

# Package ‘mapsf’

September 5, 2023

**Title** Thematic Cartography

**Version** 0.7.1

**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, Rcpp, sf, stats, utils, grDevices

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

**LinkingTo** Rcpp

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**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)

**Maintainer** Timothée Giraud <[timothee.giraud@cnrs.fr](mailto:timothee.giraud@cnrs.fr)>

**Repository** CRAN

**Date/Publication** 2023-09-05 12:50:03 UTC

## 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_mtq . . . . .	9
mf_get_pal . . . . .	10
mf_get_ratio . . . . .	11
mf_graticule . . . . .	12
mf_init . . . . .	13
mf_inset_on . . . . .	14
mf_label . . . . .	15
mf_layout . . . . .	16
mf_legend . . . . .	17
mf_map . . . . .	19
mf_raster . . . . .	22
mf_scale . . . . .	23
mf_shadow . . . . .	24
mf_theme . . . . .	24
mf_title . . . . .	26
mf_worldmap . . . . .	27

<b>Index</b>	<b>29</b>
--------------	-----------

---

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).

### Author(s)

**Maintainer:** [Timothée Giraud <timothee.giraud@cnrs.fr>](mailto:timothee.giraud@cnrs.fr) ([ORCID](#))

Other contributors:

- [Diego Hernangómez](#) ([ORCID](#)) [contributor]
- [Hugues Pecout](#) [contributor]
- [Ronan Ysebaert](#) [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]
- [Florian Zenoni](#) (Orthographic projection visualisation) [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 <a href="#">text</a> arguments.

## Value

No return value, an annotation is displayed.

**Examples**

```
mtq <- mf_get_mtg()
mf_map(mtg)
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, 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_mtg()
mf_map(mtg)
mf_arrow(pos = "topright")
```

---

mf_background	<i>Plot a background image</i>
---------------	--------------------------------

---

**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 <a href="#">rasterImage</a>

**Value**

No return value, a background image is displayed.

**Examples**

```
mtq <- mf_get_mtg()
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	<i>Plot credits</i>
------------	---------------------

---

**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\_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 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 (deprecated)
export	deprecated

**Value**

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

**Examples**

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

---

mf\_get\_breaks

*Get class intervals*


---

**Description**

A function to classify continuous variables.

**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", "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 <a href="#">classIntervals</a>

## 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.

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, 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

## Note

This function is mainly a wrapper of [classIntervals](#) + "arith", "em", "q6", "geom" and "msd" methods.

## See Also

[classIntervals](#)

## Examples

```
mtq <- mf_get_mtg()
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_mtg()
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\_mtg

*Get the 'mtq' dataset*

---

**Description**

Import the mtq dataset (Martinique municipalities).

**Usage**

```
mf_get_mtg()
```

**Details**

This is 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 ( $\geq 1$ ) to be in the palette.
palette	a valid palette name (one of <code>hcl.pals()</code> ). 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 <code>hsv</code> and <code>hcl</code> , 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\_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_mtg()
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_mtg()
mf_map(mtg, expandBB = c(0, .1, .1, 0))
mf_graticule(mtg)

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(mtg, add = TRUE)
```

---

mf\_init

*Initialize a map with a specific extent*


---

### Description

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

Always use `add = TRUE` in `mf_map` calls following an `mf_init` call. This function is similar to `mf_map(x, col = NA, border = NA)`.

### Usage

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

### Arguments

<code>x</code>	object of class <code>sf</code> , <code>sfc</code> or <code>SpatRaster</code>
<code>expandBB</code>	fractional values to expand the bounding box with, in each direction (bottom, left, top, right)
<code>theme</code>	apply a theme (deprecated)

**Value**

No return value, a map is initiated.

**Examples**

```
mtq <- mf_get_mtg()
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**

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

mf_inset_off()
```

**Arguments**

x	an sf object, or "worldmap" to use with <a href="#">mf_worldmap</a> .
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 <code>?par()</code> )

**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**

```

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,
  ...
)

```

**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
...	further <a href="#">text</a> arguments.

**Value**

No return value, labels are displayed.

**Examples**

```
mtq <- mf_get_mtg()
mf_map(mtg)
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 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**

```
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 <a href="#">hcl.colors</a> )
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**

```

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"),
  pt_pch = 21:23, pt_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**

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 for each map types are detailed in specific functions, see [Details](#).

**Usage**

```

mf_map(
  x,
  var,
  type = "base",
  breaks,
  nbreaks,
  pal,
  alpha,
  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,
    expandBB,
    add,
    ...
  )

```

### Arguments

x	object of class <code>sf</code> or <code>sfc</code>
var	name(s) of the variable(s) to plot
type	<ul style="list-style-type: none"> <li>• <b>base</b>: base maps</li> <li>• <b>prop</b>: proportional symbols maps</li> <li>• <b>choro</b>: choropleth maps</li> <li>• <b>typo</b>: typology maps</li> <li>• <b>ymb</b>: symbols maps</li> <li>• <b>grad</b>: graduated symbols maps</li> <li>• <b>prop_choro</b>: proportional symbols maps with symbols colors based on a quantitative data classification</li> <li>• <b>prop_typo</b>: proportional symbols maps with symbols colors based on qualitative data</li> <li>• <b>ymb_choro</b>: symbols maps with symbols colors based on a quantitative data classification</li> </ul>
breaks	either a numeric vector with the actual breaks, or a classification method name (see <a href="#">mf_get_breaks</a> and Details)
nbreaks	number of classes
pal	a set of colors or a palette name (from <a href="#">hcl.colors</a> )
alpha	if <code>pal</code> is a <a href="#">hcl.colors</a> 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	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
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)
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)
...	further parameters from <a href="#">plot</a> for sfc objects

## Details

Relevant arguments, default values and examples for each map types are detailed in specific functions:

- **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](#)

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]`. The "jenks" method is an exception and has to be right-closed. Jenks breaks computed as `c(2, 5, 10, 15, 20)` will be mapped as `[2 - 5], ]5 - 10], ]10 - 15], ]15 - 20]`.

**Value**

x is (invisibly) returned.

**Examples**

```
mtq <- mf_get_mtq()
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, expandBB = rep(0, 4), add = FALSE, ...)
```

**Arguments**

x	a SpatRaster
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).
...	bgalpha, smooth, maxcell or other arguments passed to be passed to <a href="#">plotRGB</a> or <a href="#">plot</a>

**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	<i>Plot a scale bar</i>
----------	-------------------------

---

**Description**

Plot a scale bar.

**Usage**

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

**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 does not work on unprojected (long/lat) maps.

**Examples**

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

---

mf_shadow	<i>Plot a shadow</i>
-----------	----------------------

---

### 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_mtg()
mf_shadow(mtg)
mf_map(mtg, add = TRUE)
```

---

mf_theme	<i>Set a theme</i>
----------	--------------------

---

### 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_mtg()

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

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

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

```
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(mtg)
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	<i>Plot a point on a world map</i>
-------------	------------------------------------

---

**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_mtg()
mf_worldmap(mtg)
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, [18](#), [20](#)  
hcl.pals, [10](#)

mapsf, [2](#)  
mapsf-package (mapsf), [2](#)  
mf\_annotation, [3](#)  
mf\_arrow, [4](#), [16](#)  
mf\_background, [5](#)  
mf\_base, [21](#)  
mf\_choro, [21](#)  
mf\_credits, [5](#), [16](#)  
mf\_export, [6](#)  
mf\_get\_breaks, [7](#), [20](#)  
mf\_get\_links, [8](#)  
mf\_get\_mtg, [9](#)  
mf\_get\_pal, [10](#)  
mf\_get\_ratio, [11](#)  
mf\_grad, [21](#)  
mf\_graticule, [12](#)  
mf\_init, [13](#)  
mf\_inset\_off (mf\_inset\_on), [14](#)  
mf\_inset\_on, [14](#)  
mf\_label, [15](#)  
mf\_layout, [16](#)  
mf\_legend, [17](#)  
mf\_legend\_c, [17](#)  
mf\_legend\_gl, [17](#)  
mf\_legend\_p, [17](#)  
mf\_legend\_pl, [17](#)  
mf\_legend\_s, [17](#)  
mf\_legend\_t, [17](#)  
mf\_map, [19](#)  
mf\_prop, [21](#)  
mf\_prop\_choro, [21](#)  
mf\_prop\_typo, [21](#)  
mf\_raster, [22](#)  
mf\_scale, [16](#), [23](#)  
mf\_shadow, [24](#)  
mf\_symb, [21](#)  
mf\_symb\_choro, [21](#)  
mf\_theme, [24](#)  
mf\_title, [16](#), [24](#), [26](#)  
mf\_typo, [21](#)  
mf\_worldmap, [14](#), [27](#)

plot, [21](#), [22](#)  
plotRGB, [22](#)

quantile, [8](#)

rasterImage, [5](#)

st\_graticule, [13](#)

text, [3](#), [16](#)