 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 the bin averages corresponding to cells without data to NA. |
| objective | character; optimal criterion to be used ("CV", "GCV" or "MASE"). |
| cov.dat | covariance matrix of the data or semivariogram model (of class extending svarmod). Defaults to the identity matrix (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 pmax(25, rule.binning(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`, `rule.binning`.

## 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)

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

---

mask *Mask methods*

---

## Description

Filters the data that satisfy a condition.

## Usage

```
mask(x, ...)

## Default S3 method:
mask(x, tol.mask = 0, ...)

## S3 method for class 'data.grid'
mask(x, mask = NULL, window = NULL, set.NA = FALSE, warn = FALSE, ...)

## S3 method for class 'bin.den'
mask(
  x,
  mask = mask.default(x$binw, npsp.tolerance(2)),
  window = NULL,
  set.NA = FALSE,
  warn = TRUE,
  ...
)

## S3 method for class 'bin.data'
mask(
  x,
  mask = NULL,
  window = NULL,
  set.NA = FALSE,
  warn = FALSE,
  filter.lp = TRUE,
  ...
)

## S3 method for class 'locpol.bin'
mask(
  x,
  mask = mask.default(x$binw, npsp.tolerance(2)),
```

```
  window = NULL,
  set.NA = FALSE,
  warn = TRUE,
  filter.lp = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| x | object used to select a method (binned data, ...). |
| ... | further arguments passed to or from other methods |
| tol.mask | tolerance. |
| mask | logical; vector (or array) indicating the selected values (not masked). |
| window | spatial window (values outside this window will be masked), currently an sp-object of class extending SpatialPolygons. |
| set.NA | logical; If TRUE, the values corresponding to masked cells are set to NA. |
| warn | logical; If TRUE a warning message is generated when original data is masked. |
| filter.lp | logical; If TRUE, masked nodes will be leaved out in local polynomial estimation. |

## Value

mask.default returns the logical vector x > tol.mask.

mask.bin.den, mask.bin.data and mask.locpol.bin return an object of the same class as x with the additional component $mask and optionally $window.

## See Also

locpol, locpolhcv, binning, np.svar, npsp.tolerance.

## Examples

```
mask(1:10, 5)
bin <- binning(aquifer[,1:2], aquifer$head, nbin = c(41,41), set.NA = TRUE)
str(mask(bin, mask(bin$binw), warn = TRUE))
str(mask(bin, mask(bin$binw, 1)))
```

---

# 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; only the upper triangular part of h is used). |
| 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:<br>• degree degree of the polinomial.<br>• h bandwidth matrix.<br>• rm residual mean (of the escaled bin counts).<br>• rss sum of squared residuals (of the escaled bin counts).<br>• 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, h.cv, data.grid.

## Examples

```
bin.den <- binning(earthquakes[, c("lon", "lat")], nbin = c(30,30))
h.den <- h.cv(bin.den)
den <- np.den(bin.den, h = h.den$h)
plot(den, main = 'Estimated log(density)')
```

---

# np.fitgeo *Fit a nonparametric geostatistical model*

---

## Description

Fits a nonparametric (isotropic) geostatistical model (jointly estimates the trend and the variogram) by calling locpol, np.svariso.corr (or np.svariso ) and fitsvar.sb.iso iteratively. At each iteration, the trend estimation bandwith is updated by a call to h.cv.

## Usage

```
np.fitgeo(x, ...)

## Default S3 method:
np.fitgeo(
  x,
  y,
  nbin = NULL,
```

```
  iter = 2,
  h = NULL,
  tol = 0.05,
  set.NA = FALSE,
  h.svar = NULL,
  corr.svar = iter > 0,
  maxlag = NULL,
  nlags = NULL,
  dk = 0,
  svm.resid = FALSE,
  hat.bin = corr.svar,
  warn = FALSE,
  plot = FALSE,
  window = NULL,
  ...
)

## S3 method for class 'locpol.bin'
np.fitgeo(
  x,
  svm,
  iter = 1,
  tol = 0.05,
  h.svar = svm$esv$locpol$h,
  dk = 0,
  corr.svar = TRUE,
  svm.resid = FALSE,
  hat.bin = corr.svar,
  warn = FALSE,
  plot = FALSE,
  ...
)

## S3 method for class 'fitgeo'
np.fitgeo(
  x,
  iter = 1,
  tol = 0.05,
  h.svar = x$svm$esv$locpol$h,
  dk = x$svm$par$dk,
  corr.svar = TRUE,
  svm.resid = FALSE,
  hat.bin = corr.svar,
  warn = FALSE,
  plot = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| x | a (data) object used to select a method. |
| ... | further arguments passed to h.cv (trend bandwidth selection parameters). |
| y | vector of data (response variable). |
| nbin | vector with the number of bins on each dimension. |
| iter | maximum number of iterations (of the whole algorithm). |
| h | initial bandwidth matrix for trend estimation (final bandwidth if iter = 1). |
| tol | relative convergence tolerance (semivariogram). |
| set.NA | logical. If TRUE, sets the bin averages corresponding to cells without data to NA. |
| h.svar | bandwidth matrix for variogram estimation. |
| corr.svar | logical; if TRUE (default), a bias-corrected semivariogram estimate is computed (see np.svariso.corr). If FALSE the (uncorrected) residual variogram is computed (the traditional approach in geostatistics). |
| maxlag | maximum lag. Defaults to 55% of largest lag. |
| nlags | number of lags. Defaults to 101. |
| dk | dimension of the Shapiro-Botha variogram model (see fitsvar.sb.iso). |
| svm.resid | logical; if TRUE, the fitted (uncorrected) residual semivariogram model is computed and returned (this parameter has no effect when corr.svar = FALSE). |
| hat.bin | logical; if TRUE, the hat matrix of the binned data is returned. |
| warn | logical; sets the handling of warning messages in bandwidth selection (h.cv). |
| plot | logical; if TRUE, semivariogram estimates obtained at each iteration are plotted. |
| window | spatial window (values outside this window will be masked), currently an sp-object of class extending SpatialPolygons. |
| svm | (fitted) variogram model (object of class fitsvar or svarmod). |

## Details

Currently, only isotropic semivariogram estimation is supported.

If parameter h is not specified, h.cv is called with the default values (modified CV) to set it. If parameter h.svar is not specified, is set to 1.5*h.cv.svar.bin()$h.

Setting corr.svar = TRUE may be very slow (and memory demanding) when the number of data is large (note also that the bias in the residual variogram decreases when the sample size increases).

## Value

Returns an object of class fitgeo (extends np.geo). A locpol.bin object with the additional (some optional) 3 components:

| | |
|---|---|
| svm | fitted variogram model (object of class fitsvar). |
| svm0 | (if requested) fitted residual variogram model (object of class fitsvar). |
| residuals | model residuals. |

## See Also

locpol, fitsvar.sb.iso, np.svar, np.svariso.corr, np.geo.

## Examples

```
geomod <- np.fitgeo(aquifer[,1:2], aquifer$head, svm.resid = TRUE)
plot(geomod)


# Uncorrected variogram estimator
geomod0 <- np.fitgeo(aquifer[,1:2], aquifer$head, iter = 0, corr.svar = FALSE)
plot(geomod0)

# Additional iteration with bias-corrected variogram estimator
geomod1 <- np.fitgeo(geomod0, corr.svar = TRUE, svm.resid = TRUE)
plot(geomod1)
```

---

# np.geo *Nonparametric geostatistical model (S3 class "np.geo")*

---

## Description

Defines a nonparametric geostatistical model (not intended to be used regularly; see np.fitgeo). Constructor function of the np.geo and fitgeo S3 classes.

## Usage

```
np.geo(lp, svm, svm0 = NULL, nbin = lp$grid$n)
```

## Arguments

| | |
|---|---|
| lp | local polynomial estimate of the trend function (object of class locpol.bin). |
| svm | (fitted) variogram model (object of class fitsvar or svarmod). |
| svm0 | (fitted) residual variogram model (object of class fitsvar or svarmod). |
| nbin | number of bins on each dimension. |

## Value

Returns an object of class np.geo (extends locpol.bin), the lp argument with the others and the vector of residuals as additional components.

## See Also

np.fitgeo, locpol, fitsvar.sb.iso.

`np.kriging` *Nonparametric (residual) kriging*

## Description

Compute simple kriging or residual kriging predictions (and also the corresponding simple kriging standard errors ). Currently, only global (residual) simple kriging is implemented.

## Usage

```
np.kriging(object, ...)

## Default S3 method:
np.kriging(
  object,
  svm,
  lp.resid = NULL,
  ngrid = object$grid$n,
  intermediate = FALSE,
  ...
)

## S3 method for class 'np.geo'
np.kriging(object, ngrid = object$grid$n, intermediate = FALSE, ...)

kriging.simple(x, y, newx, svm, intermediate = FALSE)
```

## Arguments

| | |
|---|---|
| `object` | object used to select a method: local polynomial estimate of the trend (class `locpol.bin`) or nonparametric geostatistical model (class extending `np.geo`). |
| `...` | further arguments passed to or from other methods. |
| `svm` | semivariogram model (of class extending `svarmod`). |
| `lp.resid` | residuals (defaults to `residuals(object)`). |
| `ngrid` | number of grid nodes in each dimension. |
| `intermediate` | logical, determines whether the intermediate computations are included in the output (component `kriging`; see Value). These calculations can be reused, e.g. for bootstrap. |
| `x` | vector/matrix with data locations (each component/row is an observation location). |
| `y` | vector of data (response variable). |
| `newx` | vector/matrix with the (irregular) locations to predict (each component/row is a prediction location). or an object extending `grid.par-class` (`data.grid`). |

## Value

`np.kriging()`, and `kriging.simple()` when `newx` defines gridded data (extends `grid.par` or `data.grid` classes), returns an S3 object of class `krig.grid` (kriging results + grid par.). A `data.grid` object with the additional (some optional) components:

| | |
|---|---|
| `kpred` | vector or array (dimension `$grid$n`) with the kriging predictions. |
| `ksd` | vector or array with the kriging standard deviations. |
| `kriging` | (if requested) a list with 4 components: |

- `lambda` matrix of kriging weights (columns correspond with predictions and rows with data)).
- `cov.est` (estimated) covariance matrix of the data.
- `chol` Cholesky factorization of `cov.est`.
- `cov.pred` matrix of (estimated) covariances between data (rows) and predictions (columns).

When `newx` is a matrix of coordinates (where each row is a prediction location), `kriging.simple()` returns a list with the previous components (`kpred`, `ksd` and, if requested, `kriging`).

## See Also

`np.fitgeo`, `locpol`, `np.svar`.

## Examples

```
geomod <- np.fitgeo(aquifer[,1:2], aquifer$head)
krig.grid <- np.kriging(geomod, ngrid = c(96, 96)) # 9216 locations
old.par <- par(mfrow = c(1,2))
simage(krig.grid, 'kpred', main = 'Kriging predictions',
       xlab = "Longitude", ylab = "Latitude", reset = FALSE )
simage(krig.grid, 'ksd', main = 'Kriging sd', xlab = "Longitude",
       ylab = "Latitude" , col = hot.colors(256), reset = FALSE)
par(old.par)
```

---

# 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,
  loss = c("MRSE", "MRAE", "MSE", "MAE"),
  ncv = 1,
  warn = FALSE,
  ...
```

```
)

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,
  verbose = plot,
  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). |
| h | (full) bandwidth matrix (controls the degree of smoothing; only the upper triangular part of h is used). |
| maxlag | maximum lag. Defaults to 55% of largest lag. |
| nlags | number of lags. Defaults to 101. |
| minlag | minimun lag. |
| 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 semivariances 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. |
| loss | character; CV error. See "Details" bellow. |
| warn | logical; sets the handling of warning messages (normally due to the lack of data in some neighborhoods). If FALSE all warnings are ignored. |
| 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. |

| | |
|---|---|
| verbose | logical; if TRUE, the errors (averages of the relative squared differences) at each iteration are printed. |
| ylim | y-limits of the plot (if plot == TRUE). |

## 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:<br>• degree degree of the local polinomial used.<br>• h smoothing matrix.<br>• rm mean of residual semivariances.<br>• rss sum of squared residual semivariances.<br>• ncv number of cells ignored in each direction.<br>• hat (if requested) hat matrix of the binned semivariances.<br>• 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). |
| ... | further arguments passed to or from other methods. |
| m | variogram model (e.g. svarmod object). |

## 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).

## Value

as.variogram() returns an object of the (not fully documented) **geoR**-class variogram.

as.variomodel() returns an object of the **geoR**-class variomodel.

## 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.

## Value

A variogramModel-class object of the **gstat** package.

## See Also

vgm, svarmod.

## npsp.tolerance *npsp Tolerances*

### Description

Returns a (convergence, taper, approximation,...) tolerance. Defaults to `.Machine$double.eps^(1/level)`, typically about `1e-8`.

### Usage

```
npsp.tolerance(level = 2, warn = TRUE)
```

### Arguments

| | |
|---|---|
| `level` | numerical, |
| `warn` | logical; If `TRUE` (the default) a warning message is issued when `level < 1`. |

### Value

Returns `.Machine$double.eps^(1/level)` if `level >= 1`, in other case `1 - .Machine$double.eps`.

### See Also

`.Machine`

### Examples

```
curve(npsp.tolerance, 1, 1000)
abline(h = npsp.tolerance(0, FALSE), lty = 2)
```

## plot.fitgeo *Plot a nonparametric geostatistical model*

### Description

Plots the trend estimates and the fitted variogram model.

### Usage

```
## S3 method for class 'fitgeo'
plot(x, y = NULL, main.trend = "Trend estimates", main.svar = NULL, ...)
```

### Arguments

| | |
|---|---|
| x | a nonparametric geostatistical model object. Typically an output of np.fitgeo. |
| y | ignored argument. |
| main.trend | title for the trend plot. |
| main.svar | title for the semivariogram plot. |
| ... | additional graphical parameters (to be passed to simage for trend plotting). |

### Value

No return value, called for side effects (generate the plot).

### See Also

np.fitgeo.

### Examples

```
geomod <- np.fitgeo(aquifer[,1:2], aquifer$head)
plot(geomod)
```

---

## precipitation *Precipitation data*

---

### Description

The data set consists of total precipitations during March 2016 recorded over 1053 locations on the continental part of USA.

### Format

A SpatialPointsDataFrame with 1053 observations on the following 6 variables:

**y** total precipitations (square-root of rainfall inches),

**WBAN** five-digit Weather station identifier,

**state** factor containing the U.S. state,

and the following attributes:

**labels** list with data and variable labels,

**border** SpatialPolygons with the boundary of the continental part of USA,

**interior** SpatialPolygons with the U.S. state boundaries.

### Source

National Climatic Data Center:
https://www.ncdc.noaa.gov/cdo-web/datasets.

## References

Fernandez-Casal R., Castillo-Paez S. and Francisco-Fernandez M. (2017) Nonparametric geostatistical risk mapping, *Stoch. Environ. Res. Ris. Assess.*, doi:10.1007/s004770171407y.

Fernandez-Casal R., Castillo-Paez S. and Garcia-Soidan P. (2017) Nonparametric estimation of the small-scale variability of heteroscedastic spatial processes, *Spa. Sta.*, doi:10.1016/j.spasta.2017.04.001.

## Examples

```
summary(precipitation)
scattersplot(precipitation)
```

---

`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 "`data.grid`").

## Usage

```
## S3 method for class 'data.grid'
image(
  x,
  data.ind = 1,
  xlab = NULL,
  ylab = NULL,
  useRaster = all(dim(x) > dev.size("px")),
  ...
)

## 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 "`data.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]`. |
| `useRaster` | logical; if `TRUE` a bitmap raster is used to plot the image instead of polygons. |

| | |
|---|---|
| ... | additional graphical parameters (to be passed to main plot function). |
| zlab | label for the z axis, defaults to names(x)[data.ind]. |
| filled | logical; if FALSE (default), function contour is called, otherwise filled.contour. |

## Value

image() and contour() do not return any value, call for secondary effects (generate the corresponding plot). persp() invisibly returns 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.

## 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)
```

---

# rule *npsp Rules*

---

## Description

Compute the number of classes for a histogram, the number of nodes of a binning grid, etc.

## Usage

```
rule(x, d = 1, rule = c("Rice", "Sturges", "scott", "FD"), ...)

rule.binning(x, ...)

## Default S3 method:
rule.binning(x, d = ncol(x), a = 2, b = d + 1, ...)

rule.svar(x, ...)

## Default S3 method:
rule.svar(x, d = ncol(x), a = 2, b = d + 1, ...)

## S3 method for class 'bin.den'
rule.svar(x, ...)
```

## Arguments

| | |
|---|---|
| x | data vector or object used to select a method. |
| d | (spatial) dimension. |
| rule | character; rule to be used. |
| ... | further arguments passed to or from other methods. |
| a | scale values. |
| b | exponent values. |

## Details

The Rice Rule, $m = \lceil 2n^{1/3} \rceil$, is a simple alternative to Sturges's rule (nclass.Sturges).

## Value

The rule values (vector or scalar).

rule.binning returns a vector with the suggested number of bins on each dimension.

rule.binning.default returns rep(ceiling(a * nrow(x) ^ (1 / b)), d).

rule.svar returns the suggested number of bins for variogram estimation.

rule.svar.default returns ceiling(a * (nrow(x)^2 / 4) ^ (1 / b)).

## See Also

hist, nclass.Sturges, nclass.scott, nclass.FD, binning, np.den, bin.den.

scattersplot *Exploratory scatter plots*

## Description

Draws (in a 2 by 2 layout) the following plots: a scatter plot with a color scale, the scatter plots of the response against the (first two) coordinates and the histogram of the response values.

## Usage

```
scattersplot(x, ...)

## Default S3 method:
scattersplot(
  x,
  z,
  main,
  xlab,
  ylab,
  zlab,
  col = hot.colors(128),
  lowess = TRUE,
  density = FALSE,
  omd = c(0.05, 0.95, 0.01, 0.95),
  ...
)

## S3 method for class 'SpatialPointsDataFrame'
scattersplot(
  x,
  data.ind = 1,
  main,
  xlab,
  ylab,
  zlab,
  col = hot.colors(128),
  lowess = TRUE,
  density = FALSE,
  omd = c(0.05, 0.95, 0.01, 0.95),
  ...
)
```

## Arguments

| | |
|---|---|
| x | object used to select a method. |
| ... | additional graphical parameters (to be passed to spoints). |

| | |
|---|---|
| z | vector of data (response variable). |
| main | an overall title for the plot. |
| xlab | a title for the axis corresponding to the first coordinate. |
| ylab | a title for the axis corresponding to the second coordinate. |
| zlab | a title for the axis corresponding to the response. |
| col | color table used to set up the color scale (see spoints). |
| lowess | logical. If TRUE, a lowess smooth is added to the plots of the response against the coordinates. |
| density | logical. If TRUE, a kernel density estimate is added to the histogram. |
| omd | a vector of the form c(x1, x2, y1, y2) giving the region inside outer margins in normalized device coordinates (i.e. fractions of the device region). |
| data.ind | integer (or character) with the index (or name) of the data component. |

## Details

Standard generic function with a default method, in which argument x is a matrix with the spatial coordinates (each row is a point).

scattersplot.SpatialPointsDataFrame sets default values for some of the arguments from attributes of the object x (if present; see e.g. precipitation).

## Value

No return value, called for side effects (generate the plot).

## See Also

splot, spoints, lowess, density

---

# 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 = 1,
  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,
  reset = TRUE,
  xlab = NULL,
  ylab = NULL,
  asp = NA,
  ...
)

## 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),
  tolerance = npsp.tolerance(),
  reset = TRUE,
  ...
)
```

## 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. |
| ... | 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). |
| 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. |
| 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. |
| 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). |
| 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). |
| reset | logical; if FALSE 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 (default) the plot parameters will be reset 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. |
| asp | the y/x aspect ratio, see plot.window. |
| 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. |
| tolerance | tolerance value (lower values are masked). |

## 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 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', reset = FALSE)
simage(lp, main = 'Estimated trend', reset = FALSE)
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,
  reset = TRUE,
  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. |
| ... | 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). |
| 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. |
| 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. |
| 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). |
| 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). |
| reset | logical; if FALSE 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 (default) the plot parameters will be reset 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. |
| 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")). |
| data.ind | integer (or character) with the index (or name) of the component containing the z values to be plotted. |

## 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 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', reset = FALSE)
spersp(lp, main = 'Estimated trend', zlab = 'y', reset = FALSE)
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. For instance, sxxxx functions (spoints, simage and spersp) draw the corresponding high-level plot (xxxx), after calling splot, to include 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 (the graphical parameters are not changed). |
| 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 splot, 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

```
# Plot equivalent to spoints():
scale.range <- range(aquifer$head)
res <- splot(slim = scale.range)
with( aquifer, plot(lon, lat, col = scolor(head, slim = scale.range),
       pch = 16, cex = 1.5, main = "Wolfcamp aquifer data"))
par(res$old.par) # restore graphical parameters
# 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 a scatter plot 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 = 1,
  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,
  reset = TRUE,
  pch = 16,
  cex = 1.5,
  xlab = NULL,
  ylab = NULL,
  asp = NA,
  ...
)

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

## S3 method for class 'SpatialPointsDataFrame'
spoints(x, data.ind = 1, main, xlab, ylab, legend.lab, ...)
```

## Arguments

| | |
|---|---|
| x | object used to select a method. In the default method, it provides the x coordinates for the plot (and optionally the y coordinates; any reasonable way of defining the coordinates is acceptable, see the function xy.coords for details). |
| ... | 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). |
| y | y coordinates. Alternatively, a single argument x can be provided. |
| s | numerical vector containing the values used for coloring the points. |
| 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. |
| 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). |

| | |
|---|---|
| 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, and legend is TRUE, these will be determined within the function. |
| smallplot | plot coordinates for legend strip. If not passed, and legend is TRUE, 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 scatter plot is just added to the existing plot. |
| reset | logical; if FALSE 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 (default) the plot parameters will be reset 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. |
| asp | the y/x aspect ratio, see plot.window. |
| data.ind | integer (or character) with the index (or name) of the data component. |
| main | an overall title for the plot. |

## Details

spoints.SpatialPointsDataFrame sets default values for some of the arguments from attributes of the object x (if present; see e.g. precipitation).

## 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 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 'svar.grid'
sv(x, h, ...)

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

### Arguments

| | |
|---|---|
| x | variogram model (svarmod object). |
| h | vector (isotropic case) or matrix of lag values. |
| ... | further arguments passed to or from other methods. |
| discretize | logical. If TRUE the variogram is previously discretized. |

### 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 (i.e. computes a binned sample variogram).

### 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. |
| y | vector of data (response variable). |
| maxlag | maximum lag. Defaults to 55% of largest lag. |
| nlags | number of lags. Defaults to max(12, rule.svar(x)). |
| minlag | minimun lag. |
| estimator | character, estimator name (e.g. "classical"). See "Details" below. |

## Details

Currently, only isotropic semivariogram estimation is supported.

If parameter nlags is not specified is set to max(12, rule.svar(x)).

## 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:<br>• x argument x.<br>• y argument y.<br>• med (weighted) mean of the (binned) semivariances. |
| svar | a list of 2 components:<br>• type character, type of estimation (e.g. "isotropic").<br>• estimator character, estimator name (e.g. "classical"). |

## See Also

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

svar.grid *Discretize a (semi)variogram model*

## Description

Discretizes a variogram model (to speed up variogram evaluation). Constructor function of the svar.grid-class.

## Usage

```
svar.grid(svar, log = TRUE, ...)

## S3 method for class 'svarmod'
svar.grid(
  svar,
  log = TRUE,
  n = 256,
  min = 10 * .Machine$double.eps,
  max = 1.1 * svar$range,
  ...
)
```

## Arguments

| | |
|---|---|
| svar | (fitted) variogram model (a svarmod or fitsvar object). |
| log | logical. If TRUE, the variogram is discretized in (base 2) logarithmic scale. |
| ... | further arguments passed to or from other methods. |
| n | number of lags. Defaults to 256. |
| min | minimun lag. Defaults to 10*.Machine$double.eps. |
| max | maximum lag. Defaults to 1.1*svar$range. |

## Value

A svar.grid-class object extending svarmod, bin.den and data.grid classes.

## See Also

svarmod, bin.den, data.grid.

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",
  xlim = NULL,
  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",
  xlim = NULL,
  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",
  xlim = NULL,
  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"). |
| xlim | x-limits. |
| ylim | y-limits. |
| lwd | line widths for points (estimates) and lines (fitted model) respectively. |
| add | logical; if TRUE the semivariogram plot is just added to the existing plot. |
| ... | additional graphical parameters (see par). |

## Value

No return value, called for side effects (generate the 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,
  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 $\sigma^2$ 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 $\nu_0$ (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,
  discretize = nrow(coords) > 256,
  ...
)

## 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. |
| discretize | logical. If TRUE (default), the variogram is (previously) discretized. |

## Value

The covariance matrix of the data.

## See Also

sv, covar.

# Index