Package: globe (via r-universe)

August 23, 2024

Type Package

Title Plot 2D and 3D Views of the Earth, Including Major Coastline

Version 1.2-0

Date 2016-05-12

Maintainer Adrian Baddeley <adrian.baddeley@curtin.edu.au>

Depends graphics, stats

Description Basic functions for plotting 2D and 3D views of a sphere, by default the Earth with its major coastline, and additional lines and points.

License GPL (>= 2.0)

LazyData yes

ByteCompile yes

Repository https://baddstats.r-universe.dev

RemoteUrl https://github.com/baddstats/globe

RemoteRef HEAD

RemoteSha f2af47e917376daa13d4ba478de79bbde4a8dcf4

Contents

`globe-package` *Simple 2D and 3D plots of Spheres including Earth*

Description

A simple package to plot 2D and 3D views of a sphere. Options include drawing Earth with its major coastline, plotting the lines of longitude and latitude, and plotting points and curves at any locations on the sphere.

Details

This package provides very basic tools for plotting lines and points on a sphere. It does not require installation of any other libraries.

The major functions are `globeearth` to plot a 3D view of a sphere (by default the Earth with its major coastline) and `flatearth` to plot a 2D projection of the sphere.

Additional tools include `globelines` and `globepoints` to draw lines and points on the sphere.

Author(s)

Adrian Baddeley and Tom Lawrence

Examples

```
globeearth(eye=place("nedlands"), top=place("northpole"))
flatearth(projection="atlas")
flatearth(projection="cylindrical")
```

`cross` *Cross Product*

Description

Computes the cross-product of two vectors in 3D.

Usage

```
cross(a, b)
```

Arguments

a, b	Numeric vectors of length 3.

Details

Computes the cross product of the two vectors.

Value

A vector of length 3 representing the cross product. If the input vectors have length greater than 3, only the first 3 elements will be used in this calculation.

Author(s)

Adrian Baddeley and Tom Lawrence

See Also

`dot`

Examples

```
a <- c(1,0,0)
b <- c(0,1,0)
cross(a, b)
```

`dot` *Dot Product*

Description

Computes the dot product of two vectors.

Usage

```
dot(a, b)
```

Arguments

`a, b`	Two vectors of equal length.

Details

Vectors can be of any length provided they are equal.

Value

A vector of length 1.

Author(s)

Adrian Baddeley and Tom Lawrence

See Also

`cross`

Examples

```
dot(c(2,1),c(-3,4))
```

`earth` *Major Coastline of Earth*

Description

Coordinates of the coastline of continents and major islands on Earth.

Usage

```
data("earth")
```

Format

A list with two components: `coords` is a two-column matrix containing (longitude, latitude) coordinates of the coastline, in degrees; `runlen` is an integer vector giving the number of vertices for each connected polygon in the coastline.

Source

The **maps** package.

Examples

```
data(earth)
globeearth(earth$coords, earth$runlen)
```

`ensure` *Checking and converting coordinates*

Description

This is used as a checking mechanism for other functions to ensure data is given in the appropriate format, and if it is not, converting to the appropriate format where possible.

Usage

```
ensure3d(x, single = FALSE)
ensurelonlat(x)
```

Arguments

`x`	A vector, matrix or dataframe to be checked
`single`	Logical indicating whether a single point is expected.

Details

ensure3d checks to ensure that the given data are one or more sets of 3D Cartesian coordinates, and converts them to numeric vectors if they are not already. If longitudes and latitudes are given as input into *ensure3d*, they will be converted to 3D Cartesian coordinates, on the unit sphere. *ensurelonlat* checks to ensure that the given data are one or more pairs of longitude and latitude and converts the data to a pair of lists if they are not already. Vectors and single rows/columns of matrices can be used as input for both functions; multiple rows/colums of matrices and pairs of lists can also be used as input for *ensurelonlat*.

Value

ensure3d A numeric vector or matrix *ensurelonlat*

`$lon`	List of Longitudes
`$lat`	List of Latitudes

Author(s)

Adrian Baddeley, Ege Rubak and Tom Lawrence

See Also

`spatialpos`

Examples

```
ensure3d(c(1,4,2))
ensure3d(matrix(1:3, ncol = 3))
ensure3d(data.frame(x = 1:2, y = 3:4, z = 5:6), single = FALSE)
ensure3d(data.frame(lon = c(0,180), lat = c(-45,45)), single = FALSE)
ensurelonlat(c(145, -90))
```

`flatearth` *Plot the Earth as a 2D Projection*

Description

Plots the Earth as specified 2D projection, with a map of the major coastline.

Usage

```
flatearth(projection = c("atlas", "cylindrical"), gdata, runlen, asp = NULL,
          ..., do.plot=TRUE)
```

Arguments

`projection`	The type of 2D projection to be performed.
`gdata`	Two-column matrix of latitude, longitude coordinates of coastline vertices. Defaults to `earth$coords`. If `NULL`, no coastline is plotted.
`runlen`	Integer vector giving the number of vertices in each connected polygon in the coastline. Defaults to `earth$runlen`.
`asp`	Optional. Aspect ratio of the longitude and latitude scales.
`...`	Optional arguments passed to `segments` to control the plotting of the coastline.
`do.plot`	Logical value indicating whether to actually perform the plotting, or just to return the calculated coordinates.

Details

In the `atlas` projection, the continents are plotted in longitude, latitude coordinates without any correction.

In the `cylindrical` projection, the latitude is transformed so that equal areas on the sphere are transformed onto equal areas on the plot.

Value

(Invisibly) a 4-column matrix containing the projected (x, y) coordinates of the segments of the coastline.

Author(s)

Adrian Baddeley and Tom Lawrence

See Also

`globeearth`.

Use `flatpoints` to plot points on the image.

Examples

```
flatearth("atlas")
flatearth("cylindrical")
```

`flatpoints` *Plot Points on a 2D Projection of the Earth*

Description

Plots points on a 2D projection of the Earth created using `flatearth`, taking into account the projection used.

Usage

```
flatpoints(loc, projection = c("atlas", "cylindrical"), ..., do.plot)
```

Arguments

`loc`	A data frame containing latitudes and longitudes of points to be plotted.
`projection`	The projection that was used in creating the 2D image.
`...`	Other arguments usually used when plotting points.
`do.plot`	Logical value indicating whether to actually perform the plotting, or just to return the calculated coordinates.

Details

If the value of projection used in this function is not the one used to create the image, then the points will not be plotted.

Value

(Invisibly) a `list(x,y)` giving the plotted positions of the points.

Author(s)

Adrian Baddeley and Tom Lawrence

See Also

`flatearth`

Examples

```
flatearth("atlas")
flatpoints(place("nedlands"))
```

globedrawlat *Drawing lines of latitude and longitude*

Description

These functions respectively draw lines of latitude and longitude on an image of the Earth create using globeearth.

Usage

```
globedrawlat(lat, eye, top, ...)
globedrawlong(lon, eye, top, ...)
```

Arguments

lat	A list showing lines of latitude to be drawn
lon	A list showing lines of longitude to be drawn
eye	Viewpoint. Should not be specified under normal circumstances since it is set by a previous call to globeearth (details documented there).
top	Top of plot (commonly the North Pole). Should not be specified under normal circumstances since it is set by a previous call to globeearth (details documented there).
...	Additional arguments passed to segments to control the plotting of lines.

Value

Lines of latitude/longitude are plotted on the existing image.

Author(s)

Adrian Baddeley and Tom Lawrence

See Also

globeearth.

Use globepoints to add points to this plot, globelines to add lines to this plot, and globearrows to add arrows to this plot.

Examples

```
globeearth()
globeearth(eye=place("madrid"))
globedrawlat(lat=seq(-90, 90, 15))
globedrawlong(lon=seq(-180,180,30))
```

`globeearth` *Plot Earth as 3D Globe*

Description

Plots the Earth as a 3D sphere, seen from a specified viewpoint, with a map of the major coastline (by default).

Usage

```
globeearth(gdata, runlen, eye, top, ..., do.plot=TRUE)
```

Arguments

`gdata`	Two-column matrix of latitude, longitude coordinates of coastline vertices. Defaults to `earth$coords`. If `NULL`, no coastline is plotted.
`runlen`	Integer vector giving the number of vertices in each connected polygon in the coastline. Defaults to `earth$runlen`.
`eye`	Viewpoint. A vector of length 3 (or a `list(lon,lat)`) determining a position in space. If unspecified the last value from a call to `globeearth` will be used. If this is the first call to `globeearth` the default value of zero degrees latitude and longitude will be used.
`top`	Vector of length 3 (or a `list(lon,lat)`) determining a position in space. The plot will be rotated so that this position appears to be directly above the centre of the earth. If unspecified the last value from a call to `globeearth` will be used. If this is the first call to `globeearth` the default value of 90 degrees latitude and zero degrees longitude (i.e. the North Pole) will be used.
`...`	Arguments passed to `segments` controlling the plotting of the coastline.
`do.plot`	Logical value indicating whether to actually perform the plotting, or just to return the calculated coordinates.

Details

The globe is drawn as it would be seen by a viewer at position `eye`, with the location `top` at the top of the plot. Only those parts of the coastline that are visible from `eye` (on the side of the globe facing `eye`) will be plotted.

Value

(Invisibly) a 4-column matrix containing the projected (x, y) coordinates of the segments of the coastline.

Author(s)

Adrian Baddeley and Tom Lawrence

See Also

flatearth.

Use globepoints to add points to this plot, globelines to add lines to this plot, globearrows to add arrows to this plot, and globedrawlat or globedrawlong to draw latitude and longitude curves.

Examples

```
globeearth()
globeearth(eye=place("madrid"))
```

---

globeplot — *Plot points, lines and arrows on a globe*

Description

Plot points, lines and arrows on an plot of the Earth created by globeearth

Usage

```
globepoints(loc, eye, top, ..., do.plot=TRUE)
globelines(loc, eye, top, ..., do.plot=TRUE)
globearrows(loc, eye, top, len=0.3, ..., do.plot=TRUE)
```

Arguments

`loc`	A matrix or list of points to plot (`globepoints`), draw lines between (`globelines`) or draw arrows at (`globearrows`), in order.
`eye`	Viewpoint. Should not be specified under normal circumstances since it is set by a previous call to globeearth (details documented there).
`top`	Top of plot (commonly the North Pole). Should not be specified under normal circumstances since it is set by a previous call to globeearth (details documented there).
`len`	Length of arrows to be plotted (`globearrows` only)
`...`	Optional additional arguments passed to points (`globepoints`) or segments (`globelines`, `globearrows`).
`do.plot`	Logical value indicating whether to actually perform the plotting, or just to return the calculated coordinates.

Details

globeearth needs to be invoked first before any of these functions can be used. The values of eye and top in globeearth should be identical to the values used in these functions to get meaningful results.

globepoints plots points at the specified locations on the globe.

globelines draws line between the specified locations in the order that they are written, but not between the first and last points e.g. if we list three different points A, B, C in *loc*, then lines will be drawn from A to B, and from B to C, but not C to A.

globearrows plots lines at the specified locations, that extend away from the centre of the globe and are of the specified length.

Value

The return value is invisible. For globepoints, the return value is a list(x,y) giving the plotted coordinates of the points. For globelines and globearrows, the return value is a 4-column matrix giving the plotted coordinates of the lines or arrows as plotted by segments.

See Also

globeearth.

Use globedrawlat or globedrawlong to draw latitude and longitude curves.

Examples

```
ex1 <- matrix(nrow=3, ncol=2)
ex1[1,1] <- -80
ex1[1,2] <- 45
ex1[2,1] <- -60
ex1[2,2] <- 45
ex1[3,1] <- -50
ex1[3,2] <- 50
globeearth(eye=place("newyorkcity"))
globepoints(loc=ex1)
globelines(loc=ex1)
globearrows(loc=ex1)
```

---

orthogproj *Orthogonal Projection*

Description

Project points from the unit sphere onto a plane orthogonal to the viewing direction.

Usage

```
orthogproj(eye, top, loc)
```

Arguments

`eye`	Viewpoint. A vector of length 3 (or a `list(lon,lat)`) determining a position in space.
`top`	Top point. A location which will be projected onto the y axis. Vector of length 3 (or a `list(lon,lat)`) determining a position in space.
`loc`	Vector of length 3, or matrix with 3 columns, or `list(lon,lat)`. The points on the sphere to be projected.

Details

This function is used to obtain orthogonal projections of points on the sphere, for use in plotting 3D views of the sphere.

Each point of `loc` is expressed in an orthonormal coordinate system determined by the arguments `eye` and `top`. The coordinate system is such that the third (z) axis passes through the eye of the viewer and the centre of the sphere.

Value

A vector (or matrix) giving the transformed coordinates of each point, with the first two coordinates giving the orthogonal projection.

Author(s)

Adrian Baddeley and Tom Lawrence

See Also

`globeearth`

Examples

```
orthogproj(place("newyorkcity"), place("northpole"), c(1,0,0))
```

`place` *Locations of Some Places on Earth*

Description

This function gives the locations of selected places on Earth.

Usage

```
place(placename)
```

Arguments

`placename`	Character string giving the name of a place.

Details

`aarhus` and `aalborg` are cities in Denmark; `madrid` is the capital of Spain; `perth` is the capital of Western Australia; `curtin` is the location of Curtin University, Western Australia; `titanic` is the location of the wreck of the *Titanic*; `casey` and `mawson` are bases in the Australian Antarctic Territory; `newyorkcity`, `pyongyang`, `everest`, `kilimanjaro`, `northpole` and `southpole` are self explanatory.

Value

A `list(lon,lat)` giving the longitude and latitude in degrees.

Source

General knowledge.

Examples

```
place("aarhus")
```

`runifsphere` *Random Points on a Sphere*

Description

These functions generate random points on a sphere using different rules.

Usage

```
runifsphere(n)
runifsphere.wrong(n)
```

Arguments

`n`	The number of points to be simulated.

Details

`runifsphere` generates uniformly-distributed random points on the sphere.

`runifsphere.wrong` generates random points which are uniformly distributed in (longitude, latitude) coordinates. These are not uniformly distributed on the sphere.

Value

A data frame containing two columns of coordinates: the first column for longitude, the second column for latitude.

Author(s)

Adrian Baddeley and Tom Lawrence

Examples

```
runifsphere(10)
runifsphere.wrong(10)
```

---

spatialpos *Convert Geographical to Cartesian Coordinates*

Description

Converts latitudes and longitudes on the sphere into 3D Cartesian coordinates.

Usage

```
spatialpos(lon, lat)
```

Arguments

`lon`	A vector, matrix column, or list of longitudes
`lat`	A vector, matrix column, or list of longitudes

Details

The longitudes and latitudes can be input as a single entity (i.e. a 2 column matrix, or pair of lists) rather than as separate entities.

Value

A matrix with one row per set of Cartesian coordinates

Author(s)

Adrian Baddeley and Tom Lawrence

See Also

ensure3d ensures that the given data is a vector 3D coordinates, and where possible converts the data if they are not in this format. *ensurelonlat* ensures that the given data are a pair of lists, one list of for longitude, one for latitude, and where possible converts the data if they are not in this format.

Examples

```
spatialpos(30, 60)
spatialpos(place("nedlands"))
```

Index