Package 'icosa'

April 18, 2017

Title Global Triangular and Penta-Hexagonal Grids Based on Tessellated Icosahedra

Version 0.9.81

Description Employs triangular tessellation to refine icosahedra defined in 3d space. The procedures can be set to provide a grid with a custom resolution. Both the primary triangular and their inverted penta-hexagonal grids are available for implementation. Additional functions are provided to position points (latitude-longitude data) on the grids, to allow 2D and 3D plotting, use raster data and shapefiles.

Depends R (>= 3.2.2), rgl

Date 2017-04-17

License GPL-3

Encoding UTF-8

LazyData true

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 5.0.1

LinkingTo Rcpp

Imports Rcpp, sp, raster, igraph, rgdal, methods

NeedsCompilation yes

Author Adam T. Kocsis [aut, cre]

Maintainer Adam T. Kocsis <adam.kocsis@outlook.com>

Repository CRAN

Date/Publication 2017-04-18 16:04:55 UTC

R topics documented:

arcdist *Calculation of distances along arcs*

Description

This function calculates the shortest arc distance between two points.

Usage

```
arcdist(p1, p2, output = "distance", origin = c(0, 0, 0),
  radius = authRadius)
```

Arguments

p1	Numeric vector, XYZ or longitude-latitude coordinates of the first point along the arc.
p2	Numeric vector, XYZ or longitude-latitude coordinates of the last point along the arc.
output	Character value, the type of the output value. "distance" will give the distance in the metric that was fed to the function for the coordinates or the radius. "deg" will output the the distance in degrees, "rad" will do so in radians.
origin	Numeric vector, the center of the circle in XYZ coordinates (default is 0,0,0).
radius	Numeric value, the radius of the circle in case the input points have polar coordinates only. Unused when XYZ coordinates are entered. Defaults to the authalic radius of Earth ca. 6371.007km.

Value

A single numeric value.

Examples

```
# coordinates of two points
point1<- c(0,0)
point2<- c(180,0)
arcdist(point1,point2,"distance")
```

`arcdistmat` *Calculation of distance matrices along arcs*

Description

This function calculates the shortest arc distance matrix between two set of points.

Usage

```
arcdistmat(points1, points2 = NULL, origin = c(0, 0, 0),
  output = "distance", radius = authRadius)
```

Arguments

`points1`	Numeric matrix, XYZ or longitude-latitude coordinates of the first set of points.
`points2`	Numeric matrix, XYZ or longitude-latitude coordinates (matrix) of the second set of points. Leave this empty if you want all the arc distances between a set of points
`origin`	Numeric vector, the center of the circle in XYZ coordinates (default is 0,0,0).
`output`	Character value, the type of the output value. `"distance"` will give back the distance in the metric that was fed to the function in the coordinates or the radius. `"deg"` will output the the distance in degrees, `"rad"` will do so in radians.
`radius`	Numeric value, the radius of the circle in case the input points have polar coordinates only. Unused when XYZ coordinates are entered. Defaults to the authalic radius of Earth ca. 6371.007km.

Details

This function will create all possible shortest arc distances between points in the two sets, but not between the points within the sets. The function is useful for great circle distance calculations. For a symmetrical distance matrix leave the `points2` argument empty.

Value

A single numeric value.

Examples

```
g <- trigrid(c(4))
res <- arcdistmat(g@vertices)

rand<-rpsphere(500)
res2 <- arcdistmat(g@vertices, rand)
```

arcpoints *Calculation of points along an arc*

Description

This function calculates points along an arc between two points and and a circle center.

Usage

```
arcpoints(p1, p2, breaks = 2, origin = c(0, 0, 0), onlyNew = FALSE,
  output = "cartesian", radius = authRadius)
```

Arguments

p1	Numeric vector, XYZ or longitude-latitude coordinates of the first point along the arc.
p2	Numeric vector, XYZ or longitude-latitude coordinates of the last point along the arc.
breaks	Single positive integer, the number of points inserted between p1 and p2.
origin	Numeric vector, The center of the circle in XYZ coordinates (default is 0,0,0).
onlyNew	Logical value whether of p1 and p2 should be omitted from the result or not.
output	Character value, the coordinate system of the output points. Can either be "polar" for longitude-latitude or "cartesian" for XYZ data.
radius	Numeric value, the radius of the circle in case the input points have only polar coordinates. Unused when XYZ coordinates are entered. Defaults to the authalic radius of Earth ca. 6371.007km.

Details

The function always returns the smaller arc, with angle alpha < pi.

Value

Either an XYZ or a long-lat numeric matrix.

Examples

```
# empty plot
plot(NULL, NULL, xlim=c(-180, 180), ylim=c(-90,90))
# then endpoints of the arc
point1<-c(-45,-70)
point2<-c(130,65)
points(arcpoints(point1, point2, breaks=70, output="polar"))
```

CarToPol *Conversion of 3d Cartesian coordinates to polar coordinates*

Description

The function uses basic trigonometric relationships to transform xyz coordinates to polar coordinates

Usage

```
CarToPol(matXYZ, norad = FALSE, origin = c(0, 0, 0))
```

Arguments

matXYZ	3-column numeric matrix containing xyz coordinates in a Cartesian space
norad	Logical value, toggles whether the rho coordinate (distance from origin) should be omitted or not.
origin	Numeric vector of length 3, defining the center of the sphere. Defaults to c(0,0,0).

Value

A 3-column or 2-column numeric matrix with longitude, latitude and, if set accordingly, radius data.

Examples

```
# some random points
xyz <- rbind(
 c(6371, 0,0),
 c(0, 6371,0),
 c(1000,1000,1000)
)

# conversions
  CarToPol(xyz)
```

centers *The face centers of an icosahedral grid object*

Description

Shorthand function to return the faceCenters slot of an icosahedral grid or a grid linked to a face-layer.

Shorthand function to return the faceCenters slot of an icosahedral grid .

Shorthand function to return the faceCenters slot of the linked icosahedral grid .

Usage

```
centers(x, ...)

## S4 method for signature 'trigrid'
centers(x, output = "polar")

## S4 method for signature 'facelayer'
centers(x, output = "polar")
```

Arguments

x	The grid or facelayer object.
...	arguments passed to the class specific methods.
output	the coordinate system of the output. Either "polar" or "cartesian".

---

chullsphere *Spherical convex hull*

Description

This function calculates a possible implementation of the spherical convex hull

Usage

```
chullsphere(data, center = c(0, 0, 0), method = "centroidprojection",
  param = 200, inner = 10)
```

Arguments

data	Numeric matrix, XYZ or longitude-latitude coordinates of the set of points.
center	Numeric vector, The center of the sphere in XYZ coordinates (default is 0,0,0).
method	Character value, indicating the method to create the spherical convex hulls.
param	Single positive integer, indicates the number of divisions in the centroidprojection method. The higher the number, the closer the replacement points are to #' the centroid.
inner	Single positive integer, the number of points inserted between every two points of the spherical centroid. Heavilly impacts the performance.

Details

With the method `centroidprojection` the function calls the surfacecentroid() function to get the a reference point from the shape. Then all the points are 'projected' close to this point using the great circles linking them to the reference point. Each such great circle will be devided to an equal number of points and the closest will replace the original point coordinates in the convex hull algorithm implemented in `grDevices::chull()`.

Value

Either an XYZ or a long-lat numeric vector.

Examples

```
# generate some random points
allData <- rpsphere(1000)
# select only a subset
points<-allData[allData[,1]>3000,]
chullsphere(points)
```

`edgelength` *Lengths of grid edges*

Description

This function will return the length of all edges in the specified grid object.

Usage

```
edgelength(gridObj, ...)
```

Arguments

`gridObj`	A `trigrid` or `hexagrid` class object.
`...`	arguments passed to the class specific methods.

Value

A named numeric vector.

Examples

```
g <- trigrid(3)
edges <- edgelength(g, output="deg")
edges
```

edgelength,trigrid-method *Lengths of grid edges*

Description

This function will return the length of all edges in the specified grid object.

Usage

```
## S4 method for signature 'trigrid'
edgelength(gridObj, output = "distance")
```

Arguments

gridObj	A trigrid or hexagrid class object.
output	Character value, the type of the output. "distance" will give back the distance in the metric that was fed to the function in the coordinates or the radius. "deg" will output the the distance in degrees, "rad" will do so in radians.

edges *The edges of a 3d object*

Description

Shorthand function to get the edges slot of an icosahedral grid or a grid linked to a facelayer.

Usage

```
edges(x)

## S4 method for signature 'obj3d'
edges(x)

## S4 method for signature 'facelayer'
edges(x)

## S4 method for signature 'facelayer'
edges(x)
```

Arguments

x	The grid or facelayer object.

facelayer *Container for data storage using the faces on icosahedral grid*

Description

Container for data storage using the faces on icosahedral grid

Arguments

gridObj	A hexagrid or trigrid object.
value	The facelayer will be initialized with these values/this value

Examples

```
g <- trigrid(c(4,4))
fl <- facelayer(g, 1:length(g))
faces3d(fl)
```

faces *The faces of a 3d object*

Description

Shorthand function to get the faces slot of an icosahedral grid or a grid linked to a facelayer.

Usage

```
faces(x)

## S4 method for signature 'trigrid'
faces(x)

## S4 method for signature 'gridlayer'
faces(x)

## S4 method for signature 'facelayer'
faces(x)
```

Arguments

x	The grid or facelayer object.

faces3d *Methods of 3d face plotting.*

Description

This is a generic function used to plot the faces of either a trigrid or a hexagrid object in 3d space.

This is a generic function used to plot the faces of either a trigrid or a hexagrid object in 3d space.

This is a generic function used to plot the faces of either a trigrid or a hexagrid object in 3d space.

This is a generic function used to plot the faces of a facelayer type object.

Usage

```
faces3d(x, ...)

## S4 method for signature 'trigrid'
faces3d(x, ...)

## S4 method for signature 'hexagrid'
faces3d(x, ...)

## S4 method for signature 'facelayer'
faces3d(x, col = "red", ...)
```

Arguments

x	The trigrid, hexagrid or facelayer object to be plotted.
...	Further graphical parameters passed to (see plot3d) and the heatMapLegend() function.
col	graphical parameter indicating the colours of the faces. A single value is accepted for logical values. Multiple colors will be passed to grDevices::colorRampPalette(), to create palettes for heat maps in case of numeric values. The defaul plotting method in this case is the reversed grDevices::heat.colors (). In case of categorical data, random colors will be chosen.

Details

The function is built on the openGL renderer of the R package rgl.

The function is built on the openGL renderer of the R package rgl.

The function is built on the openGL renderer of the R package rgl.

The function is built on the openGL renderer of the R package rgl.

Value

The function does not return any value.

The function does not return any value.

The function does not return any value.

The function does not return any value.

Examples

```
# create a hexagonal grid
    g <- hexagrid(c(2,2))
# plot the grid in 3d space
    plot3d(g)
# make a subset to select faces
   subG <- subset(g, c("F5", "F2"))
# plot the subset defined above
    faces3d(subG, col="orange")
```

---

gridgraph *Create or instantiate an 'igraph' class graph from the faces of an icosahedral grid*

Description

The function can be applied to both grids and facelayers.

The function can be applied to a trigrid class object.

The function can be applied to a hexagrid class object.

The function can be applied to a facelayer class object of logical values. The resulting graph will have the characteristics of the original grid (directed/undirected etc.).

Usage

```
gridgraph(x, ...)

## S4 method for signature 'trigrid'
gridgraph(x, directed = FALSE, distances = FALSE)

## S4 method for signature 'hexagrid'
gridgraph(x, directed = FALSE, distances = FALSE)

## S4 method for signature 'facelayer'
gridgraph(x)
```

Arguments

x	the icosahedral grid or facelayer.
...	arguments passed to the class specific methods.
directed	logical value, defaults to FALSE creating an undirected graph. If TRUE than the graph will be directed.
distances	logical values, defaults to FALSE. If TRUE than the distances between the linked faces will be calculated and will be rendered to the edges as [["dist"]].

Value

The function returns an undirected igraph graph.

The function returns an 'igraph' graph.

The function returns an 'igraph' graph.

The function returns an 'igraph' graph.

---

gridlabs *Labels of grid vertices, faces and edges*

Description

This function will show where the grid elements are located.

Usage

```
gridlabs(gridObj, type = "f", projargs = NULL, ...)
```

Arguments

gridObj	a trigrid or hexagrid class icosahedral grid.
type	The type of element to be plotted: either "f" (faces), "v" (vertices) or "e" (edges).
projargs	a projection string for the transformation fo coordinates. Accepts both a CRS class object and a character string that will be transformed to the CRS class.
...	Arguments passed to the text() function.

Value

the function has no return value

gridlabs3d *Display the names of the grid elements in 3d plots.*

Description

This function will display the names of vertices, faces and edges on 3d plots.

Usage

```
gridlabs3d(gridObj, ...)

## S4 method for signature 'trigrid'
gridlabs3d(gridObj, type = "f", ...)

## S4 method for signature 'hexagrid'
gridlabs3d(gridObj, type = "f", ...)
```

Arguments

gridObj	A trigrid or hexagrid object to be plotted.
...	further arguments passed to text3d of the rgl package.
type	A character vector containing either "f", "e" or "v", rendering the names of either the faces, edges or vertives respectively.

Value

The function does not return any value.

Examples

```
# create a hexagonal grid
    g <- hexagrid(c(2,2))
# plot the grid in 3d space
   plot3d(g, guides=FALSE)
# plot the names of the faces
gridlabs3d(g, type="f", col="red")
# plot the names of the vertices
    gridlabs3d(g, type="v", col="blue", cex=0.6)
```

guides3d *Guides for 3d spherical plotting.*

Description

This function plots 3d guidelines for navigation on the surface of the sphere, includings the drawing of the rotational axis and a polar coordinate system.

Usage

```
guides3d(axis = 1.5, polgrid = c(30, 30), textPG = FALSE, res = 1,
  origin = c(0, 0, 0), radius = authRadius, drad = 1.1, ...)
```

Arguments

axis	Numeric argument draws the -90(lat. deg.) +90 (lat. deg.) axis. The plotted radius will be 'axis' times the authalic radius ca. 6371km.
polgrid	numeric argument with the length of 2, where the first argument specifies the size of the longitudinal and the second the latitudinal divisions (degrees).
textPG	logical value, indicating whether the coordinate values should be added to the 3d render.
res	numeric argument for graphical resolution of the curves: the distance in degrees between the points of the rendered guides.
origin	Numeric vector of length=3. Indicates the center of the guiding sphere.
radius	Numeric values indicating the radius of the guiding sphere. Defaults to the R2 radius of Earth (6371.007km).
drad	Numeric value, indicates the position of coordinate 3d text relative to the guiding sphere radius.
...	additional arguments passed to rgl::segments3d(), rgl::lines3d() and rgl::text3d().

Details

The function is built on the openGL renderer of the R package `rgl`.

Value

The function does not return any value.

Examples

```
# create a hexagonal grid
    g <- hexagrid(c(2,2))
# plot the grid in 3d space
   plot3d(g, guides=FALSE)
# plot the rotational axis in blue
guides3d(axis=2, polgrid=NULL, col="blue")
```

```
# plot the polar grid at 10 degree resolution
    guides3d(axis=NULL, polgrid=c(10,10), col="red")
# plot some coordinates
   guides3d(axis=NULL, polgrid=c(30,30), textPG=TRUE, col="orange", cex=1.4)
```

---

heatMapLegend *Legend for a heatmap with predefined colors.*

Description

This function will invoke the plot function to draw a heatmap legend.

Usage

```
heatMapLegend(cols, minVal, varName, tick.text, maxVal, ticks = 5,
  tick.cex = 1.5, barWidth = 3, barHeight = 50, tickLength = 1,
  xLeft = 88, yBot = 25, add = FALSE, ...)
```

Arguments

cols	Character vector, containnig the ordered colors that are used for the heatmap.
minVal	If tick.text is missing, the lowest value in the heatmap
varName	The label of the variable name plotted to the heatmap.
tick.text	The values on the heatmap legend. If missing, will be calculated with minVal and maxVal. Should have the length as 'ticks'.
maxVal	If tick.text is missing, the highest value in the heatmap
ticks	The number of ticks/values in the heatmap legend (the bar will be divided to this number).
tick.cex	Letter size of the values on the legend.
barWidth	The width (percent) of the bar featuring the colors of the heatmap.
barHeight	The height (percent)of the bar featuring the colors of the heatmap.
tickLength	The length (percent) of the ticks at the bars.
xLeft	the x coordinate of the lower left hand corner of the bar.
yBot	the y coordinate of the lower left hand corner of the bar.
add	indicates wheter a new plot should be drawn or not. Defaults to FALSE.
...	arguments passed to the plot() function.

Details

The 'percents' refer to the plotting area measured from the lower left corner.

hexagrid *A penta-hexagonal icosahedral grid*

Description

hexagrid creates a hexa-pentagonal grid based on the inversion of a tessellated icosahedron.

Usage

```
## S4 method for signature 'hexagrid'
initialize(.Object, tessellation = 1, sp = FALSE,
  graph = TRUE, center = origin, radius = authRadius)
```

Arguments

.Object	non-argument pointing to self.
tessellation	An integer vector with the tessellation values. Each number describes the number of new edges replacing one original edge. Multiple series of tessellations are possible this way. The total tessellation is the product of the tessellation vector. Higher values result in more uniform cell sizes, but the larger number of tessellation series, increases the speed of lookup functions
sp	A logical value indicating whether the 'SpatialPolygons' class representation of the grid should be added to the object when the grid is calculated. If set to true the SpPolygons() function will be run with with the resolution parameter set to 25. The resulting object will be stored in slot @sp. As the calculation of this object can increase the grid creation time substantially by default this argument has a value FALSE. This can be added on demand by running the function newsp.
graph	A logical value indicating whether the 'igraph' class representation of the grid should be added to the object when the grid is calculated. This argument defaults to TRUE because this option has only minor performance load on the grid constructor function. For familiarization with the object structure, however, setting this parameter to FALSE might help, as invoking str() on the 'igraph' class slot of the class might flood the console.
center	The origin of the grid in the reference Cartesian coordinate system. Defaults to (0,0,0).
radius	The radius of the grid. Defaults to the authalic radius of Earth.

Details

Inherits from the trigrid class.

The grid structure functions as a frame for data graining, plotting and calculations. Data can be stored in layers that are linked to the grid object. In the current version only the facelayer class is implemented which allows the user to render data to the cells of the grid which are called faces. The grid 'user interface' is made up of four primary tables: the @vertices table for the coordinates of

the vertices, the `faceCenters` for the coordinates of the centers of faces, the `faces` and the `edges` tables that contain which vertices form which faces and edges respectively. In these tables, the faces and vertices are sorted to form spirals that go from the north pole in a counter-clockwise direction. In case grid subsetting is performed these tables get truncated.

At finer resolutions, the large number of spatial elements render all calculations very resource demanding and slow, therefore the hierarchical structure created during the tessellation procedure is retained for efficient implementations. These data are stored in a list in the slot `@skeleton` and are 0-indexed integer tables for Rccp-based functions. `$v` stores vertex, `$f` the edge, and `$e` contains the edge data for plotting and calculations. In these tables the original hierarchy based orderings of the units are retained, during subsetting, additional vectors are used to indicate deactivation of these units. Any sort of meddling with the @skeleton object will lead to unexpected behavior.

Value

A hexagonal grid object, with class `hexagrid`.

Slots

`vertices` Matrix of the vertex coordinates.

`faces` Matrix of the verticies forming the faces

`edges` Matrix of the vertices forming the edges.

`tessellation` Contains the tessellation vector.

`orientation` Contains the grid orientation in xyz 3d space, values in radian.

`center` The xyz coordinates of the grid's origin/center.

`div` Contains the number of faces that a single face of the previous tessellation level is decomposed to.

`faceCenters` Contains the xyz coordinates of the centers of the faces on the surface of the sphere.

Examples

```
g <- hexagrid(c(8))
g1 <- hexagrid(c(2,3,4))
```

---

icosa *Global Triangular and Hexa-Pentagonal Grids Based on Tessellated Icosahedra*

Description

The **icosa** package provides tools to aggregate and analyze geographic data using grids based on tessellated icosahedra. The procedures can be set to provide a grid with a custom resolution. Both the primary triangular and their inverted penta- hexagonal grids are available for implementation. Additional functions are provided to position points (latitude-longitude data) on the grids, to allow 2D and 3D plotting, use raster data and shapefiles.

Details

This is the 0.9 (Beta) version. Notes about found bugs and suggestions are more than welcome!

Author(s)

Adam T. Kocsis (adam.kocsis@outlook.com)

Examples

```
# Create a triangular grid
tri <- trigrid(c(2,2))
```

---

length,trigrid-method *The length of a trigrid or, hexagrid class object.*

Description

The length of the object is interpreted as the number of faces it contains.

This function returns the number of values present in the gridlayer.

Usage

```
## S4 method for signature 'trigrid'
length(x)

## S4 method for signature 'gridlayer'
length(x)
```

Arguments

x	the object.

Value

An integer value

lines,trigrid-method *Lines method for the trigrid and hexagrid classes*

Description

This function will invoke the `lines()` method of the SpatialPolygons class.

Usage

```
## S4 method for signature 'trigrid'
lines(x, projargs = NULL, ...)
```

Arguments

`x`	the trigrid or hexagrid class object.
`projargs`	a projection string for the transformation fo coordinates. Accepts both a CRS class object and a character string that will be transformed to the CRS class.
`...`	arguments passed to the sp::lines() method of the SpatialPolygons class.

lines3d,Line-method *3d plotting method of a Line class object*

Description

The function draws the segments of a Line clas object in 3d.

The function draws the segments of a Line clas object in 3d.

lines3d method for the SpatialLines class

lines3d method for the SpatialLinesDataFrame

lines3d method for the Polygon class

lines3d method for the Polygon sclass

lines3d method for the SpatialPolygons sclass

lines3d method for the SpatialPolygonsDataFrame sclass

Usage

```
## S4 method for signature 'Line'
lines3d(x, y = NULL, z = NULL, ...)

## S4 method for signature 'Lines'
lines3d(x, y = NULL, z = NULL, ...)

## S4 method for signature 'SpatialLines'
```

```
lines3d(x, y = NULL, z = NULL, ...)

## S4 method for signature 'SpatialLinesDataFrame'
lines3d(x, y = NULL, z = NULL, ...)

## S4 method for signature 'Polygon'
lines3d(x, y = NULL, z = NULL, ...)

## S4 method for signature 'Polygons'
lines3d(x, y = NULL, z = NULL, ...)

## S4 method for signature 'SpatialPolygons'
lines3d(x, y = NULL, z = NULL, ...)

## S4 method for signature 'SpatialPolygonsDataFrame'
lines3d(x, y = NULL, z = NULL, ...)
```

Arguments

x	the SpatialLines object
y	unused argument of the lines3d() generic.
z	unused argument of the lines3d() generic.
...	arguments passed to rgl::lines3d()

---

lines3d,trigrid-method

Methods of 3d line plotting.

Description

This is a generic function used to plot the edge lines of either a `trigrid` or a `hexagrid` object in 3d space. The method is also implemented for the object classes defined by the package 'sp'.

Usage

```
## S4 method for signature 'trigrid'
lines3d(x, arcs = FALSE, ...)
```

Arguments

x	The `trigrid`, `hexagrid`, `facelayer` or `sp` object to be plotted.
arcs	Logical value setting whether great circle arcs or segments shall be drawn betwenn the points of the grid.
...	Further graphical parameters passed to (see `plot3d`).

Details

The function is built on the openGL renderer of the R package `rgl`.

Value

The function does not return any value.

Examples

```
# create a hexagonal grid
    g <- hexagrid(c(2,2))
# plot the grid in 3d space
    plot3d(g, col="blue")
# make a subset to select faces
   subG <- subset(g, c("F5", "F2"))
# plot the subset defined above
    plot3d(subG, type="f", col=c("orange"), add=TRUE, lwd=1)
```

---

locate *Basic lookup function of coordinates on an icosahedral grid*

Description

Basic lookup function of coordinates on an icosahedral grid

Usage

```
locate(gridObj, ...)

## S4 method for signature 'trigrid'
locate(gridObj, data, randomborder = FALSE,
  output = "ui")

## S4 method for signature 'hexagrid'
locate(gridObj, data, output = "ui",
  randomborder = FALSE, forceNA = FALSE)
```

Arguments

`gridObj`	a trigrid or hexagrid class object.
`...`	arguments passed to class specific methods.
`data`	Coordinates of individual points. Can be either a two-dimensional matrix of long-lat coordinates, a three-dimensional matrix of XYZ coordinates, or a set of points with class 'SpatialPoints'.
`randomborder`	Logical value. Defaults to FALSE. If TRUE, then the points falling on vertices and edges will be randomly assigned, otherwise they will be kept as NAs.

output	Character value either "ui" or "skeleton". ui returns the face names used in the user interface, while "skeleton" returns their indices used in back-end procedures.
forceNA	logical value, suppressing the recursive lookup of points falling on subface boundaries.

Value

The function returns the cell names where the input coordinates fall.

Examples

```
# create a grid
g <- trigrid(4)
# some random points
randomPoints<-rpsphere(4, output="polar")
# cells
locate(g, randomPoints)
```

`names,gridlayer-method`

The face names in a gridlayer class object

Description

Function to extract the registered face names to which the gridlayer renders information.

Usage

```
## S4 method for signature 'gridlayer'
names(x)
```

Arguments

x	A gridlayer class object.

Value

Character vector, the names of the faces.

newgraph *Add an igraph object to a predefined slot in a trigrid or hexagrid object*

Description

Add an igraph object to a predefined slot in a trigrid or hexagrid object

Add an igraph object to a predefined slot in a trigrid or hexagrid object

Usage

```
newgraph(gridObj, ...)

## S4 method for signature 'trigrid'
newgraph(gridObj, ...)
```

Arguments

gridObj	an icosahedral grid.
...	arguments passed to the gridgraph() function.

Examples

```
#create a grid
g<-trigrid(4, graph=FALSE)
g<-newgraph(g)

#create a grid
g<-trigrid(4, graph=FALSE)
g<-newgraph(g)
```

newsp *Add SpatialPolygons object to a predefined slot in a trigrid or hexagrid object*

Description

Add SpatialPolygons object to a predefined slot in a trigrid or hexagrid object

Usage

```
newsp(gridObj, res = 50)

## S4 method for signature 'trigrid'
newsp(gridObj, res = 50)
```

Arguments

gridObj	an icosahedral grid.
res	integer, the number of points inserted between two vertices, passed to SpPolygons().

Value

A trigrid or hexagrid class object.

Examples

```
a<-trigrid(4)
a<-newsp(a)
plot(a)
```

---

occupied *Faces occupied by the specified object*

Description

This function will return a facelayer class object showing which faces are occupied by the input object.

Usage

```
occupied(gridObj, data, ...)
```

Arguments

gridObj	a trigrid or hexagrid object.
data	the queried data.
...	arguments passed to the class specific methods

Details

This is a wrapper function on the OccupiedFaces methods that are specific to grid class and input data. The function creates a link between the facelayer and the relevant grid.

Value

returns a facelayer object

Examples

```
# create a grid
g <- trigrid(8, sp=TRUE)
# create random points
randPoints <- rpsphere(100,output="polar")
# the facelayer occupied by these points
randomLayer <- occupied(g, randPoints)
plot(randomLayer)
points(randPoints, col="blue", pch="+")
```

`orientation` *Extracting the grid orientation*

Description

Extracting the grid orientation

Extracting the grid orientation

Setting the orientation of a trigrid or hexagrid object

Setting the orientation of a trigrid or hexagrid object

Usage

```
orientation(gridObj, ...)

## S4 method for signature 'trigrid'
orientation(gridObj, display = "deg", ...)

orientation(gridObj) <- value

## S4 replacement method for signature 'trigrid'
orientation(gridObj) <- value
```

Arguments

`gridObj`	The grid object.
`...`	values passed on to the rotate() function.
`display`	The output units. In case it is set to "deg" the output will be in degrees, in case it is "rad" then radians.
`value`	the vector of rotation. Passed as the angle argument of rotate().

plot,facelayer,ANY-method

2d plotting of a facelayer class object This function will invoke the 2d plotting methods of a grid so data stored in a facelayer object can be displayed.

Description

The function passes arguments to the plot method of the SpatialPolygons class (€ref). In case a heatmap is plotted and the windows plotting device gets resized, some misalignments can happen. If you want to use a differently sized window, use windows() to set the height and width before running the function.

This function will invoke the `plot()` method of the SpatialPolygons class.

Usage

```
## S4 method for signature 'facelayer,ANY'
plot(x, projargs = NULL, col = "red",
  border = NA, alpha = "", frame = FALSE, ...)

## S4 method for signature 'trigrid,ANY'
plot(x, projargs = NULL, ...)
```

Arguments

`x`	The facelayer object to be plotted.
`projargs`	a projection string for the transformation fo coordinates. Accepts both a CRS class object and a character string that will be transformed to the CRS class.
`col`	Character vector. Colors passed to a grDevices::colorRampPalette() in case of the facelayer contains logical values, a single value is required (defaults to red).
`border`	Character value specifying the color of the borders of the cells.
`alpha`	Character value of two digits for the fill colors, in hexadecimal value between 0 and 255.
`frame`	Logical value, if TRUE the grid boundaries will be drawn with black.
`...`	arguments passed to the sp::plot() function.

plot3d,facelayer-method *3d plotting of a facelayer of an icosahedral grid or its subset*

Description

The function is built on the openGL renderer of the R package `rgl`. The default plotting window size is 800x800 pixels. In case you want to override this, please use the function with 'defaultPar3d=FALSE' after running 'rgl::par3d(windowRect=<>)'.

Usage

```
## S4 method for signature 'facelayer'
plot3d(x, type = "f", frame = TRUE, guides = TRUE,
  defaultPar3d = TRUE, ...)
```

Arguments

`x`	The `facelayer` object to be plotted.
`type`	A character value specifying the part of the grid to be plotted by the call of the function. `"l"` plots the grid lines (only when frame=FALSE). `"f"` draws the grid faces. `FALSE` does not plot the sphere.
`frame`	If set to TRUE the grid line structure will be plotted.
`guides`	If set to TRUE the guides3d() function will be run with col="green" and default settings.
`defaultPar3d`	Logical value, whether the default settings for par3d() are to be used (windowRect = c(50, 60, 800, 800), zoom=0.8).
`...`	Further graphical parameters passed to (see `plot3d`).

plot3d,trigrid-method *3d plotting of an icosahedral grid or its subset*

Description

This is a generic function used to plot either a `trigrid` or a `hexagrid` object or their `facelayer` in 3d space.

3d plotting of an icosahedral grid or its subset

Usage

```
## S4 method for signature 'trigrid'
plot3d(x, type = c("l"), sphere = NULL, add = FALSE,
  guides = TRUE, ...)

## S4 method for signature 'hexagrid'
plot3d(x, type = c("l"), sphere = NULL,
  color = "gray70", add = FALSE, guides = TRUE, ...)
```

Arguments

x	The trigrid, hexagrid or facelayer object to be plotted.
type	A character value specifying the part of the grid to be plotted by the call of the function. "v" plots the grid vertex points. "e" draws the grid edges. "f" draws the grid faces. "c" draws the face centers of the grid.
sphere	Defaults to NULL, adding a central white sphere to the plot. Assigning a numeric value will draw a new sphere with the given radius, FALSE does not plot the sphere.
add	Logical value indicating whether a new plot shall be drawn, or the currently plotted information should be added to the active rgl device.
guides	Logical value indicating whether the guidelines of the polar coordinate system shall be plotted.
...	Further graphical parameters passed to (see plot3d).
color	Only for the hexagrid plotting: character value/values, passed to the faces3d() function instead of col.

Details

The function is built on the openGL renderer of the R package rgl.

Value

The function does not return any value.

Examples

```
# create a hexagonal grid
    g <- hexagrid(c(2,2))
# plot the grid in 3d space
    plot3d(g, col="blue")
# make a subset to select faces
   subG <- subset(g, c("F5", "F2"))
# plot the subset defined above
    plot3d(subG, type="f", col=c("orange"), add=TRUE, lwd=1)
```

PolToCar *Conversion of spherical coordinates to 3d Cartesian coordinates*

Description

The function uses basic trigonometric relationships to transform longitude/latitude coordinates on a sphere to xyz Cartesian coordinates.

Usage

```
PolToCar(longLatMat, radius = authRadius, origin = c(0, 0, 0))
```

Arguments

longLatMat	A 2-column numerical matrix with the longitude/latitude data.
radius	Single numeric value, indicating the radius of the sphere. Defaults to the R2 radius of Earth (6371.007km).
origin	Numeric vector of length 3, the xyz coordinates of the sphere center.

Details

The authalic mean radius of Earth (6371.007 km) is used by this function as a default while the origin is c(0,0,0). The precision of these conversions is not exact (see example c(0,90) below), but should be considered acceptable when applied at a reasonable scale (e.g. for global analyses using data above 10e-6 meters of resolution).

Value

An xyz 3-column numeric matrix.

Examples

```
longLat <- rbind(
    c(0,0),
    #note the precision here!
    c(0, 90),
    c(-45,12)
)

xyz <- PolToCar(longLat)
```

pos *Position of face centers and vertices on a grid*

Description

This function will retrieve the position of a vertex or a face on a `hexagrid` or `trigrid` object.

Usage

```
pos(gridObj, names, output = "polar")
```

Arguments

`gridObj`	a `hexagrid` or `trigrid` object.
`names`	A (character) vector of the names that are to be looked up.
`output`	The coordinate system in which the names are to be shown: use `"polar"` for longitude-latitude and `"cartesian"` for XYZ output.

Details

Vertex and face names can be mixed in a single `names` argument.

Value

A numerical matrix.

Examples

```
g <- trigrid(c(4,4))
pos(g, c("F2", "P6", "dummyname"))
```

resample,Raster,trigrid-method *Resampling of data to a trigrid or a hexagrid object.*

Description

The function is used to resolve and resample data stored in RasterLayers and facelayers so they can be fitted to and can be plotted by using trigrid or hexagrid objects.

This function will calculate values for a different resolution icosahedral grid from a facelayer.

Usage

```
## S4 method for signature 'Raster,trigrid'
resample(x, y, method = "ngb", na.rm = TRUE)

## S4 method for signature 'facelayer,trigrid'
resample(x, y, method = NULL, res = 5)
```

Arguments

x	a facelayer class object.
y	a hexagrid or trigrid object.
method	Character string stating the name of the algorithm used for resampling.
na.rm	logical value. If a face contains a missing value, should its value be NA as well (FALSE) or calculate the mean anyway (TRUE).
res	Numeric value, indicating the precision of area estimation during the upscaling. In case the "ebaa" method is chosen, the variable indicate the number of breaking points on an edge.

Details

This method is necessary to utilize rasterized data in the icosa package. The only method currently implemented upscales the raster data and then resolve the values to the trigrid or hexagrid values, using averages. In the case of resampling rasterlayers, the method argument will be passed to the raster::resample() function.

The function applies different resampling algorithms. Currently there are only two implemented methods, one for upscaling and one for downscaling. The downscaling method "average" will tabluate all face centers from the high resolution grid that fall on a coarse resolution cell and average them. The upscaling method "ebaa" (edge breakpoint area approximation) will estimate the areas covered by the high resolution cells using the number of edge breakpoints.

Value

A named numeric vector.

Examples

```
g <- trigrid(c(4,4))
fl <- facelayer(g)
fl@values<-rnorm(length(fl))
h <- trigrid(4)
res <- resample(fl, h)
fl2<-facelayer(h)
fl2@values[] <- res
```

rotate,trigrid-method *Rotation method of trigrid and hexagrid objects*

Description

Rotation method of trigrid and hexagrid objects

Usage

```
## S4 method for signature 'trigrid'
rotate(x, angles = "random", pivot = NA)
```

Arguments

x	trigrid or hexagrid class object.
angles	The vector of rotation (length=3) in radians. If set to "random", the rotation will be random (default).
pivot	The pivot point of the rotation, numeric vector of xyz coordinates. Defaults to NA indicating that the rotation will be around the center of the grid.

Value

Another trigrid or hexagrid class object.

rpsphere *Random point generation on the surface of a sphere*

Description

This function will create a predefined number of points randomly distributed on the surface of a sphere with a given radius.

Usage

```
rpsphere(n = 1, output = "cartesian", radius = authRadius, origin = c(0,
  0, 0))
```

Arguments

n	The number of random points to be created.
output	The coordinate system of the new points. Can either be "cartesian" for XYZ coordiates or "polar" for spherical, longitude-latitudes coordinates.
radius	The radius of the sphere
origin	The center of the sphere (XYZ coordinates).

Details

The function uses a three dimension normal distribution to generate points, which are then projected to the surface of the sphere.

Value

A 3-column (XYZ) or a 2-column (long-lat) numeric matrix.

Examples

```
randomPoints <- rpsphere(20000)
points3d(randomPoints)
```

SpLines *SpatialLines class object from an icosahedral grid*

Description

SpatialLines class object from an icosahedral grid

The function will create an approporiate 2d render of the grid in the SpatialLines object format defined in the package 'sp'.

Usage

```
SpLines(gridObj, ...)

## S4 method for signature 'trigrid'
SpLines(gridObj, dateLine = "break", res = 15)
```

Arguments

gridObj	Either a trigrid or a hexagrid class object.
...	specific details of the new SpatialLines object.
dateLine	Specifies that NAs should be introduced at the dateline to break the boundaries of the faces. Can be switched off by setting it to FALSE.
res	Integer value, specifies the number of points (resolution) to be inserted between two vertices.

Value

an object of class SpatialLines.

SpPolygons *Spatial polygons from an icosahedral grid*

Description

The function will create a SpatialPolygons class 2d representation of the icosahedral grid.

The function will create a SpatialPolygons class 2d representation of trigrid class object.

The function will create a SpatialPolygons class 2d representation of the hexagrid class object

Usage

```
SpPolygons(gridObj, res)

## S4 method for signature 'trigrid'
SpPolygons(gridObj, res = 50)

## S4 method for signature 'hexagrid'
SpPolygons(gridObj, res = 50)
```

Arguments

gridObj	an icosahedral grid.
res	Integer value, the number of points inserted between two vertices.

subset,trigrid-method *Subsetting an icosahedral grid*

Description

This is a generic function used to access data from either a triangular or hexagonal grid using the names of the faces, integers or logical vectors.

Shorthand for the subset function.

The function extracts parts of the gridlayer depending on different criteria.

Shorthand to the subset() function.

Shorthand to the subset() function.

Usage

```
## S4 method for signature 'trigrid'
subset(x, i)

## S4 method for signature 'hexagrid'
subset(x, i)

## S4 method for signature 'trigrid,ANY,ANY'
x[i]

## S4 method for signature 'gridlayer'
subset(x, subsetVector)

## S4 method for signature 'gridlayer,ANY,missing'
x[i]

## S4 method for signature 'gridlayer,Extent,missing'
x[i]
```

Arguments

x	The gridlayer object to be subsetted.
i	A subscript vector, specifying the names of the face that are used for subsetting.
subsetVector	Vector object indicating the faces to be subsetted.

Details

The function returns subsets of the grid pertaining to the specified faces that can be used for additional operations (e.g. plotting). The subscript vector can be either a logical, character or numeric one. The character vector should contain the names of faces, the logical subscript should have the same length as the number of faces in the order in which the faces are present in the `faces` slot. The numeric vector can either refer to indices to the rownames of faces in the faces slot, or to surfaces bounded by longitude/latitude data. In the latter case, the the vector should contain an element with a names of at least one of the "lomax", "lamax", "lomin" or "lamin" strings (lo for longitude, la: latitude, min: minimum, max: maximum). In case a subset around the dateline is needed a larger longitude to a smaller longitude value is needed (e.g. between 150° to -150°).

The following methods are incorporated into the function. If the subsetVector argument is a vector of integers, they will be interpreted as indices. If the numeric subsetVector contains either the lamin, lamax, lomin or lomax names, the subsetting will be done using the latitude-longitude coordinates outlined by these 4 values. Logical subsetting and subsetting by face names are also possible.

Value

Subset of the input grid. The class of the original object is retained, the `@skeleton` slot contains all previous information.

Examples

```
    #create a triangular grid
        g <- trigrid(c(2,2))
    #make a subset pertaining to the faces
        subG1 <- subset(g, c("F1", "F33"))

   #additional way of subsetting
subG2 <- g[1:15] # selects faces F1 through F15
    logicalSub<-sample(c(TRUE,FALSE), nrow(g@faces), replace=TRUE)
subG3 <- g[logicalSub]
    #plot the subset in 3d space
        plot3d(subG3)
# previously mentioned case around the dateline
gDateLine<-g[c(lomax=-150, lomin=150)]
plot3d(gDateLine)
```

---

surfacearea *Areas of grid cell surfaces*

Description

This function will return the areas of all cells in the specified grid object.

Usage

```
surfacearea(gridObj)

## S4 method for signature 'trigrid'
surfacearea(gridObj)

## S4 method for signature 'hexagrid'
surfacearea(gridObj)
```

Arguments

gridObj	A trigrid or hexagrid object. in the metric that was fed to the function in the coordinates or the radius. "deg" will output the the distance in degrees, "rad" will do so in radians.

Value

A named numeric vector.

Examples

```
g <- trigrid(3)
surfaces <- surfacearea(g)
surfaces
```

surfacecentroid *Surface centroid point of a spherical point cloud*

Description

This function the projected place of the centroid from a pointset on the sphere

Usage

```
surfacecentroid(data, output = "polar", center = c(0, 0, 0),
  radius = authRadius, inner = 20)
```

Arguments

data	Numeric matrix, XYZ or longitude-latitude coordinates of the set of points.
output	Character value, the coordinate system of the output points. Can either be "polar" for longitude-latitude or "cartesian" for XYZ data.
center	Numeric vector, The center of the sphere in XYZ coordinates (default is 0,0,0).
radius	Numeric value, the radius of the circle in case the input points have only polar coordinates. Unused when XYZ coordinates are entered. Defaults to the authalic radius of Earth ca. 6371.007km.
inner	Single positive integer, the number of points inserted between every two points of the spherical centroid. Heavilly impacts the performance.

Details

The function implements great circle calculations to infer on the place of the centroid, which makes it resource demanding. This is necessary to avoid a particual error that frequently occurrs with other methods for centroid calculation, namely that the place of the centroid is right, but on the opposite hemisphere.

Value

Either an XYZ or a long-lat numeric vector.

Examples

```
# generate some random points
allData <- rpsphere(1000)
# select only a subset
points<-allData[allData[,1]>3000,]
# the spherical centroid
sc <- surfacecentroid(points)
sc

#3d plot
plot3d(points)
```

```
points3d(sc[1], sc[2], sc[3], col="red", size=5)
```

`translate` *Translating the grid object in 3d Cartesian space*

Description

The function translates the coordinates of a grid object with the specified 3d vector.

Usage

```
translate(gridObj, vec)

## S4 method for signature 'trigrid,numeric'
translate(gridObj, vec)

## S4 method for signature 'hexagrid,numeric'
translate(gridObj, vec)
```

Arguments

`gridObj`	A `trigrid` or `hexagrid` class object.
`vec`	A numeric vector of length 3. This is the translation vector.

Value

The same grid structure as the input, but with translated coordinates.

Examples

```
 # create a grid and plot it
g <- trigrid(3)
lines3d(g)
# translate the grid to (15000,15000,15000)
 g2 <- translate(g, c(15000,15000,15000))
 lines3d(g2)
```

trigrid *A triangular icosahedral grid*

Description

trigrid creates a triangular grid based on the tessellation of an icosahedron.

Usage

```
## S4 method for signature 'trigrid'
initialize(.Object, tessellation = 1, sp = FALSE,
  graph = TRUE, radius = authRadius, center = origin)
```

Arguments

.Object	non-argument pointing to self.
tessellation	An integer vector with the tessellation values. Each number describes the number of new edges replacing one original edge. Multiple series of tessellations are possible this way. The total tessellation is the product of the tessellation vector. Higher values result in more uniform cell sizes, but the larger number of tessellation series increases the speed of lookup functions.
sp	A logical value indicating whether the 'SpatialPolygons' class representation of the grid should be added to the object when the grid is calculated. If set to TRUE the SpPolygons() function will be run with with the resolution parameter set to 25. The resulting object will be stored in slot @sp. As the calculation of this object can substantially increase the grid creation time, by default this argument has a value of FALSE. The 'SpatialPolygons' class representation can be added on demand by running the function newsp.
graph	A logical value indicating whether the 'igraph' class representation of the grid should be added to the object when the grid is calculated. This argument defaults to TRUE because this option has only minor performance load on the grid constructor function. For familiarization with the object structure, however, setting this parameter to FALSE might help, as invoking str() on the 'igraph' class slot of the class might flood the console.
radius	The radius of the grid. Defaults to the authalic radius of Earth.
center	The origin of the grid in the reference Cartesian coordinate system. Defaults to (0,0,0).

Details

The grid structure functions as a frame for data graining, plotting and spatial calculations. Data can be stored in layers that are linked to the grid object. In the current version only the facelayer class is implemented, which allows the user to render data to the cells of the grid, which are usually referred to as faces. The grid 'user interface' is made up of four primary tables: the @vertices table for the coordinates of the vertices, the faceCenters for the coordinates of the centers of faces, the faces and the edges tables that contain which vertices form which faces and edges respectively.

In these tables, the faces and vertices are sorted to form spirals that go from the north pole in a counter-clockwise direction. In case grid subsetting is performed these tables get truncated.

At finer resolutions, the large number of spatial elements render all calculations resource demanding and slow, therefore the hierarchical structure created during the tessellation procedure is retained for efficient implementation. These data are stored in a list in the slot `@skeleton` and are 0-indexed integer tables for Rccp-based functions. `$v` stores vertex, `$f` the edge, and `$e` contains the edge data for plotting and calculations. In these tables the original hierarchy based orderings of the units are retained, during subsetting, additional vectors are used to indicate deactivation of these units. Any sort of meddling with the @skeleton object will lead to unexpected behavior.

Value

A triangular grid object, with class `trigrid`.

Slots

`vertices` Matrix of the vertex XYZ coordinates.

`faces` Matrix of the verticies forming the faces.

`edges` Matrix of the vertices forming the edges.

`tessellation` Contains the tessellation vector.

`orientation` Contains the grid orientation in xyz 3d space, values in radian relative to the (0,1,0) direction.

`center` is the xyz coordinates of the grids origin/center.

`div` vector contains the number of faces that a single face of the previous tessellation level is decomposed to.

`faceCenters` contains the xyz coordinates of the centers of the faces on the surface of the sphere.

`belts` Vector of integers indicating the belt the face belongs to.

`edgeLength` the length of an average edge in km and degrees.

`graph` an 'igraph' class graph object.

`length` integer vector of length=3. The number of vertices, edges and faces in this order.

`proj4string` a CRS class object indicating the model in the PROJ.4 system

`r` the radius of the grid

`sp` The SpatialPolygons representation of the grid. If missing, it can be created with newsp().

`skeleton` data tables with sequential indexing for the C functions.

Examples

```
# single tessellation value
g <- trigrid(c(8))
g
# series of tessellations
g1 <- trigrid(c(2,3,4))
g1
```

trishape *Shape distortions of the triangular faces and subfaces*

Description

This function will return a value that is proportional to the irregularity of a triangonal face or subface.

Usage

```
trishape(gridObj)

## S4 method for signature 'trigrid'
trishape(gridObj)

## S4 method for signature 'hexagrid'
trishape(gridObj)
```

Arguments

gridObj	A trigrid or hexagrid object.

Details

The value is exactly 1 for an equilateral triangle, and becomes 0 as one of the edges approach 0.

Value

A named numeric vector.

Examples

```
g <- trigrid(3)
shape <- trishape(g)
trishape
```

values,gridlayer-method *Extract values from a gridlayer*

Description

The function will get the values slot of a gridlayer object

Shorthand function to replace all values of a gridlayer object.

Usage

```
## S4 method for signature 'gridlayer'
values(x)

## S4 replacement method for signature 'gridlayer'
values(x) <- value
```

Arguments

x	a gridlayer derived object.
value	replacement values.

`vertices` *The vertices of an icosahedral grid object*

Description

Shorthand function to return the vertices slot of an icosahedral grid or a grid linked to a facelayer.

Usage

```
vertices(x, output)

## S4 method for signature 'trigrid,character'
vertices(x, output = "polar")

## S4 method for signature 'facelayer,character'
vertices(x, output = "polar")
```

Arguments

x	The grid or facelayer object.
output	the coordinate system of the output.

`vicinity` *The neighbouring faces of faces in an icosahedral grid*

Description

This function will return neighbouring faces of the input faces.

Usage

```
vicinity(gridObj, faces, ...)

## S4 method for signature 'trigrid,character'
vicinity(gridObj, faces, order = 1,
  output = "vector", self = TRUE, ...)
```

Arguments

gridObj	A trigrid or hexagrid class object.
faces	A character vector specifying names of faces.
...	arguments passed to the igraph::ego() function
order	Passed to the igraph::ego() function, an integer value specifying the size of the neighborhood around a face.
output	Characater value, the type of the output. The default "vector" will give back the names of the faces that adjacent to the faces specified, including themselves. "list" will return a list.
self	logical value indicating whether the input faces should be in the output. For the "list" output option, the input face names will be omitted only from those character vectors that contain face names that are related to the face in question.

Value

A character vector or a list of character vectors.

Examples

```
g <- trigrid(3)
ne <- vicinity(g, c("F4", "F10"))
ne
```

[<-,gridlayer,ANY,ANY-method

Replacement of elements in a gridlayer object.

Description

Function to replace specific elements in a gridlayer object

Usage

```
## S4 replacement method for signature 'gridlayer,ANY,ANY'
x[i] <- value
```

Arguments

x	the gridlayer.
i	the subsetting vector, as in subset().
value	the replacement values.

Details

All these methods are implementing direct replacement in the values slot of a layer, depending on criteria used for subsetting.

Index