Package ‘icosa’

July 22, 2025

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

Version 0.11.1

Collate zzz.R data.R utils-conversion.R utils-spherical.R utils-vectors.R grid-build.R grid-lookup.R grid-move.R grid-subset.R grid-attributes.R grid-graphs.R grid-sp-lines.R grid-sp-polygons.R grid-sf-polygons.R grid-resample.R data-gridlayer-basic.R data-gridlayer-attributes.R data-gridlayer-groupgen.R data-gridlayer-subset.R data-facelayer-basic.R data-facelayer-graphs.R data-facelayer-resample.R plot-legend.R plot-2d-grid.R plot-2d-data.R plot-rgl-util.R plot-rgl-grid.R plot-rgl-facelayer.R plot-rgl-sp3d.R

Description Implementation of icosahedral grids in three dimensions. The spherical-triangular tessellation can be set to create grids with custom resolutions. Both the primary triangular and their inverted penta-hexagonal grids can be calculated. Additional functions are provided that allow plotting of the grids and associated data, the interaction of the grids with other raster and vector objects, and treating the grids as a graphs.

Depends R (>= 3.5.0)

Date 2024-08-16

License GPL-3

Encoding UTF-8

LazyData true

Suggests knitr, rmarkdown, terra, rgl

URL https://icosa-grid.github.io/R-icosa/

VignetteBuilder knitr

RoxygenNote 7.3.0

LinkingTo Rcpp

Imports Rcpp, sp, igraph, methods, stats, sf

NeedsCompilation yes

BugReports https://github.com/icosa-grid/R-icosa/issues

Maintainer Adam T. Kocsis <adam.t.kocsis@gmail.com>

Author Adam T. Kocsis [cre, aut] (ORCID:
<https://orcid.org/0000-0002-9028-665X>),
Deutsche Forschungsgemeinschaft [fnd],
FAU GeoZentrum Nordbayern [fnd]

Repository CRAN

Date/Publication 2024-08-16 17:40:06 UTC

Contents

`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`) 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 `c(0,0,0)`).
`radius`	(`numeric`) 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 sets 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 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 c(0,0,0)).
output	(character) 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) 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 point coordinates along an arc*

Description

This function calculates points along an arc between two points 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	(integer) The number of points inserted between p1 and p2. Has to be positive.
origin	(numeric) vector, The center of the circle in XYZ coordinates (default is c(0,0,0)).
onlyNew	(logical) Should p1 and p2 be omitted from the result?
output	(character) The coordinate system of the output points. Can either be "polar" for longitude-latitude or "cartesian" for XYZ data.
radius	(numeric) Single 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"))
```

`arcs` *Function to plot a set of great circle arcs between points*

Description

Low level plotting of great circle arcs with lines

Usage

```
arcs(x, ...)

## S4 method for signature 'matrix'
arcs(x, breaks = 100, breakAtDateline = TRUE, plot = TRUE, ...)
```

Arguments

`x`	A matrix of longitude and latitude points (WGS 84 longlat)
`...`	Arguments passed to lines (par)
`breaks`	the number of points inserted between every points to draw great circle arcs.
`breakAtDateline`	Logical to indicate whether the lines are to be broken at the dateline.
`plot`	Logical value whether the plotting should be done at all (in case returned values are needed).

Value

Invisible return of a matrix of coordinates. If `breakAtDateline = TRUE`, then `NA` missing values will be inserted between coordinates where the lines cross the dateline.

Examples

```
# generate random points
set.seed(0)
example <- rpsphere(10, output="polar")

# plotting
plot(NULL, NULL, xlim=c(-180, 180), ylim=c(-90,90))
points(example)
text(label=1:nrow(example), example, pos=2)
arcs(example, col="red", breaks=200)
```

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(x, ...)

## S4 method for signature 'matrix'
CarToPol(x, norad = FALSE, origin = c(0, 0, 0))

## S4 method for signature 'numeric'
CarToPol(x, norad = FALSE, origin = c(0, 0, 0))

## S4 method for signature 'data.frame'
CarToPol(x, norad = FALSE, origin = c(0, 0, 0))
```

Arguments

`x`	(`matrix`, `data.frame`, `numeric`) A 3 column data matrix with XYZ coordinates in Cartesian space.
`...`	Arguments passed to class-specific methods.
`norad`	(`logical`). Toggles whether the rho coordinate (distance from origin) should be omitted from the output.
`origin`	(`numeric`) Vector with length `3`, the XYZ coordinates of the sphere center.

Value

A 3-column or 2-column `numeric`, `matrix` or `data.frame` 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)
```

`cellocator` *Locate grid faces based on their positions on a map*

Description

The function returns which grid faces contain the points clicked in a plot.

Usage

```
cellocator(gridObj, n, output = "faces", ...)
```

Arguments

`gridObj`	(`trigrid` or `hexagrid`) The grid object.
`n`	(`integer`) The number of points to be looked up.
`output`	(`character`) Type of output: `"faces"` returns only the face names of the points, `"full"` returns the coordinates as well.
`...`	Arguments passed to the `locator` function.

Value

A vector of `character` values, each corresponding to a face identifier.

`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 facelayer.

Usage

```
centers(x, ...)

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

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

Arguments

`x`	(`trigrid`, `hexagrid` or `facelayer`). The grid or linked data layer object.
`...`	Arguments passed to the class specific methods.
`output`	(`character`) The coordinate system of the output. Either `"polar"` or `"cartesian"`.

Value

The coordinates of the face centers as a `numeric` matrix.

Examples

```
a <- trigrid()
centers(a)
```

---

chullsphere *Spherical convex hull.*

Description

This function calculates a possible implementation of the spherical convex hull.

Usage

```
chullsphere(
  data,
  center = c(0, 0, 0),
  radius = authRadius,
  param = 200,
  strict = TRUE
)
```

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).
`radius`	(`numeric`) Single value, indicating the radius of the sphere. Defaults to the R2 radius of Earth (6371.007km).
`param`	(`numeric`) Single positive integer, indicates the number of divisions in the centroid projection method. The higher the number, the closer the replacement points are to the centroid.
`strict`	(`logical`) Strictly convex output is required.

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 `chull`.

Value

The indices of the data points forming the convex hull as a (`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, ...)

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

Arguments

`gridObj`	(`trigrid` or `{hexagrid}`) A grid object.
`...`	Arguments passed to the class specific methods.
`output`	(`character`) 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.

Value

A named `numeric` vector.

Examples

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

`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)
```

Arguments

`x`	(`trigrid`, `hexagrid` or `facelayer`) The grid or linked data object.

Value

The edges of the grid, as a `character` matrix.

facelayer-class *A facelayer linked to a trigrid or hexagrid object*

Description

The grids themselves are scaffolds for the assigned data. The data are stored in containers which are linked to the grids.

Arguments

gridObj	(hexagrid or trigrid) The linked grid object.
value	(logical,numeric or character) The facelayer will be initialized with these values/this value

Value

A facelayer class object.

Examples

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

faces *The face names of a trigrid or hexagrid object*

Description

Shorthand function to get the face names 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)
```

Arguments

x	(trigrid, hexagrid or facelayer) The grid or facelayer object.

Value

The names of the faces of the grid as a `character` vector.

Examples

```
ball <- hexagrid(2)
faces(ball)
```

`faces3d` *Methods of 3D face plotting.*

Description

This function is used to plot the faces of either a `trigrid`, `hexagrid` or `facelayer` object in 3D space.

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 = "heat", breaks = NULL, inclusive = TRUE, legend = TRUE, ...)
```

Arguments

`x`	The `trigrid`, `hexagrid` or `facelayer` object to be plotted.
`...`	Further graphical parameters passed to (see `plot3d`) and the `heatMapLegend` function.
`col`	(`character`) Graphical parameter indicating the colours of the faces. A single value is accepted for `logical` values. Multiple colors will be passed to `colorRampPalette`, to create palettes for heat maps in case of `numeric` values. The default plotting method in this case is the reversed `heat.colors`. In case of categorical data, random colors will be chosen.
`breaks`	(`numeric`) Vector stating the breakpoints between the plotted levels. The argument is passed to the `cut` function.
`inclusive`	(`logical`): If there are values beyond the limits of breaks, should these be represented in the plot (`TRUE`) or left out completely `FALSE`?
`legend`	(`logical`) Should the heatmap legend be plotted?

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
# faces3d(g)
h <- hexagrid(8)
b <- facelayer(h)
values(b)<- rnorm(length(b))
```

---

gridensity — *Icosahedral grid-based density estimation*

Description

Spatial density estimation algorithm based on rotation of icosahedral grids.

Usage

```
gridensity(x, y, out, trials = 100, FUN = mean)
```

Arguments

`x`	Matrix of longitude, latitude data, `sf` class, or `SpatialPoints` Point cloud.
`y`	`trigrid` or `hexagrid` An icosahedral grid.
`out`	`trigrid`, `hexagrid` or `SpatRaster`output structure.
`trials`	`numeric` value, the number of iterations.
`FUN`	`function` The function to be applied on the iteration results.

Details

Any points set can be binned to an icosahedral grid (i.e. number of incidences can be counted), which will be dependent on the exact positions of grid cells. Rotating the grid in 3d space will result in a different distribution of counts. This distribution can be resampled to a standard orientation structure. The size of the icosahedral grid cells act as a bandwidth parameter.

The implemented algorithm 1) takes a point cloud (`x`)) and an icosahedral grid `y` 2) randomly rotates the icosahedral grid, 3) looks up the points falling on grid cells, 4) resamples the grid to a constant orientation object (either `trigrid`, `hexagrid` or `SpatRaster`). Steps 2-4 are repeated `trial` times, and then `FUN` is applied to every vector of values that have same spatial position.

Value

Either named numeric vector, or a SpatRaster object. If FUN is set to NULL, the output will be either a matrix or SpatRaster.

Examples

```
# example to be run if terra is present
if(requireNamespace("terra", quietly=TRUE)){

 # randomly generated points
 x <- rpsphere(100, output="polar")

 # bandwidth grid
 y <- hexagrid(deg=13)

 # output structure
 out <- terra::rast(res=5)

 # the function
 o <- gridensity(x, y, out, trials=7)

 # visualize results
 terra::plot(o)
 points(x, pch=3)
}
```

---

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

Description

The function can be applied to both grids and to 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	(trigrid, hexagrid or facelayer) The icosahedral grid or facelayer.
...	Arguments passed to the class specific methods.
directed	logical Defaults to FALSE, creating an undirected graph. If TRUE, then the graph will be directed.
distances	logical Defaults to FALSE. If TRUE, then 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.

---

gridlabs *Labels of grid vertices, faces and edges.*

Description

This function will show where the grid elements are located.

Usage

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

Arguments

gridObj	(trigrid, hexagrid) An icosahedral grid.
type	(character) The type of element to be plotted: either "f" (faces), "v" (vertices) or "e" (edges).
crs	(character or crs) A coordinate system for the transformation of coordinates.
...	Arguments passed to the text function.

Value

The function has no return value.

Examples

```
gr <- hexagrid(sp=TRUE)
plot(gr)
gridlabs(gr)
```

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	(trigrid, hexagrid) An icosahedral grid.
...	Additional arguments passed to text3d function of the rgl package.
type	(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
# lines3d(g, guides=FALSE)
# labels
# gridlabs3d(g)
```

guides3d — *Guides for 3d spherical plotting.*

Description

This function plots 3d guidelines for navigation on the surface of the sphere, includings 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) Draws the -90(lat. deg.) +90 (lat. deg.) axis. The plotted radius will be axis times the authalic radius, ca. 6371km.
polgrid	(numeric) with the length of 2, where the first argument specifies the size of the longitudinal and the second the latitudinal divisions (degrees). Setting this argument to NULL will turn this feature off.
textPG	(logical) Flag indicating whether the coordinate values should be added to the 3d render.
res	(numeric) 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 segments3d, lines3d and 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,
  vals,
  varName,
  tick.text = NULL,
  tick.cex = 1.5,
  barWidth = 3,
  barHeight = 50,
  tickLength = 1,
  xLeft = 88,
  yBot = 25,
  add = FALSE,
  bounds = c(FALSE, FALSE),
  ...
)
```

Arguments

cols	(character) Vector, containnig the ordered colors that are used for the heatmap.
vals	(numeric) If tick.text is missing, the lowest value in the heatmap
varName	(character) The label of the variable name plotted to the heatmap.
tick.text	(numeric) The values on the heatmap legend. If missing, will be calculated with minVal and maxVal.
tick.cex	(numeric) Letter size of the values on the legend.
barWidth	(numeric) The width (percent) of the bar featuring the colors of the heatmap.
barHeight	(numeric) The height (percent)of the bar featuring the colors of the heatmap.
tickLength	(numeric) The length (percent) of the ticks at the bars.
xLeft	(numeric) The x coordinate of the lower left hand corner of the bar.
yBot	(numeric) The y coordinate of the lower left hand corner of the bar.
add	(logical) Indicates wheter a new plot should be drawn or not. Defaults to FALSE.
bounds	(logical) Vector (length 2) indicating whether open intervals should be indicated for the legend.
...	Arguments passed to the plot function.

Details

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

Value

The function does not return any value.

---

hexagrid-class *Construct a penta-hexagonal icosahedral grid*

Description

The `hexagrid` function constrcucts a hexa-pentagonal grid based on the inversion of a tessellated icosahedron.

Arguments

`tessellation`	(`numeric`) 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.
`deg`	(`numeric`) The target edge length of the grid in degrees. If provided, the function will select the appropriate tessellation vector from the `hexguide`-table, which is closest to the target. Note that these are unlikely to be the exact matches.
`sp`	(`logical`) Flag 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`	(`logical`) Flag 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`	(`numeric`) The radius of the grid. Defaults to the authalic radius of Earth.
`center`	(`numeric`) The origin of the grid in the reference Cartesian coordinate system. Defaults to `c(0,0,0)`.
`verbose`	(`logical`) Should messages be printed during grid creation?

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), sf=TRUE)
# based on approximate size (4 degrees edge length)
g1 <- hexagrid(deg=4)
```

hexguide *Tessellation guide to hexagrid objects*

Description

The table includes basic properties of hexagrids described with specific tessellation parameters

Usage

```
hexguide
```

Format

A data.frame with 120 observations and 18 variables:

- total The total tessellation of the grid, the number of points inserted between icosahedron vertices along an edge.
- level1 Level 1 tessellation.
- level2 Level 2 tessellation - second value of the tessellation vector.
- level3 Level 3 tessellation - third value of the tessellation vector.
- level4 Level 4 tessellation - four value of the tessellation vector.
- faces The number of faces in the grid.
- vertices The number of vertices in the grid.
- meanEdgeLength_deg Mean edge length in degrees.
- sdEdgeLength_deg Standard deviation of edge length in degrees.
- meanEdgeLength_km Mean edge length in kilometers.
- sdEdgeLength_km Standard devation of edge length in kilometers.
- meanArea_km2 Mean face area in square-kilometers.
- sdArea_km2 Standard deviation of face area in square-kilometers.
- time Time to compute grid with an Intel Xeon E-1650 prcessor.
- time_sp Time to compute grid with an Intel Xeon E-1650 prcessor, with the 'sp' member.
- size The size of the grid in bytes.
- size_sp The size of the grid object in bytes, with the 'sp' member.
- timeLocate_5000 Time to locate 5000 points with an Intel Xeon E-1650 processor in seconds.

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 and vector spatial data.

Details

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

Author(s)

Adam T. Kocsis (adam.t.kocsis@gmail.com)

See Also

Useful links:

- https://icosa-grid.github.io/R-icosa/
- Report bugs at https://github.com/icosa-grid/R-icosa/issues

Examples

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

length,trigrid-method — *The number of faces in a* `trigrid` *or* `hexagrid` *class object.*

Description

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

Usage

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

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

Arguments

x	(trigrid, hexagrid or facelayer) The object.

Value

An integer value.

---

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

Description

This function will invoke the method of the SpatialPolygons class. This function will invoke the lines method of the sf or the SpatialPolygons class.

Usage

```
## S4 method for signature 'trigrid'
lines(x, crs = NULL, col = 1, lwd = 1, lty = 1, ...)
```

Arguments

x	(trigrid, hexagrid) Object.
crs	(character or crs) A coordinate system for the transformation of coordinates.
col	Line colors - as in par
lwd	Line thickness - as in par
lty	Line type - as in par
...	Arguments passed to the sp.lines method.

Value

The function has no return value.

lines3d *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, a `facelayer`, or `Spatial` objects in 3d space. The method is also implemented for the object classes defined by the package 'sp'.

Usage

```
lines3d

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

## S4 method for signature 'Line'
lines3d(x, radius = authRadius, ...)

## S4 method for signature 'Lines'
lines3d(x, radius = authRadius, ...)

## S4 method for signature 'SpatialLines'
lines3d(x, radius = authRadius, ...)

## S4 method for signature 'SpatialLinesDataFrame'
lines3d(x, radius = authRadius, ...)

## S4 method for signature 'Polygon'
lines3d(x, radius = authRadius, ...)

## S4 method for signature 'Polygons'
lines3d(x, radius = authRadius, ...)

## S4 method for signature 'SpatialPolygons'
lines3d(x, radius = authRadius, ...)

## S4 method for signature 'SpatialPolygonsDataFrame'
lines3d(x, radius = authRadius, ...)
```

Arguments

`x`	(`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`).

radius (numeric) Used for plotting objects that inherit from Spatial*. The radius of the sphere the sp objects are plotted with. Default to the authalic (R2) radius of Earth.

Format

An object of class nonstandardGenericFunction of length 1.

Details

The function is built on the openGL renderer of the R package rgl, which needs to be installed for the function to run. Although the function is works without attaching rgl, note that if you want to attach both icosa and rgl,the rgl package has to be loaded ifrst otherwise the function will not be usable.

Value

The function does not return any value.

Examples

```
# create a hexagonal grid
  g <- hexagrid(c(2,2))
# plot the grid in 3d space
#  lines3d(g, col="blue")
```

---

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

Description

Basic lookup function of coordinates on an icosahedral grid

Usage

```
locate(x, y, ...)

## S4 method for signature 'trigrid,matrix'
locate(x, y, randomborder = FALSE, output = "ui")

## S4 method for signature 'trigrid,numeric'
locate(x, y, ...)

## S4 method for signature 'trigrid,data.frame'
locate(x, y, ...)

## S4 method for signature 'trigrid,sf'
locate(x, y, ...)
```

```
## S4 method for signature 'trigrid,SpatialPoints'
locate(x, y, ...)

## S4 method for signature 'trigrid,SpatialPointsDataFrame'
locate(x, y, ...)

## S4 method for signature 'hexagrid,matrix'
locate(x, y, output = "ui", randomborder = FALSE, forceNA = FALSE)
```

Arguments

x	(trigrid, hexagrid) Icosahedral grid object.
y	(matrix, data.frame, numeric or Spatial) 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 or SpatialPointsDataFrame.
...	Arguments passed to class specific methods.
randomborder	(logical) 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) 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) Suppressing the recursive lookup of points falling on subface boundaries.

Value

The function returns the cell names (as character) 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 facelayer class object

Description

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

Usage

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

Arguments

`x`	(`facelayer`) Object.

Value

A vector of `character` values, the names of the faces.

---

newgraph — *Add an igraph object to a predefined slot in an icosahedral grid*

Description

Add an igraph object to a predefined slot in an icosahedral grid

Usage

```
newgraph(gridObj, ...)

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

Arguments

`gridObj`	(`trigrid`, `hexagrid`) An icosahedral grid.
`...`	Arguments passed to the `gridgraph` function.

Value

A new (`trigrid` or `hexagrid`) object with the recalculated graph.

Examples

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

newsf — *Add a* `sf` *object to a predefined slot in a* `trigrid` *or* `hexagrid` *object*

Description

Add a `sf` object to a predefined slot in a `trigrid` or `hexagrid` object

Usage

```
newsf(x, res = NULL)

## S4 method for signature 'trigrid'
newsf(x, res = NULL)
```

Arguments

`x`	(`trigrid` or `hexagrid`) An icosahedral grid.
`res`	(`numeric`) The number of points inserted between two vertices, passed to `SpPolygons`.

Value

A `trigrid` or `hexagrid` object with the new `@sf` slot.

Examples

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

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

Description

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

Usage

```
newsp(gridObj, res = NULL)

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

Arguments

`gridObj`	(`trigrid` or `hexagrid`) An icosahedral grid.
`res`	(`numeric`) The number of points inserted between two vertices, passed to `SpPolygons`.

Value

A `trigrid` or `hexagrid` object with the new `@sp` slot.

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, out = "logical", ...)
```

Arguments

`gridObj`	(`trigrid` or `hexagrid`) An icoshedral grid.
`data`	(`matrix`, `data.frame`, `sf` or `Spatial`) The queried data.
`out`	(`character`) What shall be the output class? Can be either `facelayer` or `logical` (default.)
`...`	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.

Value

The function Returns a `facelayer`-class object.

Examples

```
# create a grid
g <- trigrid(8, sf=TRUE)

# create random points
randPoints <- rpsphere(100,output="polar")

# the faces occupied by these points
occ <- occupied(g, randPoints)

 # plot using sf slot independently
plot(g@sf[occ,"geometry"])
points(randPoints, col="red", pch="+")
```

`orientation` *Extracting and setting the grid orientation*

Description

Extracting and setting the grid orientation

Usage

```
orientation(x, ...)

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

orientation(x) <- value

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

Arguments

`x`	(`trigrid` or `hexagrid`): Input grid.
`...`	Values passed on to the `rotate` function.
`display`	(`character`) The output unit. In case it is set to `"deg"` the output will be in degrees, in case it is `"rad"`, then radians.
`value`	(`numeric`) The vector of rotation. Passed as the `angles` argument of `rotate`.

Value

In case the function returns does, it returns the orientation angles of the grid (as `numeric`).

plot *Plot method for the trigrid, hexagrid or facelayer classes*

Description

This function will invoke the plot method of the sf or the SpatialPolygons class.

The function passes arguments to the plot method of the SpatialPolygons class. In case a heatmap is plotted and the plotting device gets resized, some misalignments can happen. If you want to use a differently sized window, use x11 to set the height and width before running the function.

The function matches data referred to the grid and plots it with sf's plotting methods.

Usage

```
plot

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

## S4 method for signature 'facelayer,ANY'
plot(
  x,
  crs = NULL,
  col = "heat",
  border = NA,
  alpha = NULL,
  frame = FALSE,
  legend = TRUE,
  breaks = NULL,
  inclusive = TRUE,
  discrete = FALSE,
  ...
)

## S4 method for signature 'trigrid,numeric'
plot(x, y, crs = NULL, main = "", ...)

## S4 method for signature 'trigrid,array'
plot(x, y, crs = NULL, main = "", ...)

## S4 method for signature 'trigrid,table'
plot(x, y, crs = NULL, main = "", ...)

## S4 method for signature 'trigrid,character'
plot(x, y, crs = NULL, main = "", ...)

## S4 method for signature 'trigrid,logical'
plot(x, y, crs = NULL, main = "", ...)
```

Arguments

x	(trigrid, hexagrid or facelayer) The object to be plotted.
crs	(character or crs) A coordinate system for the transformation of coordinates.
...	Arguments passed to the plot function.
col	(character) Colors passed to a colorRamp in case of the facelayer contains logical values, a single value is required (defaults to "red").
border	(character) Specifies the color of the borders of the cells.
alpha	(character) Two digits for the fill colors, in hexadecimal value between 0 and 255.
frame	(logical) If TRUE the grid boundaries will be drawn with black.
legend	(logical): Should the legend be plotted?
breaks	(numeric) The number of breakpoints between the plotted levels. The argument is passed to the cut function.
inclusive	(logical): If there are values beyond the limits of breaks, should these be represented in the plot (TRUE) or left out completely FALSE?
discrete	(logical): Do the heatmaps symbolize a discrete or a continuous variable? This argument only affects the legend of the heatmap.
y	Data or part of the grid to be plotted. If it is an unnamed character vector, then it is expected to be a set of faces, which will be treated as a subscript that indicates the faces to be plotted. If it is a logical vector, then it is expeced to be subscript, indicating a similar operation. If it is a named logical or character vector, table with names, or single-dimensional named array then the names are expected to refer to faces of the grid x. The default sf-based plotting method will apply to the data type.
main	The main title of the plot

Format

An object of class standardGeneric of length 1.

Value

The function has no return value.

Examples

```
# A simple grid, with sf-representation
gr <- hexagrid(4, sf=TRUE)
dat <- 1:nrow(gr@faces)
names(dat) <- paste0("F", dat)
plot(x=gr, y=dat)
```

plot3d *3d plotting of an icosahedral grid, its subset or a data layer*

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 `par3d(windowRect=`<>`)`.

Usage

```
plot3d(x,...)

## S3 method for class 'trigrid'
plot3d(x, type = c("l"), sphere = NULL, add = FALSE, guides = TRUE, ...)

## S3 method for class 'hexagrid'
plot3d(
  x,
  type = c("l"),
  sphere = NULL,
  color = "gray70",
  add = FALSE,
  guides = TRUE,
  ...
)

## S3 method for class 'facelayer'
plot3d(x, type = "f", frame = TRUE, guides = TRUE, defaultPar3d = TRUE, ...)
```

Arguments

`x`	(`trigrid`, `hexagrid` or `facelayer`) Object to be plotted.
`type`	(`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`	(`numeric`) 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`	(`character`) Only for the hexagrid plotting: value/values passed to the `faces3d` function instead of `col`.

`frame`	(`logical`) If set to TRUE the grid line structure will be plotted.
`defaultPar3d`	(`logical`) Flag indicating whether the default settings for `par3d` are to be used (`windowRect = c(50, 60, 800, 800), zoom=0.8`).

Format

An object of class `function` of length 1.

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 polar 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(x, ...)

## S4 method for signature 'matrix'
PolToCar(x, radius = authRadius, origin = c(0, 0, 0))

## S4 method for signature 'numeric'
PolToCar(x, radius = authRadius, origin = c(0, 0, 0))

## S4 method for signature 'data.frame'
PolToCar(x, radius = authRadius, origin = c(0, 0, 0), long = NULL, lat = NULL)
```

Arguments

x	(matrix, numeric, data.frame) A 2-column numeric matrix with the longitude/latitude data.
...	Arguments passed to class-specific methods.
radius	(numeric) The radius of the sphere. Defaults to the R2 radius of Earth (6371.007km).
origin	(numeric) Vector with length 3, the XYZ coordinates of the sphere center.
long	(character) If x is a data.frame, then the column used as longitudes.
lat	(character) If x is a data.frame, then the column used as latitudes.

Details

The authalic mean radius of Earth (6371.007 km) is used by this function as a default. 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, data.frame or numeric, depending on the class of x.

Examples

```
longLat <- rbind(
  c(0,0),
  #note the precision here!
  c(0, 90),
  c(-45,12)
)
# matrix-method
xyz <- PolToCar(longLat)
# numeric-method
xyz2 <- PolToCar(longLat[1,])
# data.frame method
xyz3 <- PolToCar(as.data.frame(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) Icosahedral grid object.
names	(character) Vector of the names that are to be looked up.
output	(character) 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 numeric matrix.

Examples

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

---

resample — *Resampling of data involving a trigrid or a hexagrid object.*

Description

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

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.

Usage

```
resample

## S4 method for signature 'SpatRaster,trigrid'
resample(x, y, method = "near", na.rm = TRUE)

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

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

Arguments

- `x` — (`SpatRaster`, `facelayer`) Object to resample.
- `y` — (`hexagrid` or `trigrid`) Object describing the target structure.
- `method` — (`character`) The name of the algorithm used for resampling.
- `na.rm` — (`logical`) 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 (`facelayer`-method). In case the `"ebaa"` method is chosen, the variable indicate the number of breaking points on an edge.

Format

An object of class `standardGeneric` of length 1.

Details

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

Value

A named `numeric` vector.

Examples

```
# create a grid
g <- trigrid(c(4,4))
# create a data layer
fl <- facelayer(g)
fl@values<-rnorm(length(fl))
# target structure
h <- trigrid(4)
# resampling
res <- resample(fl, h)
fl2<-facelayer(h)
fl2@values[] <- res
```

---

rotate — *Rotation of `trigrid` and `hexagrid` objects*

Description

Rotation of `trigrid` and `hexagrid` objects

Usage

```
rotate

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

Arguments

x (trigrid or hexagrid) Input grid.

angles (numeric): The vector of rotation in radians (three values in each dimension). If set to "random", the rotation will be random (default).

pivot (numeric): The pivot point of the rotation, vector of xyz coordinates. Defaults to NA indicating that the rotation will be around the center of the grid.

Format

An object of class standardGeneric of length 1.

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 (numeric) The number of random points to be created.

output (character) The coordinate system of the new points. Can either be "cartesian" for XYZ coordiates or "polar" for spherical, longitude-latitudes coordinates.

radius (numeric) The radius of the sphere

origin (numeric) 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(2000, output="polar")
# observe latitudinal pattern
 plot(randomPoints, xlim=c(-180, 180), ylim=c(-90, 90))
```

SpLines *Create a* `SpatialLines` *class object from an icosahedral grid*

Description

Create a `SpatialLines` class object from an icosahedral grid

Usage

```
SpLines(gridObj, ...)

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

Arguments

`gridObj`	(`trigrid` or `hexagrid`) Icosahedral grid object.
`...`	Specific details of the new `SpatialLines` object.
`dateLine`	(`logical`)Specifies whether `NA`s should be introduced at the dateline to break the boundaries of the faces. Can be switched off by setting it to `FALSE`.
`res`	(`numeric`) The number of points inserted between two vertices, or `NULL`, if this is to be set by the package. The default method increases resolution wiht lower tessellation values, and is higher for higher absolute latitudes.

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.

Usage

```
SpPolygons(gridObj, ...)

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

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

Arguments

`gridObj`	(`trigrid` or `hexagrid`) An icosahedral grid.
`...`	Arguments passed to class-specific methods.
`res`	(`numeric`) The number of points inserted between two vertices, or `NULL`, if this is to be set by the package. The default method increases resolution with lower tessellation values, and is higher for higher absolute latitudes.

Value

A `SpatialPolygons` class object.

Examples

```
a <- trigrid()
sp <- SpPolygons(a)
```

subset — *Subsetting an icosahedral grid or data layers organized with them*

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.

The function extracts subsets of the `gridlayer` depending on different criteria.

Usage

```
subset

## 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, i)
```

Arguments

x	(trigrid, hexagrid or facelayer) The object to be subsetted.
i	(logical, numeric or character) The subscript vector, specifying the faces that are used for subsetting. As in subset.

Format

An object of class standardGeneric of length 1.

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 i argument is a vector of integers, they will be interpreted as indices. If the numeric i 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	(trigrid or hexagrid) Object.

Value

A named numeric vector, in the metric that was given to the function in the coordinates or the radius. "deg" will output the the distance in degrees, "rad" will do so in radians.

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(x, ...)

## S4 method for signature 'matrix'
surfacecentroid(x, output = "polar", center = c(0, 0, 0), radius = authRadius)

## S4 method for signature 'data.frame'
surfacecentroid(x, ...)

## S4 method for signature 'SpatialPoints'
surfacecentroid(x, ...)
```

Arguments

`x`	(`matrix` or `data.frame`) Numeric data, XYZ or longitude-latitude coordinates of the set of points.
`...`	Arguments passed to the `matrix`-method.
`output`	(`character`) The coordinate system of the output points. Can either be `"polar"` for longitude-latitude or `"cartesian"` for XYZ data.
`center`	(`numeric`) The center of the sphere in XYZ coordinates (default is 0,0,0).
`radius`	(`numeric`) 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 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[,3]>1500,]
# transform to 2d
 points2 <- CarToPol(points, norad=TRUE)
# the spherical centroid
sc <- surfacecentroid(points2, output="polar")
sc

#3d plot
plot(points2, xlim=c(-180, 180), ylim=c(-90, 90))
points(sc[1], sc[2], col="red", cex=5, pch=3)
```

`translate` *Translating an icosahedral 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`	(`trigrid` or `hexagrid`) Icosahedral grid object.
`vec`	(`numeric`) A 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-class` *A triangular icosahedral grid*

Description

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

Arguments

`tessellation`	(`numeric`) 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.
`deg`	(`numeric`) The target edge length of the grid in degrees. If provided, the function will select the appropriate tessellation vector from the `triguide`-table, which is closest to the target. Note that these are unlikely to be the exact matches.
`sp`	(`logical`) Flag 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`	(`logical`) Flag 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`	(`numeric`) The radius of the grid. Defaults to the authalic radius of Earth.
`center`	(`numeric`) The origin of the grid in the reference Cartesian coordinate system. Defaults to (0,0,0).
`verbose`	(`logical`) Should messages be printed during grid creation?

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.

`crs` a CRS class object, by design this is the authalic sphere (ESRI:37008)

`r` the radius of the grid

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

`sf` The sf representation of the grid. If missing, it can be created with newsf().

`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
# based on approximate size (4 degrees edge length)
g2 <- trigrid(deg=4)
```

---

triguide *Tessellation guide to trigrid objects*

Description

The table includes basic properties of trigrids described with specific tessellation parameters

Usage

```
triguide
```

Format

A data.frame with 120 observations and 18 variables:

- `total` The total tessellation of the grid, the number of points inserted between icosahedron vertices along an edge.
- `level1` Level 1 tessellation.
- `level2` Level 2 tessellation - second value of the tessellation vector.
- `level3` Level 3 tessellation - third value of the tessellation vector.
- `level4` Level 4 tessellation - four value of the tessellation vector.
- `faces` The number of faces in the grid.
- `vertices` The number of vertices in the grid.
- `meanEdgeLength_deg` Mean edge length in degrees.
- `sdEdgeLength_deg` Standard deviation of edge length in degrees.
- `meanEdgeLength_km` Mean edge length in kilometers.
- `sdEdgeLength_km` Standard devation of edge length in kilometers.
- `meanArea_km2` Mean face area in square-kilometers.
- `sdArea_km2` Standard deviation of face area in square-kilometers.
- `time` Time to compute grid with an Intel Xeon E-1650 prcessor.
- `time_sp` Time to compute grid with an Intel Xeon E-1650 prcessor, with the 'sp' member.
- `size` The size of the grid in bytes.
- `size_sp` The size of the grid object in bytes, with the 'sp' member.
- `timeLocate_5000` Time to locate 5000 points with an Intel Xeon E-1650 processor in seconds.

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. The ratio of the lengths of the shortest and the longest edges.

Usage

```
trishape(gridObj)

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

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

Arguments

gridObj (`trigrid`, `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, one value for every face of the grid.

Examples

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

values *Extract and replace values from a gridlayer-derived object (e.g.* `link{facelayer}`*).*

Description

The function will get the `@values` slot of a `facelayer` object.

Usage

```
values(x,...)

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

values(x) <- value

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

Arguments

`x`	(`facelayer`) Object.
`value`	(`logical`, `character` or `numeric`) Replacement values.
`...`	Arguments passed to class-specific methods. (Not used.)

Format

An object of class `standardGeneric` of length 1.

An object of class `standardGeneric` of length 1.

`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, ...)

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

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

Arguments

`x`	(`trigrid`, `hexagrid` or `facelayer`) The icosahedral grid, or linked data object.
`...`	Additional arguments passed to class-specific methods.
`output`	(`character`) The coordinate system of output.

Examples

```
a <- trigrid(1)
vertices(a)
```

---

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,
  namedorder = FALSE,
  ...
)
```

Arguments

gridObj	(trigrid or hexagrid) Icosahedral grid object.
faces	(character) A vector specifying names of faces.
...	Arguments passed to the ego function.
order	(numeric) Passed to the ego function, an integer value specifying the size of the neighborhood around a face.
output	(character) 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) Flag 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.
namedorder	(logical) Should the orders of the neighbouring cells be reported (TRUE) or just the names of the cells (default, FALSE).

Value

A character vector or a list of character vectors.

Examples

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

[,gridlayer,ANY,missing-method *Extraction from a gridlayer using indices*

Description

Shorthand to the `subset` function.

Function to replace specific elements in a gridlayer object

Usage

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

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

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

Arguments

x	(`facelayer`) The object to be subsetted.
i	(`logical`, `numeric` or `SpatExtent`) The subscript vector, or extent, specifying the faces that are used for subsetting. 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.

Value

The extraction methods return `facelayer`-class objects.

Index