Package ‘maps’
February 20, 2015

Title Draw Geographical Maps
Version 2.3-9
Date 2014-09-22
Author Original S code by Richard A. Becker and Allan R. Wilks.
R version by Ray Brownrigg <Ray.Brownrigg@ecs.vuw.ac.nz>. Enhancements by Thomas P Minka <tpminka@media.mit.edu>
Description Display of maps. Projection code and larger maps are in separate packages (mapproj and mapdata).
Depends R (>= 2.10.0)
LazyLoad yes
Suggests mapproj (>= 1.2-0)
License GPL-2
Maintainer Ray Brownrigg <Ray.Brownrigg@ecs.vuw.ac.nz>
NeedsCompilation yes
Repository CRAN
Date/Publication 2014-09-22 08:37:18

R topics documented:

area.map ......................................................... 2
canada.cities .................................................... 3
county ............................................................ 4
county.fips ....................................................... 5
france ............................................................ 5
identify.map ...................................................... 6
italy ............................................................. 7
map .............................................................. 8
map.axes .......................................................... 12
map.cities ........................................................ 12
map.scale ........................................................ 14
map.text ........................................................ 15
map.where ....................................................... 16

1
Description
Computes the areas of regions in a projected map.

Usage
area.map(m, regions = ".", sqmi=TRUE, ...)

Arguments
m
a map object containing named polygons (created with fill = TRUE).
regions
a character vector naming one of more regions, as in map.
sqmi
If TRUE, measure area in square miles. Otherwise keep the units of m.
...
additional arguments to match.map

Details
The area of each matching region in the map is computed, and regions which match the same element of regions have their areas combined. Each region is assumed planar, with vertices specified by the x and y components of the map object.

The correct use of this function is to first use map to create polygons and project the coordinates onto a plane, then apply area.map to compute the area of the projected regions. If the projection is area-preserving (such as albers), then these areas will match the area on the globe, up to a constant. To get an absolute area in square miles, the sqmi option will scale the result, depending on the projection.

The coordinates from map are affected by its resolution argument, so use resolution=0 for the most accurate areas.
Value

a named vector of region areas.

NOTE

The sqmi option assumes the coordinates have been projected with the mapproject function.

Author(s)

Tom Minka

See Also

area.polygon, apply.polygon

Examples

# because the projection is rectangular, these are not true areas on the globe.
m = map("state", fill = TRUE, plot = FALSE)
area.map(m)
area.map(m, ".*dakota")
area.map(m, c("North Dakota", "South Dakota"))

if(require(mapproj)) {
    # true areas on the globe
    m = map("state", proj="bonne", param=45, fill=TRUE, plotFALSE)
    # North Dakota is listed as 70,704 square miles
    area.map(m, "North Dakota")
}

---

canada.cities

Database of Canadian cities

Description

This database is of Canadian cities of population greater than about 1,000. Also included are province capitals of any population size.

Format

A list with 6 components, namely "name", "country.etc", "pop", "lat", "long", and "capital", containing the city name, the province abbreviation, approximate population (as at January 2006), latitude, longitude and capital status indication (0 for non-capital, 1 for capital, 2 for provincial capital).

NOTE

Some of the city names may be out of date. Please send any corrections to the package maintainer.
See Also

map.cities

---

county	United States County Map

Description

This database produces a map of the counties of the United States mainland generated from US Department of the Census data (see the reference).

Usage

data(countyMapEnv)

Format

The data file is merely a character string which specifies the name of an environment variable which contains the base location of the binary files used by the map drawing functions. This environment variable (R_MAP_DATA_DIR for the datasets in the maps package) is set at package load time if it does not already exist. Hence setting the environment variable before loading the package can override the default location of the binary datasets.

References


See Also

map.

Examples

map('county', 'iowa', fill = TRUE, col = palette())
county.fips

FIPS county codes for US County Map

Description

A database matching FIPS codes to maps package county and state names.

Usage

data(county.fips)

Format

A list with 2 components, namely "fips" and "polynname", containing the FIPS number and respective state or county polygon name.

See Also

state.fips

france

France Map

Description

This france database comes from the NUTS III (Tertiary Administrative Units of the European Community) database of the United Nations Environment Programme (UNEP) GRID-Geneva data sets. These were prepared around 1989, and so may be somewhat out of date.

Users of data sets supplied through UNEP/GRID are requested to incorporate in output products and reports acknowledgements to the originator of the data and to the fact that they were acquired through UNEP/GRID. Appropriate wording may be "UNESCO (1987) through UNEP/GRID-Geneva".

Usage

data(franceMapEnv)

Format

The data file is merely a character string which specifies the name of an environment variable which contains the base location of the binary files used by the map drawing functions. This environment variable (R_MAP_DATA_DIR for the datasets in the maps package) is set at package load time if it does not already exist. Hence setting the environment variable before loading the package can override the default location of the binary datasets.
References


See Also

map

Examples

```r
map('france', fill = TRUE, col = 1:10)
```

**identify.map**  
Identify regions on a map

Description

Identifies the map regions clicked by the user.

Usage

```r
## S3 method for class 'map'
identify(x, n = 1, index = FALSE, ...)
```

Arguments

- `x` a map object containing named polygons.
- `n` the number of clicks to wait for.
- `index` If TRUE, returns the index of the polygon, rather than its name.
- `...` additional arguments passed to `identify.default`.

Details

The current algorithm is somewhat crude — selects the region whose centroid is closest to the click. A more sophisticated approach would use `map.where`.

Value

A character vector of length `n`, naming the selected regions.

Author(s)

Tom Minka


See Also

identify, map.where

Examples

```r
identify(map("state", fill = TRUE, col = 0))
if(require(mapproj))
  identify(map("world", proj = "lagrange", fill = TRUE, col = 0))
```

---

**italy**  

**Italy Map**

**Description**

This italy database comes from the NUTS III (Tertiary Administrative Units of the European Community) database of the United Nations Environment Programme (UNEP) GRID-Geneva data sets. These were prepared around 1989, and so may be somewhat out of date.

Users of data sets supplied through UNEP/GRID are requested to incorporate in output products and reports acknowledgements to the originator of the data and to the fact that they were acquired through UNEP/GRID. Appropriate wording may be "UNESCO (1987) through UNEP/GRID-Geneva".

**Usage**

```r
data(italyMapEnv)
```

**Format**

The data file is merely a character string which specifies the name of an environment variable which contains the base location of the binary files used by the map drawing functions. This environment variable (`r_map_data_dir` for the datasets in the maps package) is set at package load time if it does not already exist. Hence setting the environment variable before loading the package can override the default location of the binary datasets.

**References**


**See Also**

map

**Examples**

```r
map('italy', fill = TRUE, col = 1:10)
```
**Map**

*Draw Geographical Maps*

**Description**

Draw lines and polygons as specified by a map database.

**Usage**

```r
call = map(database = "world", regions = ".", exact = FALSE, boundary = TRUE,
           interior = TRUE, projection = ".", parameters = NULL, orientation = NULL,
           fill = FALSE, col = 1, plot = TRUE, add = FALSE, namesonly = FALSE,
           xlim = NULL, ylim = NULL, wrap = FALSE, resolution = if(plot) 1 else 0,
           type = "l", bg = par("bg"), mar = c(4.1, 4.1, par("mar")[3], 0.1),
           myborder = 0.01, ...)```

**Arguments**

- **database**: character string naming a geographical database, or a list of x, y, and names obtained from a previous call to `map`. The string choices include a world map, three USA databases (`usa`, `state`, `county`), and more (see the package index). The location of the map databases may be overridden by setting the `R_MAP_DATA_DIR` environment variable. Note that the world (and worldhires) databases are derived from a very old 1908's CIA database for which updates have not been made freely available. In particular the individual Russian states are still defined collectively as "USSR". See world for further details.

- **regions**: character vector that names the polygons to draw. Each database is composed of a collection of polygons, and each polygon has a unique name. When a region is composed of more than one polygon, the individual polygons have the name of the region, followed by a colon and a qualifier, as in `michigan:north` and `michigan:south`. Each element of regions is matched against the polygon names in the database and, according to exact, a subset is selected for drawing. The default selects all polygons in the database.

- **exact**: If TRUE, only exact matches with regions are selected for drawing. If FALSE, each element of regions is matched as a regular expression against the polygon names in the database and all matches are selected for drawing.

- **boundary**: If FALSE, boundary segments are not drawn. A boundary segment is a line segment of the map that bounds only one of the polygons to be drawn. This argument is ignored if fill is TRUE.

- **interior**: If FALSE, interior segments are not drawn. An interior segment is a line segment of the map that bounds two of the polygons to be drawn. This argument is ignored if fill is TRUE.

- **projection**: character string that names a map projection to use. See mapproject (in the mapproj library). The default is to use a rectangular projection with the aspect ratio chosen so that longitude and latitude scales are equivalent at the center of the picture.
parameters numeric vector of parameters for use with the projection argument. This argument is optional only in the sense that certain projections do not require additional parameters. If a projection does require additional parameters, these must be given in the parameters argument.

orientation a vector c(latitude, longitude, rotation) describing where the map should be centered and a clockwise rotation (in degrees) about this center.

fill logical flag that says whether to draw lines or fill areas. If FALSE, the lines bounding each region will be drawn (but only once, for interior lines). If TRUE, each region will be filled using colors from the col = argument, and bounding lines will not be drawn.

col vector of colors. If fill is FALSE, the first color is used for plotting all lines, and any other colors are ignored. Otherwise, the colors are matched one-one with the polygons that get selected by the region argument (and are reused cyclically, if necessary). A color of NA causes the corresponding region to be deleted from the list of polygons to be drawn. Polygon colors are assigned after polygons are deleted due to values of the xlim and ylim arguments.

plot logical flag that specifies whether plotting should be done. If plot is TRUE the return value of map will not be printed automatically.

add logical flag that specifies whether to add to the current plot. If FALSE, a new plot is begun, and a new coordinate system is set up.

namesonly If TRUE, the return value will be a character vector of the names of the selected polygons. See the Value section below.

xlim two element numeric vector giving a range of longitudes, expressed in degrees, to which drawing should be restricted. Longitude is measured in degrees east of Greenwich, so that, in particular, locations in the USA have negative longitude. If fill = TRUE, polygons selected by region must be entirely inside the xlim range. The default value of this argument spans the entire longitude range of the database.

ylim two element numeric vector giving a range of latitudes, expressed in degrees, to which drawing should be restricted. Latitude is measured in degrees north of the equator, so that, in particular, locations in the USA have positive latitude. If fill = TRUE, polygons selected by region must be entirely inside the ylim range. The default value of this argument spans the entire latitude range of the database.

wrap If TRUE, lines that cross too far across the map (due to a strange projection) are omitted.

resolution number that specifies the resolution with which to draw the map. Resolution 0 is the full resolution of the database. Otherwise, just before polylines are plotted they are thinned: roughly speaking, successive points on the polyline that are within resolution device pixels of one another are collapsed to a single point (see the Reference for further details).

type character string that controls drawing of the map. Aside from the default type = "1", the value type = "n" can be used to set up the coordinate system and projection for a map that will be added to in later calls.

bg background color.
mar margins, as in par. Defaults allow for map.axes().
myborder scalar or vector of length 2 specifying the proportion of the plot to add to the defined or computed limits as borders.
... Extra arguments passed to polygon or lines.

Details

The simplest form of use of this function is:

map(mymap)

where mymap is the returned value from a previous call to map().

Value

If plot = TRUE, a plot is made where the polygons selected from database, through the regions, xlim, and ylim arguments, are outlined (fill is FALSE) or filled (fill is TRUE) with the colors in col.

The return value is a list with x, y, range, and names components. This object can be used as a database for successive calls to map and functions. If fill is FALSE, the x and y vectors are the coordinates of successive polylines, separated by NAs. If fill is TRUE, the x and y vectors have coordinates of successive polygons, again separated by NAs. Thus the return value can be handed directly to lines or polygon, as appropriate.

When namesonly is TRUE, only the names component is returned.

After a call to map for which the projection argument was specified there will be a global variable .Last.projection containing information about the projection used. This will be consulted in subsequent calls to map which use projection = "".

References


See Also

map.text, map.axes, map.scale, map.grid (in the mapproj library)

Examples

map() # low resolution map of the world
map("usa") # national boundaries
map("county", "new jersey") # county map of New Jersey
map("state", region = c("new york", "new jersey", "penn")) # map of three states
map("state", ".*dakota", myborder = 0) # map of the dakotas
map.axes() # show the effect of myborder = 0
if(require(mapproj))
  mapproj('state', proj = 'bonne', param = 45) # Bonne equal-area projection of states

# names of the San Juan islands in Washington state
map('county', 'washington,san', names = TRUE, plot = FALSE)

# national boundaries in one linetype, states in another
# (Figure 5 in the reference)
map("state", interior = FALSE)
map("state", boundary = FALSE, lty = 2, add = TRUE)

# plot the ozone data on a base map
# (Figure 4 in the reference)
data(ozone)
map("state", xlim = range(ozone$x), ylim = range(ozone$y))
text(ozone$x, ozone$y, ozone$median)
box()
if(require(mapproj)) { # mapproj is used for projection="polyconic"
  # color US county map by 2009 unemployment rate
  # match counties to map using FIPS county codes
  # Based on J's solution to the "Choropleth Challenge"

  # load data
  # unemp includes data for some counties not on the "lower 48 states" county
  # map, such as those in Alaska, Hawaii, Puerto Rico, and some tiny Virginia
  # cities
  data(unemp)
data(county.fips)

  # define color buckets
colors = c("#F1EEF6", "#D4B9DA", "#C994C7", "#DF65B0", "#DD1C77", "#980043")
unemp$colorBuckets <- as.numeric(cut(unemp$unemp, c(0, 2, 4, 6, 8, 10, 100)))
leg.txt <- c("<2%", "2-4%", "4-6%", "6-8%", "8-10%", ">10%")

  # align data with map definitions by (partial) matching state,county
  # names, which include multiple polygons for some counties
  cnty.fips <- county.fips$fips[match(map("county", plot=FALSE)$names,
     county.fips$fips$polyname)]
colorsmatched <- unemp$colorBuckets [match(cnty.fips, unemp$fips)]

  # draw map
  map("county", col = colors[colorsmatched], fill = TRUE, resolution = 0,
      lty = 0, projection = "polyconic")
  map("state", col = "white", fill = FALSE, add = TRUE, lty = 1, lwd = 0.2,
      projection="polyconic")
title("unemployment by county, 2009")
  legend("topright", leg.txt, horiz = TRUE, fill = colors)

  # Choropleth Challenge example, based on J's solution, see:
  # To see the faint county boundaries, use RGui menu: File/SaveAs/PDF
}
map.axes

**Draw Axes on Geographical Maps**

**Description**

Draws a set of axes on an existing map.

**Usage**

```r
map.axes()
```

**Side Effects**

x- and y-axes are drawn for the currently displayed map. These will display in longitude and latitude (if no projection= has been specified in the map() call).

**Examples**

```r
map("state")
map.axes()
```

---

map.cities

**Add Cities to Existing Map**

**Description**

Adds city locations and (optionally) names to an existing map using a specified database.

**Usage**

```r
map.cities(x = world.cities, country = "", label = NULL, minpop = 0, maxpop = Inf, capitals = 0, cex = par("cex"), projection = FALSE, parameters = NULL, orientation = NULL, pch = 1, ...)
```

**Arguments**

- **x**
  - Name of database. See `world.cities` to determine the structure of the database.
- **country**
  - If the string country is specified, limit the displayed cities to be from within the specified country, province or state (depending on how the database has been constructed).
- **label**
  - If TRUE, label all cities. If NULL, the cities will be labelled unless there are 20 or more.
- **minpop**
  - The minimum value of population below which a particular city will not be shown.
map.cities

maxpop  The maximum value of population above which a particular city will not be shown.
capitals Selection of capitals-only display. Capitals may be 1 (country capital), 2 (provincial, state, or regional capital) or 3 (local capital). See world.cities for further information.
cex The value of cex acts to override the current value of character size expansion.
projection Boolean or character value. If FALSE (the default), no projection is assumed, if TRUE, the previous projection is used, otherwise a character string that names a map projection to use. See mapproject (in the mapproj library).
parameters numeric vector of parameters for use with the projection argument. This argument is optional only in the sense that certain projections do not require additional parameters. If a projection does require additional parameters, these must be given in the parameters argument.
orientation a vector c(latitude, longitude, rotation) describing where the map should be centered and a clockwise rotation (in degrees) about this center.
pch plotting character to use for marking city location. See points for options.
... Further plotting parameters may be specified as for the commands points and text.

Details

The database is searched for all cities matching the specified criteria and fitting within the limits of the plot currently displayed. The default database is of all cities that have a population greater than a certain threshold or which are capital cities of a country or island territory. The threshold varies from country to country, but in general is no higher than about 40,000. The data are copyright Stefan Helders but freely available from his website http://www.world-gazetteer.com.

There are three supplied databases, world.cities (the default), us.cities and canada.cities. The latter two, which need to be made available by using a 'data()' call, include the state or province name with the city name (thanks to John Woodruff <jpwoodruff@irisinternet.net> for the state and province information).

Note that if the underlying map is "Pacific-centric", i.e. longitudes exceed 180 degrees, and a projection is used, then the map.cities data must be transformed appropriately.

Value

No value is returned from map.cities.

Side Effects

All cities within the boundaries of the plot containing the current map are added to the plot. Note that it is possible that the boundaries of the plot exceed the boundaries of the map requested, and so more cities than were expected might be shown.

See Also

world.cities, canada.cities, us.cities
Examples

```r
map("world", "China")
map.cities(country = "China", capitals = 2)
map("state", "New Jersey")
data(us.cities)
map.cities(us.cities, country="NJ")
```

---

**map.scale**

*Add Scale to Existing Unprojected Map*

**Description**

Adds a scale to an existing map, both as a ratio and a distance gauge.

**Usage**

```r
map.scale(x, y, relwidth = 0.15, metric = TRUE, ratio = TRUE, ...)
```

**Arguments**

- **x**
- **y**
- **relwidth**
- **metric**
- **ratio**
- **...**

**Details**

The scale is calculated from the displayed graph’s plotting parameters, and the latitude of the location at which the distance gauge will be displayed.

**Value**

The exact calculated scale is returned.

**NOTE**

This function is meaningful only if no projection= has been specified in the call to map().

**Side Effects**

A scale is added to the currently displayed map. This takes the form of an approximate 1:n scale (containing 2-3 significant digits), above a distance gauge which is reasonably accurate for the latitude at which it appears. The circumference at the given latitude is interpolated from a radius of 6356.78 km at the pole and 6378.16 km at the equator.
See Also

map.axes

Examples

map("world", "China")
map.scale()

map.text  Draw a map with labeled regions

Description

Like map, but labels the regions.

Usage

map.text(database, regions = ".", exact = FALSE, labels, cex = 0.75,
add = FALSE, move = FALSE, ...)

Arguments

database character string naming a geographical database, or a list of x, y, and names
obtained from a previous call to map.

regions character vector that names the polygons to draw.

exact If ‘TRUE’, only exact matches with ’regions’ are selected for drawing.

labels character vector of labels, one for each region selected. Defaults to the names in
the database.

cex character expansion factor.

add If FALSE, a map is drawn, then labels placed on top. If TRUE, labels are added to
the existing map.

move If TRUE, labels are moved so that they don’t overlap. Requires the mining library
(not in CRAN, contact tpminka@media.mit.edu).

... Other arguments are the same as in map.

Value

If add = FALSE, a map is drawn by calling map. Then the label for each region is placed at the
centroid of the region polygon.

The return value is a map object, as from map.

Author(s)

Tom Minka
Examples

map.text("world", "ira")  # iran and iraq
map.text("state", "penn")
map.text("county", "penn")  # Pennsylvania counties
map.text("county", "new jersey")  # New Jersey counties

map.where

Locate points on a map

Description

Returns the region names containing given locations.

Usage

map.where(database = "world", x, y)

Arguments

database character string naming a geographical database, or a list of x, y, and names.
See the documentation for map for more details.

x vector of longitudes.

y vector of latitudes.

Value

A list of character strings, naming the map region that each (longitude, latitude) pair falls into.

Note

Because maps are currently organized as flat files, this function can be slow.

Author(s)

Tom Minka

See Also

in.polygon
match.map

Examples

# NYC
map.where("state", -73.8, 41)
# Auckland
map.where("nz", 174.6, -36.92)
# find both in the world (takes a while)
map.where(x = c(174.6, -73.8), y = c(-36.92, 41))
# with a map object:
m = map("state", "new york", fill = TRUE, plot = FALSE)
map.where(m, -73.8, 41)

match.map

Index map regions

Description

Assigns an index to each map region, useful for map coloring.

Usage

match.map(database, regions, exact = FALSE, warn = TRUE)

Arguments

database character string naming a geographical database, or a map object. See the documentation for map for more details.

regions a vector of names, or more generally regular expressions to match against the map region names.

exact If TRUE, only exact matches with regions are considered. Otherwise each element of regions is assumed to be a regular expression. Matches are always case-insensitive.

warn If TRUE, a warning is printed when an element of regions matches nothing in the map.

Value

Returns an integer vector giving an index to each region in the database. The index is the index of the string in regions which matches the region name. Matching is done as in map. More specifically, all regions r whose name matches regions[i] will have index i. Unmatched regions will have index NA. Overlapping matches cause an error.

This behavior differs from pmatch because a single entry in regions may match several entries in the map.

Author(s)

Tom Minka


References


See Also
grep

Examples

# filled map showing Republican vote in 1900
# (figure 6 in the reference)
data(state, package = "datasets")
data(votes.repub)
state.to.map <- match.map("state", state.name)
x <- votes.repub[state.to.map, "1900"]
gray.colors <- function(n) gray(rev(0:(n - 1))/n)
color <- gray.colors(100)[floor(x)]
map("state", fill = TRUE, col = color); map("state", add = TRUE)

nz

New Zealand Basic Map

Description

This database produce a map of New Zealand at a basic level of detail. The "nz" database includes the 3 main Islands and 19 smaller coastal islands.

Usage

data(nzMapEnv)

Format

The data file is merely a character string which specifies the name of an environment variable which contains the base location of the binary files used by the map drawing functions. This environment variable (R_MAP_DATA_DIR for the datasets in the maps package) is set at package load time if it does not already exist. Hence setting the environment variable before loading the package can override the default location of the binary datasets.

References


**ozone**

See Also

map

Examples

map('nz')
map('nz', xlim = c(166, 179), ylim = c(-48, -34))

---

**Description**

Datasets used to illustrate map functions.

**Usage**

data(ozone)
data(unemp)
data(votes.repub)

---

**smooth.map**

Smooth out aggregated data

**Description**

Increases the resolution of data aggregated over map regions, by either smoothing or interpolation. Also fills in missing values.

**Usage**

smooth.map(m, z, res = 50, span = 1/10, averages = FALSE, type = c("smooth", "interp"), merge = FALSE)

**Arguments**

- **m** a map object
- **z** a named vector
- **res** a vector of length two, specifying the resolution of the sampling grid in each dimension. If a single number, it is taken as the vertical resolution, with double taken as the horizontal resolution.
- **span** kernel parameter (larger = smoother). span = Inf is a special case which invokes the cubic spline kernel. span is automatically scaled by the map size, and is independent of res.
averages

If TRUE, the values in z are interpreted as averages over the regions. Otherwise they are interpreted as totals.

type

see details.

merge

If TRUE, a region named in z includes all matching regions in the map (according to match.map). If FALSE, a region named in z is assumed to refer to exactly one region on the map.

Details

For type = "smooth", the region totals are first converted into point measurements on the sampling grid, by dividing the total for a region among all sample points inside it. Then it is a regular kernel smoothing problem. Note that the region totals are not preserved.

The prediction \( z_o \) for location \( x_o \) (a vector) is the average of \( z \) for nearby sample points:

\[
z_o = \frac{\sum_x k(x, x_o)z(x)}{\sum_x k(x, x_o)}
\]

\( k(x, x_o) = e^{\lambda ||x - x_o||^2} \)

\( \lambda \) is determined from span. Note that \( x_o \) is over the same sampling grid as \( x \), but \( z_o \) is not necessarily the same as \( z(x_o) \).

For type = "interp", the region totals are preserved by the higher-resolution function. The function is assumed to come from a Gaussian process with kernel \( k \). The measurement \( z[r] \) is assumed to be the sum of the function over the discrete sample points inside region \( r \). This leads to a simple formula for the covariance matrix of \( z \) and the cross-covariance between \( z_o \) and \( z \). The prediction is the cross-covariance times the inverse covariance times \( z \). Unlike Tobler’s method, the predictions are not constrained to live within the original data range, so there tends to be “ringing” effects.

See the references for more details.

Value

A data frame with columns \( x, y, \) and \( z \) giving the smoothed value \( z \) for locations \( (x, y) \). Currently the \( (x, y) \) values form a grid, but this is not guaranteed in the future.

Author(s)

Tom Minka

References


Examples

```r
# compare to the example for match.map
data(state, package = "datasets")
data(votes.repub)
z = votes.repub[, "1900"]
m = map("state", fill = TRUE, plot = FALSE)
# use a small span to fill in, but not smooth, the data
# increase the resolution to get better results
fit = smooth.map(m, z, span = 1/100, merge = TRUE, ave = TRUE)
mat = tapply(fit$z, fit[1:2], mean)
gray.colors <- function(n) gray(rev((0:(n - 1))/n))
par(bg = "blue")
filled.contour(mat, color.palette = gray.colors, nlev = 32, asp = 1)
# another way to visualize:
image(mat, col = gray.colors(100))

# for a higher degree of smoothing:
# fit = smooth.map(m, z, merge = TRUE, ave = TRUE)
# interpolation, state averages are preserved:
# fit = smooth.map(m, z, merge = TRUE, ave = TRUE, type = "interp")
```

Description

This database produces a map of the states of the United States mainland generated from US Department of the Census data (see the reference).

Usage

```r
data(stateMapEnv)
```

Format

The data file is merely a character string which specifies the name of an environment variable which contains the base location of the binary files used by the map drawing functions. This environment variable (R\_MAP\_DATA\_DIR for the datasets in the maps package) is set at package load time if it does not already exist. Hence setting the environment variable before loading the package can override the default location of the binary datasets.

References

state.carto

See Also

map.

Examples

map('state', fill = TRUE, col = palette())

---

state.carto | United States State Population Cartogram Map

Description

This database produces a cartogram of the states of the United States mainland based on CartoDraw, roughly proportional to population (see references).

state.carto.center are coordinates of the state centers for annotation purposes.

Usage

data(stateMapEnv)
data(state.carto.center)

Format

The data file is merely a character string which specifies the name of an environment variable which contains the base location of the binary files used by the map drawing functions. This environment variable (R_MAP_DATA_DIR for the datasets in the maps package) is set at package load time if it does not already exist. Hence setting the environment variable before loading the package can override the default location of the binary datasets.

References


See Also

map.

Examples

map('state.carto', fill = TRUE, col = palette())
**state.fips**

*FIPS state codes for US 48 State Map*

**Description**

A database matching FIPS codes to maps package state names.

**Usage**

```r
data(state.fips)
```

**Format**

A list with 6 components, namely "fips", "ssa", "region", division", "abb" and "polynome", containing the US Census Bureau FIPS, SSA, REGION and DIVISION numbers, the standard state abbreviation and the respective state polygon name.

**See Also**

`county.fips`

---

**state.vbm**

*United States State Visibility Base Map*

**Description**

This database produces a map of the states of the United States mainland. The Visibility Base Map was created by Mark Monmonier to provide simplified state shapes with sufficient areas to allow annotations in even the small states.

*state.vbm.center* are coordinates of the state centers for annotation purposes.

**Usage**

```r
data(stateMapEnv)
data(state.vbm.center)
```

**Format**

The data file is merely a character string which specifies the name of an environment variable which contains the base location of the binary files used by the map drawing functions. This environment variable (R_MAP_DATA_DIR for the datasets in the maps package) is set at package load time if it does not already exist. Hence setting the environment variable before loading the package can override the default location of the binary datasets.
References


See Also

map.

Examples

```r
map('state.vbm', fill = TRUE, col = palette())
```

---

**us.cities**

*Database of US cities*

Description

This database is of US cities of population greater than about 40,000. Also included are state capitals of any population size.

Format

A list with 6 components, namely "name", "country.etc", "pop", "lat", "long", and "capital", containing the city name, the state abbreviation, approximate population (as at January 2006), latitude, longitude and capital status indication (0 for non-capital, 1 for capital, 2 for state capital).

NOTE

Some of the city names may be out of date. Please send any corrections to the package maintainer.

See Also

map.cities
United States Coast Map

Description

This database produces a map of the United States mainland generated from US Department of the Census data (see the reference).

Usage

\`\`data(usaMapEnv)\`

Format

The data file is merely a character string which specifies the name of an environment variable which contains the base location of the binary files used by the map drawing functions. This environment variable (\`R_MAP_DATA_DIR\` for the datasets in the maps package) is set at package load time if it does not already exist. Hence setting the environment variable before loading the package can override the default location of the binary datasets.

References


See Also

map.

Examples

\`\`map('usa')\`

Description

This world database comes from a thinned cleaned-up version of the CIA World Data Bank II data and contains approximately 30,000 points representing the world coastlines and national boundaries. The full database of nearly 2 million points is found in the mapdata package.

Usage

data(worldMapEnv)

Format

The data file is merely a character string which specifies the name of an environment variable which contains the base location of the binary files used by the map drawing functions. This environment variable (R_MAP_DATA_DIR for the datasets in the maps package) is set at package load time if it does not already exist. Hence setting the environment variable before loading the package can override the default location of the binary datasets.

Source

The CIA World Data Bank II is currently (mid-2003) available from http://www.evl.uic.edu/pape/data/WDB/.

References


See Also

map

Examples

map('world', fill = TRUE, col = 1:10)
world.cities

Description

This database is primarily of world cities of population greater than about 40,000. Also included are capital cities of any population size, and many smaller towns.

Usage

data(world.cities)

Format

A list with 6 components, namely "name", "country.etc", "pop", "lat", "long", and "capital", containing the city name, the country name, approximate population (as at January 2006), latitude, longitude and capital status indication (0 for non-capital, 1 for capital, 2 for China Municipalities, and 3 for China Provincial capitals)

NOTE

Some of the country names and city names may be out of date. Please send any corrections to the package maintainer.

See Also

map.cities

world2

Pacific Centric Low resolution World Map

Description

This world database comes from a thinned cleaned-up version of the CIA World Data Bank II data and contains approximately 30,000 points representing the world coastlines and national boundaries. The full database of nearly 2 million points is found in the mapdata package. This map is based on latitudes [0, 360), which then has the Pacific Ocean in the centre of the map.

Usage

data(world2MapEnv)
Format

The data file is merely a character string which specifies the name of an environment variable which contains the base location of the binary files used by the map drawing functions. This environment variable (R\_MAP\_DATA\_DIR for the datasets in the maps package) is set at package load time if it does not already exist. Hence setting the environment variable before loading the package can override the default location of the binary datasets.

Source

The CIA World Data Bank II is currently (mid-2003) available from [http://www.evl.uic.edu/pape/data/WDB/](http://www.evl.uic.edu/pape/data/WDB/).

References


See Also

map, world

Examples

```r
map('world2', xlim = c(100, 300))
map.axes()
```
Index

*Topic **datasets**
canada.cities, 3
county, 4
county.fips, 5
france, 5
italy, 7
nz, 18
ozone, 19
state, 21
state.carto, 22
state.fips, 23
state.vbm, 23
us.cities, 24
usa, 25
world, 26
world.cities, 27
world2, 27
*Topic **dplot**
area.map, 2
match.map, 17
smooth.map, 19
*Topic **hplot**
map, 8
map.axes, 12
map.cities, 12
map.scale, 14
map.text, 15
*Topic **iplot**
identify.map, 6
map.where, 16
area.map, 2

canada.cities, 3, 13
county, 4, 8
county.fips, 5, 23
countyMapEnv (county), 4
france, 5
franceMapEnv (france), 5
grep, 18
identify, 7
identify.default, 6
identify.map, 6
italy, 7
italyMapEnv (italy), 7
map, 2, 4, 6, 7, 8, 15–17, 19, 22, 24–26, 28
map.axes, 10, 12, 15
map.cities, 4, 12, 24, 27
map.grid, 10
map.scale, 10, 14
map.text, 10, 15
map.where, 6, 7, 16
mapproject, 3, 8, 13
match.map, 2, 17, 20
nz, 18
nzMapEnv (nz), 18
ozone, 19
par, 10
pmatch, 17
points, 13
smooth.map, 19
state, 8, 21
state.carto, 22
state.cartoMapEnv (state.carto), 22
state.fips, 5, 23
state.vbm, 23
state.vbmMapEnv (state.vbm), 23
stateMapEnv (state), 21
text, 13
unemp (ozone), 19
us.cities, 13, 24
usa, 8, 25
usaMapEnv (usa), 25
votes.repub (ozone), 19
world, 8, 26, 28
world.cities, 12, 13, 27
world2, 27
world2MapEnv (world2), 27
worldHires, 8
worldMapEnv (world), 26