Package 'mapsf'

October 22, 2024

Title Thematic Cartography

Version 0.12.0

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)

URL https://riatelab.github.io/mapsf/

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

Depends R (>= 3.6.0)

Imports classInt, graphics, maplegend, s2, sf, stats, utils, grDevices

Suggests terra, Ckmeans.1d.dp, png, jpeg, lwgeom, knitr, rmarkdown,
tinytest, covr

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Timothée Giraud [cre, aut] (<https://orcid.org/0000-0002-1932-3323>),
Hugues Pecout [ctb] (<https://orcid.org/0000-0002-0246-0954>, Logo),
Ronan Ysebaert [ctb] (<https://orcid.org/0000-0002-7344-5911>, Cheat
sheet),
Ian Fellows [cph] (No overlap algorithm for labels, from wordcloud
package),
Jim Lemon [cph] (Arc drawing algorithm for annotations, from plotrix
package)

Maintainer Timothée Giraud <timothee.giraud@cnrs.fr>

Repository CRAN

Date/Publication 2024-10-22 16:10:02 UTC

Contents

---

mapsf   *Package description*

Description

Create and integrate thematic maps in your workflow. This package helps to design various carto-graphic 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.

A "Get Started" **vignette** contains commented scripts on how to create various maps: vignette(topic = "mapsf", package = "mapsf")

Symbology

These functions display cartographic layers.

- mf_map() Plot a map
- mf_label() Plot labels
- mf_raster() Plot a raster
- mf_graticule() Plot graticules

Map layout

These functions are dedicated to the map layout design.

- mf_theme() Set a theme
- mf_shadow() Plot a shadow
- mf_background() Plot a background image
- mf_annotation() Plot an annotation
- mf_arrow() Plot a north arrow
- mf_credits() Plot credits
- mf_layout() Plot a map layout
- mf_title() Plot a title
- mf_scale() Plot a scale bar
- mf_inset_on() / mf_inset_off() Plot an inset
- mf_worldmap() Plot a point on a world map
- mf_legend() Plot a legend

Utility functions

- mf_export() Export a map
- mf_distr() Plot a distribution
- mf_get_links() Get a link layer from a data.frame of links
- mf_get_pal() Get color palettes
- mf_get_breaks() Get class intervals
- mf_get_mtq() Get the 'mtq' dataset
- mf_get_ratio() Get map width and height values
- mf_get_pencil() Get a pencil layer from polygons
- mf_get_borders() Get a border layer from polygons

Author(s)

Maintainer: Timothée Giraud <timothee.giraud@cnrs.fr> (ORCID)

Other contributors:

- Hugues Pecout (ORCID) (Logo) [contributor]
- Ronan Ysebaert (ORCID) (Cheat sheet) [contributor]
- Ian Fellows (No overlap algorithm for labels, from wordcloud package) [copyright holder]
- Jim Lemon (Arc drawing algorithm for annotations, from plotrix package) [copyright holder]

See Also

Useful links:

- <https://riatelab.github.io/mapsf/>
- Report bugs at <https://github.com/riatelab/mapsf/issues/>

---

mf_annotation   *Plot an annotation*

Description

Plot an annotation on a map.

Usage

```
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)) or "interactive"
txt	the text to display
pos	position of the text, one of "topleft", "topright", "bottomright", "bottomleft"
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

```
mtq <- mf_get_mtq()
mf_map(mtq)
mf_annotation(
  x = c(711167.8, 1614764),
  txt = "Look!\nImportant feature\nhere!",
  pos = "bottomleft", cex = 1.2, font = 2,
  halo = TRUE, s = 1.5
)

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

---

mf_arrow   *Plot a north arrow*

Description

Plot a north arrow.

Usage

```
mf_arrow(
  pos = "topleft",
  col = getOption("mapsf.fg"),
  cex = 1,
  adj = c(0, 0),
  align,
  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
cex	arrow size
adj	adjust the postion of the north arrow in x and y directions
align	object of class sf or sfc used to adjust the arrow to the real north
adjust	deprecated, see align

Value

No return value, a north arrow is displayed.

Examples

```
mtq <- mf_get_mtq()
mf_map(mtq)
mf_arrow(pos = "topright")
```

---

mf_background   *Plot a background image*

Description

Plot a background image on an existing plot

Usage

```
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

```
mtq <- mf_get_mtq()
mf_map(mtq, col = NA, border = NA)
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` *Plot credits*

Description

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

Usage

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

Arguments

txt	text of the credits, use '\n' to add line breaks
pos	position, one of 'bottomleft', 'bottomright' or 'rightbottom'
col	color
cex	cex of the credits
font	font of the credits
bg	background color

Value

No return value, credits are displayed.

Examples

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

---

mf_distr   *Plot a distribution*

Description

This function displays a histogram, a box plot, a strip chart and a density curve on the same plot.

Usage

```
mf_distr(x, nbins, bw)
```

Arguments

x	a numeric variable
nbins	number of bins in the histogram
bw	bandwidth of the density curve

Value

The number of bins of the histogram and the bandwidth of the density curve are (invisibly) returned in a list.

Examples

```
(mf_distr(rnorm(1000)))
mf_distr(rbeta(1000, .6, 7))
mf_distr(rbeta(1000, 5, .6))
```

---

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.
Always use add = TRUE in mf_map calls following an mf_export call.
Use dev.off to finish the export (see Examples).

Usage

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

Arguments

x	object of class `sf`, `sfc` or `SpatRaster`
filename	path to the exported file. If the file extention 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 (deprecated)
export	deprecated

Value

No return value, a map file is initiated (in PNG or SVG format).

Examples

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

---

mf_get_borders   *Get a border layer from polygons*

Description

This function extracts borders between contiguous polygons.

Usage

```
mf_get_borders(x)
```

Arguments

x   an sf object of POLYGONS, using a projected CRS

Value

An sf object (MULTILINESTRING) of borders is returned.

Note

If the polygon layer contains topology errors (such as contiguous polygons not sharing exactly the same boundary) the function may not return all boundaries correctly. It is possible to use st_snap() or other functions to try and correct these errors.

Examples

```
mtq <- mf_get_mtq()
mtq_b <- mf_get_borders(mtq)
mf_map(mtq)
mf_map(mtq_b, col = 1:5, lwd = 4, add = TRUE)
```

---

mf_get_breaks   *Get class intervals*

Description

A function to classify continuous variables.

This function is a wrapper for [classIntervals](#) with some additional methods.

Usage

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

Arguments

x	a vector of numeric values. NA and Inf values are not used in the classification.
nbreaks	a number of classes
breaks	a classification method; one of "fixed", "sd", "equal", "pretty", "quantile", "kmeans", "hclust", "bclust", "fisher", "jenks", "dpih", "q6", "Q6", geom", "arith", "em", "msd" or "ckmeans" (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

Details

"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.

The "jenks", "fisher" and "ckmeans" methods are based on the same concept of **natural breaks** and and produce similar groupings.

- The "jenks" method produces class boundaries falling on data points and is slow.
- The "fisher" method produces class boundaries located more conveniently between data points, and is faster than the "jenks" method.
- The "ckmeans" method produces exactly the same class boundaries as the "fisher" method, but is much faster. It uses the optimal univariate k-means method from the Ckmeans.1d.dp package. If the "ckmeans" method is selected but the Ckmeans.1d.dp package is not installed then the "fisher" method is used.

The relative speeds of these three methods may vary depending on the number of data points and the number of classes.

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

The "Q6" method uses the following quantile probabilities: 0, 0.05, 0.25, 0.5, 0.75, 0.95, 1.

The "geom" method is based on a geometric progression along the variable values, all values must be strictly greater than zero.

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 nbreaks 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

See Also

[classIntervals](#)

Examples

```
mtq <- mf_get_mtq()
mf_get_breaks(x = mtq$MED, nbreaks = 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)
```

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

```
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

```
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_get_pencil *Get a pencil layer from polygons*

Description

Create a pencil layer. This function transforms a POLYGON or MULTIPOLYGON sf object into a MULTILINESTRING one.

Usage

```
mf_get_pencil(x, size = 100, buffer = 0, lefthanded = TRUE, clip = FALSE)
```

Arguments

x	an sf object, a simple feature collection (POLYGON or MULTIPOLYGON).
size	density of the penciling. Median number of points used to build the MULTI-LINESTRING.
buffer	buffer around each polygon. This buffer (in map units) is used to take sample points. A negative value adds a margin between the penciling and the original polygons borders
lefthanded	if TRUE the penciling is done left-handed style.
clip	if TRUE, the penciling is cut by the original polygon.

Value

A MULTILINESTRING sf object is returned.

Examples

```
mtq <- mf_get_mtq()
mtq_pencil <- mf_get_pencil(x = mtq, clip = FALSE)
mf_map(mtq)
mf_map(mtq_pencil, add = TRUE)
```

---

mf_get_ratio   *Get map width and height values*

Description

This function is to be used to get width and height values for maps created in reports (*.Rmd, *.qmd).
It uses the width / height ratio of a spatial object bounding box to find a matching ratio for the map.
If width is specified, then height is deduced from the width / height ratio of x, figure margins and title size.
If height is specified, then width is deduced from the width / height ratio of x, figure margins and title size.

Usage

```
mf_get_ratio(
  x,
  width,
  height,
  res = 96,
  expandBB = rep(0, 4),
  theme = mf_theme()
)
```

Arguments

x	object of class sf, sfc or SpatRaster
width	width of the figure (inches), use only one of width or height
height	height of the figure (inches), use only one of width or height
res	resolution
expandBB	fractional values to expand the bounding box with, in each direction (bottom, left, top, right)
theme	theme used for the map

Value

Width and height are returned in inches.

Examples

```
mtq <- mf_get_mtq()
mf_get_ratio(x = mtq, width = 5)
```

---

mf_graticule   *Plot graticules*

Description

Display graticules and labels on a map.

Usage

```
mf_graticule(
  x,
  col = col,
  lwd = 1,
  lty = 1,
  expandBB = rep(0, 4),
  label = TRUE,
  pos = c("top", "left"),
  cex = 0.7,
  add = TRUE
)
```

Arguments

x	object of class sf, sfc or SpatRaster
col	graticules and label color
lwd	graticules line width
lty	graticules line type

expandBB	fractional values to expand the bounding box with, in each direction (bottom, left, top, right)
label	whether to add labels (TRUE) or not (FALSE)
pos	labels positions ("bottom", "left", "top" and / or "right")
cex	labels size
add	whether to add the layer to an existing plot (TRUE) or not (FALSE)

Value

An (invisible) layer of graticules is returned (LINESTRING).

Use of graticules

From [st_graticule](): "In cartographic visualization, the use of graticules is not advised, unless the graphical output will be used for measurement or navigation, or the direction of North is important for the interpretation of the content, or the content is intended to display distortions and artifacts created by projection. Unnecessary use of graticules only adds visual clutter but little relevant information. Use of coastlines, administrative boundaries or place names permits most viewers of the output to orient themselves better than a graticule."

Examples

```
mtq <- mf_get_mtq()
mf_map(mtq, expandBB = c(0, .1, .1, 0))
mf_graticule(mtq)

mf_graticule(
  x = mtq,
  col = "coral4",
  lwd = 2,
  lty = 2,
  expandBB = c(.1, 0, 0, .1),
  label = TRUE,
  pos = c("right", "bottom"),
  cex = .8,
  add = FALSE
)
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

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

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.

Examples

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

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()
```

---

mf_label	*Plot labels*

Description

Put labels on a map.

Usage

```
mf_label(
  x,
  var,
  col,
  cex = 0.7,
  overlap = TRUE,
  lines = TRUE,
  halo = FALSE,
  bg,
  r = 0.1,
  q = 1,
  ...
)
```

Arguments

x	object of class sf
var	name(s) of the variable(s) to plot
col	labels color, it can be a single color or a vector of colors
cex	labels cex, it can be a single size or a vector of sizes
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, a 'halo' is displayed around the text and additional arguments bg and r can be modified to set the color and width of the halo.
bg	halo color, it can be a single color or a vector of colors
r	width of the halo, it can be a single value or a vector of values
q	quality of the non overlapping labels placement. Possible values are 0 (quick results), 1 (reasonable quality and speed), 2 (better quality), 3 (insane quality, can take a lot of time).
...	further text arguments.

Value

No return value, labels are displayed.

Examples

```
mtq <- mf_get_mtq()
mf_map(mtq)
mtq$cex <- c(rep(.8, 8), 2, rep(.8, 25))
mf_label(
  x = mtq, var = "LIBGEO",
  col = "grey10", halo = TRUE, cex = mtq$cex,
  overlap = FALSE, lines = FALSE
)
```

mf_layout   *Plot a map layout*

Description

Plot a map layout (title, credits, scalebar, north arrow, frame).

This function uses mf_title, mf_credits, mf_scale and mf_arrow with default values.

Usage

```
mf_layout(
  title = "Map Title",
  credits = "Authors & Sources",
  scale = TRUE,
  arrow = TRUE,
  frame = FALSE
)
```

Arguments

title	title of the map
credits	credits
scale	display a scale bar
arrow	display an arrow
frame	display a frame

Value

No return value, a map layout is displayed.

Examples

```
mtq <- mf_get_mtq()
mf_map(mtq)
mf_layout()
```

mf_legend	*Plot a legend*

Description

Plot different types of legend. The "type" argument defines the legend type. Please note that some arguments are available for all types of legend and some others are only relevant for specific legend types (see Details). mf_legend() is a wrapper for maplegend::leg().

Usage

```
mf_legend(
  type,
  val,
  pos = "left",
  pal = "Inferno",
  alpha = 1,
  col = "tomato4",
  inches = 0.3,
  symbol = "circle",
  self_adjust = FALSE,
  lwd = 0.7,
  border = "#333333",
  pch = seq_along(val),
  cex = rep(1, length(val)),
  title = "Legend Title",
  title_cex = 0.8 * size,
  val_cex = 0.6 * size,
  val_rnd = 0,
  col_na = "white",
  cex_na = 1,
  pch_na = 4,
  no_data = FALSE,
  no_data_txt = "No Data",
  box_border = "#333333",
  box_cex = c(1, 1),
  horiz = FALSE,
  frame_border,
  frame = FALSE,
  bg,
  fg,
  size = 1,
  return_bbox = FALSE,
  adj = c(0, 0),
  pt_pch,
  pt_cex,
  pt_cex_na,
```

```
    pt_pch_na
)
```

Arguments

`type`	type of legend:

- **prop** for proportional symbols,
- **choro** for choropleth maps,
- **cont** for continuous maps (e.g. raster),
- **typo** for typology maps,
- **symb** for symbols maps,
- **prop_line** for proportional lines maps,
- **grad_line** for graduated lines maps.

`val`	vector of value(s) (for "prop" and "prop_line", at least c(min, max) for "cont"), vector of categories (for "symb" and "typo"), break labels (for "choro" and "grad_line").
`pos`	position of the legend. 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)).
`pal`	a color palette name or a vector of colors
`alpha`	if `pal` is a hcl.colors palette name, the alpha-transparency level in the range [0,1]
`col`	color of the symbols (for "prop") or color of the lines (for "prop_line" and "grad_line")
`inches`	size of the largest symbol (radius for circles, half width for squares) in inches
`symbol`	type of symbols, 'circle' or 'square'
`self_adjust`	if TRUE values are self-adjusted to keep min, max and intermediate rounded values
`lwd`	width(s) of the symbols borders (for "prop" and "symb"), width of the largest line (for "prop_line"), vector of line width (for "grad_line")
`border`	symbol border color(s)
`pch`	type(s) of the symbols (0:25)
`cex`	size(s) of the symbols
`title`	title of the legend
`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
`cex_na`	size of the symbols for missing values
`pch_na`	type of the symbols for missing values
`no_data`	if TRUE a "missing value" box is plotted
`no_data_txt`	label for missing values

`box_border`	border color of legend boxes
`box_cex`	width and height size expansion of boxes, (or offset between circles for "prop" legends with horiz = TRUE)
`horiz`	if TRUE plot an horizontal legend
`frame_border`	border color of the frame
`frame`	if TRUE the legend is plotted within a frame
`bg`	background color of the legend
`fg`	foreground color of the legend
`size`	size of the legend; 2 means two times bigger
`return_bbox`	return only bounding box of the legend. No legend is plotted.
`adj`	adjust the postion of the legend in x and y directions
`pt_pch`	deprecated
`pt_cex`	deprecated
`pt_cex_na`	deprecated
`pt_pch_na`	deprecated

Details

Some arguments are available for all types of legend: `val, pos, title, title_cex, val_cex, frame, bg, fg, size, adj, return_bbox`).

Relevant arguments for each specific legend types:

- `mf_legend(type = "prop", val, inches, symbol, col, lwd, border, val_rnd, self_adjust, horiz)`

- `mf_legend(type = "choro", val, pal, val_rnd, col_na, no_data, no_data_txt, box_border, horiz)`

- `mf_legend(type = "cont", val, pal, val_rnd, col_na, no_data, no_data_txt, box_border, horiz)`

- `mf_legend(type = "typo", val, pal, col_na, no_data, no_data_txt, box_border)`

- `mf_legend(type = "symb", val, pal, pch, cex, lwd, pch_na, cex_na, col_na, no_data, no_data_txt)`

- `mf_legend(type = "prop_line", val, col, lwd, val_rnd)`

- `mf_legend(type = "grad_line", val, col, lwd, val_rnd)`

Value

No value is returned, a legend is displayed (except if `return_bbox` is used).

Examples

```
mtq <- mf_get_mtq()
mf_map(mtq)
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")
)
mf_legend(
  type = "typo", pos = "topleft", val = c("A", "B", "C", "D"),
  pal = hcl.colors(4, "Dynamic")
)
mf_legend(
  type = "symb", pos = "bottomleft", val = c("A", "B", "C"),
  pch = 21:23, cex = c(1, 2, 2),
  pal = hcl.colors(3, "Dynamic")
)
mf_legend(
  type = "grad_line", pos = "top", val = c(1, 2, 3, 4, 10, 15),
  lwd = c(0.2, 2, 4, 5, 10)
)
mf_legend(type = "prop_line", pos = "bottom", lwd = 20, val = c(5, 50, 100))
```

---

mf_map	*Plot a map*

Description

mf_map() is the main function of the package, it displays map layers on a georeferenced plot.

mf_map() has three main arguments:

- x, an sf object;
- var, the name(s) of a variable(s) to map;
- type, the map layer type.

Many parameters are available to fine tune symbologies and legends.

Relevant arguments and default values are different for each map type and are described in the "Details" section.

Usage

```
mf_map(x, var, type = "base",
       breaks, nbreaks, pal, alpha, rev, inches, val_max, symbol, col,
       lwd_max, val_order, pch, cex, border, lwd, col_na, cex_na, pch_na,
       expandBB, add,
       leg_pos, leg_title, leg_title_cex, leg_val_cex, leg_val_rnd,
       leg_no_data, leg_frame, leg_frame_border, leg_horiz, leg_adj, leg_bg,
       leg_fg, leg_size, leg_border, leg_box_border, leg_box_cex, ...)
```

Arguments

x	object of class sf or sfc
var	name(s) of the variable(s) to plot
type	• **base**: base maps
	• **prop**: proportional symbols maps
	• **choro**: choropleth maps
	• **typo**: typology maps
	• **symb**: symbols maps
	• **grad**: graduated symbols maps
	• **prop_choro**: proportional symbols maps with symbols colors based on a quantitative data classification
	• **prop_typo**: proportional symbols maps with symbols colors based on qualitative data
	• **symb_choro**: symbols maps with symbols colors based on a quantitative data classification
breaks	either a numeric vector with the actual breaks, or a classification method name (see [mf_get_breaks](#) and Details)
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]
rev	if pal is a [hcl.colors](#) palette name, whether the ordering of the colors should be reversed (TRUE) or not (FALSE)
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	point type
cex	point size
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
expandBB	fractional values to expand the bounding box with, in each direction (bottom, left, top, right)
add	whether to add the layer to an existing plot (TRUE) or not (FALSE)

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 onthe 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)
leg_frame_border	
	border color of the legend frame
leg_horiz	display the legend horizontally (for proportional symbols and choropleth types)
leg_adj	adjust the postion of the legend in x and y directions
leg_bg	color of the legend backgournd
leg_fg	color of the legend foreground
leg_size	size of the legend; 2 means two times bigger
leg_border	symbol border color(s)
leg_box_border	border color of legend boxes
leg_box_cex	width and height size expansion of boxes
...	further parameters from plot for sfc objects

Details

Relevant arguments and default values for each map types::

base: displays sf objects geometries.

```
mf_map(x, col = "grey80", pch = 20, cex = 1, border = "grey20", lwd = 0.7,
       expandBB, add = FALSE, ...)
```

prop: displays symbols with areas proportional to a quantitative variable (stocks). inches is used to set symbols sizes.

```
mf_map(x, var, type = "prop", inches = 0.3, val_max, symbol = "circle",
       col = "tomato4", lwd_max = 20, border = getOption("mapsf.fg"),
       lwd = 0.7, expandBB, add = TRUE,
       leg_pos = mf_get_leg_pos(x), leg_title = var,
       leg_title_cex = 0.8, leg_val_cex = 0.6, leg_val_rnd = 0,
       leg_frame = FALSE, leg_frame_border = getOption("mapsf.fg"),
       leg_horiz = FALSE, leg_adj = c(0, 0),
       leg_bg = getOption("mapsf.bg"), leg_fg = getOption("mapsf.fg"),
       leg_size = 1)
```

choro: areas are shaded according to the variation of a quantitative variable. Choropleth maps are used to represent ratios or indices. nbreaks, and breaks allow to set the variable classification. Colors palettes, defined with pal, can be created with mf_get_pal() or can use palette names from hcl.pals().

```
mf_map(x, var, type = "choro", breaks = "quantile", nbreaks, pal = "Mint",
       alpha = 1, rev = FALSE, pch = 21, cex = 1,
       border = getOption("mapsf.fg"), lwd = 0.7, col_na = "white",
       cex_na = 1, pch_na = 4, expandBB, add = FALSE,
       leg_pos = mf_get_leg_pos(x), leg_title = var, leg_title_cex = 0.8,
       leg_val_cex = 0.6, leg_val_rnd = 2, leg_no_data = "No data",
       leg_frame = FALSE, leg_frame_border = getOption("mapsf.fg"),
       leg_horiz = FALSE, leg_adj = c(0, 0), leg_bg = getOption("mapsf.bg"),
       leg_fg = getOption("mapsf.fg"), leg_size = 1,
       leg_box_border = getOption("mapsf.fg"), leg_box_cex = c(1, 1))
```

typo: displays a typology map of a qualitative variable. val_order is used to set modalities order in the legend.

```
mf_map(x, var, type = "typo", pal = "Dynamic", alpha = 1, rev = FALSE,
       val_order,border = getOption("mapsf.fg"), pch = 21, cex = 1,
       lwd = 0.7, cex_na = 1, pch_na = 4, col_na = "white",
       leg_pos = mf_get_leg_pos(x), leg_title = var, leg_title_cex = 0.8,
       leg_val_cex = 0.6, leg_no_data = "No data", leg_frame = FALSE,
       leg_frame_border = getOption("mapsf.fg"), leg_adj = c(0, 0),
       leg_size = 1, leg_box_border = getOption("mapsf.fg"),
       leg_box_cex = c(1, 1), leg_fg = getOption("mapsf.fg"),
       leg_bg = getOption("mapsf.bg"), add = FALSE)
```

symb: displays the different modalities of a qualitative variable as symbols.

```
mf_map(x, var, type = "symb", pal = "Dynamic", alpha = 1, rev = FALSE,
       border = getOption("mapsf.fg"), pch, cex = 1, lwd = 0.7,
       col_na = "grey", pch_na = 4, cex_na = 1, val_order,
       leg_pos = mf_get_leg_pos(x), leg_title = var, leg_title_cex = 0.8,
       leg_val_cex = 0.6, leg_val_rnd = 2, leg_no_data = "No data",
       leg_frame = FALSE, leg_frame_border = getOption("mapsf.fg"),
       leg_adj = c(0, 0), leg_fg = getOption("mapsf.fg"),
       leg_bg = getOption("mapsf.bg"), leg_size = 1, add = TRUE)
```

grad: displays graduated symbols. Sizes classes are set with breaks and nbreaks. Symbol sizes are set with cex.

```
mf_map(x, var, type = "grad", breaks = "quantile", nbreaks = 3, col = "tomato4",
       border = getOption("mapsf.fg"), pch = 21, cex, lwd,
       leg_pos = mf_get_leg_pos(x), leg_title = var, leg_title_cex = 0.8,
       leg_val_cex = 0.6, leg_val_rnd = 2, leg_frame = FALSE,
       leg_adj = c(0, 0), leg_size = 1, leg_border = border,
       leg_box_cex = c(1, 1), leg_fg = getOption("mapsf.fg"),
```

```
              leg_bg = getOption("mapsf.bg"), leg_frame_border = getOption("mapsf.fg"),
              add = TRUE)
```

prop_choro: displays symbols with sizes proportional to values of a first variable and colored to reflect the classification of a second quantitative variable.

```
mf_map(x, var, type = "prop_choro", inches = 0.3, val_max, symbol = "circle",
        pal = "Mint", alpha = 1, rev = FALSE, breaks = "quantile", nbreaks,
        border = getOption("mapsf.fg"), lwd = 0.7, col_na = "white",
        leg_pos = mf_get_leg_pos(x, 1), leg_title = var,
        leg_title_cex = c(0.8, 0.8), leg_val_cex = c(0.6, 0.6),
        leg_val_rnd = c(0, 2), leg_no_data = "No data",
        leg_frame = c(FALSE, FALSE), leg_frame_border = getOption("mapsf.fg"),
        leg_horiz = c(FALSE, FALSE), leg_adj = c(0, 0),
        leg_fg = getOption("mapsf.fg"), leg_bg = getOption("mapsf.bg"),
        leg_size = 1, leg_box_border = getOption("mapsf.fg"),
        leg_box_cex = c(1, 1), add = TRUE)
```

prop_typo: displays symbols with sizes proportional to values of a first variable and colored to reflect the modalities of a second qualitative variable.

```
mf_map(x, var, type = "prop_typo", inches = 0.3, val_max, symbol = "circle",
        pal = "Dynamic", alpha = 1, rev = FALSE, val_order,
        border = getOption("mapsf.fg"), lwd = 0.7, lwd_max = 15,
        col_na = "white",
        leg_pos = mf_get_leg_pos(x, 1), leg_title = var,
        leg_title_cex = c(0.8, 0.8), leg_val_cex = c(0.6, 0.6),
      leg_val_rnd = c(0), leg_no_data = "No data", leg_frame = c(FALSE, FALSE),
        leg_frame_border = getOption("mapsf.fg"), leg_horiz = FALSE,
        leg_adj = c(0, 0), leg_fg = getOption("mapsf.fg"),
        leg_bg = getOption("mapsf.bg"), leg_size = 1,
        leg_box_border = getOption("mapsf.fg"), leg_box_cex = c(1, 1),
        add = TRUE)
```

symb_choro: displays the different modalities of a first qualitative variable as symbols colored to reflect the classification of a second quantitative variable.

```
mf_map(x, var, type = "symb_choro", pal = "Mint", alpha = 1, rev = FALSE,
        breaks = "quantile", nbreaks, border = getOption("mapsf.fg"),
        pch, cex = 1, lwd = 0.7, pch_na = 4, cex_na = 1, col_na = "white",
        val_order,
        leg_pos = mf_get_leg_pos(x, 1), leg_title = var,
        leg_title_cex = c(0.8, 0.8), leg_val_cex = c(0.6, 0.6),
        leg_val_rnd = 2, leg_no_data = c("No data", "No data"),
        leg_frame = c(FALSE, FALSE), leg_frame_border = getOption("mapsf.fg"),
        leg_horiz = FALSE, leg_adj = c(0, 0), leg_fg = getOption("mapsf.fg"),
        leg_bg = getOption("mapsf.bg"), leg_size = 1,
        leg_box_border = getOption("mapsf.fg"), leg_box_cex = c(1, 1),
        add = TRUE)
```

Class boundaries:

Breaks defined by a numeric vector or a classification method are left-closed: breaks defined by c(2, 5, 10, 15, 20) will be mapped as [2 - 5[, [5 - 10[, [10 - 15[, [15 - 20].

Value

x is (invisibly) returned.

Examples

```
library(mapsf)
mtq <- mf_get_mtq()
# basic examples
# type = "base"
mf_map(mtq)
# type = "prop"
mf_map(mtq)
mf_map(mtq, var = "POP", type = "prop")
# type = "choro"
mf_map(mtq, var = "MED", type = "choro")
# type = "typo"
mf_map(mtq, "STATUS", "typo")
# type = "symb"
mf_map(mtq)
mf_map(mtq, "STATUS", "symb")
# type = "grad"
mf_map(mtq)
mf_map(mtq, var = "POP", type = "grad")
# type = "prop_choro"
mf_map(mtq)
mf_map(mtq, var = c("POP", "MED"), type = "prop_choro")
# type = "prop_typo"
mf_map(mtq)
mf_map(mtq, var = c("POP", "STATUS"), type = "prop_typo")
# type = "symb_choro
mf_map(mtq)
mf_map(mtq, var = c("STATUS", "MED"), type = "symb_choro")




# detailed examples
# type = "base"
mf_map(mtq, type = "base", col = "lightblue", lwd = 1.5, lty = 2)

# type = "prop"
mf_map(mtq)
mf_map(
  x = mtq, var = "POP", type = "prop",
  inches = .4, symbol = "circle", val_max = 90000,
```

```
  col = "lightblue", border = "grey", lwd = 1,
  leg_pos = "right", leg_title = "Population",
  leg_title_cex = 1, leg_val_cex = .8, leg_val_rnd = 0,
  leg_frame = TRUE, add = TRUE
)

# type = "choro"
mtq[6, "MED"] <- NA
mf_map(
  x = mtq, var = "MED", type = "choro",
  col_na = "grey80", pal = "Cividis",
  breaks = "quantile", nbreaks = 4, border = "white",
  lwd = .5, leg_pos = "topleft",
  leg_title = "Median Income", leg_title_cex = 1.1,
  leg_val_cex = 1, leg_val_rnd = -2, leg_no_data = "No data",
  leg_frame = TRUE, leg_adj = c(0, -3)
)

# type = "typo"
mtq[4, "STATUS"] <- NA
mf_map(
  x = mtq, var = "STATUS", type = "typo",
  pal = c("red", "blue", "yellow"), lwd = 1.1,
  val_order = c("Prefecture", "Sub-prefecture", "Simple municipality"),
  col_na = "green", border = "brown",
  leg_pos = "bottomleft",
  leg_title = "Status", leg_title_cex = 1.1,
  leg_val_cex = 1, leg_no_data = "No data",
  leg_frame = TRUE, add = FALSE
)

# type = "symb"
mf_map(mtq)
mf_map(
  x = mtq, var = "STATUS", type = "symb",
  pch = c(21:23), pal = c("red1", "tan1", "khaki1"),
  border = "grey20", cex = c(2, 1.5, 1), lwd = .5,
  val_order = c("Prefecture", "Sub-prefecture", "Simple municipality"),
  pch_na = 24, col_na = "blue", leg_frame = TRUE
)

# type = "grad"
mf_map(mtq)
mf_map(
  x = mtq, var = "POP", type = "grad",
  pch = 22, breaks = "quantile", nbreaks = 4, lwd = 2, border = "blue",
  cex = c(.75, 1.5, 3, 5), col = "lightgreen"
)

# type = "prop_choro"
mf_map(mtq)
mf_map(
  x = mtq, var = c("POP", "MED"), type = "prop_choro",
```

```
    inches = .35, border = "tomato4",
    val_max = 90000, symbol = "circle", col_na = "white", pal = "Cividis",
    breaks = "equal", nbreaks = 4, lwd = 4,
    leg_pos = "bottomleft",
    leg_title = c("Population", "Median Income"),
    leg_title_cex = c(0.8, 1),
    leg_val_cex = c(.7, .9),
    leg_val_rnd = c(0, 0),
    leg_no_data = "No data",
    leg_frame = c(TRUE, TRUE),
    add = TRUE
)

# type = "prop_typo"
mf_map(mtq)
mf_map(
    x = mtq, var = c("POP", "STATUS"), type = "prop_typo",
    inches = .35, border = "tomato4",
    val_max = 90000, symbol = "circle", col_na = "white", pal = "Dynamic",
    lwd = 2,
    leg_pos = c("bottomright", "bottomleft"),
    leg_title = c("Population", "Municipality\nstatus"),
    leg_title_cex = c(0.9, 0.9),
    leg_val_cex = c(.7, .7),
    val_order = c("Prefecture", "Sub-prefecture", "Simple municipality"),
    leg_no_data = "No dada",
    leg_frame = c(TRUE, TRUE),
    add = TRUE
)

# type = "symb_choro"
mf_map(mtq)
mf_map(
    x = mtq, c("STATUS", "MED"), type = "symb_choro",
    pal = "Reds 3", breaks = "quantile", nbreaks = 4,
    pch = 21:23, cex = c(3, 2, 1),
    pch_na = 25, cex_na = 1.5, col_na = "blue",
    val_order = c(
        "Prefecture",
        "Sub-prefecture",
        "Simple municipality"
    )
)
```

---

mf_raster   *Plot a raster*

Description

Plot a raster object (SpatRaster from terra).

Usage

```
mf_raster(
  x,
  type,
  nbreaks,
  breaks = "equal",
  val_order,
  pal,
  expandBB = rep(0, 4),
  alpha = 1,
  rev = FALSE,
  leg_pos = "right",
  leg_title = names(x),
  leg_title_cex = 0.8,
  leg_val_cex = 0.6,
  leg_val_rnd = 1,
  leg_frame = FALSE,
  leg_frame_border = getOption("mapsf.fg"),
  leg_horiz = FALSE,
  leg_adj = c(0, 0),
  leg_box_border = "#333333",
  leg_box_cex = c(1, 1),
  leg_fg = getOption("mapsf.fg"),
  leg_bg = getOption("mapsf.bg"),
  leg_size = 1,
  add = FALSE,
  ...
)
```

Arguments

x	a SpatRaster
type	type of raster map, one of "continuous", "classes", or "interval". Default type for a numeric and categorial raster are "continuous" and "classes" respectively.
nbreaks	number of classes
breaks	either a numeric vector with the actual breaks (for type = "continuous" and type = "interval"), or a classification method name (for type = "interval" only; see [mf_get_breaks](#) for classification methods)
val_order	values order, a character vector that matches var modalities
pal	a set of colors or a palette name (from [hcl.colors](#))
expandBB	fractional values to expand the bounding box with, in each direction (bottom, left, top, right)
alpha	if pal is a [hcl.colors](#) palette name, the alpha-transparency level in the range [0,1]
rev	if pal is a [hcl.colors](#) palette name, whether the ordering of the colors should be reversed (TRUE) or not (FALSE)

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 onthe 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_frame	whether to add a frame to the legend (TRUE) or not (FALSE)
leg_frame_border	
	border color of the legend frame
leg_horiz	display the legend horizontally
leg_adj	adjust the postion of the legend in x and y directions
leg_box_border	border color of legend boxes
leg_box_cex	width and height size expansion of boxes
leg_fg	color of the legend foreground
leg_bg	color of the legend backgournd
leg_size	size of the legend; 2 means two times bigger
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

x is (invisibly) returned.

Examples

```
if (require("terra")) {
  # multi band
  logo <- rast(system.file("ex/logo.tif", package = "terra"))
  mf_raster(logo)

  # one band
  elev <- rast(system.file("ex/elev.tif", package = "terra"))

  ## continuous
  mf_raster(elev)
  mf_raster(elev,
    pal = "Burg", expandBB = c(.2, 0, 0, 0),
    leg_pos = "bottom", leg_horiz = TRUE
  )

  ## continuous with colors and breaks
  mf_raster(elev,
    type = "continuous",
```

```
    breaks = c(141, 400, 547),
    pal = c("darkseagreen1", "black", "red")
  )

  ## interval
  mf_raster(elev,
    type = "interval",
    nbreaks = 5, breaks = "equal", pal = "Teal"
  )

  ## classes
  elev2 <- classify(elev, c(140, 400, 450, 549))
  lev_evel <- data.frame(ID = 0:2, elevation = c("Low", "High", "Super High"))
  levels(elev2) <- lev_evel
  mf_raster(elev2)
  mf_raster(elev2,
    pal = c("salmon4", "olivedrab", "yellow3"),
    val_order = c("Super High", "High", "Low")
  )
}
```

---

mf_scale   *Plot a scale bar*

Description

Plot a scale bar.

Usage

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

Arguments

size   size of the scale bar in scale units (scale_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	line width of the scale bar
cex	size of the scale bar text
col	color of the scale bar (line and text)
crs_units	units used in the CRS of the currently plotted layer. Possible values are "m" and "ft" (see Details).
scale_units	units used for the scale bar. Can be "mi" for miles, "ft" for feet, "m" for meters, or "km" for kilometers (default).
x	object of class crs, sf or sfc. If set, the CRS of x will be used instead of `crs_units` to define CRS units.
unit	deprecated, use scale_units instead

Details

Most CRS use the meter as unit. Some US CRS use feet or US survey feet. If unsure of the unit used in the CRS you can use the x argument of the function. Alternatively, you can use `sf::st_crs(zz, parameters = TRUE)$units_gdal` to see which units are used in the `zz` layer.

This scale bar does not work on unprojected (long/lat) maps.

Value

No return value, a scale bar is displayed.

Examples

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

library(sf)
nc <- st_read(system.file("shape/nc.shp", package = "sf"))[1, ]

nc_foot <- st_transform(nc, 2264) # NC state plane, US foot
mf_map(nc_foot)
mf_scale(size = 5, crs_units = "ft", scale_units = "mi")
mf_map(nc_foot)
mf_scale(size = 5, x = nc_foot, scale_units = "mi")

nc_meter <- st_transform(nc, 32119) # NC state plane, m
mf_map(nc_meter)
mf_scale(size = 5, crs_units = "m", scale_units = "mi")
mf_scale(size = 5, crs_units = "m", scale_units = "km", pos = "bottomleft")
```

---

mf_shadow   *Plot a shadow*

Description

Plot the shadow of a polygon layer.

Usage

```
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)
```

---

mf_theme   *Set a theme*

Description

This function set a map theme. The parameters set by this function are the figure margins, background and foreground colors and some [mf_title](#) options. Use mf_theme(NULL) or mf_theme('default') to reset to default theme settings.

Usage

```
mf_theme(x, 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".
bg	background color
fg	foreground color
mar	margins
tab	if TRUE the title is displayed as a 'tab'
pos	title 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). `mf_theme()` returns the current theme settings.

Value

The (invisible) list of theme parameters is returned.

Examples

```
mtq <- mf_get_mtq()

# Choosing a theme by name:
mf_theme("default")
mf_map(mtq)
mf_title()

# Specifying some values directly:
mf_theme(bg = "darkslategrey", fg = "lightgrey")
mf_map(mtq)
mf_title()

# Using a mix of the above:
mf_theme("brutal", fg = "lightgreen", pos = "center", font = 2, tab = FALSE)
mf_map(mtq)
mf_title()

# Specifying a list with theme values:
theme <- mf_theme("default")
theme$mar <- c(1, 1, 3, 1)
theme$line <- 2
theme$cex <- 1.5
mf_theme(theme)
mf_map(mtq)
```

```
mf_title()

# or
theme <- list(
  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(theme)
mf_map(mtq)
mf_title()

# Obtaining a list of parameters for the current theme:
mf_theme()

# Removing the current theme:
mf_theme(NULL)
# or
mf_theme("default")
```

---

mf_title   *Plot a title*

Description

Plot a title

Usage

```
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

```
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

```
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/ef23faf6d1ada5e4a91c9ef23b0`

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