Package 'GADMTools'

November 20, 2018

Type Package

Title Easy Use of 'GADM' Shapefiles

Version 3.0-1

Date 2018-11-15

Description Manipulate, assemble, export <http://www.gadm.org> shapefiles. Create choropleth, heatmaps, dots plot, proportional dots and more.

License GPL-3

Depends sp, ggplot2, dplyr, classInt, rgdal

Imports RColorBrewer, maptools, stringr, raster, rosm, lattice, jsonlite, gridExtra, rgeos, ggmap

Suggests knitr, rmarkdown

Encoding UTF-8

VignetteBuilder knitr

NeedsCompilation no

Author Jean Pierre Decorps [aut, cre], Morgane Vallee [ctb]

Maintainer Jean Pierre Decorps <jp.decorps@epiconcept.fr>

Repository CRAN

Date/Publication 2018-11-20 05:50:12 UTC

R topics documented:

`GADMTools-package` *Easy use of GADM shapefiles*

Description

See; https://gadm.org/

GADM is a spatial database of the world's adminstrative boundaries for use in **GIS** and similar software. Administrative areas in this database are countries and lower level subdivisions such as provinces, departments, cantons, etc.

With **GADMTools**, a wrapper for **GADM** shapefiles, you can easily manipulate, assemble, and create subsets of these objects.

Details

Package:	GADMTools
Type:	Package
Version:	3.0-1
Date:	2018-11-15
License:	GPL-3

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

Maintainer: Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

~~ Literature or other references for background information ~~

See Also

~~ Optional links to other man pages, e.g. ~~

Examples

```
## NOTHING
```

`choropleth` *Draw a choropleth on selected regions*

Description

Drawing a choropleth (colored regions based on data values) with GADMTools is straightforward. You just have to select your shape(s) file(s) with *gadm.loadcountries*, load your data from a csv file for example, and call the choropleth function with the right arguments.

Usage

```
choropleth (x, data, value=NULL, breaks = NULL, steps = 5, adm.join=NULL,
            legend = NULL, labels = NULL, palette=NULL,
            title="")
```

Arguments

`x`	**Object** GADMWrapper
`data`	**data.frame** - data to plot
`value`	**String** - the name of the column in the data.frame we want to plot (eg: an incidence in epidemiology studies)
`breaks`	**Vector** of breaks values or a **Sring** name of a function from *classIntervals* (one of "sd", "equal", "pretty", "quantile", "kmeans", "hclust", "bclust", "fisher", or "jenks")
`steps`	**Integer** - number of breaks. Default = 5. If *breaks* is **NOT NULL** this value is used internally with cut().
`adm.join`	**String** - the name in GADM spdf dataset which will be joined with a column of the data.
`legend`	**String** - legend title. Default **NULL**.
`labels`	**String vector** labels for the legend. Default **NULL**
`palette`	**String** - An RColorBrewer palette name or a **String vector** vector of colors. Default **NULL**.
`title`	**String** - Title of the plot. Default is an empty string.

Details

—-

Value

Object ggplot2

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

classIntervals

Examples

```
# library(GADMTools)
# library(sp)
# library(dplyr)

# MAP <- gadm.loadCountries("BEL", level = 3, simplify=0.01)
# DAT = read.csv2("BE_clamydia_incidence.csv")

# Here is the main trick !
# ------------------------------------------------------
# DAT <- rename(DAT, NAME_3 = district)

# choropleth(MAP, DAT,
#            adm.join = "NAME_3",
#            value = "rate03",
#            breaks = "sd",
#            palette="Oranges",
#            legend = "Incidence",
#            title="Chlamydia incidence by Belgian district for 2003)")
```

`classDots` *Plot dots on a map with values between different fixed classes.*

Description

Plot values as discretized scale circles on a map.

Usage

```
classDots(x, data, color="red", value = NULL, breaks = NULL,
                      steps = 5, labels = NULL, opacity = 0.5, title="",
                      note=NULL, legend = NULL)
```

Arguments

`x`	**Object** GADMWrapper
`data`	**Object** data.frame with columns 'latitude' and 'longitude'
`color`	a valid color
`value`	**Character** Name of a column of the data.frame.
`breaks`	**vector** of breaks
`steps`	unused
`labels`	**vector** of labels
`opacity`	**float** Background opacity of the filled circles
`title`	**Character** The title of the plot
`note`	**Character** Add an annotation
`legend`	**Character** The title of the legend

Details

—-

Value

Object ggplot2

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

—-

Examples

```
# todo
```

`dots` *Plot dots on a map*

Description

Plot points on a map with different colors and shapes.

Usage

```
dots(x, points, color="red", size = 8, value = NULL,
    breaks = NULL, steps = 5, palette = NULL, labels = NULL, strate = NULL,
    title="", legend = NULL, note=NULL)
```

Arguments

`x`	**Object** GADMWrapper
`points`	**Object** data.frame with columns 'latitude' and 'longitude'
`color`	a valid color
`size`	**integer** size of point
`value`	**Character** Name of a column in the data.frame. If is not null, colored dots are displayed according to the value.
`breaks`	**vector** of breaks
`steps`	**Integer** Number of breaks for the value field.
`palette`	a valid palette
`labels`	**vector** of labels
`strate`	**Character** name of a column in the data.frame. If is not null, display dots with different shapes according to the value.
`title`	**Character** The title of the plot
`legend`	**Character** The title of the legend
`note`	**Character** Add an annotation

Details

—-

Value

Object ggplot2

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

RColorBrewer

Examples

```
# library(GADMTools)
# library(sp)
# data(quakes)
# M <- gadm.loadCountries("FJI", basefile="./")
# colnames(quakes)[1] <- "latitude"
# colnames(quakes)[2] <- "longitude"
# dots(M, quakes, value="mag")
```

---

fast.choropleth *Draw a choropleth on selected regions with lattice.*

Description

Drawing a choropleth (colored regions based on data values) with GADMTools is straightforward. You just have to select your shape(s) file(s) with *gadm.loadcountries*, load your data from a csv file for example, and call the fast.choropleth function with the right arguments. fast.choropleth does not use ggplot2 but lattice, so it is very fast.

Usage

```
fast.choropleth (x, data, value=NULL, breaks = NULL, steps = 5,
  adm.join=NULL, legend = NULL, labels = NULL, palette=NULL,
  title="")
```

Arguments

x	**Object** GADMWrapper
data	**data.frame** - data to plot
value	**String** - the name of the column in the data.frame we want to plot (eg: an incidence in epidemiology studies)
breaks	
steps	**Integer** - number of breaks. Default = 5. If *breaks* is **NOT NULL** this value is used internally with cut().
adm.join	**String** - the name in GADM spdf dataset which will be joined with a column of the data.
legend	**String** - legend title. Default **NULL**.
labels	**String vector** labels for the legend. Default **NULL**
palette	**String** - An RColorBrewer palette name or a **String vector** vector of colors. Default **NULL**.
title	**String** - Title of the plot. Default is an empty string.

Details

—-

Value

Object a lattice plot of class "trellis"

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

classIntervals

Examples

```
# MAP <- gadm.loadCountries("BEL", level = 3, simplify=0.01)
# DAT = read.csv2("BE_clamydia_incidence.csv")

# DAT <- rename(DAT, NAME_3 = district)

# fast.choropleth(MAP, DAT,
#                 adm.join = "NAME_3",
#                 value = "rate03",
#                 steps = 4,
#                 breaks = "jenks",
#                 palette="Greens",
#                 legend = "Incidence",
#                 title="Chlamydia incidence by Belgian district (2003)")
```

---

gadm.getBackground *Gets tiles with 'rosm' from OpenStreetMap*

Description

Load tiles from OpenStreetMap create and save a .tif file with assembled tiles. The bounding box is automatically retrieved from the GADM shapefile passed as argument. The .tif file is stored in the working directory.

Usage

```
gadm.getBackground(x, name, type="osm", clip=TRUE)
```

Arguments

x	**GADMWrapper Object** of the region that you want to add a background.
name	**character** the name of the TIFF file generated by this function. The .tif extension is automaticaly added.
type	**Character** type (default "osm') of the map provided by osm.types().
clip	**boolean** if TRUE (the default), background is clipped by the the external border of the spatial object. If FALSE, spatial object is drawn upper the background using the full bounding box.

Value

Object GADMWrapper

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

osm.types

Examples

```
# library(GADMTools)
# library(rosm)
# FRA = gadm.loadCountries("FRA", 2, basefile = "./")
# BRE = GADMTools::subset(FRA, level=1, regions=c("Bretagne"))
# BRE2 <- gadm.getBackground(BRE, "BRE", "osm")
# plotmap(BRE2, title = "Map of Bretagne (FRANCE)")
```

---

gadm.loadCountries *Load one or more GADM shapefiles*

Description

Load one or more GADM shapefiles from a local path or from a remote repository.

Usage

```
gadm.loadCountries(fileNames, level = 0, basefile=GADM_BASE,
                   baseurl=GADM_URL, simplify=NULL)
```

Arguments

fileNames	**Character vector** of named regions. An ISO-3166-1 code or a custom name. You don't have to specify the suffix (admX) nor the file extension (.rds).
level	**Integer** the level of the administrative boundaries (0 is the country, higher values equal finer divisions)
basefile	**Character vector** the path of the directory where shapefiles are stored. Default is "./GADM"
baseurl	**Character vector** The url of GADM files. Default is "http://biogeo.ucdavis.edu/data/gadm2.8/rds/"
simplify	**Numeric** Numerical tolerance value to be used by the Douglas-Peuker algorithm. Higher values use less polygon points (and less memory) and lower values use more polygon points (and more memory). We suggest not going higher than 0.01 in order for intra-country boundaries to align.

Value

Object GADMWrapper

ISO-3166-1

See : https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3

"ABW","AFG","AGO","AIA","ALA","ALB","AND","ANT","ARE","ARG", "ARM","ASM","ATA","ATF","ATG","AUS","

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

—-

Examples

```
# library(GADMTools)
# library(sp)
# Belgium = gadm.loadCountries("BEL", level=2, basefile="./")
# plotmap(Belgium)
```

---

gadm.loadStripped *Load one GADM stripped shapefile*

Description

Load one GADM stripped shapefiles from a local path for use with ggplot2.

Usage

```
gadm.loadStripped(name, level, basefile='./')
```

Arguments

name	**Character vector** of a named region. An ISO-3166-1 code or a custom name. You don't have to specify the suffix (admX) nor the file extension (.rds).
level	**Integer** the level of the administrative boundaries (0 is the country, higher values equal finer divisions)
basefile	**Character vector** the path of the directory where shapefiles are stored. Default is "./"

Value

Object GADMWrapper with stripped properties == TRUE

ISO-3166-1

See : https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3

```
"ABW","AFG","AGO","AIA","ALA","ALB","AND","ANT","ARE","ARG", "ARM","ASM","ATA","ATF","ATG","AUS","
```

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

—-

Examples

```
# library(GADMTools)
# library(sp)
# BE <- gadm.loadStripped('BEL', level=2)
# plotmap(BE)
```

gadm.longTo360 *Converts longitudes from -180° - 0° - 180° to 0° - 360°*

Description

Converts longitudes of a GADM shapefile to a range of 0° - 360° using the modulo R function.

Usage

```
gadm.longTo360(x)
```

Arguments

x	**GADMWrapper Object** to convert.

Value

Object GADMWrapper

Note

The transformation is done only when rendering a graph. The original data are not modified.

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

Examples

```
# library(GADMTools)
# MAP <- gadm.loadCountries("FJI", level = 0)
# plotmap(MAP)
# MAP <- gadm.longTo360(MAP)
# plotmap(MAP)
```

`gadm.saveStripped` *Save a stripped GADM object*

Description

Save a stripped (with stripSP()) GADM object for later use it with ggplot2.

Usage

```
gadm.saveStripped(x,fname, basefile = './')
```

Arguments

`x`	**Object** GADMWrapper with stripped property == TRUE
`fname`	**String** file name of a region. You don't have to specify the suffix (admX) nor the file extension (.rds).
`basefile`	**Character vector** the path of the directory where shapefiles are stored. Default is "./"

Value

Boolean TRUE

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

—-

Examples

```
# library(GADMTools)
# library(sp)
# BE <- gadm.loadCountries('BEL', level=2)
# S_BE <- stripSP(BE)
# gadm.saveStripped(S_BE, "BEL")
```

gadm.union *Merges regions*

Description

This function merges regions by removing common borders.

Usage

```
gadm.union(x)
```

Arguments

x	**GADMWrapper Object** containing regions.

Value

Object GADMWrapper

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

Examples

```
# library(GADMTools)
# FRA <- gadm.loadCountries("FRA", 2, basefile = "./")
# BRE <- GADMTools::subset(FRA, level=1, regions=c("Bretagne"))
# plotmap(BRE)
# BRE <- gadm.union(BRE)
# plotmap(BRE)
```

grid.map *Arrange maps on a grid*

Description

Allows you to arrange multiple maps into one image. This is useful for showing a country together with its territories in other parts of the world (ex: showing France and Reunion island) or placing two or more countries side by side.

Usage

```
grid.map(left, right, center=NULL, title=NULL)
```

Arguments

left	**Object** GADMWrapper
right	**data.frame** - data to plot
center	**String** - an RColorBrewer palette name or a **String vector** vector of colors. Default **NULL**.
title	**String** - plot title. Default is an empty string.

Details

—-

Value

Object ggplot2

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

—-

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--  or do  help(data=index)  for the standard data sets.
```

`isopleth` *Draw an isopleth on selected regions*

Description

Drawing an isopleth (also known as heat maps) with GADMTools is straightforward. You just have to select your shape(s) file(s) with *gadm.loadcountries*, load your data from a csv file for example, and call the isoopleth function with the right arguments.

Usage

```
isopleth(x, data, palette=NULL, title="", subtitle = "", caption = "")
```

Arguments

`x`	**Object** GADMWrapper
`data`	**data.frame** - data to plot
`palette`	**String** - An RColorBrewer palette name or a **String vector** vector of colors. Default **NULL**.
`title`	**String** - Plot title. Default is an empty string.
`subtitle`	**String** - Plot subtitle. Default is an empty string.
`caption`	**String** - Plot caption. Default is an empty string.

Details

—-

Value

Object ggplot2

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

—-

Examples

```
# library(GADMTools)
# library(sp)

# France = gadm.loadCountries("FRA", level=1, simplify=0.01)
# W <- read.csv2("wepi.csv")
# W$lieux_lat <- as.double(as.character(W$lieux_lat))
# W$lieux_long <- as.double(as.character(W$lieux_long))
# colnames(W)[1] <- "latitude"
# colnames(W)[2] <- "longitude"
# Region = subset(France, regions=c("Ile-de-France","Haute-Normandie"), level=1)
# isopleth(Region, W)
```

---

json.choropleth *Create a geojson choropleth of selected regions.*

Description

Drawing a choropleth (colored regions based on data values) with GADMTools is straightforward. You just have to select your shape(s) file(s) with *gadm.loadcountries*, load your data from a csv file for example, and call the json.choropleth function with the right arguments. json.choropleth creaate a GEOJSON file (output.json) that can be used with Leaflet library.

Usage

```
json.choropleth (x, data, value=NULL, breaks = NULL, steps = 5,
  adm.join=NULL, legend = NULL, labels = NULL, palette=NULL,
  title="")
```

Arguments

x	**Object** GADMWrapper
data	**data.frame** - data to plot
value	**String** - the name of the column in the data.frame we want to plot (eg: an incidence in epidemiology studies)
breaks	
steps	**Integer** - number of breaks. Default = 5. If *breaks* is **NOT NULL** this value is used internally with cut().
adm.join	**String** - the name in GADM spdf dataset which will be joined with a column of the data.
legend	**String** - legend title. Default **NULL**.
labels	**String vector** labels for the legend. Default **NULL**
palette	**String** - An RColorBrewer palette name or a **String vector** vector of colors. Default **NULL**.
title	**String** - Title of the plot. Default is an empty string.

Details

—-

Value

Object a lattice plot of class "trellis"

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

classIntervals

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.
```

---

listNames *List the region names for a specific administrative level*

Description

Returns a list of the names associated with the particular administration level.

Usage

```
listNames(x, level = 0)
```

Arguments

x	**Object** - a GADMWrapper object
level	**Integer** - the value of the administration level to list. Attention: only the administrative levels that have been loaded in the loadCountries object can be listed. Names are given in the country's language or English.

Details

Some GADM country maps provide five or more administrative levels.

Value

Character vector of names

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

—-

Examples

```
# library(GADMTools)
# library(sp)
# MAP <- gadm.loadCountries("FRA", level=1, basefile="./")
# listNames(MAP, level=1)
```

---

plotmap *Draw a shapefile from a GADMWrapper object*

Description

Given the graphic object of a map, this function will display it with a title.

Usage

```
plotmap(x, title="")
```

Arguments

x	**Object** GADMWrapper
title	**String** - Title of the plot. Default is an empty string

Details

—-

Value

Object ggplot2

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

—-

Examples

```
# library(GADMTools)
# library(sp)
# map <- gadm.loadCountries(c("FRA"), level=1, basefile="./")
# plotmap(map, title="carte de la France")
```

---

propDots *Plot proportionnal circles (dots) on a map*

Description

Plot values as proportionnal circles on a map.

Usage

```
propDots(x, data, value, breaks=NULL, range=NULL,
         labels=NULL, color="red", title="", note=NULL)
```

Arguments

x	**Object** GADMWrapper
data	**Object** data.frame with columns 'latitude' and 'longitude'
value	**Character** Name of a column of the data.frame.
breaks	a vector of breaks
range	vector min, max
labels	vector of labels

color	a valid color
title	**Character** The title of the plot
note	**Character** A note associated with the plot

Details

—-

Value

Object ggplot2

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

—-

Examples

```
# library(GADMTools)
# library(sp)
# data(quakes)
# M <- gadm.loadCountries("FJI", basefile="./")
# colnames(quakes)[1] <- "latitude"
# colnames(quakes)[2] <- "longitude"
# propDots(M, quakes, value="mag")
```

---

remove *Remove one or more regions from a map in a GADMWrapper object*

Description

Remove the shapes of one or more regions in a GADMWrapper object.

Usage

```
remove(x, level=NULL, regions=NULL)
```

Arguments

x	**Object** GADMWrapper
level	**Integer** - level from which shapes are removed. If NULL, curent level is used.
regions	**String** - vector of regions to be removed

Details

—-

Value

Object - A GADMWrapper object.

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

—-

Examples

```
# library(GADMTools)
# library(sp)
# map <- gadm.loadCountries(c("FRA"), level=1, basefile="./")
# remove(map, regions="Auvergne")
```

---

saveas *Save your own GADM shapefile as an rds file*

Description

Save a GADM shapefile (.rds)

Usage

```
saveas(x, name = NULL)
```

Arguments

x	**Object** - GADMWWrapper
name	**String** - File path

Details

—-

Value

—-

Note

Do not specify the rds extension, it is added automatically.

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

—-

Examples

```
# library(GADMTools)
# library(sp)
# France = gadm.loadCountries("FRA", level=1, basefile="./")
# Auvergne = subset(France,regions = "Auvergne", level=1)
# saveas(Auvergne, "./AUVERGNE")
# AUV <- gadm.loadCountries("AUVERGNE", level=1, basefile="./")
# plotmap(AUV)
```

---

saveAsStripped *Strip a GADMWrapper object*

Description

Strip a GADMWrapper object (with property 'stripped' == FALSE) and save it stripped (with property 'stripped' == TRUE).

Usage

```
saveAsStripped(x, fname, name= NULL, basefile = './')
```

Arguments

x	**Object** GADMWrapper with stripped property == FALSE
fname	**String** file name of the region to save. You don't have to specify the suffix (admX) nor the file extension (.rds).
name	**String** the name of the field in spdf, like "NAME_1".
basefile	**String** the path of the directory where shapefiles are stored. Default is "./"

Value

Object GADMWrapper with stripped property == TRUE

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

—-

Examples

```
# library(GADMTools)
# library(sp)
# BE <- gadm.loadCountries('BEL', level=2)
# saveAsStripped(BE, "BEL", level=1)
```

---

strippedExists *Test if a stripped GADMWrapper object exists*

Description

Test if a stripped GADMWrapper object exists on the file system in the directory 'basefile'

Usage

```
strippedExists(name, level, basefile = './')
```

Arguments

name	**Character vector** of a named region. An ISO-3166-1 code or a custom name. You don't have to specify the suffix (admX) nor the file extension (.rds).
level	**Integer** the level of the administrative boundaries (0 is the country, higher values equal finer divisions)
basefile	**Character vector** the path of the directory where shapefiles are stored. Default is "./"

Value

Boolean TRUE if the file exists, FALSE if not

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

—-

Examples

```
# library(GADMTools)
# library(sp)
# if (strippedExists('BEL', level = 2) {
#   BE <- gadm.loadStripped("BEL", level=2)
# }
```

---

stripSP *Strip a GADMWrapper object*

Description

Strip a GADMWrapper object (with property 'stripped' == FALSE) and return a stripped GADMWrapper object (with property 'stripped' == TRUE)

Usage

```
stripSP(x, level=NULL)
```

Arguments

x	**Object** GADMWrapper with property 'stripped' == FALSE
level	**Int** admin level to be stripped/extracted. If NULL, the current level is selected

Value

Object GADMWrapper with property 'stripped' == TRUE

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

—-

Examples

```
# library(GADMTools)
# library(sp)
# BE <- gadm.loadCountries('BEL', level=2)
# Belgique <- stripSP(BE, level=2)
```

---

subset *Extract regions*

Description

With subset you can extract one or more regions from a country at the current level.

Usage

```
subset(x, level = NULL, regions = NULL, usevar = NULL)
```

Arguments

x	**Object** GADMWrapper
level	**Integer** the level at which the regions are extracted from
regions	**character vector** of named regions
usevar	**character** name of an other var of spdf@data

Details

—-

Value

Object GADMWrapper

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

listNames

Examples

```
# library(GADMTools)
# library(sp)
# France = gadm.loadCountries("FRA", level=1, basefile="./")
# Auvergne = subset(France,regions = "Auvergne", level=1)
# plotmap(Auvergne, title="Auvergne - level 1")
```

---

vignette *Create a vignette*

Description

Vignette will superimpose a region map over a larger (lower level) map.

Usage

```
vignette(main, region, maincolor = "black",
        regioncolor = "white", mainfill = "grey",
        regionfill = "black",
        mainsize = 1, regionsize = 0.5)
```

Arguments

main	**Object** GADMWrapper
region	**Object** GADMWrapper
maincolor	a valid color
regioncolor	a valid color
mainfill	a valid color
regionfill	a valid color
mainsize	**Numeric** border size
regionsize	**Numeric** border size

Details

—-

Value

Object ggplot2

Note

—-

Author(s)

Jean Pierre Decorps <jp.decorps@epiconcept.fr>

References

—-

See Also

—-

Examples

```
# library(GADMTools)
# library(sp)
# library(ggplot2)
# FR <- gadm.loadCountries("FRA", level=1, basefile="./")
# AU <- subset(FR, regions="Auvergne", level=1)
# vignette(FR, AU)
```

Index