

Package ‘rworldmap’

April 25, 2012

Type Package

Title For mapping global data : rworldmap

Version 0.1310

Date 2012-04-24

Author Andy South, with contributions from Joe Scutt-Phillips, Barry Rowlingson, Roger Bivand & Pru Foster

Maintainer Andy South <southandy@gmail.com>

Description Enables mapping of country level and gridded user datasets.

License GPL (>= 2)

Depends R (>= 2.10.0), sp, maptools, foreign, fields, spam, maps

Suggests RColorBrewer, classInt, cshapes, ncdf, raster

Repository CRAN

Date/Publication 2012-04-25 04:39:30

R topics documented:

rworldmap-package	2
addMapLegend	3
addMapLegendBoxes	6
aggregateHalfDegreeGridToCountries	8
barplotCountryData	9
country2Region	11
countryExData	13
countryRegions	16
getMap	17
gridCountriesNumeric	18
gridExData	19
identifyCountries	20

joinCountryData2Map	22
joinData2Map	23
mapBars	25
mapBubbles	27
mapByRegion	30
mapCountryData	32
mapDevice	34
mapGriddedData	36
mapHalfDegreeGridToCountries	38
mapPies	40
mapPolys	42
rwmGetClassBreaks	44
rwmGetColours	45
rwmNewMapPlot	46
rworldmapExamples	47
setMapExtents	47
wrld_coarse	48
wrld_simpl	49
wrld_simpl_lessIslands	50
wrld_simpl_Mollweide	52

Index	54
--------------	-----------

rworldmap-package *For mapping global data.*

Description

Enables mapping of country level and gridded user datasets by facilitating joining to world maps and visualisation options.

Details

Package: rworldmap
 Type: Package
 Version: 0.1310
 Date: 2012-04-24
 License: GPL (>= 2)

Country Level Data can be joined to a map using [joinCountryData2Map](#), then mapped using [mapCountryData](#). Gridded data can be mapped using [mapGriddedData](#).

More generic functions allow the user to provide their own polygon map using [joinData2Map](#) and [mapPolys](#).

Bubble, bar and pie charts can be added to maps using [mapBubbles](#), [mapBars](#) and [mapPies](#).

Try the new method [barplotCountryData](#) for producing a ranked bar plot of country data with

country names that can provide a useful companion to maps.

Options are provided for categorising data, colouring maps and symbols, and adding legends.

Type vignette('rworldmap') to access a short document showing a few examples of the main rworldmap functions to get you started.

Author(s)

Andy South

with contributions from Joe Scutt-Phillips, Barry Rowlingson, Roger Bivand and Pru Foster

Maintainer: <southandy@gmail.com>

References

Stable version : <http://cran.r-project.org/web/packages/rworldmap>

Development version : <http://code.google.com/p/rworld>

Examples

```
#mapping country level data, with no file specified it uses internal example data
mapCountryData()
#specifying region
mapCountryData(mapRegion="asia")
#mapping gridded data, with no file specified it uses internal example data
mapGriddedData()
#specifying region
mapGriddedData(mapRegion="africa")
#aggregating gridded data to country level
#with no file specified it uses internal example data
mapHalfDegreeGridToCountries()
```

addMapLegend

Add a legend to a map

Description

Creates a colour bar legend, showing the range of colours and the values the colours correspond to. Relies heavily on image.plot() from the package fields. For simple use, simply use addLegend=TRUE in a rworldmap map function. Or users can call addMapLegend seperately to fine tune the legend. The user should insure that data, catMethod,numCats and colourPalette match the values used in the plot. The legend is designed to be useful for the variety of classification methods that exist.

Usage

```

addMapLegend(
    colourVector=""
    ,cutVector=""

    ,legendLabels="limits"
    ,labelFontSize=1
    ,legendWidth=1.2
    ,legendShrink=0.9
    ,legendMar=3
    ,horizontal=TRUE
    ,legendArgs=NULL
    ,tcl=-.5
    ,mgp=c(3,1,0)
    ,sigFigs=4
    ,digits=3
    ,legendIntervals='page'

    ,plottedData=""
    ,catMethod="pretty"
    ,colourPalette="heat"
    #,missingCountryCol="white"
)

```

Arguments

cutVector	the categories or breaks used in the map
colourVector	colours used in the map
legendLabels	Controls the style of the labels on the legend. Choose "none" for no labels, "limits" for the two end values, and "all" to show all the break values if they fit.
labelFontSize	Controls font size of the labels. A multiplier, so use 2 to double the size, 0.5 to halve it, etc.
legendWidth	Controls the width of the colour bar.
legendShrink	Controls the length of the colour bar. 1 means full width of the plot.
legendMar	Moves the legend away from the side of the plot. Measured in character widths.
horizontal	If TRUE the legend is horizontal, if FALSE, vertical.
legendArgs	For producing titles and labels. A list of arguments to be passed to mtext.
tcl	Controls the length of the tick marks. Useful when labelFontSize is changed.
mgp	Numeric vector length 3. The second element controls the distance between labels and the axis. Useful when labelFontSize is changed.
sigFigs	The number of significant figures for legend labels.
digits	An argument to the formatting of the labels
legendIntervals	"page" or "data". Controls the division of the colour bar, "page" sets the intervals equal on the page, "data" sets them to be equal in the units of the data.

plottedData	Numeric vector. The data which was plotted
catMethod	Controls the classification scheme. Usual options of "pretty", "quantiles", "fixed-Width", "logFixedWidth" or a vector of breaks.
colourPalette	Controls the colour scheme. Usual options of "palette", "heat", "topo", ...

Details

The default legend is a horizontal colour bar, with labels only at the extremes.

Can use a parameter list returned from mapping functions, e.g. `mapCountryData()`. `mapCountryData(addLegend=TRUE)` produces same results as: `mapParams <- mapCountryData(addLegend=FALSE)`
`do.call(addMapLegend, mapParams)`

Using the following allows the modification of the legend : `mapParams <- mapCountryData(addLegend=FALSE)`
`do.call(addMapLegend, c(mapParams, legendLabels="all", legendWidth=0.5))`

Value

Adds a legend to a plot.

Note

Can have the unintentional effect of modifying graphical parameters, e.g. `mfcol` reverts to `mfrow`.

Author(s)

Andy South & Matthew Staines

See Also

`mapCountryData`, `mapGriddedData`, `image.plot`

Examples

```
#Set up the plot so the world map uses the full width.
mapDevice()
#join example data to a map
data("countryExData",envir=environment())
sPDF <- joinCountryData2Map(countryExData
  , joinCode = "ISO3"
  , nameJoinColumn = "ISO3V10"
  , projection = "none"
)
#map the data with no legend
mapParams <- mapCountryData( sPDF
  , nameColumnToPlot="BIODIVERSITY"
  , addLegend='FALSE'
)

#add a modified legend using the same initial parameters as mapCountryData
do.call( addMapLegend, c( mapParams
  , legendLabels="all"
```

```
, legendWidth=0.5
))
```

addMapLegendBoxes *Add a legend of coloured boxes to a map*

Description

Creates a colour box legend, showing the range of colours and the values the colours correspond to. This works well for categorical data with relatively few categories.

Usage

```
addMapLegendBoxes(
  cutVector=""
  , colourVector = ""
  , x='bottomleft'
  , horiz=FALSE
  , title="category"
  , cex=1
  , pt.cex=2
  , col="gray"
  , bg="white"
  , legendText=""

  , catMethod="categorical"

  , plottedData=""
  , colourPalette="heat"
  , sigFigs=2
  , missingCountryCol="white"

  , ...
)
```

Arguments

cutVector	the categories or breaks used in the map
colourVector	colours used in the map
x	positioning of legend e.g. 'bottomleft', 'topright'
horiz	if TRUE horizontal legend
title	title for Legend
cex	controls the font size, default is 1

pt.cex	controls size of colour boxes relative to cex, default is 2
col	colour for boundary of colour boxes, default is "gray"
bg	colour for legend background, default is "white", NA makes the legend background transparent
legendText	the text to put against each legend box, if left blank cutVector is used, needs to be a vector the same length as length cutVector
catMethod	the categorisation method used influences what text added to legend elements, for 'categorical' just the category names are used for other options limits are used
plottedData	not used yet but maybe in future
colourPalette	not used yet but maybe in future
sigFigs	not used yet but maybe in future
missingCountryCol	not used yet but maybe in future
...	to allow other params to be set in legend

Details

This creates a legend with separate boxes of colour rather than addMapLegend() which creates a colour bar. This method is used as the default for categorical data.

See the examples for how to use a parameter list returned from mapping functions.

Value

Adds a legend to a plot.

Author(s)

Andy South

See Also

addMapLegend, mapCountryData, mapGriddedData

Examples

```
#Set up the plot so the world map uses the full width.
mapDevice()
#map example categorical data with no legend
mapParams <- mapCountryData(nameColumnToPlot='REGION'
  , catMethod='categorical'
  , addLegend='FALSE'
  )

#add default legend using the same parameters as mapCountryData
do.call( addMapLegendBoxes, c( mapParams))

#adding a modified legend by specifying extra parameters
```

```
do.call( addMapLegendBoxes, c(mapParams,x='bottom',horiz=TRUE,title="Region"))

#user defined map colour scene
mapParams <- mapCountryData(nameColumnToPlot='REGION'
                             , catMethod='categorical'
                             , addLegend='FALSE'
                             , colourPalette=c('white','green','red','yellow','blue','black')
                             )
#changing legendText
mapParams$legendText <- c('antarctic','africa','oceania','americas','s.asia','eurasia')
do.call( addMapLegendBoxes, c(mapParams,x='bottom',title="Region",horiz=TRUE))

#or this way
#do.call( addMapLegendBoxes
#         , c(mapParams
#             ,list(legendText=c('antarctic','africa','oceania','americas','s.asia','eurasia')
#                 ,x='bottom',title="Region",horiz=TRUE)))
```

aggregateHalfDegreeGridToCountries

Aggregates global half degree gridded data to countries

Description

Aggregates global half degree gridded data to countries (options for sum, mean, min, max). Uses a very simple grid map defining a single country identity for each half degree cell. (other more sophisticated approaches dividing cells between multiple countries will be investigated in future). The country identity at each cell is specified in data(gridCountriesNumeric).

Usage

```
aggregateHalfDegreeGridToCountries(inFile = "", aggregateOption = "sum")
```

Arguments

`inFile` either a gridascii filename or an sp SpatialGridDataFrame object specifying a global half degree grid dataset

`aggregateOption` how to aggregate the data ('sum','mean','min','max')

Value

a dataframe with 2 columns : numeric country codes and the aggregated value for each country

Author(s)

andy south

See Also

[mapHalfDegreeGridToCountries](#)

Examples

```
data(gridExData,envir=environment(),package="rworldmap")
gridExData <- get("gridExData")
#aggregating the gridded data to countries
dF <- aggregateHalfDegreeGridToCountries(gridExData)
#joining the aggregated data to a country map
sPDF <- joinCountryData2Map(dF, nameJoinColumn='UN', joinCode='UN')
#plotting the map
mapCountryData(sPDF,nameColumnToPlot='sum_pa2000.asc')
```

barplotCountryData *Barplot country-level data.*

Description

Draw a barplot of country-level data, ranking the countries to allow easy comparison. This is useful for comparing with maps created by [mapCountryData](#) and accepts many of the same arguments for categorising and colouring.

Usage

```
barplotCountryData( dF
                    , nameColumnToPlot = ""
                    , nameCountryColumn = "NAME"
                    , numPanels = 4
                    , scaleSameInPanels = FALSE
                    , main=nameColumnToPlot
                    , numCats = 5
                    , catMethod="quantiles"
                    , colourPalette= "heat"
                    , addLegend=TRUE
                    , toPDF = FALSE
                    , outFile = ""
                    , decreasing = TRUE
                    , na.last = TRUE
                    , ...
                    )
```

Arguments

<code>df</code>	a dataframe containing at least one column with numeric data and one with country names or other labels
<code>nameColumnToPlot</code>	name of column containing the data you want to plot
<code>nameCountryColumn</code>	name of column containing country names (or other labels to be used in plot)
<code>numPanels</code>	the number of layout panels in the plot
<code>scaleSameInPanels</code>	whether to set the scale the same in each panel TRUE/FALSE, default=FALSE allowing more of the variability in the data to be viewed
<code>main</code>	title for the plot
<code>numCats</code>	number of categories to put the data in, may be modified if this number is incompatible with the <code>catMethod</code> chosen
<code>catMethod</code>	method for categorisation of data "pretty", "fixedWidth", "diverging", "logfixedWidth", "quantiles", "categorical", or a numeric vector defining breaks
<code>colourPalette</code>	a string describing the colour palette to use, choice of : <ol style="list-style-type: none"> 1. ="palette" for the current palette 2. a vector of valid colours, e.g. =c('red','white','blue') or output from RColourBrewer 3. = one of "heat","diverging","white2Black","black2White","topo","rainbow","terrain","negpos8","neq"
<code>addLegend</code>	NOT YET WORKING whether to add a legend or not, TRUE/FALSE
<code>toPDF</code>	whether to output the plot to a pdf rather than the screen, TRUE/FALSE
<code>outFile</code>	output filename if toPDF=TRUE
<code>decreasing</code>	logical. Should the sort order be increasing or decreasing?
<code>na.last</code>	for controlling the treatment of NAs. If TRUE, missing values in the data are put last; if FALSE, they are put first; if NA, they are removed.
<code>...</code>	other arguments to pass to <code>barplot</code>

Details

Finer control can be achieved by [addMapLegend](#).

Value

invisibly returns a list containing the data and main options used for the map, the list can be passed to [addMapLegend](#) or [addMapLegendBoxes](#) along with additional options to allow greater flexibility in legend creation.

Warning

will generate unhelpful errors in data categorisation if inappropriate options are chosen, e.g. with `catMethod:Quantiles` if `numCats` too high so that unique breaks cannot be defined.

Author(s)

andy south

See Also

classInt, RColorBrewer

Examples

```
#default uses popn data in the default map
barplotCountryData()

data("countryExData",envir=environment(),package="rworldmap")

barplotCountryData( countryExData
                    , nameColumnToPlot="BIODIVERSITY"
                    , nameCountryColumn = "Country"
                    )
```

country2Region

Produce regional data from country level data

Description

A function to aggregate country level data into regional data. For example finding the total population of Asia, Europe,etc, from country level populations. As well as sums, other functions can be used, like mean, median, min, max, etc. There are currently 8 choices of region and 4 choices of country code.

Usage

```
country2Region(regionType, inFile, nameDataColumn, joinCode,
               nameJoinColumn, FUN = 'mean', ...)
```

Arguments

regionType	Must be one of: "GEO3", "GEO3major", "IMAGE24", "GLOCAF", "Stern", "SRES", "SRESmajor" or "GBD"
inFile	a data frame
nameDataColumn	The name of the data column to aggregate
joinCode	The type of code to join with. Must be one of: "ISO2", "ISO3", "Numeric" or "FIPS"
nameJoinColumn	The name of a column of inFile. Contains joining codes.

FUN A function to apply to each region, e.g. 'mean'
 ... further arguments to be passed to FUN, e.g. na.rm=TRUE

Details

The user must specify 'nameJoinColumn' from their data which contains country codes, and joinCode which specifies the type of code. regionType specifies which regions to aggregate the data to. Using FUN='identity' will return the names of the countries within each region.

Value

If FUN returns a single value, country2Region returns a data frame, with value of FUN for each region.

If FUN returns more than one value, country2Region will return a list, with one element for each region.

See Also

For producing maps of regional data from aggregated country level data, see [mapByRegion](#)

Examples

```
data(countryExData)

#to report which countries make up regions
country2Region(regionType="Stern")

#Using country2Region to calculate mean Environmental Health index in Stern regions.
sternEnvHealth <- country2Region(inFile=countryExData
, nameDataColumn="ENVHEALTH"
, joinCode="ISO3"
, nameJoinColumn="ISO3V10"
, regionType="Stern"
, FUN='mean'
)

print(sternEnvHealth)

#A simple plot of this data.
#dotchart(sort(sternEnvHealth))
dotchart(sort(sternEnvHealth[,1]))

#use FUN='identity' to see which countries in your data belong to which region.
country2Region(inFile=countryExData
, nameDataColumn="Country"
, joinCode="ISO3"
, nameJoinColumn="ISO3V10"
, regionType="Stern"
, FUN='identity'
)
```

```
#Change FUN to length, to count the number of countries in each region.
country2Region(inFile=countryExData
, nameDataColumn="Country"
, joinCode="ISO3"
, nameJoinColumn="ISO3V10"
, regionType="Stern"
, FUN='length'
)
```

countryExData	<i>Example dataset for country level data (2008 Environmental Performance Index)</i>
---------------	--

Description

A dataframe containing example country level data for 149 countries. This is the 2008 Environmental Performance Index (EPI) downloaded from <http://epi.yale.edu/>. Used here with permission, further details on the data can be found there. The data are referenced by ISO 3 letter country codes and country names.

Usage

```
data(countryExData)
```

Format

A data frame with 149 observations on the following 80 variables.

ISO3V10 a character vector
Country a character vector
EPI_regions a character vector
GEO_subregion a character vector
Population2005 a numeric vector
GDP_capita.MRYA a numeric vector
landlock a numeric vector
landarea a numeric vector
density a numeric vector
EPI a numeric vector
ENVHEALTH a numeric vector
ECOSYSTEM a numeric vector
ENVHEALTH.1 a numeric vector
AIR_E a numeric vector

WATER_E a numeric vector
BIODIVERSITY a numeric vector
PRODUCTIVE_NATURAL_RESOURCES a numeric vector
CLIMATE a numeric vector
DALY_SC a numeric vector
WATER_H a numeric vector
AIR_H a numeric vector
AIR_E.1 a numeric vector
WATER_E.1 a numeric vector
BIODIVERSITY.1 a numeric vector
FOREST a numeric vector
FISH a numeric vector
AGRICULTURE a numeric vector
CLIMATE.1 a numeric vector
ACSAT_pt a numeric vector
WATSUP_pt a numeric vector
DALY_pt a numeric vector
INDOOR_pt a numeric vector
PM10_pt a numeric vector
OZONE_H_pt a numeric vector
SO2_pt a numeric vector
OZONE_E_pt a numeric vector
WATQI_pt a numeric vector
WATSTR_pt a numeric vector
WATQI_GEMS.station.data a numeric vector
FORGRO_pt a numeric vector
CRI_pt a numeric vector
EFFCON_pt a numeric vector
AZE_pt a numeric vector
MPAEEZ_pt a numeric vector
EEZTD_pt a numeric vector
MTI_pt a numeric vector
IRRSTR_pt a numeric vector
AGINT_pt a numeric vector
AGSUB_pt a numeric vector
BURNED_pt a numeric vector
PEST_pt a numeric vector

GHGCAP_pt a numeric vector
CO2IND_pt a numeric vector
CO2KWH_pt a numeric vector
ACSAT a numeric vector
WATSUP a numeric vector
DALY a numeric vector
INDOOR a numeric vector
PM10 a numeric vector
OZONE_H a numeric vector
SO2 a numeric vector
OZONE_E a numeric vector
WATQI a numeric vector
WATQI_GEMS.station.data.1 a numeric vector
WATSTR a numeric vector
FORGRO a numeric vector
CRI a numeric vector
EFFCON a numeric vector
AZE a numeric vector
MPAEEZ a numeric vector
EEZTD a numeric vector
MTI a numeric vector
IRRSTR a numeric vector
AGINT a numeric vector
AGSUB a numeric vector
BURNED a numeric vector
PEST a numeric vector
GHGCAP a numeric vector
CO2IND a numeric vector
CO2KWH a numeric vector

Details

2008 Environmental Performance Index (EPI) data downloaded from : <http://epi.yale.edu/Downloads>

Disclaimers This 2008 Environmental Performance Index (EPI) tracks national environmental results on a quantitative basis, measuring proximity to an established set of policy targets using the best data available. Data constraints and limitations in methodology make this a work in progress. Further refinements will be undertaken over the next few years. Comments, suggestions, feedback, and referrals to better data sources are welcome at: <http://epi.yale.edu> or epi@yale.edu.

Source

<http://epi.yale.edu/Downloads>

References

Esty, Daniel C., M.A. Levy, C.H. Kim, A. de Sherbinin, T. Srebotnjak, and V. Mara. 2008. 2008 Environmental Performance Index. New Haven: Yale Center for Environmental Law and Policy.

Examples

```
data(countryExData,envir=environment(),package="rworldmap")
str(countryExData)
```

countryRegions	<i>Regional Classification Table</i>
----------------	--------------------------------------

Description

A number of regional classifications exist, e.g. SRES, Stern, etc. This table can be used to find which grouping a country belongs to, given its country code. A variety of different codes or groupings can be used.

Usage

```
data(countryRegions)
```

Format

A data frame with 235 observations on the following 13 variables.

Name Country Name

ISO2 ISO 2 letter country code

FIPS FIPS country code

Numeric UN numeric country code

ISO3 ISO 3 letter country code

GEO3major Global Environment Outlook GEO3 major region names

GEO3 Global Environment Outlook GEO3 major region names

IMAGE24 Image24 region names

GLOCAF GLOCAF region names

Stern Stern report region names

SRESmajor SRES major region names

SRES SRES region names

GBD Global Burden of Disease GBD region names

AVOIDnumeric numeric codes for AVOID regions

AVOIDname AVOID regions
 Vulnerability.Name country names with spaces replaced by '.'
 GIS.Country ESRI country names
 CNTRY_NAME more slightly different country names
 GMI_CNTRY more slightly different country names
 LDC UN Least Developed Countries
 SID UN Small Island Developing states
 LLDC UN Landlocked Developing Countries

Details

Used by [country2Region](#) and [mapByRegion](#).

Examples

```

data(countryRegions,envir=environment(),package="rworldmap")
str(countryRegions)

#joining example data onto the regional classifications
data(countryExData,envir=environment(),package="rworldmap")
dF <- merge(countryExData,countryRegions,by.x='ISO3V10',by.y='ISO3')
#plotting ENVHEALTH for Least Developed Countries (LDC) against others
plot( dF$ENVHEALTH ~ dF$LDC)
points( y=dF$ENVHEALTH, x=dF$LDC)

```

getMap

A simple way to access maps stored in the package.

Description

A simple way to access maps stored in the package.

Usage

```
getMap(resolution = "low", projection = "none")
```

Arguments

resolution	options "coarse", "low" or "medium", for projection="none" only
projection	Options "none" or "equalArea"

Value

A SpatialPolygonsDataFrame object.

Author(s)

Barry Rowlingson & Andy South

Examples

```
plot(getMap())
```

gridCountriesNumeric *A gloabl half degree grid specifying the country at each cell*

Description

A grid covering the globe at half degree resolution, specifying the country (UN numeric code) at each cell.

Usage

```
data(gridCountriesNumeric)
```

Format

The format is:

```
Formal class 'SpatialGridDataFrame' [package "sp"] with 6 slots
..@ data      : 'data.frame': 259200 obs. of  1 variable:
.. ..$ country.asc: num [1:259200] NA NA NA NA NA NA NA NA NA NA NA ...
..@ grid      : Formal class 'GridTopology' [package "sp"] with 3 slots
.. .. ..@ cellcentre.offset: num [1:2] -179.8 -89.8
.. .. ..@ cellsize          : num [1:2] 0.5 0.5
.. .. ..@ cells.dim         : int [1:2] 720 360
..@ grid.index : int(0)
..@ coords     : num [1:2, 1:2] -179.8 179.8 -89.8 89.8
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:2] "coords.x1" "coords.x2"
..@ bbox       : num [1:2, 1:2] -180 -90 180 90
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:2] "coords.x1" "coords.x2"
.. .. ..$ : chr [1:2] "min" "max"
..@ proj4string: Formal class 'CRS' [package "sp"] with 1 slots
.. .. ..@ projargs: chr " +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0"
```

Details

Uses a simple grid map defining a single country identity for each half degree cell. (sp, Spatial-GridDataFrame), used by the function `aggregateHalfDegreeGridToCountries()`

Source

IIASA

References<http://www.iiasa.ac.at/Research/GGI/DB/>**Examples**

```
data(gridCountriesNumeric)
```

gridExData	<i>Example half degree grid data : population estimates for 2000 from IIASA</i>
------------	---

Description

Example half degree grid data : people per cell estimates for 2000 from IIASA (International Institute for Applied System Analysis) (sp, SpatialGridDataFrame).

Usage

```
data(gridExData)
```

Format

The format is:

```
Formal class 'SpatialGridDataFrame' [package "sp"] with 6 slots
..@ data      : 'data.frame': 259200 obs. of  1 variable:
.. ..$ pa2000.asc: num [1:259200] NA NA NA NA NA NA NA NA NA NA NA ...
..@ grid      : Formal class 'GridTopology' [package "sp"] with 3 slots
.. .. ..@ cellcentre.offset: num [1:2] -179.8  -89.8
.. .. ..@ cellsize         : num [1:2] 0.5 0.5
.. .. ..@ cells.dim        : int [1:2] 720 360
..@ grid.index : int(0)
..@ coords    : num [1:2, 1:2] -179.8  179.8  -89.8   89.8
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : NULL
.. .. ..$ : chr [1:2] "coords.x1" "coords.x2"
..@ bbox      : num [1:2, 1:2] -180 -90 180 90
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:2] "coords.x1" "coords.x2"
.. .. ..$ : chr [1:2] "min" "max"
..@ proj4string: Formal class 'CRS' [package "sp"] with 1 slots
.. .. ..@ projargs: chr " +proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0"
```

Details

From International Institute for Applied System Analysis (IIASA) GGI Scenario Database, 2007 Available at: <http://www.iiasa.ac.at/Research/GGI/DB/> The data are made available for individual, academic research purposes only and on a "as is" basis, subject to revisions without further notice. Commercial applications are not permitted.

The data is used as the default dataset in other functions, e.g. `mapGriddedData()`, when no data file is given.

Source

<http://www.iiasa.ac.at/web-apps/ggi/GgiDb/dsd?Action=htmlpage&page=about>

References

Grubler, A., O'Neill, B., Riahi, K., Chirkov, V., Goujon, A., Kolp, P., Prommer, I., Scherbov, S. & Slentoe, E. (2006) Regional, national and spatially explicit scenarios of demographic and economic change based on SRES. *Technological Forecasting and Social Change* doi:10.1016/j.techfore.2006.05.023

Examples

```
data(gridExData)
```

<code>identifyCountries</code>	<i>a function that will print country name and attribute values when a user clicks on the map</i>
--------------------------------	---

Description

An interactive function that will print on a map the nearest country name to a user mouse click. The user can specify nothing and the function will use a map from the package. Alternatively the user can specify a data frame or `SpatialPolygonsDataFrame` in which case they need to define the column containing the country names (`nameCountryColumn`) and optionally a 2nd attribute column to print (`nameColumnToPlot`).

Usage

```
identifyCountries(df = ""  
                 , nameCountryColumn = "NAME"  
                 , nameX = "LON"  
                 , nameY = "LAT"  
                 , nameColumnToPlot = ""  
                 , plotSelected = FALSE  
                 , ...)
```

Arguments

<code>df</code>	data frame or <code>SpatialPolygonsDataFrame</code>
<code>nameCountryColumn</code>	name of column containing country names to be printed on the map (could also be set to any other attribute the user wants to query)
<code>nameX</code>	name of column containing the X variable (longitude), not needed if <code>df</code> is a <code>SpatialPolygonsDataFrame</code>
<code>nameY</code>	name of column containing the Y variable (latitude), not needed if <code>df</code> is a <code>SpatialPolygonsDataFrame</code>
<code>nameColumnToPlot</code>	name of an attribute column in the data frame the value of which will be appended to the country name when it is printed
<code>plotSelected</code>	if set to <code>TRUE</code> a blue outline will be printed around the countries selected when the selection process is finished
<code>...</code>	other parameters that can be passed to <code>identify()</code>

Details

Uses the `identify()` function, which waits for the user to click on the map, and stops when the user right clicks and selects 'stop'.

It uses country centroids, and will give a warning if one is too far away (default value of 0.25 inches).

Value

a vector of the indices of the countries selected

Author(s)

andy south

See Also

`identify()`

Examples

```
#mapCountryData()
#identifyCountries()

#identifyCountries(nameColumnToPlot = "POP2005", plotSelected = TRUE)
```

joinCountryData2Map *Joins user country referenced data to a map*

Description

Joins user data referenced by country codes or names to an internal map, ready for plotting using [mapCountryData](#). Reports join successes and failures.

Usage

```
joinCountryData2Map(dF
, joinCode = "ISO3"
, nameJoinColumn = "ISO3V10"
, nameCountryColumn = "Country"
, suggestForFailedCodes = FALSE
, projection="none" #options "none", "EqualArea"
, mapResolution="low"
, verbose = FALSE
)
```

Arguments

dF	R data frame with at least one column for country reference and one column of data
joinCode	how countries are referenced options "ISO2", "ISO3", "FIPS", "NAME", "UN" = numeric codes
nameJoinColumn	name of column containing country referencing
nameCountryColumn	optional name of column containing country names (used in reporting of success/failure)
suggestForFailedCodes	NOT YET ENABLED T/F whether you want system to suggest for failed codes
projection	which map projection to use, options 'none', 'EqualArea' latter uses Mollweide
mapResolution	resolution of the borders in the internal map, only for projection='none' : options 'low', 'medium'
verbose	if set to FALSE it doesn't print progress messages to console

Details

Joins data referenced by country codes to an internally stored map to enable plotting. The user specifies which country code their data are referenced by, and the name of the column in their data containing that referencing data. The user can choose from different map resolutions and projections, uses the function [getMap](#) to retrieve the map. The function reports on how many countries successfully join to the map. Data can then be plotted using [mapCountryData](#).

Value

An R 'SpatialPolygonsDataFrame' [package "sp"] object with the data joined to it

Author(s)

andy south

See Also

[mapCountryData](#), [getMap](#)

Examples

```
data("countryExData", envir=environment(), package="rworldmap")

sPDF <- joinCountryData2Map(countryExData
  , joinCode = "IS03"
  , nameJoinColumn = "IS03V10"
  , projection = "none"
  )
mapCountryData( sPDF
  , nameColumnToPlot="BIODIVERSITY"
  )
```

joinData2Map

Joins user polygon attribute data to a map

Description

Joins user polygon attribute data to a map of polygon boundaries. The map can either be one stored in the package or provided by the user. Returns a `spatialPolygonsDataFrame` ready for plotting using `mapPolys`. Reports join successes and failures.

Usage

```
joinData2Map(dF
  , nameMap = ""
  , nameJoinIDMap = "IS03"
  , nameJoinColumnData = "IS03V10"
  , nameNameColumnData = "Country"
  , suggestForFailedCodes = FALSE
  , projection="none"
  , mapResolution="low"
  , verbose = FALSE
  )
```

Arguments

df	R data frame with at least one column of polygon IDs and one column of data
nameMap	the map to join the attribute data too
nameJoinIDMap	the name of the joinIDs in the map
nameJoinColumnData	name of column in the data containing country referencing
nameNameColumnData	optional name of column in the data containing polygon names (used in reporting of success/failure)
suggestForFailedCodes	NOT YET ENABLED T/F whether you want system to suggest for failed codes
projection	which map projection to use, options 'none', 'EqualArea' latter uses Mollweide
mapResolution	resolution of the borders in the internal map, only for projection='none' : options 'low', 'medium'
verbose	if set to FALSE progress messages to console are restricted

Details

Joins user polygon attribute data provided in a 'data frame' to a map of polygon boundaries. The map can either be one stored in the package or provided by the user. Returns a `spatialPolygonsDataFrame` ready for plotting using `mapPolys`. Reports join successes and failures.

The user specifies the name of the column in their data containing polygon referencing.

The user can choose from different internal map resolutions and projections. Uses the function `getMap` to retrieve the map.

Value

An R 'SpatialPolygonsDataFrame' [package "sp"] object with the data joined to it

Author(s)

andy south

See Also

[mapPolys](#), [getMap](#)

Examples

```
## this example uses downloaded files
## to run it download the files
## and remove the comment symbols '#' from all the lines starting with a single '#'

## US states map downloaded from :
## http://www2.census.gov/cgi-bin/shapefiles2009/national-files
```

```

#inFile <- 'tl_2009_us_stateec.shp'
#sPDF <- readShapePoly(inFile)
#str(sPDF@data)

#####
## use mapPolys to map the sPDF
#mapPolys(sPDF,nameColumnToPlot = "ALANDEC")
#mapPolys(sPDF,nameColumnToPlot = "AWATEREC",mapRegion='North America')

#####
## join some other data to it
## education data downloaded from here as xls then saved as csv
## http://nces.ed.gov/ccd/drpcompstatelvl.asp

#dataFile <- 'SDR071A_xls.csv'
#dF <- read.csv(dataFile,as.is=TRUE)
#str(dF)
## STATENAME
## DRP912 Dropout Rate, Grades 9 through 12

## joining the data to the map
## based upon state names (column NAMEEC in map, and STATENAME in the data)
#sPDF2 <- joinData2Map(dF
#      , nameMap = sPDF
#      , nameJoinIDMap = "NAMEEC"
#      , nameJoinColumnData = "STATENAME")

#####
## plot one of the attribute variables
#mapDevice()# to set nice shape map window
#mapPolys(sPDF2,nameColumnToPlot = "DRP912",mapRegion='North America')

```

mapBars

function to produce bar plots on a map

Description

The function will produce a map with bars centred on country centroids (or other chosen points). The length of the bars is determined by the sum of the attribute columns and each section is coloured.

Usage

```

mapBars(dF = ""
        , nameX="longitude", nameY="latitude"
        , nameZs=c(names(dF)[3],names(dF)[4])
        , zColours=c(1:length(nameZs))

        , barWidth = 1

```

```

, barOrient = 'vert'

, ratio = 1
, addCatLegend = TRUE
, addSizeLegend = TRUE

, symbolSize = 1
, maxZVal=NA

, xlim=c(-160,160)
, ylim=c(-80,90)
, mapRegion = "world"
, borderCol = "grey"
, oceanCol=NA
, landCol=NA
, add=FALSE
, main=''
,... )

```

Arguments

dF	data frame or SpatialPolygonsDataFrame
nameX	name of column containing the X variable (longitude), not needed if dF is a SpatialPolygonsDataFrame
nameY	name of column containing the Y variable (latitude), not needed if dF is a SpatialPolygonsDataFrame
nameZs	name of columns containing numeric variables to determine bar sections
zColours	colours to apply to the bar section for each attribute column
barWidth	multiple for the width of bar symbols, relative to barOrient see below
barOrient	orientation of bars, options 'horiz' and 'vert'
ratio	the ratio of Y to N in the output map, set to 1 as default
addCatLegend	whether to add a legend for categories
addSizeLegend	whether to add a legend for symbol size
symbolSize	multiplier of default symbol size
maxZVal	the attribute value corresponding to the maximum symbol size, this can be used to set the scaling the same between multiple plots
xlim	map extents c(west,east), can be overridden by mapRegion
ylim	map extents c(south,north), can be overridden by mapRegion
mapRegion	a country name from getMap()[['NAME']] or 'world', 'africa', 'oceania', 'eurasia', 'uk' sets map extents, overrides xlim,ylim
borderCol	the colour for country borders
oceanCol	a colour for the ocean
landCol	a colour to fill countries

add	whether to add the symbols to an existing map, TRUE/FALSE
main	title for the map
...	any extra arguments to points()

Details

Horizontal or vertical bars can be achieved by using the barOrient argument 'horiz' or 'vert'.

Value

currently doesn't return anything

Author(s)

andy south

Examples

```
#getting example data
dF <- getMap()@data

## these examples repeat the same column in 'nameZs' to show that equal sized bars are created

#mapBars( dF,nameX="LON", nameY="LAT",nameZs=c('AREA','AREA') )

#mapBars( dF,nameX="LON", nameY="LAT",nameZs=c('AREA','AREA'),mapRegion='africa' )

#mapBars( dF,nameX="LON", nameY="LAT"
#         , nameZs=c('AREA','AREA','AREA','AREA'),mapRegion='africa' )

#mapBars( dF,nameX="LON", nameY="LAT"
#         , nameZs=c('AREA','AREA','AREA','AREA'),mapRegion='africa',symbolSize=2 )

mapBars( dF, nameX="LON", nameY="LAT"
        , nameZs=c('AREA','AREA','AREA','AREA')
        , mapRegion='africa'
        , symbolSize=2
        , barOrient = 'horiz' )
```

Description

The function will produce a map with bubbles (circles) centred on country centroids (or other chosen points). Bubbles can be sized and coloured according to specified attribute values.

Usage

```
mapBubbles(dF = ""
  , nameX = "longitude"
  , nameY = "latitude"
  , nameZSize = ""
  , nameZColour = ""

  , fill = TRUE
  , pch = 21
  , symbolSize = 1
  , maxZVal = NA

  , main = nameZSize

  , numCats = 5
  , catMethod = "categorical"
  , colourPalette = "heat"

  , mapRegion = "world"
  , borderCol = "grey"
  , oceanCol = NA
  , landCol = NA

  , addLegend = TRUE
  , legendBg = "white"
  , legendVals = ""
  , legendPos = "bottomright"
  , legendHoriz = FALSE
  , legendTitle = nameZSize
  , addColourLegend = TRUE
  , colourLegendPos = "bottomleft"
  , colourLegendTitle = nameZColour
  , add = FALSE
  , plotZeroVals = TRUE
  , ...)
```

Arguments

dF	data frame or SpatialPolygonsDataFrame
nameX	name of column containing the X variable (longitude), not needed if dF is a SpatialPolygonsDataFrame
nameY	name of column containing the Y variable (latitude), not needed if dF is a SpatialPolygonsDataFrame

nameZSize	name of column containing numeric variable to set symbol size
nameZColour	name of column containing variable to set symbol colour
fill	whether or not to fill symbols TRUE/FALSE
pch	symbol type, default of 21 for circles, will work with other filled symbol types e.g. 22=square, 23=diamond, 24=triangle
symbolSize	multiplier of default symbol size
maxZVal	the attribute value corresponding to the maximum symbol size, this can be used to set the scaling the same between multiple plots
main	title for the map, set to nameZSize by default
numCats	number of categories to put the data in, may be modified if this number is in- compatible with the catMethod chosen
catMethod	method for categorisation of data "pretty", "fixedWidth", "diverging", "logfixed- Width", "quantiles", "categorical", or a numeric vector defining breaks
colourPalette	a string describing the colour palette to use, choice of : <ol style="list-style-type: none"> 1. ="palette" for the current palette 2. a vector of valid colours, e.g. =c('red', 'white', 'blue') or output from RColour- Brewer 3. = one of "heat", "diverging", "white2Black", "black2White", "topo", "rainbow", "terrain", "negpos8", "ne-
mapRegion	a country name from getMap()\$NAME or 'world', 'africa', 'oceania', 'eurasia', 'uk' sets map extents, overrides xlim,ylim
borderCol	the colour for country borders
oceanCol	a colour for the ocean
landCol	a colour to fill countries
addLegend	whether to add a legend for symbol sizes
legendBg	background colour for the legend, NA=transparent
legendVals	allows user to set values & hence symbol sizing in legend
legendPos	positioning of legend e.g. 'bottomleft', 'topright'
legendHoriz	whether to arrange legend elements horizontally TRUE/FALSE
legendTitle	title for the symbol size legend
addColourLegend	whether to add a legend for symbol colour
colourLegendPos	positioning of colour legend e.g. 'bottomleft', 'topright'
colourLegendTitle	title for the colour size legend
add	whether to add the symbols to an existing map, TRUE/FALSE
plotZeroVals	whether to plot zero values as a cross, TRUE/FALSE
...	any extra arguments to points()

Details

By default separate legends are added for symbol size and colouring on either side of the plot, these can be modified by altering legend parameters.

Value

currently doesn't return anything

Author(s)

andy south

Examples

```
mapBubbles()
#square symbols
mapBubbles(pch=22)

mapBubbles(dF=getMap(), nameZSize="POP2005", nameZColour="REGION")

#change colour
mapBubbles(dF=getMap(), nameZSize="POP2005", nameZColour="REGION"
           ,colourPalette='rainbow', oceanCol='lightblue', landCol='wheat')

data("countryExData",envir=environment(),package="rworldmap")
sPDF <- joinCountryData2Map(countryExData,joinCode = "ISO3"
                           ,nameJoinColumn = "ISO3V10")

mapBubbles(sPDF, nameZSize="AREA",nameZColour="BIODIVERSITY"
           ,colourPalette='topo',numCats=5,catMethod="quantiles")
```

mapByRegion

Produce maps of regional level data from country level data

Description

This function will produce maps of regional statistics by aggregating country level data. For example mapping the total population of Asia, Europe, etc, from country level population data. As well as sums, other functions can be used, like mean, median, min, max, etc. There are currently 8 choices of region and 4 choices of country code.

Usage

```
mapByRegion(inFile, nameDataColumn, joinCode,
            nameJoinColumn, regionType, FUN = 'mean', na.rm=TRUE, mapTitle = '', ...)
```

Arguments

<code>inFile</code>	a data frame
<code>nameDataColumn</code>	The name of a column of <code>inFile</code> . This is data is aggregated by FUN
<code>joinCode</code>	The type of code to join with. Must be one of: "ISO2", "ISO3", "Numeric" or "FIPS"
<code>nameJoinColumn</code>	The name of a column of <code>inFile</code> . Contains joining codes.
<code>regionType</code>	Must be one of: "GEO3", "GEO3major", "IMAGE24", "GLOCAF", "Stern", "SRES", "SRESmajor", "GBD", "AVOIDname"
<code>FUN</code>	A function to apply to each region
<code>na.rm</code>	Only used for certain values of FUN. See details section below.
<code>mapTitle</code>	a title to be printed above the map
<code>...</code>	further arguments to be passed to mapCountryData

Details

The function is very similar to `country2Region`. The first difference is that the output is a map, rather than statistics. The second is the behaviour of extra arguments. In `country2Region` the extra arguments go to FUN, here they go to `mapCountryData`.

The `na.rm` argument is used when FUN has one of the following values: "mean", "min", "max", "median", "range", "var", "sd", "m" or "IQR". This reduces the problem of not being able to supply extra arguments to FUN.

Value

`invisibly` returns a list containing the data and main options used for the map, the list can be passed to [addMapLegend](#) along with additional options to allow greater flexibility in legend creation.

See Also

An alternative tool to [country2Region](#). The plotting is done by [mapCountryData](#)

Examples

```
data(countryExData)

mapByRegion(inFile=countryExData
            ,nameDataColumn="CLIMATE"
            ,joinCode="ISO3"
            ,nameJoinColumn="ISO3V10"
            ,regionType="Stern"
            ,FUN='mean'
            )
```

mapCountryData	<i>Map country-level data.</i>
----------------	--------------------------------

Description

Draw a map of country-level data, allowing countries to be coloured, from an object created in `joinCountryData2Map()`.

Usage

```
mapCountryData(
  mapToPlot =      ""
  , nameColumnToPlot = ""
  , numCats =      7
  , xlim =         c(-180,180)
  , ylim =         c(-80,90)
  , mapRegion =    "world"
  , catMethod =    "quantiles"
  , colourPalette = "heat"
  , addLegend =    TRUE
  , borderCol =    'grey'
  , mapTitle =     'columnName'
  , oceanCol =     NA
  , aspect =       1
  , missingCountryCol = NA
  , add =          FALSE
  , nameColumnToHatch = ""
)
```

Arguments

<code>mapToPlot</code>	a spatial polygons dataframe from <code>joinCountryData2Map()</code> containing country polygons and data, if none specified an internal example data is used
<code>nameColumnToPlot</code>	name of column containing the data you want to plot
<code>numCats</code>	number of categories to put the data in, may be modified if this number is incompatible with the <code>catMethod</code> chosen
<code>xlim</code>	map extents <code>c(west,east)</code> , can be overridden by <code>mapRegion</code>
<code>ylim</code>	map extents <code>c(south,north)</code> , can be overridden by <code>mapRegion</code>
<code>mapRegion</code>	a country name from <code>getMap()[['NAME']]</code> or <code>'world'</code> , <code>'africa'</code> , <code>'oceania'</code> , <code>'eurasia'</code> , <code>'uk'</code> sets map extents, overrides <code>xlim,ylim</code>
<code>catMethod</code>	method for categorisation of data "pretty", "fixedWidth", "diverging", "logfixed-Width", "quantiles", "categorical", or a numeric vector defining breaks
<code>colourPalette</code>	string describing the colour palette to use, choice of: <ol style="list-style-type: none"> 1. "palette" for the current palette

	2. a vector of valid colours, e.g. =c('red','white','blue') or output from RColourBrewer
	3. one of "heat", "diverging", "white2Black", "black2White", "topo", "rainbow", "terrain", "negpos8", "negpos4"
addLegend	whether to add a legend or not
borderCol	the colour for country borders
mapTitle	title to add to the map, any string or 'columnName' to set it to the name of the data column
oceanCol	a colour for the ocean
aspect	aspect for the map, defaults to 1, if set to 'variable' uses same method as plot.Spatial in sp
missingCountryCol	a colour for missing countries
add	whether to add this map on top of an existing map, TRUE/FALSE
nameColumnToHatch	allows hatching of country fills (e.g. to represent uncertainty) , specify a column containing numeric data , highest values will be solid and lower values will have a decreasing density of hatching , new feature more documentation will be added soon

Details

Certain `catMethod` and `colourPalette` options go well together. e.g. "diverging" and "diverging", "categorical" and "rainbow"

There are two styles of legend available. If `catMethod='categorical'` or the packages `fields` and `spam` are not installed a simple legend with coloured boxes is created. Otherwise a colour bar legend is created. Finer control can be achieved by [addMapLegendBoxes](#) or [addMapLegend](#) respectively.

Value

`mapCountryData` invisibly returns a list containing the data and main options used for the map, the list can be passed to [addMapLegend](#) or [addMapLegendBoxes](#) along with additional options to allow greater flexibility in legend creation.

Warning

will generate unhelpful errors in data categorisation if inappropriate options are chosen, e.g. with `catMethod:Quantiles` if `numCats` too high so that unique breaks cannot be defined.

Author(s)

andy south

See Also

`classInt`, `RColorBrewer`

Examples

```

mapCountryData()
data("countryExData",envir=environment(),package="rworldmap")
sPDF <- joinCountryData2Map(countryExData
  , joinCode = "ISO3"
  , nameJoinColumn = "ISO3V10"
  , projection = "none"
  )
mapCountryData( sPDF
  , nameColumnToPlot="BIODIVERSITY"
  )

#user defined map colour scheme for categorical data
mapParams <- mapCountryData(nameColumnToPlot='REGION'
  , catMethod='categorical'
  , addLegend='FALSE'
  , colourPalette=c('white','green','red','yellow','blue','black')
  )
#changing legendText
mapParams$legendText <- c('antarctic','africa','oceania','americas','s.asia','eurasia')
do.call( addMapLegendBoxes, c(mapParams,x='bottom',title="Region",horiz=TRUE))

##showing how rworldmap can be used with the classInt and RColorBrewer packages
library(classInt)
library(RColorBrewer)
#getting example data and joining to a map
data("countryExData",envir=environment(),package="rworldmap")
sPDF <- joinCountryData2Map(countryExData,joinCode = "ISO3"
  ,nameJoinColumn = "ISO3V10")
#getting class intervals using a 'jenks' classification in classInt package
classInt <- classIntervals( sPDF$EPI, n=5, style="jenks")
catMethod = classInt$brks
#getting a colour scheme from the RColorBrewer package
colourPalette <- brewer.pal(5,'RdPu')
#calling mapCountryData with the parameters from classInt and RColorBrewer
mapParams <- mapCountryData( sPDF, nameColumnToPlot="EPI", addLegend=FALSE
  , catMethod = catMethod, colourPalette=colourPalette )
do.call(addMapLegend, c(mapParams
  ,legendLabels="all"
  ,legendWidth=0.5
  ,legendIntervals="data"))

```

mapDevice

Creates a plot device set up for maps

Description

Creates a plot device suited for rworldmap plotting functions.

Usage

```
mapDevice(
  device = "dev.new"
  , rows = 1
  , columns = 1
  , plotOrder="rows"
  , projection = "none"
  , width = NULL
  , height = NULL
  , titleSpace = NULL
  , mai = c(0, 0, 0.2, 0)
  , mgp = c(0, 0, 0)
  , xaxs = "i"
  , yaxs = "i"
  , ...)
```

Arguments

device	Character string which controls the type of plot default. The default uses your standard plot device. Giving the name of a plotting device function will use that instead. e.g. "pdf", "png", etc.
rows	The number of rows. Default 1
columns	The number of columns. Default 1
plotOrder	Option of 'rows' or 'columns'. For multiple plots whether to plot in row or column order. However, note that addMapLegend can have the effect of reverting order to rows.
projection	The projection which you intend to use for the maps. Ignored if you specify both width and height.
width	The width of a single plot. This includes the margins. If you do not specify both width and height, suitable values will be calculated
height	The height of a single plot. This includes the margins. If you do not specify both width and height, suitable values will be calculated
titleSpace	The height in inches of the gap at the plot.
mai	The margin sizes in inches. If titleSpace is given this overrides mai[3].
mgp	As per par(mgp) in the graphics package
xaxs	As per par(xaxs) in the graphics package
yaxs	As per par(yaxs) in the graphics package
...	Further arguments to the device function

Value

Used for the side effect of creating a plot device, and setting graphical parameters for the device.

See Also

mapCountryData, mapGridAscii

Examples

```
## Not run:
#Basic Usage
mapDevice()
mapCountryData()

#2 by 2 plot
mapDevice(rows=2,columns=2)
columns<-c("BIODIVERSITY","EPI","ENVHEALTH","Population2005")
for(i in columns){
  mapCountryData(nameColumnToPlot=i)
}
#Creating a pdf that is 5 inches wide
mapDevice(device="pdf",width=5,file=tempfile())
mapCountryData()
dev.off()

## End(Not run)
```

mapGriddedData

Produce maps of global gridded data at half degree resolution

Description

Produce maps of global gridded data at half degree resolution

Usage

```
mapGriddedData( dataset =      ""
                , nameColumnToPlot = ""
                , numCats =      5
                , catMethod =    "quantiles"
                , colourPalette = "heat"
                , xlim =         c(-180,180)
                , ylim =         c(-80,90)
                , mapRegion =    "world"
                , addLegend =    TRUE
                , addBorders =   'low'
                , borderCol =    'grey'
                , oceanCol =     NA
                , landCol =      NA
                , plotData =     TRUE
                , aspect =       1
                )
```

Arguments

dataset	gridded data either as a : <ol style="list-style-type: none"> 1. SpatialGridDataFrame (R object defined in package sp) 2. file name of a GridAscii file - this is an Esri format 3. 2D R matrix or array (rows by columns)
nameColumnToPlot	name of column containing the data to plot
numCats	number of categories to put the data in, may be overridden if catMethod = 'pretty'
catMethod	method for categorisation of data "pretty", "fixedWidth", "diverging", "logfixed-Width", "quantiles", "categorical", or a numeric vector defining breaks
colourPalette	a string describing the colour palette to use, choice of : <ol style="list-style-type: none"> 1. "palette" for the current palette 2. a vector of valid colours, e.g. =c('red', 'white', 'blue') or output from RColour-Brewer 3. one of "heat", "diverging", "white2Black", "black2White", "topo", "rainbow", "terrain", "negpos8", "negp
xlim	map extents c(west,east), can be overridden by mapRegion
ylim	map extents c(south,north), can be overridden by mapRegion
mapRegion	a country name from getMap()[['NAME']] or 'world', 'africa', 'oceania', 'eurasia', 'uk' sets map extents, overrides xlim,ylim
addLegend	whether to add a legend or not
addBorders	options for country borders, 'low', 'coarse' = low or coarse resolution, 'coasts' = coasts only, 'none' or NA for none
borderCol	the colour for country borders
oceanCol	a colour for the ocean if the grid values are NA
landCol	a colour to fill countries if the grid values are NA over land
plotData	whether to plotData, if FALSE a legend can be added on its own
aspect	aspect for the map, defaults to 1, if set to 'variable' uses same method as plot.Spatial in sp

Details

Plots a map of global half degree gridded data, allowing classification, colours and regions to be set.

Certain catMethod and colourPalette options go well together. e.g. "diverging" and "diverging", "categorical" and "rainbow"

Value

invisibly returns a list containing the data and main options used for the map, the list can be passed to [addMapLegend](#) along with additional options to allow greater flexibility in legend creation.

Author(s)

andy south and matthew staines

See Also

classInt, RColorBrewer

Examples

```
## mapping continuous data
data(gridExData,envir=environment(),package="rworldmap")
gridExData <- get("gridExData")
mapGriddedData(gridExData)

## reclassing continuous data to categorical & mapping
data(gridExData,envir=environment(),package="rworldmap")
#find quartile breaks
cutVector <- quantile(gridExData@data[,1],na.rm=TRUE)
#classify the data to a factor
gridExData@data$categories <- cut( gridExData@data[,1]
                                   , cutVector, include.lowest=TRUE)

#rename the categories
levels(gridExData@data$categories) <- c('low', 'med', 'high', 'vhigh')
#mapping
mapGriddedData( gridExData, nameColumnToPlot= 'categories'
                , catMethod='categorical')
```

mapHalfDegreeGridToCountries

Maps user half degree gridded data at country level by first aggregating.

Description

Maps user half degree gridded data at country level by first aggregating.

Usage

```
mapHalfDegreeGridToCountries(
  inFile=""
  , aggregateOption="sum"
  , nameCountryColumn = ""
  , suggestForFailedCodes = FALSE
  , projection="none"
  , mapResolution="low"
  , numCats = 7
  , xlim=c(-160,160)
  , ylim=c(-80,90)
  , mapRegion = "world"
  , catMethod="quantiles")
```

```

    , colourPalette= "heat"
    , addLegend=TRUE
  )

```

Arguments

inFile	either a gridascii filename or an sp SpatialGridDataFrame object specifying a global half degree grid dataset, if none specified an internal example data is used
aggregateOption	how to aggregate the data ('sum', 'mean', 'min', 'max')
nameCountryColumn	optional name of column containing country names (used in reporting of success/failure)
suggestForFailedCodes	T/F whether you want system to suggest for failed codes NOT YET WORKING
projection	'none' or 'EqualArea'
mapResolution	options low, medium, only for projection='none' initially
numCats	number of categories, may be overided e.g. if catMethod ='pretty'
xlim	map extents c(west,east), can be overridden by mapRegion
ylim	map extents c(south,north), can be overridden by mapRegion
mapRegion	'world', 'africa', 'oceania', 'eurasia', 'uk' sets map extents, overrides we,ea etc.
catMethod	method for categorisation of data "pretty", any vector defining breaks, "fixed-Width", "quantiles"
colourPalette	"heat", "white2Black", "palette":for current palette
addLegend	whether to add a legend or not T/F

Details

Aggregates half degree gridded data to countries using the option specified in 'aggregateOption' then maps at a country level.

Value

invisibly returns a list containing the data and main options used for the map, the list can be passed to [addMapLegend](#) along with additional options to allow greater flexibility in legend creation.

Author(s)

andy south

See Also

[aggregateHalfDegreeGridToCountries](#)

nameY	name of column containing the Y variable (latitude), not needed if dF is a SpatialPolygonsDataFrame
nameZs	name of columns containing numeric variables to determine pie sections
zColours	colours to apply to the pie section for each attribute column
ratio	the ratio of Y to N in the output map, set to 1 as default
addCatLegend	whether to add a legend for categories
addSizeLegend	whether to add a legend for symbol size
symbolSize	multiplier of default symbol size
maxZVal	the attribute value corresponding to the maximum symbol size, this can be used to set the scaling the same between multiple plots
xlim	map extents c(west,east), can be overridden by mapRegion
ylim	map extents c(south,north), can be overridden by mapRegion
mapRegion	a country name from getMap()[['NAME']] or 'world','africa','oceania','eurasia','uk' sets map extents, overrides xlim,ylim
borderCol	the colour for country borders
oceanCol	a colour for the ocean
landCol	a colour to fill countries
add	whether to add the symbols to an existing map, TRUE/FALSE
main	title for the map
...	any extra arguments to points()

Details

Beware of creating plots that are difficult for the reader to interpret. More than 3 or 4 categories may be too many.

Value

currently doesn't return anything

Author(s)

andy south

Examples

```
#getting example data
dF <- getMap()@data

## these examples repeat the same column in 'nameZs'
## to show that equal sized pies are created

#mapPies( dF,nameX="LON", nameY="LAT",nameZs=c('AREA','AREA') )
```

```
#mapPies( dF,nameX="LON", nameY="LAT",nameZs=c('AREA','AREA')
#         , mapRegion='africa' )

mapPies( dF,nameX="LON", nameY="LAT"
         , nameZs=c('AREA','AREA','AREA','AREA'),mapRegion='africa' )
```

mapPolys

Map polygon data.

Description

Plot a map of polygons, from a spatialPolygonsDataFrame, coloured according to one a specified attribute column.

Usage

```
mapPolys( mapToPlot = ""
         , nameColumnToPlot = ""
         , numCats = 7
         , xlim=c(-160,160)
         , ylim=c(-80,90)
         , mapRegion = "world"
         , catMethod="quantiles"
         , colourPalette= "heat"
         , addLegend=TRUE
         , borderCol = 'grey'
         , mapTitle = 'columnName'
         , oceanCol=NA
         , aspect=1
         , missingCountryCol=NA
         , add=FALSE
         )
```

Arguments

mapToPlot	a spatial polygons dataframe (e.g. from joinData2Map()) containing polygons and associated data, if none specified an internal example data is used
nameColumnToPlot	name of column containing the data you want to plot
numCats	number of categories to put the data in, may be modified if this number is incompatible with the catMethod chosen
xlim	map extents c(west,east), can be overridden by mapRegion
ylim	map extents c(south,north), can be overridden by mapRegion
mapRegion	a country name from getMap()[['NAME']] or 'world','africa','oceania','eurasia','uk' sets map extents, overrides xlim,ylim

catMethod	for categorisation of data "pretty", "fixedWidth", "diverging", "logfixedWidth", "quantiles", "categorical", or a numeric vector defining breaks
colourPalette	string describing the colour palette to use, choice of: <ol style="list-style-type: none"> 1. "palette" for the current palette 2. a vector of valid colours, e.g. =c('red','white','blue') or output from RColourBrewer 3. one of "heat", "diverging", "white2Black", "black2White", "topo", "rainbow", "terrain", "negpos8", "negp
addLegend	whether to add a legend or not
borderCol	the colour for country borders
mapTitle	title to add to the map, any string or 'columnName' to set it to the name of the data column
oceanCol	a colour for the ocean
aspect	aspect for the map, defaults to 1, if set to 'variable' uses same method as plot.Spatial in sp
missingCountryCol	a colour for missing countries
add	whether to add this map on top of an existing map, TRUE/FALSE

Details

Certain catMethod and colourPalette options go well together. e.g. "diverging" and "diverging", "categorical" and "rainbow"

There are two styles of legend available. If catMethod='categorical' or the packages fields and spam are not installed a simple legend with coloured boxes is created. Otherwise a colour bar legend is created. Finer control can be achieved by [addMapLegendBoxes](#) or [addMapLegend](#) repectively.

Value

invisibly returns a list containing the data and main options used for the map, the list can be passed to [addMapLegend](#) or [addMapLegendBoxes](#) along with additional options to allow greater flexibility in legend creation.

Author(s)

andy south

See Also

[joinData2Map](#), [classInt](#), [RColorBrewer](#)

Examples

```
## this example uses downloaded files
## to run it download the files
## and remove the comment symbols '#' from all the lines starting with a single '#'
```

```

## US states map downloaded from :
## http://www2.census.gov/cgi-bin/shapefiles2009/national-files

#inFile <- 'tl_2009_us_stateec.shp'
#sPDF <- readShapePoly(inFile)
#str(sPDF@data)

#####
## use mapPolys to map the sPDF
#mapPolys(sPDF,nameColumnToPlot = "ALANDEC")
#mapPolys(sPDF,nameColumnToPlot = "AWATEREC",mapRegion='North America')

#####
## join some other data to it
## education data downloaded from here as xls then saved as csv
## http://nces.ed.gov/ccd/drpcompstatelvl.asp

#dataFile <- 'SDR071A_xls.csv'
#dF <- read.csv(dataFile,as.is=TRUE)
#str(dF)
## STATENAME
## DRP912 Dropout Rate, Grades 9 through 12

## joining the data to the map
## based upon state names (column NAMEEC in map, and STATENAME in the data)
#sPDF2 <- joinData2Map(dF
#           , nameMap = sPDF
#           , nameJoinIDMap = "NAMEEC"
#           , nameJoinColumnData = "STATENAME")

#####
## plot one of the attribute variables
#mapDevice()# to set nice shape map window
#mapPolys(sPDF2,nameColumnToPlot = "DRP912",mapRegion='North America')

#####
###to map US counties data (Tiger) downloaded from :
##http://www2.census.gov/cgi-bin/shapefiles2009/national-files

#inFile <- 'tl_2009_us_county.shp'
#sPDF <- readShapePoly(inFile)
#str(sPDF@data)
#mapPolys(sPDF,nameColumnToPlot='AWATER',xlim=c(-140,-65), ylim=c(25,45))

```

rwmGetClassBreaks

Internal function to set the numeric values for the breaks between data categories

Description

Sets the values that determine how a vector of continuous data is classified into categories. Called by `mapCountryData()` and `mapGriddedData()`

Usage

```
rwmGetClassBreaks(dataColumn, catMethod, numCats, verbose=TRUE, midpoint=0)
```

Arguments

<code>dataColumn</code>	the data vector to be classified, must be numeric
<code>catMethod</code>	the method to use to classify the data into categories, choice of "pretty", "fixed-Width", "diverging", "logfixedWidth", "quantiles", "categorical" or a numeric vector defining breaks
<code>numCats</code>	number of categories to put the data in, may be overridden if not possible under some classification methods
<code>verbose</code>	whether to print information messages to console TRUE/FALSE
<code>midpoint</code>	the midpoint to use if <code>catMethod='diverging'</code> , default=0

Value

A vector specifying the numeric breaks between data categories.

Author(s)

andy south and matthew staines

See Also

The `classInt` package

`rwmGetColours`

Internal function to choose map colours for classified data

Description

Internal function to choose map colours for classified data.

Usage

```
rwmGetColours(colourPalette, numColours)
```

Arguments

- `colourPalette` string describing the colour palette to use, choice of:
1. "palette" for the current palette
 2. a vector of valid colours, e.g. `=c('red','white','blue')` or output from `RColourBrewer`
 3. one of "heat", "diverging", "white2Black", "black2White", "topo", "rainbow", "terrain", "negpos8", "negp"
- `numColours` the number of colour categories desired

Details

Returns a vector of colours based upon the palette specified and number of colours specified. If `colourPalette` specifies a number of colours and this is different from `numColours`, `numColours` takes precedence and colours are interpolated to make the number fit.

Value

A vector specifying a number of colours.

Author(s)

andy south and matthew staines

See Also

`RColorBrewer`

Examples

```
rwmGetColours(colourPalette='topo', numColours=10)
```

`rwmNewMapPlot`

Internal function to set up an existing device for plotting maps

Description

Sets the region, aspect and ocean colour for a new map plot

Usage

```
rwmNewMapPlot(mapToPlot, oceanCol, mapRegion, xlim, ylim, aspect)
```

Arguments

mapToPlot	the worldmap to be plotted
oceanCol	a colour for the ocean
mapRegion	a string specifying the map region, see setMapExtents()
xlim	map extents c(west,east), can be overridden by mapRegion
ylim	map extents c(south,north), can be overridden by mapRegion
aspect	aspect for the map, defaults to 1, if set to 'variable' uses same default as plot.Spatial in sp

Details

Called by mapCountryData() and mapGriddedData()

Author(s)

andy south

rworldmapExamples *Example code for plot creation*

Description

Example code to demonstrate creation of a series of plots

Usage

```
rworldmapExamples()
```

Author(s)

andy south

setMapExtents *Internal function allowing map extents to be set from area names*

Description

Allows map extents to be set from country or area names (e.g. India, Africa)

Usage

```
setMapExtents(mapRegion = "world")
```

Arguments

mapRegion a country name from `getMap()[['NAME']]` or one of 'eurasia', 'africa', 'latin america', 'uk', 'oceania', 'asia'

Details

Can be called by `mapCountryData` and `mapGriddedData`

Value

a dataframe containing we,ea,so,no values in degrees between -180 & +180

Author(s)

andy south

Examples

```
mapCountryData( mapRegion='Africa' )
mapCountryData( mapRegion='India' )
```

wrl_d_coarse

a very coarse world map, only suitable for small global maps

Description

A 'SpatialPolygonsDataFrame' [package "sp"] object containing a very simplified world map. Polygons are attributed with country codes.

Usage

```
data(wrl_d_coarse)
```

Format

```
Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
..@ data      : 'data.frame': 174 obs. of 12 variables:
.. ..$ SP_ID  : Factor w/ 174 levels "AFG","AGO","ALB",..
.. ..$ FIPS   : Factor w/ 173 levels "AE","AF","AG",..
.. ..$ ISO2   : Factor w/ 174 levels "AD","AE","AF",..
.. ..$ ISO3   : Factor w/ 174 levels "AFG","AGO","ALB",..
.. ..$ UN     : int [1:174] 4 24 8 ...
.. ..$ NAME   : Factor w/ 174 levels "Afghanistan",..
.. ..$ AREA   : int [1:174] 65209 124670 2740 ...
.. ..$ POP2005 : num [1:174] 25067407 16095214 3153731 ...
.. ..$ REGION : int [1:174] 142 2 150 ...
.. ..$ SUBREGION: int [1:174] 34 17 39 ...
.. ..$ LON    : num [1:174] 65.22 17.54 20.07 ...
```

```

.. ..$ LAT      : num [1:174] 33.7 -12.3 41.1 ...
.. ..- attr(*, "data_types")= chr [1:12] "C" "C" "C" "C" ...
..@ polygons    :List of 174
.. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots

..@ plotOrder   : int [1:174] 8 135 28 166 31 23 10 66 85 6 ...
..@ bbox        : num [1:2, 1:2] -180 -90 180 83.6
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:2] "r1" "r2"
.. .. ..$ : chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
.. .. ..@ projargs: chr NA

```

Source

original shapefile obtained from : http://thematicmapping.org/downloads/world_borders.php Text from the site : "The original shapefile (world_borders.zip, 3.2 MB) was downloaded from the Mapping Hacks website: <http://www.mappinghacks.com/data/>

The dataset is available under a Creative Commons Attribution-Share Alike License. If you use this dataset, please provide a link to this website."

imported to R and cleaned by Roger Bivand

simplified using mapShaper (www.mapshaper.org) and reduce the size to 30 percent for faster printing

Examples

```
data(wrld_coarse)
```

wrld_simpl

World map simple

Description

A 'SpatialPolygonsDataFrame' [package "sp"] object containing a simplified world map. Polygons are attributed with country codes.

Usage

```
data(wrld_simpl)
```

Format

```

Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
..@ data      :'data.frame': 246 obs. of 11 variables:
.. ..$ FIPS   : Factor w/ 244 levels "", "AC", "AE", "AF", ..
.. ..$ ISO2   : Factor w/ 246 levels "AD", "AE", "AF", ..
.. ..$ ISO3   : Factor w/ 246 levels "ABW", "AFG", "AGO", ..

```

```

.. ..$ UN      : int [1:246] 28 12 31 8 51 24 16 32 36 48 ...
.. ..$ NAME    : Factor w/ 246 levels "Afghanistan",...
.. ..$ AREA    : int [1:246] 44 238174 8260 2740 2820 ...
.. ..$ POP2005 : int [1:246] 83039 32854159 8352021 3153731 3017661 ...
.. ..$ REGION  : int [1:246] 19 2 142 150 142 2 9 19 9 142 ...
.. ..$ SUBREGION: int [1:246] 29 15 145 39 145 17 61 5 53 145 ...
.. ..$ LON     : num [1:246] -61.78  2.63 47.40 20.07 44.56 ...
.. ..$ LAT     : num [1:246] 17.1 28.2 40.4 41.1 40.5 ...
..@ polygons   :List of 246
.. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
...
..@ plotOrder  : int [1:246] 145 175 24 209 30 21 9 71 98 8 ...
..@ bbox       : num [1:2, 1:2] -180 -90 180 90
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:2] "r1" "r2"
.. .. ..$ : chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
.. .. ..@ projargs: chr " +proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs +towgs84=0,0,0"

```

Source

shapefile obtained from : http://thematicmapping.org/downloads/world_borders.php Text from the site : "The original shapefile (world_borders.zip, 3.2 MB) was downloaded from the Mapping Hacks website: <http://www.mappinghacks.com/data/>

The dataset is available under a Creative Commons Attribution-Share Alike License. If you use this dataset, please provide a link to this website."

imported to R and cleaned by Roger Bivand

Examples

```
data(wrld_simpl)
```

```
wrld_simpl_lessIslands
```

World map simple with small islands removed to make smaller

Description

A 'SpatialPolygonsDataFrame' [package "sp"] object containing a simplified world map. Polygons are attributed with country codes.

Usage

```
data(wrld_simpl_lessIslands)
```

Format

The format is:

```

Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
..@ data      : 'data.frame': 246 obs. of  11 variables:
.. ..$ FIPS    : Factor w/ 244 levels "", "AC", "AE", "AF", ...
.. ..$ ISO2    : Factor w/ 246 levels "AD", "AE", "AF", ...
.. ..$ ISO3    : Factor w/ 246 levels "ABW", "AFG", "AGO", ...
.. ..$ UN      : int [1:246] 28 12 31 8 51 24 16 32 36 48 ...
.. ..$ NAME    : Factor w/ 246 levels "Afghanistan", ...: 9 3 15 2 11 ...
.. ..$ AREA    : int [1:246] 44 238174 8260 2740 2820 124670 20 273669 768230 71 ...
.. ..$ POP2005 : int [1:246] 83039 32854159 8352021 3153731 3017661 ...
.. ..$ REGION  : int [1:246] 19 2 142 150 142 2 9 19 9 142 ...
.. ..$ SUBREGION: int [1:246] 29 15 145 39 145 17 61 5 53 145 ...
.. ..$ LON     : num [1:246] -61.78  2.63 47.40 20.07 44.56 ...
.. ..$ LAT     : num [1:246] 17.1 28.2 40.4 41.1 40.5 ...
..@ polygons  :List of 246
.. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
...
..@ plotOrder : int [1:246] 145 175 24 209 30 21 9 71 98 8 ...
..@ bbox      : num [1:2, 1:2] -180 -90 180 90
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:2] "r1" "r2"
.. .. ..$ : chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
.. .. ..@ projargs: chr " +proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs +towgs84=0,0,0"

```

Source

shapefile obtained from : http://thematicmapping.org/downloads/world_borders.php Text from the site : "The original shapefile (world_borders.zip, 3.2 MB) was downloaded from the Mapping Hacks website: <http://www.mappinghacks.com/data/>

The dataset is available under a Creative Commons Attribution-Share Alike License. If you use this dataset, please provide a link to this website."

imported to R and cleaned by Roger Bivand small islands removed by Andy South to reduce size of map and increase speed of printing using the following code : #based on a posting to r-sig-geo by Hans-Jorg Bibiko on 29.01.2009 #see <http://www.mail-archive.com/r-sig-geo@stat.math.ethz.ch/msg04341.html>

```
data(wrlld_simpl) minIslandSizeForInclusionInMap <- 1 #prob in degrees
```

```
#for each country in the map for (i in (1:length(wrlld_simpl@polygons)))
```

```
#count the number of polygons in the country polyCnt <- length(wrlld_simpl@polygons[[i]]@Polygons)
```

```
if (polyCnt > 2)
```

```
#just select those polygons with area > 0.1 pO <- sapply(wrlld_simpl@polygons[[i]]@Polygons,
function(x) slot(x,'area') > minIslandSizeForInclusionInMap )
```

```
#get new plot order new_pO <- pO[ wrlld_simpl@polygons[[i]]@plotOrder ]
```

```
#i think for each country this then just doesn't plot all of the polygons #(despite the fact that the co-ords are perhaps still in there ?) wrlld_simpl@polygons[[i]]@plotOrder <- wrlld_simpl@polygons[[i]]@plotOrder[
new_pO ]
```

```
plot(wrlld_simpl) wrlld_simpl_lessIslands <- wrlld_simpl
```

Examples

```
data(wrlld_simpl_lessIslands)
```

```
wrlld_simpl_Mollweide  World map simple, projected using Mollweide equal area
```

Description

A 'SpatialPolygonsDataFrame' [package "sp"] object containing a simplified world map projected using Mollweide equal area. Polygons are attributed with country codes.

Usage

```
data(wrlld_simpl_Mollweide)
```

Format

The format is:

```
Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
..@ data      : 'data.frame': 246 obs. of  11 variables:
.. ..$ FIPS   : Factor w/ 244 levels "", "AC", "AE", "AF", ..
.. ..$ ISO2   : Factor w/ 246 levels "AD", "AE", "AF", ..
.. ..$ ISO3   : Factor w/ 246 levels "ABW", "AFG", "AGO", ..
.. ..$ UN     : int [1:246] 28 12 31 8 51 24 16 32 36 48 ...
.. ..$ NAME   : Factor w/ 246 levels "Afghanistan", ..
.. ..$ AREA   : int [1:246] 44 238174 8260 2740 2820 ...
.. ..$ POP2005 : int [1:246] 83039 32854159 8352021 3153731 3017661 ...
.. ..$ REGION : int [1:246] 19 2 142 150 142 2 9 19 9 142 ...
.. ..$ SUBREGION: int [1:246] 29 15 145 39 145 17 61 5 53 145 ...
.. ..$ LON    : num [1:246] -61.78  2.63  47.40  20.07  44.56 ...
.. ..$ LAT    : num [1:246] 17.1 28.2 40.4 41.1 40.5 ...
..@ polygons  :List of 246
.. ..$ :Formal class 'Polygons' [package "sp"] with 5 slots
...

..@ plotOrder : int [1:246] 145 175 24 209 30 21 9 71 98 8 ...
..@ bbox      : num [1:2, 1:2] -17702743 -9020048 17840888 8747102
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:2] "r1" "r2"
.. .. ..$ : chr [1:2] "min" "max"
..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slots
.. .. ..@ projargs: chr " +proj=moll +ellps=WGS84"
```

Source

shapefile obtained from : http://thematicmapping.org/downloads/world_borders.php Text from the site : "The original shapefile (world_borders.zip, 3.2 MB) was downloaded from the Mapping Hacks website: <http://www.mappinghacks.com/data/>

The dataset is available under a Creative Commons Attribution-Share Alike License. If you use this dataset, please provide a link to this website."

imported to R, cleaned and projected by Roger Bivand

Examples

```
data(wrld_simpl_Mollweide)
```

Index

*Topic **aplot**

- addMapLegend, 3
- addMapLegendBoxes, 6
- barplotCountryData, 9
- mapBars, 25
- mapBubbles, 27
- mapCountryData, 32
- mapHalfDegreeGridToCountries, 38
- mapPies, 40
- mapPolys, 42
- rworldmapExamples, 47

*Topic **datasets**

- countryExData, 13
- countryRegions, 16
- gridCountriesNumeric, 18
- gridExData, 19
- wrld_coarse, 48
- wrld_simpl, 49
- wrld_simpl_lessIslands, 50
- wrld_simpl_Mollweide, 52

*Topic **device**

- mapDevice, 34

*Topic **dplot**

- aggregateHalfDegreeGridToCountries, 8
- identifyCountries, 20
- joinCountryData2Map, 22
- joinData2Map, 23
- rwmGetClassBreaks, 44
- rwmGetColours, 45
- setMapExtents, 47

*Topic **hplot**

- mapByRegion, 30
- mapGriddedData, 36

*Topic **manip**

- country2Region, 11

*Topic **misc**

- getMap, 17
- rwmNewMapPlot, 46

*Topic **package**

- rworldmap-package, 2

- addMapLegend, 3, 10, 31, 33, 37, 39, 43
- addMapLegendBoxes, 6, 10, 33, 43
- aggregateHalfDegreeGridToCountries, 8, 39

- barplotCountryData, 2, 9

- country2Region, 11, 17, 31
- countryExData, 13
- countryRegions, 16

- getMap, 17, 22–24
- gridCountriesNumeric, 18
- gridExData, 19

- identifyCountries, 20

- joinCountryData2Map, 2, 22
- joinData2Map, 2, 23

- mapBars, 2, 25
- mapBubbles, 2, 27
- mapByRegion, 12, 17, 30
- mapCountryData, 2, 9, 22, 23, 31, 32, 48
- mapDevice, 34
- mapGriddedData, 2, 36, 48
- mapHalfDegreeGridToCountries, 9, 38
- mapPies, 2, 40
- mapPolys, 2, 23, 24, 42

- rwmGetClassBreaks, 44
- rwmGetColours, 45
- rwmNewMapPlot, 46
- rworldmap (rworldmap-package), 2
- rworldmap-package, 2
- rworldmapExamples, 47

- setMapExtents, 47

wrld_coarse, [48](#)
wrld_simpl, [49](#)
wrld_simpl_lessIslands, [50](#)
wrld_simpl_Mollweide, [52](#)