Package ‘mapsf’

May 30, 2022

Title Thematic Cartography

Version 0.5.0

Description Create and integrate thematic maps in your workflow. This package helps to design various cartographic representations such as proportional symbols, choropleth or typology maps. It also offers several functions to display layout elements that improve the graphic presentation of maps (e.g. scale bar, north arrow, title, labels). ‘mapsf’ maps ‘sf’ objects on ‘base’ graphics.

License GPL-3


BugReports https://github.com/riatelab/mapsf/issues/

Depends R (>= 3.6.0)

Imports classInt, graphics, methods, Rcpp, sf, stats, utils, grDevices

Suggests terra, png, jpeg, lwgeom, knitr, rmarkdown, tinytest, covr

LinkingTo Rcpp

Encoding UTF-8

RoxygenNote 7.2.0

VignetteBuilder knitr

Language en-US

NeedsCompilation yes

Author Timothée Giraud [cre, aut] (<https://orcid.org/0000-0002-1932-3323>), Diego Hernangómez [ctb] (<https://orcid.org/0000-0001-8457-4658>), Hugues Pecout [ctb], Ronan Ysebaert [ctb], Ian Fellows [cph] (No overlap algorithm for labels, from wordcloud package), Jim Lemon [cph] (Arc drawing algorithm for annotations, from plotrix package), Florian Zenoni [cph] (Orthographic projection visualisation)
R topics documented:

- mapsf .................................................. 2
- mf_annotation ........................................ 3
- mf_arrow .............................................. 4
- mf_background ......................................... 5
- mf_credits ............................................. 5
- mf_export ............................................... 6
- mf_get_breaks .......................................... 7
- mf_get_links ........................................... 8
- mf_get_ntq ............................................ 9
- mf_get_pal ............................................ 10
- mf_init .................................................. 11
- mf_inset_on ........................................... 12
- mf_label ............................................... 13
- mf_layout .............................................. 14
- mf_legend .............................................. 15
- mf_map .................................................. 17
- mf_raster ............................................... 20
- mf_scale ............................................... 20
- mf_shadow ............................................. 21
- mf_theme ................................................ 22
- mf_title ............................................... 23
- mf_worldmap .......................................... 24

Index 26

mapsf

Package description

Description

Create maps with simple features. mapsf helps to map sf objects and offers features that improve the graphic presentation of maps (scale bar, north arrow, title or legend).
Plot an annotation

Description

Plot an annotation on a map.

Usage

```r
mf_annotation(
  x,
  txt,
  pos = "topright",
  cex = 0.8,
  col_arrow,
  col_txt,
  halo = FALSE,
  bg,
  s = 1,
  ...
)
```

Arguments

- `x`: an sf object with 1 row, a couple of coordinates (c(x, y)).
- `txt`: the text to display
- `pos`: position of the text, one of "topleft", "topright", "bottomright", "bottomleft" or "interactive"
- `cex`: size of the text
- `col_arrow`: arrow color
- `col_txt`: text color
- `halo`: add a halo around the text
- `bg`: halo color
- `s`: arrow size (min=1)
- `...`: further text arguments.

Value

No return value, an annotation is displayed.
Examples

```r
mtq <- mf_get_mtz()
mf_map(mtq)
mf_annotation(
  x = c(711167.8, 1614764),
  txt = "Look! Important feature here!",
  pos = "bottomleft", cex = 1.2, font = 2,
  halo = TRUE, s = 1.5
)

mf_annotation(
  x = mtq[20, ],
  txt = "This is less important",
  cex = .7, font = 3, s = 1.3
)
```

### mf_arrow

Plot a north arrow

#### Description
Plot a north arrow.

#### Usage
```
mf_arrow(pos = "topleft", col, adjust)
```

#### Arguments
- `pos`: position. It can be one of 'topleft', 'top', 'topright', 'right', 'bottomright', 'bottom', 'bottomleft', 'left', 'interactive' or a vector of two coordinates in map units (c(x, y))
- `col`: arrow color
- `adjust`: object of class sf or sfc used to adjust the arrow to the real north

#### Value
No return value, a north arrow is displayed.

#### Examples
```
mtq <- mf_get_mtz()
mf_map(mtq)
mf_arrow(pos = "topright")
```
**mf_background**

---

**mf_background**

*Plot a background image*

### Description

Plot a background image on an existing plot.

### Usage

```r
mf_background(filename, ...)
```

### Arguments

- **filename**
  - filename of the background image, PNG or JPG/JPEG format.

- **...**
  - further parameters for `rasterImage`.

### Value

No return value, a background image is displayed.

### Examples

```r
mtq <- mf_get_mtq()
mf_init(mtq)
mf_background(system.file("img/background.jpg", package = "mapsf"))
mf_map(mtq, lwd = 3, col = NA, border = "white", add = TRUE)
mf_credits(
  txt = "Background photo by Noita Digital on Unsplash",
  col = "white"
)
```

---

**mf_credits**

---

**mf_credits**

*Plot credits*

### Description

Plot credits (sources, author, year...).

### Usage

```r
mf_credits(
  txt = "Source(s) & Author(s)",
  pos = "bottomleft",
  col,
  cex = 0.6,
  font = 3,
  bg = NA
)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>txt</td>
<td>text of the credits, use '\n' to add line breaks</td>
</tr>
<tr>
<td>pos</td>
<td>position, one of 'bottomleft', 'bottomright' or 'rightbottom'</td>
</tr>
<tr>
<td>col</td>
<td>color</td>
</tr>
<tr>
<td>cex</td>
<td>cex of the credits</td>
</tr>
<tr>
<td>font</td>
<td>font of the credits</td>
</tr>
<tr>
<td>bg</td>
<td>background color</td>
</tr>
</tbody>
</table>

Value

No return value, credits are displayed.

Examples

```r
mtq <- mf_get_mtq()
mf_map(mtq)
mf_credits(txt = "Author\nSources - Year")
```

mf_export

Export a map

Description

Export a map with the extent of a spatial object. The map is exported in PNG or SVG format. If only one of width or height is set, mf_export uses the width/height ratio of x bounding box to find a matching ratio for the export.

Usage

```r
mf_export(
  x,
  filename = "map.png",
  width,
  height,
  res = 96,
  ..., 
  expandBB = rep(0, 4),
  theme,
  export = "png"
)
```
mf_get_breaks

Arguments

- **x**: object of class `sf`, `sfc` or `Raster`
- **filename**: path to the exported file. If the file extension is ".png" a png graphic device is opened, if the file extension is ".svg" a svg graphic device is opened.
- **width**: width of the figure (pixels for png, inches for svg)
- **height**: height of the figure (pixels for png, inches for svg)
- **res**: resolution (for png)
- **...**: further parameters for png or svg export
- **expandBB**: fractional values to expand the bounding box with, in each direction (bottom, left, top, right)
- **theme**: apply a theme
- **export**: deprecated

Value

No return value, a map is initiated.

Examples

```r
mtq <- mf_get_mtq()
(filenname <- tempfile(fileext = ".png"))
mf_export(mtq, filename = filenname)
mf_map(mtq, add = TRUE)
dev.off()
```

---

mf_get_breaks | Get class intervals

Description

A function to classify continuous variables.

Usage

```r
mf_get_breaks(x, nbreaks, breaks, k = 1, central = FALSE, ...)
```

Arguments

- **x**: a vector of numeric values
- **nbbreaks**: a number of classes
- **breaks**: a classification method; one of "fixed", "sd", "equal", "pretty", "quantile", "kmeans", "hclust", "bclust", "fisher", "jenks", "dpih", "q6", "geom", "arith", "em" or "msd" (see Details).
- **k**: number of standard deviation for "msd" method (see Details)
- **central**: creation of a central class for "msd" method (see Details)
- **...**: further arguments of `classIntervals`
"fixed", "sd", "equal", "pretty", "quantile", "kmeans", "hclust", "bclust", "fisher", "jenks" and "dpih" are classIntervals methods. You may need to pass additional arguments for some of them.

Jenks ("jenks" method) and Fisher ("fisher" method) algorithms are based on the same principle and give quite similar results but Fisher is much faster.

The "q6" method uses the following quantile probabilities: 0, 0.05, 0.275, 0.5, 0.725, 0.95, 1.

The "geom" method is based on a geometric progression along the variable values.

The "arith" method is based on an arithmetic progression along the variable values.

The "em" method is based on nested averages computation.

The "msd" method is based on the mean and the standard deviation of a numeric vector. The nbbreaks parameter is not relevant, use k and central instead. k indicates the extent of each class in share of standard deviation. If central=TRUE then the mean value is the center of a class else the mean is a break value.

Value
A numeric vector of breaks

Note
This function is mainly a wrapper of classIntervals + "arith", "em", "q6", "geom" and "msd" methods.

See Also
classIntervals

Examples
mtq <- mf_get_mtq()
mf_get_breaks(x = mtq$MED, nbbreaks = 6, breaks = "quantile")

mf_get_links Get a link layer from a data.frame of links.

Description
Create a link layer from a data.frame of links and an sf object.

Usage
mf_get_links(x, df, x_id, df_id)
mf_get_mtq

Arguments

- **x**: an sf object, a simple feature collection.
- **df**: a data.frame that contains identifiers of starting and ending points.
- **x_id**: name of the identifier variable in x, default to the first column (optional).
- **df_id**: names of the identifier variables in df, character vector of length 2, default to the two first columns. (optional)

Value

An sf object is returned, it is composed of df and the sfc (LINESTRING) of links.

Examples

```r
mtq <- mf_get_mtq()
mob <- read.csv(system.file("csv/mob.csv", package = "mapsf"))
# Select links from Fort-de-France (97209)
mob_97209 <- mob[mob$i == 97209, ]
# Create a link layer
mob_links <- mf_get_links(x = mtq, df = mob_97209)
# Plot the links
mf_map(mtq)
mf_map(mob_links, col = "red4", lwd = 2, add = TRUE)
```

mf_get_mtq

Get the 'mtq' dataset

Description

Import the mtq dataset (Martinique municipalities).

Usage

`mf_get_mtq()`

Details

This a wrapper around `st_read(system.file("gpkg/mtq.gpkg", package = "mapsf"), quiet = TRUE)`.

Value

an sf object of Martinique municipalities

Examples

```r
mtq <- mf_get_mtq()
```
mf_get_pal  
Get color palettes

Description

mf_get_pal builds sequential, diverging and qualitative color palettes. Diverging color palettes can be dissymmetric (different number of colors in each of the two gradients).

Usage

mf_get_pal(n, palette, alpha = NULL, rev = c(FALSE, FALSE), neutral)

Arguments

n          the number of colors (>= 1) to be in the palette.
palette    a valid palette name (one of hcl.pals()). The name is matched to the list of available palettes, ignoring upper vs. lower case, spaces, dashes, etc. in the matching.
alpha      an alpha-transparency level in the range [0,1] (0 means transparent and 1 means opaque), see argument alpha in hsv and hcl, respectively.
rev        logical indicating whether the ordering of the colors should be reversed.
neutral    a color, if two gradients are used, the 'neutral' color can be added between them.

Details

See hcl.pals to get available palette names. If two gradients are used, the 'neutral' color can be added between them.

Value

A vector of colors.

Examples

cols <- mf_get_pal(n = 10, pal = "Reds 2")
plot(1:10, rep(1, 10), bg = cols, pch = 22, cex = 4)
cols <- mf_get_pal(n = c(3, 7), pal = c("Reds 2", "Greens"))
plot(1:10, rep(1, 10), bg = cols, pch = 22, cex = 4)
cols <- mf_get_pal(n = c(5, 5), pal = c("Reds 2", "Greens"))
plot(1:10, rep(1, 10), bg = cols, pch = 22, cex = 4)
cols <- mf_get_pal(n = c(7, 3), pal = c("Reds 2", "Greens"))
plot(1:10, rep(1, 10), bg = cols, pch = 22, cex = 4)
cols <- mf_get_pal(n = c(5, 5), pal = c("Reds 2", "Greens"), neutral = "grey")
plot(1:11, rep(1, 11), bg = cols, pch = 22, cex = 4)
opar <- par(bg = "black")
cols <- mf_get_pal(n = c(7, 3), pal = c("Reds 2", "Greens"), alpha = c(.3, .7))
plot(1:10, rep(1, 10), bg = cols, pch = 22, cex = 4)
par(opar)
cols <- mf_get_pal(
  n = c(5, 5), pal = c("Reds 2", "Greens"),
  rev = c(TRUE, TRUE)
)
plot(1:10, rep(1, 10), bg = cols, pch = 22, cex = 4)

mf_init

*Initialize a map with a specific extent*

**Description**

Plot an invisible layer with the extent of a spatial object.

**Usage**

```
mf_init(x, expandBB = rep(0, 4), theme)
```

**Arguments**

- **x**: object of class `sf`, `sfc` or `Raster`
- **expandBB**: fractional values to expand the bounding box with, in each direction (bottom, left, top, right)
- **theme**: apply a theme from `mf_theme`

**Value**

No return value, a map is initiated.

**Examples**

```
mtq <- mf_get_mtq()
target <- mtq[30, ]
mf_init(target)
mf_map(mtq, add = TRUE)
```
### mf_inset_on

**Plot an inset**

**Description**

This function is used to add an inset map to the current map.

**Usage**

```r
mf_inset_on(x, pos = "topright", cex = 0.2, fig)
```

```r
mf_inset_off()
```

**Arguments**

- **x**: an sf object, or "worldmap" to use with mf_worldmap.
- **pos**: position, one of "bottomleft", "left", "topleft", "top", "bottom", "bottomright", "right", "topright"
- **cex**: share of the map width occupied by the inset
- **fig**: coordinates of the inset region (in NDC, see in ?par())

**Details**

- If x is used (with pos and cex), the width/height ratio of the inset will match the width/height ratio of x bounding box.
- If fig is used, coordinates (xmin, xmax, ymin, ymax) are expressed as fractions of the mapping space (i.e. excluding margins).
- If map layers have to be plotted after the inset (i.e after mf_inset_off()), please use add = TRUE.
- It is not possible to plot an inset within an inset.
- It is possible to plot anything (base plots) within the inset, not only map layers.

**Value**

No return value, an inset is initiated or closed.

**Note**

This function does not work when mfrow is used in par().

**Examples**

```r
mtq <- mf_get_mtq()
mf_map(mtq)
mf_inset_on(x = mtq[1, ], cex = .2)
mf_map(mtq[1, ])
mf_inset_off()
mf_inset_off()
```
**mf_label**

```r
mf_map(mtq)
mf_inset_on(x = "worldmap", pos = "bottomleft")
mf_worldmap(x = mtq)
mf_inset_off()

mf_map(mtq)
mf_inset_on(fig = c(0, 0.25, 0, 0.25))
mf_map(x = mtq)
mf_inset_off()
```

---

**Plot labels**

**Description**

Put labels on a map.

**Usage**

```r
mf_label(
  x,
  var,
  col,
  cex = 0.7,
  overlap = TRUE,
  lines = TRUE,
  halo = FALSE,
  bg,
  r = 0.1,
  ...
)
```

**Arguments**

- **x**: object of class `sf`
- **var**: name(s) of the variable(s) to plot
- **col**: labels color
- **cex**: labels cex
- **overlap**: if FALSE, labels are moved so they do not overlap.
- **lines**: if TRUE, then lines are plotted between x,y and the word, for those words not covering their x,y coordinate
- **halo**: If TRUE, then a 'halo' is printed around the text and additional arguments `bg` and `r` can be modified to set the color and width of the halo.
- **bg**: halo color
- **r**: width of the halo
- **...**: further text arguments.
mf_layout

Plot a map layout

Description
Plot a map layout (title, credits, scalebar, north arrow, frame).
This function uses \texttt{mf_title}, \texttt{mf_credits}, \texttt{mf_scale} and \texttt{mf_arrow} with default values.

Usage
\begin{verbatim}
mf_layout(
    title = "Map Title",
    credits = "Authors & Sources",
    scale = TRUE,
    arrow = TRUE,
    frame = FALSE
)
\end{verbatim}

Arguments
\begin{itemize}
\item \texttt{title}  title of the map
\item \texttt{credits}  credits
\item \texttt{scale}  display a scale bar
\item \texttt{arrow}  display an arrow
\item \texttt{frame}  display a frame
\end{itemize}

Value
No return value, a map layout is displayed.

Examples
\begin{verbatim}
mtq <- mf_get_mtq()
mf_map(mtq)
mf_label(
    x = mtq, var = "LIBGEO", halo = TRUE, cex = 0.8,
    overlap = FALSE, lines = FALSE
)
\end{verbatim}
mf_legend

Plot a legend

Description

Plot all types of legend. The "type" argument defines the legend type:

- **prop**, for proportional symbols maps, see `mf_legend_p` for arguments, default values and details;
- **choro**, for choropleth maps, see `mf_legend_c` for arguments, default values and details;
- **typo**, for typology maps, see `mf_legend_t` for arguments, default values and details;
- **symb** for symbols maps, see `mf_legend_s` for arguments, default values and details;
- **prop_line**, for proportional lines maps, see `mf_legend_pl` for arguments, default values and details;
- **grad_line** for graduated lines maps, see `mf_legend_gl` for arguments, default values and details.

Usage

```r
mf_legend(
    type,
    pos,
    val,
    pal,
    col,
    inches,
    lwd,
    border,
    symbol,
    pt_pch,
    pt_cex,
    title,
    title_cex,
    val_cex,
    val_rnd,
    col_na,
    pt_cex_na,
    pt_pch_na,
    no_data,
    no_data_txt,
    frame,
    bg,
    fg,
    cex
)
```
Arguments

- **type**: type of legend; one of "prop", "choro", "typo", "symb", "prop_line", "grad_line"
- **pos**: position. It can be one of 'topleft', 'top', 'topright', 'right', 'bottomright', 'bottom', 'bottomleft', 'left', 'interactive' or a vector of two coordinates in map units \((c(x, y))\)
- **val**: a vector of values
- **pal**: a set of colors or a palette name (from hcl.colors)
- **col**: a color
- **inches**: size of the biggest symbol (radius for circles, half width for squares) in inches.
- **lwd**: line width(s)
- **border**: border color
- **symbol**: type of symbols, 'circle' or 'square'
- **pt_pch**: pch of the symbols (0:25)
- **pt_cex**: cex of the symbols
- **title**: legend title
- **title_cex**: size of the legend title
- **val_cex**: size of the values in the legend
- **val_rnd**: number of decimal places of the values in the legend
- **col_na**: color for missing values
- **pt_cex_na**: cex of the symbols for missing values
- **pt_pch_na**: pch of the symbols for missing values
- **no_data**: if TRUE a 'missing values' box is plotted
- **no_data_txt**: label for missing values
- **frame**: whether to add a frame to the legend (TRUE) or not (FALSE)
- **bg**: background color
- **fg**: foreground color
- **cex**: size of the legend; 2 means two times bigger

Value

No return value, a legend is displayed.

Examples

```r
mtq <- mf_get_mtw()
mf_map(mfq)
mf_legend(type = "prop", pos = "topright", val = c(1, 5, 10), inches = .3)
mf_legend(
  type = "choro", pos = "bottomright", val = c(10, 20, 30, 40, 50),
  pal = hcl.colors(4, "Reds 2")
)
mfLegend
```
mf_map

Plot a map

Description

This is the main function of the package. mf_map can be used to plot all types of maps. The three main arguments are: x (sf object), var (variable to map), and type (map type).

Relevant arguments and default values are detailed in specific functions.

Maps types:

- **base**, base maps (mf_base);
- **prop**, proportional symbols maps (mf_prop);
- **choro**, choropleth maps (mf_choro);
- **typo**, typology maps (mf_typo);
- **symb**, symbols maps (mf_symb);
- **grad**, graduated symbols maps (mf_grad);
- **prop_choro**, proportional symbols maps with symbols colors based on a quantitative data classification (mf_prop_choro);
- **prop_typo**, proportional symbols maps with symbols colors based on qualitative data (mf_prop_typo);
- **symb_choro**, symbols maps with symbols colors based on a quantitative data classification (mf_symb_choro).

Usage

mf_map(
  x,
  var,
  type = "base",
  breaks,
  nbreaks,
  pal,
alpha = 1,
inches,
val_max,
symbol,
col,
lwd_max,
val_order,
pch,
cex,
border,
lwd,
col_na,
cex_na,
pch_na,
leg_pos,
leg_title,
leg_title_cex,
leg_val_cex,
leg_val_rnd,
leg_no_data,
leg_frame,
add,
...)

Arguments

x            object of class sf or sfc
var          name(s) of the variable(s) to plot
type         one of "base", "prop", "choro", "typo", "symb", "grad", "prop_choro", "prop_typo", "symb_choro"
breaks        either a numeric vector with the actual breaks, or a classification method name (see mf_get_breaks)
nbreaks       number of classes
pal           a set of colors or a palette name (from hcl.colors)
alpha         if pal is a hcl.colors palette name, the alpha-transparency level in the range [0,1]
inches        size of the biggest symbol (radius for circles, half width for squares) in inches.
val_max       maximum value used for proportional symbols
symbol        type of symbols, 'circle' or 'square'
col           color
lwd_max       line width of the largest line
val_order     values order, a character vector that matches var modalities
pch           pch (point type) for symbols
cex           cex (point size) for symbols
mf_map

border  border color
lwd     border width
col_na  color for missing values
cex_na  cex (point size) for NA values
pch_na  pch (point type) for NA values
leg_pos position of the legend, one of ‘topleft’, ‘top’, ‘topright’, ‘right’, ‘bottomright’, ‘bottom’, ‘bottomleft’, ‘left’ or a vector of two coordinates in map units (c(x, y)). If leg_pos = NA then the legend is not plotted. If leg_pos = ‘interactive’ click on the map to choose the legend position.
leg_title legend title
leg_title_cex size of the legend title
leg_val_cex size of the values in the legend
leg_val_rnd number of decimal places of the values in the legend
leg_no_data label for missing values
leg_frame whether to add a frame to the legend (TRUE) or not (FALSE)
add      whether to add the layer to an existing plot (TRUE) or not (FALSE)
...      further parameters from plot for sfc objects

Value

x is (invisibly) returned.

Examples

mtq <- mf_get_mtz()
mf_map(mtq)
mf_map(mtq, var = "POP", type = "prop")
mf_map(mtq, var = "MED", type = "choro")
mf_map(mtq, var = "STATUS", type = "typo")
mf_map(mtq)
mf_map(mtq, var = "STATUS", type = "symb")
mf_map(mtq)
mf_map(mtq, var = "POP", type = "grad")
mf_map(mtq)
mf_map(mtq, var = c("POP", "MED"), type = "prop_choro")
mf_map(mtq)
mf_map(mtq, var = c("POP", "STATUS"), type = "prop_typo")
mf_map(mtq)
mf_map(mtq, var = c("STATUS", "MED"), type = "symb_choro")
`mf_raster`  
*Plot a raster*

**Description**
Plot a raster object (SpatRaster from terra).

**Usage**

```
mf_raster(x, add = FALSE, ...)  
```

**Arguments**
- **x**: a SpatRaster
- **add**: whether to add the layer to an existing plot (TRUE) or not (FALSE).
- **...**: bgalpha, smooth, maxcell or other arguments passed to be passed to `plotRGB` or `plot`

**Value**
No return value, a map is displayed.

**Examples**

```
if (require("terra")) {
  r <- rast(system.file("ex/elev.tif", package = "terra"))
  mf_raster(r)
}
```

---

`mf_scale`  
*Plot a scale bar*

**Description**
Plot a scale bar.

**Usage**

```
mf_scale(size, pos = "bottomright", lwd = 1.5, cex = 0.6, col, unit = "km")
```

**Examples**

```
if (require("terra")) {
  r <- rast(system.file("ex/elev.tif", package = "terra"))
  mf_raster(r)
}
```
**Arguments**

- **size**: size of the scale bar in units (default to km). If size is not set, an automatic size is used (1/10 of the map width)
- **pos**: position. It can be one of 'bottomright', 'bottomleft', 'interactive' or a vector of two coordinates in map units (c(x, y)).
- **lwd**: width of the scale bar
- **cex**: cex of the text
- **col**: color
- **unit**: units used for the scale bar. Can be "mi" for miles, "m" for meters, or "km" for kilometers (default)

**Value**

No return value, a scale bar is displayed.

**Note**

This scale bar is not accurate on unprojected (long/lat) maps.

**Examples**

```r
mtq <- mf_get_mtq()
mf_map(mtq)
mf_map(mtq)
mf_scale()
```

---

**mf_shadow**

*Plot a shadow*

**Description**

Plot the shadow of a polygon layer.

**Usage**

```r
ten <- mf_shadow(x, col = "grey50", cex = 1, add = FALSE)
```

**Arguments**

- **x**: an sf or sfc polygon object
- **col**: shadow color
- **cex**: shadow extent
- **add**: whether to add the layer to an existing plot (TRUE) or not (FALSE)

**Value**

x is (invisibly) returned.
Examples

mtq <- mf_get_mtq()
mf_shadow(mtq)
mf_map(mtq, add = TRUE)

Description

This function sets a map theme. The parameters set by this function are the figure margins, background and foreground colors and some mf_title options.

Usage

mf_theme(x = "default", bg, fg, mar, tab, pos, inner, line, cex, font)

Arguments

x  
name of a map theme. One of "default", "brutal", "ink", "dark", "agolalight", "candy", "darkula", "iceberg", "green", "nevermind", "jsk", "barcelona". If x is used other parameters are ignored.

bg  
background color

fg  
foreground color

mar  
margins

tab  
if TRUE the title is displayed as a 'tab'

pos  
position, one of 'left', 'center', 'right'

inner  
if TRUE the title is displayed inside the plot area.

line  
number of lines used for the title

cex  
cex of the title

font  
font of the title

Details

It is also possible to set a custom theme using a list of arguments (see Examples). Use mf_theme('default') to reset theme settings. mf_theme() returns the current theme settings.

Value

The (invisible) list of theme parameters is returned.
Examples

```r
mtq <- mf_get_mtq()

# built-in theme
mf_theme("green")
mf_map(mtq)
mf_title()

# theme from arguments
mf_theme(
  bg = "darkslategrey", fg = "cornsilk3", mar = c(2, 2, 4, 2),
  tab = FALSE, pos = "center", inner = FALSE,
  line = 2, cex = 2, font = 4
)
mf_map(mtq)
mf_layout()

# theme from list
custom <- list(
  name = "custom",
  bg = "green",
  fg = "red",
  mar = c(2, 2, 2, 2),
  tab = TRUE,
  pos = "center",
  inner = TRUE,
  line = 2,
  cex = 1.5,
  font = 3
)
mf_theme(custom)
mf_map(mtq)
mf_title()

(mf_theme("default"))
```

---

**mf_title**

Plot a title

**Description**

Plot a title

**Usage**

```r
mf_title(txt = "Map Title", pos, tab, bg, fg, cex, line, font, inner)
```
Arguments

- **txt**: title text
- **pos**: position, one of 'left', 'center', 'right'
- **tab**: if TRUE the title is displayed as a 'tab'
- **bg**: background of the title
- **fg**: foreground of the title
- **cex**: cex of the title
- **line**: number of lines used for the title
- **font**: font of the title
- **inner**: if TRUE the title is displayed inside the plot area.

Value

No return value, a title is displayed.

Examples

```r
mtq <- mf_get_mtq()
mf_map(mtq)
mf_title()
```

**mf_worldmap**

*Plot a point on a world map*

Description

Plot a point on a world map.

Usage

```r
mf_worldmap(
  x,
  lon,
  lat,
  water_col = "lightblue",
  land_col = "grey60",
  border_col = "grey40",
  border_lwd = 0.8,
  ...
)
```
Arguments
x object of class sf or sfc
lon longitude
lat latitude
water_col color of the water
land_col color of the land
border_col color of the borders
border_lwd width of the borders
... further parameters related to the plotted point aspect (cex, pch, col...)

Value
No return value, a world map is displayed.

Note
The main part of the code is stolen from @fzenoni (https://gist.github.com/fzenoni/ef23fa6d1ada5e4a91c9ef23b0

Examples
mtq <- mf_get_mtq()
mf_worldmap(mtq)
mf_worldmap(lon = 24, lat = 39)
mf_worldmap(
  lon = 106, lat = 26,
  pch = 4, lwd = 3, cex = 2, col = "tomato4",
  water_col = "#232525", land_col = "#A9B7C6",
  border_col = "white", border_lwd = 1
)
Index

classIntervals, 7, 8
hcl.colors, 16, 18
hcl.pals, 10

mapsf, 2
mf_annotation, 3
mf_arrow, 4, 14
mf_background, 5
mf_base, 17
mf_choro, 17
mf_credits, 5, 14
mf_export, 6
mf_get_breaks, 7, 18
mf_get_links, 8
mf_get_mtg, 9
mf_get_pal, 10
mf_grad, 17
mf_init, 11
mf_inset_off (mf_inset_on), 12
mf_inset_on, 12
mf_label, 13
mf_layout, 14
mf_legend, 15
mf_legend_c, 15
mf_legend_gl, 15
mf_legend_p, 15
mf_legend_pl, 15
mf_legend_s, 15
mf_legend_t, 15
mf_map, 17
mf_prop, 17
mf_prop_choro, 17
mf_prop_typo, 17
mf_raster, 20
mf_scale, 14, 20
mf_shadow, 21
mf_symb, 17
mf_symb_choro, 17
mf_theme, 22

mf_title, 14, 22, 23
mf_typo, 17
mf_worldmap, 12, 24
plot, 19, 20
plotRGB, 20
quantile, 8
rasterImage, 5
text, 3, 13