Package ‘rworldmap’
February 20, 2015

Type Package
Title Mapping global data, vector and raster.
Version 1.3-1
Date 2013-08-22
Author Andy South
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
Imports maptools, fields
Suggests rgdal, rworldxtra, RColorBrewer, classInt, cshapes, ncdf, raster
URL http://groups.google.com/group/rworldmap,
https://r-forge.r-project.org/R/?group_id=1497,
http://andysouth.co.uk/
NeedsCompilation no
Repository CRAN
Date/Publication 2013-12-12 09:56:41

R topics documented:

rworldmap-package .................................................. 2
addMapLegend ...................................................... 4
addMapLegendBoxes ................................................. 6
aggregateHalfDegreeGridToCountries .............................. 8
barplotCountryData ............................................... 9
coastsCoarse ...................................................... 12
countriesCoarse .................................................. 12
countriesCoarseLessIslands ................................... 13
countriesLow ...................................................... 14
rworldmap-package

For mapping global data.

Description

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

Details

Package: rworldmap
Type: Package
Version: 1.3-1
Date: 2013-08-22
License: GPL (>= 2)
Version 1.01 October 2012 newly used updated Natural Earth Data for country boundaries e.g. `countriesLow`. Higher resolution boundaries are provided in a companion package `rworldxtra`. It also added country synonym matching in `joinCountryData2Map`.

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 : https://r-forge.r-project.org/projects/rworldmap/

Discussion group : http://groups.google.com/group/rworldmap

**Examples**

```r
# 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 separately 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.
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"
)
# 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"
)
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
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(gridCountriesDegreesHalf).

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='GEO3major'
  , 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 scheme
mapParams <- mapCountryData(nameColumnToPlot='GEO3major'
  , 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)))
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')
Usage

```r
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,
    cex = 0.7,
    ...)
```

Arguments

dF a dataframe containing at least one column with numeric data and one with country names or other labels

nameColumnToPlot name of column containing the data you want to plot

nameCountryColumn name of column containing country names (or other labels to be used in plot)

numPanels the number of layout panels in the plot

scaleSameInPanels 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

main title for the plot

numCats number of categories to put the data in, may be modified if this number is incompatible with the catMethod chosen

catMethod method for categorisation of data "pretty", "fixedWidth", "diverging", "logFixedWidth", "quantiles", "categorical", or a numeric vector defining breaks

colourPalette a 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", "negpos9"

addLegend NOT YET WORKING whether to add a legend or not, TRUE/FALSE

toPDF whether to output the plot to a pdf rather than the screen, TRUE/FALSE

outFile output filename if toPDF=TRUE
decreasing logical. Should the sort order be increasing or decreasing?
nan.last 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.
cex sizing of labels, default = 0.7
... other arguments to pass to barplot

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"
)
Description
A spatial lines dataframe containing world coasts at a coarse resolution.

Usage
data(coastsCoarse)

Format
The format is: Formal class 'SpatialLinesDataFrame' [package "sp"] with 4 slots

Details
Used in mapGriddedData(addBorders=’coasts’). This is the 1:110m coasts data from Natural Earth version 1.3.0.

Source
http://www.naturalearthdata.com/downloads/110m-physical-vectors/

Examples
data(coastsCoarse)
mapGriddedData(addBorders=’coasts’)
plot(coastsCoarse, add=TRUE, col=’blue’)

countriesCoarse

Description
A coarse resolution world map, a vector map of 244 country boundaries, suitable for global maps.

Usage
data(countriesCoarse)

Format
The format is: Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
Details

Derived from version 1.4.0 of Natural Earth data 1:110 m data. Added in countries from the higher resolution data.

The different country boundaries in rworldmap are processed from Natural Earth Data as follows:
All:
- rename any non-ASCII country names that cause R trouble
- rename Curacao which is particularly troublesome!
- check polygon geometries using checkPolygonsHoles
- set projections, e.g. proj4string(countriesCoarse) <- CRS("+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs")
- set polygon IDs to country names (from ADMIN field)
- replace missing ISO3 codes (6 in this version) with ADM0_A3
- check for duplicate ISO3 codes (2 in this version)
- set ISO3 for Gaza to Gaza and ‘Ashmore and Cartier Islands’ to Ashm
- replace POP_EST of -99 with NA
- join on countryRegions data

countriesCoarseLessIslands : ne_110
countriesCoarse : ne_110 plus extra countries from ne_50 plus Tuvalu from ne_10
countriesLow : ne_50 plus Tuvalu from ne_10
countriesHigh (in package rworldxtra) : ne_10

Source

http://www.naturalearthdata.com/downloads/110m-cultural-vectors/110m-admin-0-countries/

Examples

data(countriesCoarse)

countriesCoarseLessIslands

A coarse resolution world map, a vector map of 177 country boundaries, suitable for global maps

Description

A `SpatialPolygonsDataFrame` [package "sp"] object containing a simplified world map. Polygons are attributed with country codes. 177 countries. Based on Natural Earth data.

Usage

data(countriesCoarseLessIslands)
Format

The format is: Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots

Details

Derived from version 1.4.0 of Natural Earth data 1:110 m data.
The different country boundaries in rworldmap are processed from Natural Earth Data as follows:
All:
~ rename any non-ASCII country names that cause R trouble
~ rename Curacao which is particularly troublesome!
~ check polygon geometries using checkPolygonsHoles
~ set projections, e.g. proj4string(countriesCoarse) <- CRS("+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs")
~ set polygon IDs to country names (from ADMIN field)
~ copy ISO_A3 to ISO3
~ replace missing ISO3 codes (6 in this version) with ADM0_A3
~ check for duplicate ISO3 codes (2 in this version)
~ set ISO3 for Gaza to Gaza and ‘Ashmore and Cartier Islands’ to Ashm
~ replace POP_EST of -99 with NA
~ join on countryRegions data

countriesCoarseLessIslands : ne_110
countriesCoarse : ne_110 plus extra countries from ne_50 plus Tuvalu from ne_10
countriesLow : ne_50 plus Tuvalu from ne_10
countriesHigh (in package rworldxtra) : ne_10

Source

http://www.naturalearthdata.com/downloads/110m-cultural-vectors/110m-admin-0-countries/

Examples

data(countriesCoarseLessIslands)

countriesLow

a low resolution world map, a vector map of 244 country boundaries, suitable for zooming in on regions or large global maps

Description

A 'SpatialPolygonsDataFrame' [package "sp"] object containing country boundaries derived from Natural Earth data. Polygons are attributed with country codes.

Usage

data(countriesLow)
Format

The format is: Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots

Details

Derived from version 1.4.0 of Natural Earth data 1:50 m data.

The different country boundaries in rworldmap are processed from Natural Earth Data as follows:

- rename any non-ASCII country names that cause R trouble
- rename Curacao which is particularly troublesome!
- check polygon geometries using checkPolygonsHoles
- set projections, e.g. proj4string(countriesCoarse) <- CRS("+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs")
- set polygon IDs to country names (from ADMIN field)
- copy ISO_A3 to ISO3
- replace missing ISO3 codes (6 in this version) with ADM0_A3
- check for duplicate ISO3 codes (2 in this version)
- set ISO3 for Gaza to Gaza and 'Ashmore and Cartier Islands' to Ashm
- replace POP_EST of -99 with NA
- join on countryRegions data

countriesCoarseLessIslands : ne_110
countriesCoarse : ne_110 plus extra countries from ne_50 plus Tuvalu from ne_10
countriesLow : ne_50 plus Tuvalu from ne_10
countriesHigh (in package rworldxtra) : ne_10

Source

http://www.naturalearthdata.com/downloads/50m-cultural-vectors/

Examples

data(countriesLow)

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'
 )
countryExData

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  Example dataset for country level data (2008 Environmental Performance Index)

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
S02_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
EFCON_pt a numeric vector
AZE_pt a numeric vector
MPAAEZ_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
EFCON a numeric vector
AZE a numeric vector
MPAAEZ 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


Examples

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

---

countryRegions Regional Classification Table

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 the following variables.

ISO3 ISO 3 letter country code
ADMIN country name
REGION 7 region continent classification
continent 6 continents classification
GEO3major Global Environment Outlook GEO3 major region names
GEO3 Global Environment Outlook GEO3 major region names
IMAGE24 Image24 region names
countrySynonyms

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
LDC  UN Least Developed Countries
SID  UN Small Island Developing states
LLDC  UN Landlocked Developing Countries

Details

Joined onto vector country maps. 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)

countrySynonyms  Synonyms of country names for each ISO 3 letter country code to enable conversion.

Description

contains a variable number of synonyms (mostly English language) for each country

Usage

data(countrySynonyms)
Format

A data frame with 281 observations on the following 10 variables.

- **id**: a numeric vector
- **ISO3**: ISO 3 letter country code
- **name1**: country name - most common
- **name2**: country name - alternative
- **name3**: country name - alternative
- **name4**: country name - alternative
- **name5**: country name - alternative
- **name6**: country name - alternative
- **name7**: country name - alternative
- **name8**: country name - alternative

Details

This is used by `joinCountryData2Map()` when country names are used as the joinCode. Note that using ISO codes is preferable if they are available.

Source

This was derived and used with permission from the Perl Locale package. `Locale::Codes::Country_Codes`. Thanks to Sullivan Beck for pulling this together. Data sources are acknowledged here:

http://search.cpan.org/~sbeck/Locale-Codes-3.23/lib/Locale/Codes/Country.pod

Examples

```r
data(countrySynonyms)
```

---

**getMap**

* A simple way to access maps stored in the package.

Description

A simple way to access maps stored in the package.

Usage

```r
getMap(resolution = "coarse", projection=NA)
```
Arguments

resolution options "coarse","low","less islands","li","high". For "high" you need to install the package rworldxtra

projection DEPRECATED OCTOBER 2012 to reproject maps see spTransform in rgdal

Value

A SpatialPolygonsDataFrame object.

Author(s)

Barry Rowlingson & Andy South

Examples

plot(getMap())

gridCountriesDegreesHalf

A global 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(gridCountriesDegreesHalf)

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 ...
..@ 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 -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

gridCountriesDegreesHalf

A global 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(gridCountriesDegreesHalf)

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 ...
..@ 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 -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
Details

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

Source

created from getMap(resolution='low')

Examples

data(gridCountriesDegreesHalf)

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 ...
  ..@ grid : Formal class 'GridTopology' [package "sp"] with 3 slots
              ..@ cellcentre.offset: num [1:2] -179.8 -89.8
              ..@ cells.size : 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
                   89.8
  ..@ bbox : num [1:2, 1:2] -180 -90 180 90
            ..$ : chr [1:2] "coords.x1" "coords.x2"
gridExData

... - 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, SpatialGridDataFrame), used by the function aggregateHalfDegreeGridToCountries()

Source

IIASA

References

http://www.iiasa.ac.at/Research/GGI/DB/

Examples

data(gridCountriesNumeric)

data(gridExData)
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

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)

---

identifyCountries a function that will print country name and attribute values when a
user clicks on the map

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

dF data frame or SpatialPolygonsData Frame
nameCountryColumn 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)
nameX name of column containing the X variable (longitude), not needed if dF is a SpatialPolygonsData Frame
nameY name of column containing the Y variable (latitude), not needed if dF is a SpatialPolygons Data Frame
nameColumnToPlot name of an attribute column in the data frame the value of which will be appended to the country name when it is printed
plotSelected if set to TRUE a blue outline will be printed around the countries selected when the selection process is finished
... other parameters that can be passed to identify()

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() labelCountries
isoToName

Returns the country name corresponding to the passed iso code (3 letter, 2 letter or numeric).

Description

Searches getMap()@data to find the iso code. By default it returns the string in the ADMIN column. By modifying nameColumn you can also get it to return values from any other columns in getMap()@data - see the examples. Thus it can also be used to convert between ISO codes.

Usage

isoToName( iso = "", lookup = getMap()@data, nameColumn = 'ADMIN' )

Arguments

iso
  iso code to convert to a country name
lookup
  the dataframe containing iso codes and country names
nameColumn
  which column to get the name from, see examples

Details

You could optionally provide a dataframe containing alternate iso conversions using lookup=. The passed dataframe would need to contain at least one of the following columns containing 2 letter, 3 letter or numeric iso codes respectively : ISO_A2, ISO_A3, ISO_N3.

Value

The country name (or other field) associated with the ISO code passed. NA is returned if no matching code is found.

Author(s)

Andy South
Examples

isoToName('gb')
isoToName('gbr')
isoToName('gb') #generates a warning and returns NA
#beware that using nameColumn may be vulnerable to future changes
#in column names in Natural Earth data
isoToName('gb',nameColumn='ABBREV') #returns abbreviation
isoToName('gb',nameColumn='ISO_A3') #returns iso3 for this iso2
isoToName('gbr',nameColumn='continent') #returns continent for this iso3

joinCountryData2Map

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
, mapResolution="coarse"
, projection=NA #DEPRECATED
, 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 DEPRECATED JUNE 2012
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, using 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`. NEW to version 1.01 Oct 2012 : for `joinCode`='NAME' alternative country names are matched using `countrySynonyms`.

The projection argument has now been deprecated, you can project maps using package rgdal as shown below and in the FAQ.

```r
library(rgdal)
#first get countries excluding Antarctica which crashes spTransform
sPDF <- getMap()[-which(getMap()$ADMIN=="Antarctica",]
#transform to robin for the Robinson projection
sPDF <- spTransform(sPDF, CRS=CRS("+proj=robin +ellps=WGS84"))
mapCountryData( sPDF, nameColumnToPlot="REGION")
```

Value

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

Author(s)

andy south

See Also

`mapCountryData`, `getMap`

Examples

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

sPDF <- joinCountryData2Map(countryExData
  , joinCode = "ISO3"
  , nameJoinColumn = "ISO3V10"
)
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**

```r
joinData2Map(df,
  , nameMap = ""
  , nameJoinIDMap = "ISO3"
  , nameJoinColumnData = "ISO3V10"
  , nameNameColumnData = "Country"
  , suggestForFailedCodes = FALSE
  , projection=NA #DEPRECATED JUNE 2012
  , mapResolution="coarse"
  , 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`: DEPRECATED JUNE 2012
- `mapResolution`: resolution of the borders in the internal map: options 'coarse','low', 'less islands'
- `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. 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

```r
## 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/drpcmpstatevl1.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')
```

labelCountries to print country labels on a world map
**Description**

Given no arguments it will print country names stored in the ’NAME’ column of `getMap` onto an existing map at the centroids of each country polygon, stored in the ’LAT’ and ’LON’ columns. 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`). First you need to create a map plot, for example using `mapCountryData` or `mapBubbles`.

**Usage**

```r
labelCountries(dF = "",
    nameCountryColumn = "NAME",
    nameX = "LON",
    nameY = "LAT",
    nameColumnToPlot = "",
    col = 'grey',
    cex = 0.8,
    ...)```

**Arguments**

- `dF`: dataframe or `SpatialPolygonsDataFrame`
- `nameCountryColumn`: name of column containing country names to be printed on the map (could also be set to any other column in the dataframe)
- `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`
- `nameColumnToPlot`: name of an attribute column in the data frame the value of which will be appended to the country names
- `col`: colour for labels, default ’grey’, can be e.g. `rgb(1,1,0,alpha=0.5)`
- `cex`: sizing of labels, default = 0.8
- `...`: other parameters that can be passed to `text()`, e.g. `pos=4` to right, (1=below, 2=left, 3=above)

**Value**

nothing

**Author(s)**

andy south

**See Also**

identifyCountries
Examples

mapCountryData()
labelCountries()

labelCountries(nameColumnToPlot = "POP_EST")

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'
  , barRelative = TRUE

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

  , symbolSize = 1
  , maxZVal=NA

  , xlim=NA
  , ylim=NA
  , mapRegion = "world"
  , borderCol = "grey"
  , oceanCol=NA
  , landCol=NA
  , add=FALSE
  , main=''
  , lwd=0.5
  , lwdSymbols=1
  , ...)
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'
barRelative default is TRUE, each variable (column) is scaled to it's maximum value
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
lwd   line width for country borders
lwdSymbols line width for symbols
... 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
sPDF <- getMap()

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

# mapBars( df, nameX = "LON", nameY = "LAT", nameZs = c('POP_EST', 'POP_EST'))

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

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

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

mapBars( sPDF,
          , nameZs = c('POP_EST', 'GDP_MD_EST')
          , mapRegion = 'africa'
          , symbolSize = 4)

# this does work too
# mapBars( df, nameX = "LON", nameY = "LAT"
#          , nameZs = c('POP_EST', 'GDP_MD_EST')
#          , mapRegion = 'africa'
#          , symbolSize = 4)

mapBubbles function to produce bubble plots on a map, size and colour determined
by attribute data

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 = ""
mapBubbles

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

, main = nameZSize

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

, xlim = NA
, ylim = NA
, 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
, lwd = 0.5
, lwdSymbols = 1
, ...

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
mapBubbles

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 incompatible with the catMethod chosen
catMethod  method for categorisation of data "pretty", "fixedWidth", "diverging", "logFixedWidth", "quantiles", "categorical", or a numeric vector defining breaks
colourPalette  a 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", "negpos9"
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
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 horizontaly 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
lwd  line width for country borders
lwdSymbols  line width for symbols
...  any extra arguments to points()

Details

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

Value
currently doesn’t return anything

Author(s)
andy south

Examples

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

mapBubbles(df=getMap(), nameZSize="POP_EST", nameZColour="GEO3")

# change colour
mapBubbles(df=getMap(), nameZSize="POP_EST", nameZColour="GEO3"
   , colourPalette='rainbow', oceanCol='lightblue', landCol='wheat')

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

mapBubbles(sPDF, nameZSize="POP_EST", 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 = ''
   , lwd = 0.5
   , ...)

Arguments

infile a data frame
nameDataColumn The name of a column of inFile. This is data is aggregated by FUN
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.
regionType Must be one of: "GEO3", "GEO3major", "IMAGE24", "GLOCAF", "Stern", "SRES", "SRESmajor", "GBD", "AVOIDname"
FUN A function to apply to each region
na.rm Only used for certain values of FUN. See details section below.
mapTitle a title to be printed above the map
lwd line width for country borders
... 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", "mad" 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

Map country-level data.

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 = NA,
  ylim = NA,
  mapRegion = "world",
  catMethod = "quantiles",
  colourPalette = "heat",
  addLegend = TRUE,
  borderCol = 'grey',
  mapTitle = 'columnName',
  oceanCol = NA,
  aspect = 1,
  missingCountryCol = NA,
  add = FALSE,
  nameColumnToHatch = "",
  lwd = 0.5)

Arguments

mapToPlot a spatial polygons dataframe from joinCountryData2Map() containing country polygons and 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 method for categorisation of data:
  1. "categorical" - each unique value is treated as a separate category
2. for numeric data: "pretty", "fixedWidth", "diverging", "logFixedWidth", "quantiles"
3. a numeric vector defining breaks e.g. c(0:5), note that a value of 2 goes into 1-2 not 2-3, uses cut(include.lowest=TRUE)

**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", "negpos9"

**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
**lwd** line width for country borders

**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.

**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.
mapCountryData

Author(s)
andy south

See Also
classInt, RColorBrewer

Examples

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

# user defined map colour scheme for categorical data
mapParams <- mapCountryData(nameColumnToPlot="GEO3major"
                             , catMethod='categorical'
                             , addLegend='FALSE'
                             , colourPalette=c('white','green','red','yellow','blue','black')
                         )
# changing legend text
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

```r
mapDevice(
    device = "dev.new",
    rows = 1,
    columns = 1,
    plotOrder="rows",
    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.
- **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
mapGriddedData

Value

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

See Also

mapCountryData, mapGridAscii

Examples

```r
## 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

```r
mapGriddedData(  
  dataset = "",
  nameColumnToPlot = "",
  numCats = 5,
  catMethod = "quantiles",
  colourPalette = "heat",
  xlim = c(-180,180),
  ylim = c(-80,90),
  mapRegion = "world",
  addLegend = TRUE,
  addBorders = 'coarse',
  borderCol = 'grey'
)
```
Arguments

dataset gridded data either as a:
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", "logFixedWidth", "quantiles", "categorical", or a numeric vector defining breaks
colourPalette a 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", "negpos9"

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
lwd line width for country borders

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"
mapHalfDegreeGridToCountries

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"
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 deprecated june 2012
mapResolution options low, medium, only for projection=’none’ initially
numCats number of categories, may be overridden 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
lwd line width for country borders

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.
mapPies

Author(s)

andy south

See Also

aggregateHalfDegreeGridToCountries

Examples

data(gridExData, envir=environment(), package="rworldmap")
gridExData <- get("gridExData")
mapHalfDegreeGridToCountries(gridExData)

# different aggregate option
mapHalfDegreeGridToCountries(gridExData, aggregateOption="mean")

mapPies

function to produce pie charts on a map

Description

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

Usage

mapPies(dF = ""
  , nameX = "LON"
  , nameY = "LAT"
  , nameZs=c(names(dF)[3], names(dF)[4])
  , zColours=c(1:length(nameZs))
  , ratio = 1
  , addCatLegend = TRUE
  , addSizeLegend = TRUE
  , symbolSize = 1
  , maxZVal=NA
  , xlim=NA
  , ylim=NA
  , mapRegion = "world"
  , borderCol = "grey"
  , oceanCol=NA
  , landCol=NA
  , add=FALSE
  , main=""
  , lwd = 0.5
  , ...)
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 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
lwd | line width for country borders
... | 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

```r
# 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('POP_EST','POP_EST','POP_EST','POP_EST'), mapRegion='africa' )
```

mapPolys

**Map polygon data.**

Description

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

Usage

```r
mapPolys( mapToPlot = "",
          nameColumnToPlot = "",
          numCats = 7,
          xlim=NA,
          ylim=NA,
          mapRegion = "world",
          catMethod="quantiles",
          colourPalette = "heat",
          addLegend=TRUE,
          borderCol = 'grey',
          mapTitle = 'columnName',
          oceanCol=NA,
          aspect=1,
          missingCountryCol=NA,
          add=FALSE,
          lwd=0.5 )
```
mapPolys

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()

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:

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", "negpos9"

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

lwd

line width for country borders

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.
mapPolys

Author(s)

andy south

See Also

joinData2Map, classInt, RColorBrewer

Examples

```r
## 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/drpcmpstatevl.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"
##     , nameJoinColumnName = "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:
```
rwmCheckAndLoadInput

internal function to check and load input data to mapping functions

Description

Internal function checking and loading dFs or sPDFs to mapCountryData, mapPolys, mapPies, mapBubbles, mapBars.

Usage

rwmCheckAndLoadInput(
  inputData = "",
  inputNeeded = "sPDF",
  callingFunction = ""
)

Arguments

inputData a dF, sPDF or "", for latter an internal example data is used
inputNeeded "sPDF", "sPDF or dF", "dF"
callingFunction optional : name of the calling function

Details

a rworldmap internal function, unlikely to be of use to users

Value

invisibly returns a dF or sPDF

Author(s)

andy south
Description

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

Usage

\[
\text{rwmGetClassBreaks}(\text{dataColumn}, \text{catMethod}, \text{numCats}, \text{verbose=TRUE}, \text{midpoint=0})
\]

Arguments

- **dataColumn**: the data vector to be classified, must be numeric
- **catMethod**: 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
- **numCats**: number of categories to put the data in, may be overridden if not possible under some classification methods
- **verbose**: whether to print information messages to console TRUE/FALSE
- **midpoint**: the midpoint to use if catMethod='diverging', 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 \hspace{1cm} \textit{Internal function to choose map colours for classified data}

\textbf{Description}

Internal function to choose map colours for classified data.

\textbf{Usage}

\begin{verbatim}
rwmGetColours(colourPalette, numColours)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{colourPalette} \hspace{1cm} \texttt{string} describing the colour palette to use, choice of:
    \begin{enumerate}
      \item \texttt{"palette"} for the current palette
      \item \texttt{a} \hspace{0.5cm} vector of valid colours, e.g. \texttt{=c\{red\,white\,blue\}} or output from \texttt{RColourBrewer}
      \item \texttt{one} \hspace{0.5cm} of "heat", "diverging", "white2Black", "black2White", "topo", "rainbow", "terrain", "negpos8", "negpos9"
    \end{enumerate}
  \item \texttt{numColours} \hspace{1cm} \texttt{the number of colour categories desired}
\end{itemize}

\textbf{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.

\textbf{Value}

A vector specifying a number of colours.

\textbf{Author(s)}

andy south and matthew staines

\textbf{See Also}

RColorBrewer

\textbf{Examples}

\begin{verbatim}
#rwmGetColours(colourPalette='topo', numColours=10)
\end{verbatim}
Internal function for getting the ISO3 country code for a country name synonym.

Description

Searches countrySynonyms to get the ISO3 code. If the name is not found NA is returned. Allows joining of imperfect names to other country data in joinCountryData2Map(joinCode='NAME')

Usage

rwmGetISO3(oddName)

Arguments

oddName country name that user wishes to find code for

Value

the ISO3 code (3 letters) corresponding to the country name passed, or NA if one is not found

Author(s)

Andy South

References

This was derived and used with permission from the Perl Locale package. Locale::Codes::Country::Codes.
Thanks to Sullivan Beck for pulling this together.
Data sources are acknowledged here:
http://search.cpan.org/~sbeck/Locale-Codes-3.23/lib/Locale/Codes/Country.pod

Examples

rwmGetISO3("vietnam")
**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**

```r
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()`.

**Value**

A dataframe containing `xlim` and `ylim`.

**Author(s)**

andy south

---

**rworldmapExamples**  
*Example code for plot creation*

**Description**

Example code to demonstrate creation of a series of plots.

**Usage**

```r
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' )
Index

*Topic **aplot**
  - addMapLegend, 4
  - addMapLegendBoxes, 6
  - barplotCountryData, 9
  - mapBars, 34
  - mapBubbles, 36
  - mapCountryData, 41
  - mapHalfDegreeGridToCountries, 47
  - mapPies, 49
  - mapPolys, 51
  - rwmCheckAndLoadInput, 54
  - rworldmapExamples, 58

*Topic **datasets**
  - coastsCoarse, 12
  - countriesCoarse, 12
  - countriesCoarseLessIslands, 13
  - countriesLow, 14
  - countryExData, 17
  - countryRegions, 20
  - countrySynonyms, 21
  - gridCountriesDegreesHalf, 23
  - gridCountriesNumeric, 24
  - gridExData, 25

*Topic **device**
  - mapDevice, 44

*Topic **dplot**
  - aggregateHalfDegreeGridToCountries, 8
  - identifyCountries, 26
  - joinCountryData2Map, 29
  - joinData2Map, 30
  - labelCountries, 32
  - rwmGetClassBreaks, 55
  - rwmGetColours, 56
  - setMapExtents, 59

*Topic **hplot**
  - mapByRegion, 39
  - mapGriddedData, 45

*Topic **manip**
  - country2Region, 15
  - isoToName, 28
  - rwmGetISO3, 57

*Topic **misc**
  - getMap, 22
  - rwmNewMapPlot, 58

*Topic **package**
  - rworldmap-package, 2

addMapLegend, 4, 11, 40, 42, 47, 48, 52
addMapLegendBoxes, 6, 11, 42, 52
aggregateHalfDegreeGridToCountries, 8, 49
barplotCountryData, 3, 9
coastsCoarse, 12
countriesCoarse, 12
countriesCoarseLessIslands, 13
countriesLow, 3, 14
country2Region, 15, 21, 40
countryExData, 17
countryRegions, 20
countrySynonyms, 21, 30
getMap, 22, 30–33
gridCountriesDegreesHalf, 23
gridCountriesNumeric, 24
gridExData, 25
identifyCountries, 26, 33
isoToName, 28
joinCountryData2Map, 3, 29, 41
joinData2Map, 3, 30
labelCountries, 27, 32
mapBars, 3, 34, 54
mapBubbles, 3, 33, 36, 54
mapByRegion, 16, 21, 39
INDEX

mapCountryData, 3, 9, 29, 30, 33, 40, 41, 54, 59
mapDevice, 44
mapGriddedData, 3, 45, 59
mapHalfDegreeGridToCountries, 9, 47
mapPies, 3, 49, 54
mapPolys, 3, 30–32, 51, 54

rwmCheckAndLoadInput, 54
rwmGetClassBreaks, 55
rwmGetColours, 56
rwmGetISO3, 57
rwmNewMapPlot, 58
rworldmap (rworldmap-package), 2
rworldmap-package, 2
rworldmapExamples, 58

setMapExtents, 59