Package ‘sgeostat’

February 3, 2016

Title An Object-Oriented Framework for Geostatistical Modeling in S+

Version 1.0-27

Author S original by James J. Majure <majure@iastate.edu> Iowa State University, R port + extensions by Albrecht Gebhardt <albrecht.gebhardt@aau.at>

Maintainer Albrecht Gebhardt <albrecht.gebhardt@aau.at>

Description An Object-oriented Framework for Geostatistical Modeling in S+ containing functions for variogram estimation, variogram fitting and kriging as well as some plot functions. Written entirely in S, therefore works only for small data sets in acceptable computing time.

Depends stats, grDevices, graphics, R (>= 2.0.0)

License GPL (>= 2)

Date 2016-02-02

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-02-03 01:11:19

R topics documented:

est.variogram .................................................. 2
fit.trend .......................................................... 3
fit.variogram ................................................... 4
identify.point ................................................... 6
in.chull .......................................................... 7
in.polygon ......................................................... 8
krige ............................................................... 9
lagplot ........................................................... 10
maas ............................................................... 12
maas.bank ......................................................... 12
pair ............................................................... 13
plot.point ........................................................ 14
plot.variogram ................................................ 15
point ............................................................. 16
est.variogram

print.pair .............................................. 17
print.point .............................................. 18
sgeostat-internal .................................... 19
spacebox ............................................... 19
spacecloud ............................................ 20

Index 22

est.variogram Variogram Estimator

Description

Calculate empirical variogram estimates.

An object of class variogram contains empirical variogram estimates generated from a point object and a pair object. A variogram object is stored as a data frame containing six columns: lags, bins, classic, robust, med, and n. The length of each vector is equal to the number of lags in the pair object used to create the variogram object, say l. The lags vector contains the lag numbers for each lag, beginning with one (1) and going to the number of lags (l). The bins vector contains the spatial midpoint of each lag. The classic, robust, and med vectors contain the classical,

\[ \gamma_c(h) = \frac{1}{n} \sum_{(i,j) \in N(h)} (z(x_i) - z(x_j))^2 \]

robust,

\[ \gamma_m(h) = \frac{\left( \frac{1}{n} \sum_{(i,j) \in N(h)} \left( \sqrt{|z(x_i) - z(x_j)|} \right)^4 \right)}{0.457 + \frac{0.494}{n}} \]

and median

\[ \gamma_m(h) = \frac{\left( \text{median}_{(i,j) \in N(h)} \left( \sqrt{|z(x_i) - z(x_j)|} \right)^4 \right)}{0.457 + \frac{0.494}{|N(h)|}} \]

variogram estimates for each lag, respectively (see Cressie, 1993, p. 75). The n vector contains the number |N(h)| of pairs of points in each lag N(h).

Usage

est.variogram(point.obj, pair.obj, a1, a2)

Arguments

point.obj a point object generated by point()
pair.obj a pair object generated by pair()
a1 a variable to calculate semivariogram for
a2 an optional variable name, if entered cross variograms will be created between a1 and a2
Value

A variogram object:

- `lags` vector of lag identifiers
- `bins` vector of midpoints of each lag
- `classic` vector of classic variogram estimates for each lag
- `robust` vector of robust variogram estimates for each lag
- `med` vector of median variogram estimates for each lag
- `n` vector of the number of pairs in each lag

References

http://www.gis.iastate.edu/SGeoStat/homepage.html

See Also

`point.pair`

Examples

```r
maas.v<est.variogram(maas.point,maas.pair,'zinc')
```

---

**fit.trend**  
*Fit polynomial trend functions*

Description

Fits a polynomial trend function to a `point` object. Similar to functions in B. Ripleys spatial library.

Usage

```r
fit.trend(point.obj, at, np=2, plot.it=TRUE)
```

Arguments

- `point.obj` point object
- `at` name of dependent variable in `point.obj`
- `np` degree of polynom to be fitted
- `plot.it` switches generation of a contour plot

Value

- `beta` estimated parameters

...
fit.variogram

References

http://www.gis.iastate.edu/SGeoStat/homepage.html

---

**fit.variogram**

**Variogram Model Fit**

**Description**

Fit variogram models (exponential, spherical, gaussian, linear) to empirical variogram estimates. An object of class variogram.model represents a fitted variogram model generated by fitting a function to a variogram object. A variogram.model object is composed of a list consisting of a vector of parameters, parameters, and a semi-variogram model function, model.

**Usage**

```r
fit.variogram(model="exponential", v.object, nugget, sill, range, slope, ...)
```

```r
fit.exponential(v.object, c0, ce, ae, type='c',
iterations=10, tolerance=1e-06, echo=FALSE, plot.it=FALSE, weighted=TRUE)
```

```r
fit.gaussian(v.object, c0, cg, ag, type='c',
iterations=10, tolerance=1e-06, echo=FALSE, plot.it=FALSE, weighted=TRUE)
```

```r
fit.spherical(v.object, c0, cs, as, type='c', iterations=10,
tolerance=1e-06, echo=FALSE, plot.it=FALSE, weighted=TRUE, delta=0.1, verbose=TRUE)
```

```r
fit.wave(v.object, c0, cw, aw, type='c',
iterations=10, tolerance=1e-06, echo=FALSE, plot.it=FALSE, weighted=TRUE)
```

```r
fit.linear(v.object, type='c', plot.it=FALSE, iterations=1, c0=0, cl=1)
```

**Arguments**

- **model**: only available for `fit.variogram`, switches what kind of model should be fitted ("exponential", "wave", "gaussian", "spherical", "linear").
- **v.object**: a variogram object generated by `est.variogram()`
- **nugget, sill, range, slope**: only available for `fit.variogram`, initial estimates for specified variogram model (slope only for `fit.linear`)
- **c0**: initial estimate for nugget effect, valid for all variogram types, partial sill (cX) and (asymptotical) range (aX) as follows:
- **ce, ae**: initial estimates for the exponential variogram model
- **cg, ag**: initial estimates for the gaussian variogram model
- **cs, as**: initial estimates for the spherical variogram model
- **cw, aw**: initial estimates for the periodical variogram model
- **cl**: initial estimates for the linear variogram model (slope)
fit.variogram

- **type**: one of 'c' (classic), 'r' (robust), 'm' (median). Indicates to which type of empirical variogram estimate the model is to be fit.
- **iterations**: the number of iterations of the fitting procedure to execute.
- **tolerance**: the tolerance used to determine if model convergence has been achieved.
- **delta**: initial stepsize (relative) for pseudo Newton approximation, applies only to `fit.spherical`
- **echo**: if TRUE, be verbose.
- **verbose**: if TRUE, be verbose (show iteration for spherical model fit).
- **plot.it**: if TRUE, the variogram estimate will be plotted each iteration.
- **weighted**: if TRUE, the fit will be done using weighted least squares, where the weights are given in Cressie (1991, p. 99)
- ... only `fit.variogram`: additional parameters to hand through to specific model fit functions

**Value**

A variogram.model object:

- **parameters**: vector of fitted model parameters
- **model**: function implementing a valid variogram model

**Note**

`fit.exponential`, `fit.gaussian` and `fit.wave` use an iterative, Gauss-Newton fitting algorithm to fit to an exponential or gaussian variogram model to empirical variogram estimates. `fit.spherical` uses the same algorithm but with differential quotients in place of first derivatives. When `weighted` is TRUE, the regression is weighted by \( n(h)/\gamma(h)^2 \) where the numerator is the number of pairs of points in a given lag.

Setting `iterations` to 0 means no fit procedure is applied. Thus parameter values from external sources can be plugged into a variogram model object.

**References**

http://www.gis.iastate.edu/SGeoStat/homepage.html

**See Also**

`est.variogram`

**Examples**

```r
# # automatic fit:
#
maas.vmod<-fit.gaussian(maas.v,c0=60000,cg=110000,ag=800,plot.it=TRUE, iterations=30)
#```
# iterations=0, means no fit, intended for "subjective" fit
#
maas.vmod.fixed<-fit.variogram("gaussian",maas.v,nugget=60000.sill=110000,
  range=800,plot.it=TRUE,iterations=0)

**identify.point**  
Identify points on a Point Object

Description

Plot variable values next to locations after the `plot.point()` function.

Usage

```r
## S3 method for class 'point'
identify(x, v, ...)
```

Arguments

- `x`  
a point object generated by `point()`
- `v`  
use values of variable "v" as labels
- `...`  
additional arguments to identify

Value

An integer vector containing the indexes of the identified points.

References

http://www.gis.iastate.edu/SGeoStat/homepage.html

See Also

`plot.point`

Examples

```r
plot(maas.point)
# use indices as labels:
identify(maas.point)
# use values as labels:
identify(maas.point,v="zinc")
```
in.chull

Convex hull test

Description
Checks if points are in the interior of a convex hull.

Usage
in.chull(x0, y0, x, y)

Arguments
- x0: coordinates of points to check
- y0: see x0
- x: coordinates defining the convex hull
- y: see x

Details
Uses a simple points-in-polygon check combined with the chull function.

Value
- comp1: Description of ‘comp1’
- comp2: Description of ‘comp2’

Author(s)
Albrecht Gebhardt <agebhard@uni-klu.ac.at>

References
Follows an idea from algorithm 112 from CACM (available at http://www.netlib.org/tomspdf/112.pdf)

See Also
in.convex.hull, chull

Examples
in.chull(c(0,1),c(0,1),c(0,1,0,-1),c(-1,0,1,0))
# should give: TRUE FALSE
in.polygon  In-Polygon test

Description
Checks if points are in the interior of a polygon.

Usage
\texttt{in.polygon(x0, y0, x, y)}

Arguments
\begin{itemize}
  \item \texttt{x0} coordinates of points to check
  \item \texttt{y0} see \texttt{x0}
  \item \texttt{x} coordinates defining the polygon
  \item \texttt{y} see \texttt{x}
\end{itemize}

Details
Uses a simple points-in-polygon check combined with the \texttt{polygon} function.
Polygon is closed automatically.

Value
\begin{itemize}
  \item \texttt{comp1} Description of ‘comp1’
  \item \texttt{comp2} Description of ‘comp2’
\end{itemize}

Author(s)
Albrecht Gebhardt <agebhard@uni-klu.ac.at>

References
Follows an idea from algorithm 112 from CACM (available at http://www.netlib.org/tomspdf/112.pdf)

See Also
\texttt{in.convex.hull}, \texttt{polygon}, \texttt{in.chull}

Examples
\begin{verbatim}
in.polygon(c(0,1),c(0,1),c(0,1,0,-1),c(-1,0,1,0))
# should give: TRUE FALSE
\end{verbatim}
krige

Krige

Description

Carry out spatial prediction (or kriging).

Usage

\texttt{krige(s, point.obj, at, var.mod.obj, maxdist=NULL, extrap=FALSE, border)}

Arguments

- \texttt{s}: a point object, generated by \texttt{point()}, at which prediction is carried out
- \texttt{point.obj}: a point object, generated by \texttt{point()}, containing the sample points and data
- \texttt{at}: the variable, contained in \texttt{point.obj}, for which prediction will be carried out
- \texttt{var.mod.obj}: variogram object
- \texttt{maxdist}: an optional maximum distance. If entered, then only sample points (i.e., in \texttt{point.obj}) within \texttt{maxdist} of each prediction point will be used to do the prediction at that point. If not entered, then all \texttt{n} sample points will be used to make the prediction at each point.
- \texttt{extrap}: logical, indicates if prediction outside the convex hull of data points should be done, default \texttt{FALSE}
- \texttt{border}: optional polygon (list with two components \texttt{x} and \texttt{y} of same length) representing a (possibly non convex) region of interest to be used instead of the convex hull. Needs \texttt{extrap=TRUE}.

Value

A point object which is a copy of the \texttt{s} object with two new variables, \texttt{zhat} and \texttt{sigma2hat}, which are, respectively, the predicted value and the kriging variance.

References

http://www.gis.iastate.edu/SGeoStat/homepage.html

See Also

\texttt{est.variogram}, \texttt{fit.variogram}
Examples

# a single point:
prdpnt <- point(data.frame(list(x=180000,y=331000)))
prdpnt <- krige(prdpnt, maas.point, 'zinc', maas.vmod)
prdpnt

# kriging on a grid (slow!)
grid <- list(x=seq(min(maas$x), max(maas$x), by=100),
y=seq(min(maas$y), max(maas$y), by=100))
grid$x <- range(grid$x)
grid$yr <- range(grid$y)
grid$max <- max(grid$x, grid$y)
grid$xy <- data.frame(cbind(c(matrix(grid$x, length(grid$x), length(grid$y)),
c(matrix(grid$y, length(grid$x), length(grid$y)), byrow=TRUE))),
colnames(grid$xy) <- c("x", "y")
grid$point <- point(grid$xy)
data(maas.bank)
grid$krige <- krige(grid$point, maas.point, 'zinc', maas.vmod,
maxdist=1000, extrap=FALSE, border=maas.bank)
op <- par(no.readonly = TRUE)
par(pty="s")
plot(grid$xy, type="n", xlim=c(grid$x[1], grid$x[1]+grid$max),
ylim=c(grid$y[1], grid$y[1]+grid$max))
image(grid$x, grid$y,
 matrix(grid$krige$zhat, length(grid$x), length(grid$y)),
 add=TRUE)
contour(grid$x, grid$y,
 matrix(grid$krige$zhat, length(grid$x), length(grid$y)),
 add=TRUE)
data(maas.bank)
lines(maas.bank$x, maas.bank$y, col="blue")
par(op)

---

lagplot

Lag Scatter Plot

Description

Create a spatially lagged scatter plot, e.g. plot z(s) versus z(s+h), where h is a lag in a pair object.

Usage

lagplot(point.obj, pair.obj, a1, a2, lag=1, std=FALSE, query.a, xlim, ylim)
**Arguments**

- `point.obj` a point object generated by `point()`
- `pair.obj` a pair object generated by `pair()`
- `a1` a variable to plot
- `a2` an optional variable name, if entered the plot will be created between `a1` and `a2`
- `lag` the lag to plot
- `std` a logical variable indicating whether the data should be standardized to their means and standard deviations before plotting
- `query.a` an optional variable name, if entered, the value of the variable will be displayed on the graphics device for points identified by the user.
- `xlim` a vector of length 2 indicating the x limits of the graphics page
- `ylim` a vector of length 2 indicating the y limits of the graphics page

**Value**

NULL

**Note**

When `query.a` is entered, the user will be prompted to identify points on the display device. Because each point in the plot represents a pair of locations, the user must identify each point twice, once for the "from" point and once for the "to" point. Querying is ended by pressing the middle mouse button on the mouse while the cursor is in the display window.

**References**

http://www.gis.iastate.edu/SGeoStat/homepage.html

**See Also**

- `point.pair`

**Examples**

```r
opar <- par(ask = interactive() & & .Device == "X11")
lagplot(maas.point, maas.pair, 'zinc')
# with identifying pairs:
lagplot(maas.point, maas.pair, 'zinc', lag=2, query.a='zinc')
par(opar)
```
maas

maas - zinc measurements

Description

Zinc measurements as groundwater quality variable.

Usage

maas

Value

list with components x, y and zinc.

References

gstat E.J. Pebesma (E.J.Pebesma@frw.uva.nl) http://www.frw.uva.nl/~pebesma/gstat/

See Also

maas

maas.bank

maas.bank - coordinates

Description

Coordinates of maas bank. To be used together with maas.

Usage

maas.bank

Value

list with components x and y.

References

gstat E.J. Pebesma (E.J.Pebesma@frw.uva.nl) http://www.frw.uva.nl/~pebesma/gstat/

See Also

maas
pair

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a pair object from a point object. A pair object contains information defining pairs of points contained in a point object. A pair object is a list containing five vectors: <code>from</code>, <code>to</code>, <code>lags</code>, <code>dist</code>, and <code>bins</code>. The length of each of these vectors (except <code>bins</code>) is equal to the number of pairs of points being represented, say ( k ). The vectors <code>from</code> and <code>to</code> contain pointers into the vectors of a point object, pointing to each member of the pair of points (e.g., <code>from[k]</code> points to ( s_i ) and <code>to[k]</code> points to ( s_j )). The vector <code>dist</code> contains the distance between the pairs of points. The vector <code>lags</code> contains the lag number to which each pair of points has been assigned. The vector <code>bins</code> contains the spatial midpoint between each lag and is used for plotting.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pair(point.obj, num.lags=10, type='isotropic', theta=0, dtheta=5, maxdist)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>point.obj</code></td>
</tr>
<tr>
<td><code>num.lags</code></td>
</tr>
<tr>
<td><code>type</code></td>
</tr>
<tr>
<td><code>theta</code></td>
</tr>
<tr>
<td><code>dtheta</code></td>
</tr>
<tr>
<td><code>maxdist</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A pair object:</td>
</tr>
<tr>
<td><code>from</code></td>
</tr>
<tr>
<td><code>to</code></td>
</tr>
<tr>
<td><code>lags</code></td>
</tr>
<tr>
<td><code>dist</code></td>
</tr>
<tr>
<td><code>bins</code></td>
</tr>
</tbody>
</table>
NOTE

Name of this function changed from pairs to pair to avoid conflicts with R’s pairs function!!

Note

When creating an anisotropic pair object, the assumption is that the direction, as well as the distance, between pairs of points is important in describing the variation. Using the theta and dtheta arguments, pairs of points that meet direction requirements can be selected. A pair of points will be included when the angle between the positive x axis and the vector formed by the pair of points falls within the tolerance angle given by (theta-dtheta,theta+dtheta)

References

http://www.gis.iastate.edu/SGeoStat/homepage.html

See Also

point

Examples

maas.pair <- pair(maas.point,num.lags=10,maxdist=2000)
maas.pair25 <- pair(maas.point,num.lags=10,type='anisotropic',
theta=25,maxdist=500)

plot.point

Plot Point Objects

Description

Plot the spatial locations in a point object, optionally coloring by quantile.

Usage

## S3 method for class 'point'
plot(x, v, legend.pos=0,axes=TRUE,xlab='',ylab='', add=FALSE, ...)

Arguments

x a point object generated by point()

v an optional variable name, if entered will divide the points into quantiles and color using 4 colors

legend.pos position of legend (0 - none, 1 - bottom-left, 2 -bottom-right, 3 - top-right, 4 - top-left), requires Lang(v)

axes logical, whether to plot axes

xlab,ylab axes labels, default none
plot.variogram

add usefull for overlaying images with a point plot
... additional arguments for plot

Value

NULL

References

http://www.gis.iastate.edu/SGeoStat/homepage.html

See Also

point

Examples

plot(maas.point)
plot(maas.point,v='zinc')
plot(maas.point,v='zinc',xlab='easting',ylab='northing',axes=TRUE,legend.pos=4)
# plot additionally the maas bank:
data(maas.bank)
lines(maas.bank)

plot.variogram Plot Variogram

Description

Plot empirical variogram estimates, optionally plotting a fitted variogram model.

Usage

## S3 method for class 'variogram'
plot(x, var.mod.obj, title.str, ylim, type='c',N=FALSE, ...)

Arguments

x a variogram object generated by est.variogram()
var.mod.obj a variogram model object generated by a model fitting routine.
title.str optional: an user supplied plot title
type optional: which type of variogram model to plot, 'c' = classical, 'r' = robust, 'm' median
N logical, toggles printing of absolute pair counts per lag
ylim optional user supplied y dimension for the plot
... additional arguments for plot
Value

NULL

References

http://www.gis.iastate.edu/SGeoStat/homepage.html

See Also

est.variogram

Examples

# two plots
oldpar <- par(mfrow=c(2,1))
plot(maas.v)
plot(maas.v, var.mod.obj=maas.vmod)
par(oldpar)

point Point Object

Description

Create an object of class point from a data frame.

An object of class point represents the observed data of a spatial process. This includes the spatial location of sampling sites and the values observed at those sites. A point object is stored as a data frame. The data frame must contain one column for the X coordinate and one column for the Y coordinate of each point, as well as any number of columns representing data observed at the points.

Usage

point(dframe, x='x', y='y')

Arguments

dframe a data frame containing the x and y coordinates for each point and the variables observed at each point
x the name of the column in dframe that contains the x coordinate
y the name of the column in dframe that contains the y coordinate
Value

A point object:

- `x` vector of x coordinates
- `y` vector of y coordinates
- `var1` vector of the first variable
- ...
- `varm` vector of the mth variable

References

http://www.gis.iastate.edu/SGeoStat/homepage.html

See Also

point

Examples

data(maas)
maas.point <- point(maas)

print.pair  Pairs Object Description

Description

Print descriptive information about a pair object.

Usage

```r
## S3 method for class 'pair'
print(x,...)
```

Arguments

- `x` a pair object generated by `pair()`
- `...` additional arguments for `print`

Value

NULL

References

http://www.gis.iastate.edu/SGeoStat/homepage.html
See Also

pair

Examples

print(maas.pair)
# gives:
# Pairs object: maas.pair
# 
# Type: isotropic
# Number of pairs: 8370
# Number of lags: 10
# Max h: 1999.867
Examples

```r
print.point(maas.point)
# gives
# Point object: maas.point
#
# Locations: 155
#
# Attributes:
#   x
#   y
#   zinc
```

Description

Internal sgeostat functions

Details

These functions are not intended to be called by the user.

The `krige` function interfaces to `krige.*`, `pair` to `pair.*` and `fit.trend` to `trend.*`.

spacebox

Boxplot of Variogram Cloud

Description

Create boxplots of square-root or squared differences between variable values at pairs of points versus the distance between the points.

Usage

```r
spacebox(point.obj, pair.obj, a1, a2, type='r')
```

Arguments

- `point.obj`: a point object generated by `point()`
- `pair.obj`: a pairs object generated by `pair()`
- `a1`: a variable to plot
- `a2`: an optional variable name, if entered the plot will be created between `a1` and `a2`
- `type`: either 'r' for square-root differences or 's' for squared differences
spacecloud

Value
NULL

References
http://www.gis.iastate.edu/SGeoStat/homepage.html

See Also
point, pair

Examples

spacecloud(maas.point, maas.pair, 'zinc')

spacecloud Variogram Cloud

Description
Create a scatter plot of square-root or squared differences between variable values at pairs of points versus the distance between the points.

Usage

spacecloud(point.obj, pair.obj, a1, a2, type='r', query.a, ...)

Arguments

point.obj a point object generated by point()
pair.obj a pair object generated by pair()
a1 a variable to plot
a2 an optional variable name, if entered the plot will be created between a1 and a2
type either 'r' for square-root differences or 's' for squared differences
query.a an optional variable name, if entered, the value of the variable will be displayed on the graphics device for points identified by the user.
... additional arguments for plot

Value
NULL
**Note**

When `query.a` is entered, the user will be prompted to identify points on the display device. Because each point in the plot represents a pair of locations, the user must identify each point twice, once for the "from" point and once for the "to" point. Querying is ended by pressing the middle mouse button on the mouse while the cursor is in the display window.

**References**

http://www.gis.iastate.edu/SGeoStat/homepage.html

**See Also**

`point`, `pair`

**Examples**

```r
opar <- par(ask = interactive() & .Device == "X11")
spacecloud(maas.point,maas.pair,'zinc')
# identify some points:
spacecloud(maas.point,maas.pair,'zinc',query.a='zinc')
par(opar)
```
Index

*Topic datasets
  maas, 12
  maas.bank, 12

*Topic spatial
  est.variogram, 2
  fit.trend, 3
  fit.variogram, 4
  identify.point, 6
  in.chull, 7
  in.layer, 8
  krige, 9
  lagplot, 10
  pair, 13
  plot.point, 14
  plot.variogram, 15
  point, 16
  print.pair, 17
  print.layer, 18
  sgeostat.internal, 19
  spacebox, 19
  spacecloud, 20

  calcangle (sgeostat.internal), 19
  chull, 7
  est.variogram, 2, 5, 9, 16
  export.layer (sgeostat.internal), 19

  fit.exponential (fit.variogram), 4
  fit.gaussian (fit.variogram), 4
  fit.linear (fit.variogram), 4
  fit.spherical (fit.variogram), 4
  fit.trend, 3, 19
  fit.variogram, 4, 9
  fit.wave (fit.variogram), 4

  identify.point, 6
  in.chull, 7, 8
  in.convex.hull, 7, 8
  in.layer, 8

  kriging.9, 19
  kriging.all (sgeostat.internal), 19
  kriging.maxdist (sgeostat.internal), 19

  lagplot, 10
  maas, 12, 12
  maas.bank, 12, 12

  pair, 3, 11, 13, 18–21
  pair.aniso (sgeostat.internal), 19
  pair.iso (sgeostat.internal), 19
  pair.newangle (sgeostat.internal), 19
  pairs, 14
  plot.layer, 6, 14
  plot.variogram, 15
  point, 3, 11, 14, 15, 16, 17, 18, 20, 21
  polygon, 8
  prediction.matrix (sgeostat.internal), 19
  print.pair, 17
  print.layer, 18

  sgeostat.internal, 19
  spacebox, 19
  spacecloud, 20

  trend.matrix (sgeostat.internal), 19
  trend.value (sgeostat.internal), 19

  which.na (sgeostat.internal), 19