Package: maplayer (via r-universe)

November 6, 2024

Type Package

Title Make Map Layers With ggplot2

Version 0.1.0.9003

Maintainer Eli Pousson <`eli.pousson@gmail.com`>

Description Make map-making with ggplot2 and sf more convenient with layers that subset by location.

License MIT + file LICENSE

URL `https://github.com/elipousson/maplayer`, `https://elipousson.github.io/maplayer/`

BugReports `https://github.com/elipousson/maplayer/issues`

Depends R (>= 2.10)

Imports cli, cliExtras (>= 0.1.0), dplyr, getdata (>= 0.1.0), ggplot2, glue, lifecycle, papersize (>= 0.1.0.9001), rlang (>= 1.1.0), scales, sf, sfext (>= 0.1.0), vctrs

Suggests covr, figpatch, filenamr (>= 0.1.0.9002), geomtextpath, ggarchery (>= 0.4.1), ggforce, ggfx, ggpath (>= 0.0.0.9000), ggpattern, ggrepel, ggsvg (>= 0.1.11), ggtext, gridExtra, lwgeom, magick, mapboxapi, paletteer, patchwork, qpdf, roxygen2, rsvg, smoothr, testthat (>= 3.0.0), tibble, vdiffr

Remotes coolbutuseless/ggsvg, elipousson/cliExtras, elipousson/filenamr, elipousson/getdata, elipousson/papersize, elipousson/sfext, mdhall272/ggarchery

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Config/pak/sysreqs libgdal-dev gdal-bin libgeos-dev libicu-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository https://elipousson.r-universe.dev

RemoteUrl https://github.com/elipousson/maplayer

RemoteRef HEAD

RemoteSha 3e0b27255cf04e05a52e00050fdeae3dfbf67446

Contents

`labs_ext` *Add labels to a ggplot2 plot or map*

Description

A helper function that converts strings to glue strings for the title, subtitle, and caption.

Usage

```
labs_ext(
  ...,
  title = ggplot2::waiver(),
  subtitle = ggplot2::waiver(),
  caption = ggplot2::waiver(),
  tag = ggplot2::waiver(),
  alt = ggplot2::waiver(),
  alt_insight = ggplot2::waiver(),
  source_note = NULL,
  source_sep = ". ",
  source_before = "Source: ",
  source_end = ".",
  .sep = "",
  .envir = parent.frame(),
  .open = "{",
  .close = "}",
  .na = "NA",
  .null = character(),
  .comment = "#",
  .literal = FALSE,
  .transformer = glue::identity_transformer,
  .trim = TRUE
)
```

Arguments

`...`	Arguments passed on to `ggplot2::labs` `alt,alt_insight` Text used for the generation of alt-text for the plot. See get_alt_text for examples.
`title`	The text for the title.
`subtitle`	The text for the subtitle for the plot which will be displayed below the title.
`caption`	The text for the caption which will be displayed in the bottom-right of the plot by default.
`tag`	The text for the tag label which will be displayed at the top-left of the plot by default.
`alt`, `alt_insight`	Text used for the generation of alt-text for the plot. See get_alt_text for examples.
`source_note`	Data source(s) to append to caption or use as caption (if no caption is supplied). Also supports glue string interpolation.
`source_sep`, `source_before`, `source_end`	Strings used to separate caption (if supplied) and source note, add before the source note, and add after the source note.
`.sep`	[`character(1)`: ‘""’] Separator used to separate elements.

`.envir`	[`environment`: `parent.frame()`] Environment to evaluate each expression in. Expressions are evaluated from left to right. If `.x` is an environment, the expressions are evaluated in that environment and `.envir` is ignored. If `NULL` is passed, it is equivalent to `emptyenv()`.
`.open`	[`character(1)`: '\{'] The opening delimiter. Doubling the full delimiter escapes it.
`.close`	[`character(1)`: '\}'] The closing delimiter. Doubling the full delimiter escapes it.
`.na`	[`character(1)`: 'NA'] Value to replace `NA` values with. If `NULL` missing values are propagated, that is an `NA` result will cause `NA` output. Otherwise the value is replaced by the value of `.na`.
`.null`	[`character(1)`: 'character()'] Value to replace NULL values with. If `character()` whole output is `character()`. If `NULL` all NULL values are dropped (as in `paste0()`). Otherwise the value is replaced by the value of `.null`.
`.comment`	[`character(1)`: '#'] Value to use as the comment character.
`.literal`	[`boolean(1)`: 'FALSE'] Whether to treat single or double quotes, backticks, and comments as regular characters (vs. as syntactic elements), when parsing the expression string. Setting `.literal = TRUE` probably only makes sense in combination with a custom `.transformer`, as is the case with `glue_col()`. Regard this argument (especially, its name) as experimental.
`.transformer`	[`function`] A function taking two arguments, `text` and `envir`, where `text` is the unparsed string inside the glue block and `envir` is the execution environment. A `.transformer` lets you modify a glue block before, during, or after evaluation, allowing you to create your own custom `glue()`-like functions. See `vignette("transformers")` for examples.
`.trim`	[`logical(1)`: 'TRUE'] Whether to trim the input template with `trim()` or not.

`layer_arrow` *Create a layer with an arrow or segment from and to specified locations*

Description

A wrapper for ggplot2::geom_segment, `ggplot2::geom_curve()`, `ggarchery::geom_arrowsegment()`, `ggforce::geom_diagonal0()`, `ggforce::geom_link()` that makes it easier to specify the start and end of the segment using any object supported by the `sfext::as_xy()` function.

Usage

```
layer_arrow(
  mapping = NULL,
  data = NULL,
  crs = NULL,
  from,
  to,
  geom = "segment",
  ...
)
```

Arguments

`mapping`	aesthetic mapping overwritten with x, y, xend, and yend values based on provided from and to parameters.
`data`	Required if from or to are character vectors to `sfext::as_xy()`
`crs`	A character or numeric reference to a coordinate reference system supported by `sf::st_crs()` or another `sf`, `sfc`, or bbox object that is used to provide crs.
`from`, `to`	Required. Passed to x parameter of `sfext::as_xy()` (using `nm = c("xend", "yend")`) for the to parameter.
`geom`	Character string for geom to use c("segment", "curve", "arrowsegment", "diagonal0", "link") or a geom function.
`...`	Additional parameters passed to function specified by geom paramter.

See Also

`ggplot2::reexports()`, `ggplot2::geom_segment()`, `ggplot2::aes()`

Examples

```
library(ggplot2)

nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))

nc_map <- ggplot(data = nc) +
  geom_sf()

nc_map +
  layer_arrow(
    data = nc,
    from = c("xmin", "ymin"),
    to = c("xmid", "ymax"),
  )

nc_map +
  layer_arrow(
    data = nc,
    from = c("xmin", "ymin"),
    to = c("xmid", "ymax"),
    geom = "curve",
    curvature = 0.25
  )

nc_map +
  layer_arrow(
    data = nc,
    from = c("xmax", "ymin"),
    to = c("xmid", "ymax"),
    geom = "arrowsegment"
  )
```

layer_count — *Layer for counting occurrences of data in spatial relation a location or other sf object*

Description

Wraps `sfext::count_sf_ext()`. Specification of parameters for this function may be too complex and may be changed in the future.

Usage

```
layer_count(
  data,
  location = NULL,
  y = NULL,
  join = sf::st_intersects,
  largest = TRUE,
  replace_na = FALSE,
  lims = NULL,
  .id = "id",
  grid_params = list(alpha = 1, color = NA),
  show_data = FALSE,
  data_params = list(mapping = aes(), alpha = 0.75, size = 1),
  show_label = FALSE,
  label_params = NULL,
  scale_fn = ggplot2::scale_fill_continuous,
  scale_params = list(type = "viridis", breaks = scales::breaks_pretty(n = 4)),
  ...
)
```

Arguments

`data`	Data to count in relationship to y
`location`	Passed to x parameter of `sfext::count_sf_ext()`.

`y`	If `NULL` (default), y defaults to an `sf` object created by `st_make_grid_ext()` using x or data (if x is `NULL`) as the x parameter for `st_make_grid_ext()`. If not `NULL`, y must be an `sf` object that has a column with the same name as .id (defaults to "id").
`join`	geometry predicate function with the same profile as st_intersects; see details
`largest`	logical; if `TRUE`, return `x` features augmented with the fields of `y` that have the largest overlap with each of the features of `x`; see https://github.com/r-spatial/sf/issues/578
`replace_na`	If `TRUE`, replace NA values from count with 0.
`lims`	Optional numeric vector with minimum or both minimum and maximum count values. If provided, any values below the minimum are set to that minimum and any values above the maximum as set to the maximum. If only one value is provided, it is assumed to be a minimum limit.
`.id`	A name to use for the cell id column. Defaults to "id".
`grid_params`	Passed to `layer_location_data()` to style foreground grid with fill based on count.
`show_data`	If `TRUE`, add background layer with data to stack returned by function. If `TRUE` and grid_params includes a fixed aesthetic for alpha, divide alpha in half to ensure background data is visible below the filled grid.
`data_params`	Passed to `layer_location_data()` to style background layer based on data.
`show_label`	If `TRUE`, add layer with labels to stack returned by function.
`label_params`	Passed to `layer_labelled()` for foreground labels with fill based on count.
`scale_fn, scale_params`	Scale function and parameters. Defaults to `ggplot2::scale_fill_continuous()`.
`...`	Arguments passed on to `sfext::count_sf_ext` `wt` <`data-masking`> Frequency weights. Can be `NULL` or a variable: • If `NULL` (the default), counts the number of rows in each group. • If a variable, computes `sum(wt)` for each group. `sort` If `TRUE`, will show the largest groups at the top. `keep_na` If `TRUE`, filter NA values from count. Ignored if replace_na is `TRUE`. `geometry` If `TRUE` (default) return a `sf` object. If `FALSE`, return a data frame. `name` The name of the new column in the output. If omitted, it will default to `n`. If there's already a column called `n`, it will use `nn`. If there's a column called `n` and `nn`, it'll use `nnn`, and so on, adding `n`s until it gets a new name.

Examples

```
nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))
data <- sf::st_sample(nc, 75)

ggplot() +
  layer_count(data = data, location = nc)

ggplot() +
  layer_count(data = data, y = nc, .id = "FIPS")
```

`layer_frame` *Create a frame layer around a simple feature object*

Description

Create a circle or square that can be used as a frame around a simple feature object using fixed aesthetics for fill, color, size, and linetype. This function is helpful for the background of an inset map intended for use with `layer_inset()`.

Usage

```
layer_frame(
  data = NULL,
  dist = NULL,
  diag_ratio = NULL,
  unit = "meter",
  asp = NULL,
  style = "circle",
  scale = 1,
  rotate = 0,
  inscribed = FALSE,
  color = "black",
  linewidth = 0.5,
  linetype = "solid",
  fill = "white",
  neatline = TRUE,
  expand = TRUE,
  basemap = FALSE,
  union = TRUE,
  by_feature = FALSE,
  ...
)

make_frame(
  x,
  dist = NULL,
  diag_ratio = NULL,
  unit = "meter",
  asp = NULL,
  style = "circle",
  scale = 1,
  rotate = 0,
  inscribed = FALSE,
  dTolerance = 0,
  union = TRUE,
  by_feature = FALSE
)
```

Arguments

`data`, `x`	A `sf`, `sfc`, or bbox object to create the frame around.
`dist`	buffer distance in units. Optional.
`diag_ratio`	ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will be used. Ignored when `dist` is provided.
`unit`	Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"
`asp`	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, `get_asp()` returns the same value without modification.
`style`	Style of framing shape to add, "circle", "square", "rect", "buffer", or "none". If style is "buffer", the asp parameter is ignored. If style is "none", the dist, diag_ratio, and asp parameters are ignored and the input data is used as the frame.
`scale`	numeric; scale factor, Default: 1
`rotate`	numeric; degrees to rotate (-360 to 360), Default: 0
`inscribed`	If `TRUE`, the returned geometry is inscribed within x, if `FALSE` (default), the geometry is circumscribed.
`fill`, `color`, `linewidth`, `linetype`	Fixed aesthetics for frame, passed to layer_location_data.
`neatline`	If `TRUE`, return a list of layers that includes a layer_neatline
`expand`	If `TRUE`, the default, adds a small expansion factor to the limits to ensure that data and axes don't overlap. If `FALSE`, limits are taken exactly from the data or `xlim`/`ylim`.
`basemap`	Either a logical vector or ggplot object. If **logical** and `TRUE`, add x to `ggplot2::ggplot()`. If `FALSE`, return x as is. If a **ggplot**, add x to basemap object. If a **ggproto** object (or list that contains a **ggproto** object), add x and basemap object to `ggplot2::ggplot()`.
`union`	If `TRUE`, pass data to `sf::st_union()` before buffering and creating frame; defaults to `TRUE`.
`by_feature`	If `TRUE`, create a frame around each feature. If `FALSE` (default), union the provided features before creating a frame.
`...`	Arguments passed on to `layer_location_data` `layer_fn` ggplot2 geom or custom function using lambda syntax. Use for passing custom mapping functions to layer_location_data beyond the supported geom options. `label_col` Column name or id for a column with the text or labels to pass to any text geom.

`smooth_params` Optional. Logical or a list of parameters passed to `smoothr::smooth()`. If `TRUE`, apply `smoothr::smooth()` to location data using default parameters. smooth_params is ignored if data is `NULL` (inheriting data from ggplot).

`shadow_params` Optional. Logical or a list of parameters passed to `ggfx::with_shadow()`. If `TRUE`, apply `ggfx::with_shadow()` to the layer using default parameters. shadow_params is ignored if layer_fn is provided.

`location` sf object. If multiple areas are provided, they are unioned into a single sf object using `sf::st_union()`

`fileext,filetype` File extension or type to use if passing parameters to `sfext::read_sf_download()` or `sfext::read_sf_pkg()` (required for extdata and cached data).

`fn` Function to apply to data after filtering by location but before returning from function.

`crop` If `TRUE`, x is cropped to y using `sf::st_crop()`.

`trim` If `TRUE`, x is trimmed to y with `st_trim()`.

`crs` Coordinate reference system to return.

`mapping` Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

`dTolerance` numeric; tolerance parameter, specified for all or for each feature geometry. If you run `st_simplify`, the input data is specified with long-lat coordinates and `sf_use_s2()` returns `TRUE`, then the value of `dTolerance` must be specified in meters.

Details

The `make_frame()` helper function calls `sfext::st_circle()` (if `style = "circle"`), `sfext::st_square()` (if `style = "square"`), `sfext::st_bbox_ext()` (if `style = "rect"`), or `sfext::st_buffer_ext()` (if `style = "none"`).

If neatline is `TRUE`, `layer_frame()` returns a list of two geoms, the second a `layer_neatline()` layer created using the frame object as the data and the parameters `bgcolor = "none"` and `color = "none"`. asp is set to 1 if style is "circle" or "square" or the provided asp value otherwise.

Additional parameters passed through ... can include additional fixed aesthetics (e.g. alpha). If using the fn parameter, the function is applied to the frame simple feature object created by `make_frame()` (not to the original input data).

See Also

Other layer: `layer_location_data()`, `layer_neatline()`, `layer_scaled()`

Examples

```
nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))

raleigh_msa <-
  getdata::get_location(
    type = nc,
    name_col = "NAME",
    name =  c("Franklin", "Johnston", "Wake"),
    crs = 3857
    )

ggplot() +
  layer_frame(
    data = raleigh_msa,
    frame = "circle",
    fill = "lightyellow",
    inscribed = FALSE
  ) +
  layer_location_data(
    data = raleigh_msa,
    mapping = aes(fill = NAME),
    alpha = 0.5
  ) +
  ggplot2::guides(
    fill = "none"
  )
```

`layer_grouped` *Make group layers*

Description

Can be used to make multiple layers or multiple maps based on a grouping variable.

Usage

```
layer_grouped(
  data,
  mapping = NULL,
  groupname_col = "group",
  label_col = "name",
  geom = "sf",
  basemap = FALSE,
  palette = NULL,
  aesthetics = "fill",
  ...
)
```

Arguments

`data`	Character string (e.g. url, file path, or name of data from package) for a spatial data or a `sf`, `sfc`, or bbox object with geometry overlapping the location. If data is `NULL`, all unnamed parameters are passed to `sfext::read_sf_ext()` with a bbox based on location. If data is not `NULL` and not a data.frame, url, file path, or bbox, conversion to a sf object will still always be attempted with `sfext::as_sf()`.

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`groupname_col`	Group column name. Defaults to "group".
`label_col`	Column name or id for a column with the text or labels to pass to any text geom.
`geom`	A character string indicating which ggplot2 geom to use, Default: 'sf'. Options include "sf" (`ggplot2::geom_sf()`), "icon" (`layer_icon()`), "markers" (`layer_markers()`), "sf_text" (`ggplot2::geom_sf_text()`), and "sf_label" (`ggplot2::geom_sf_label` See details for a full list.
`basemap`	Either a logical vector or ggplot object. If **logical** and `TRUE`, add x to `ggplot2::ggplot()`. If `FALSE`, return x as is. If a **ggplot**, add x to basemap object. If a **ggproto** object (or list that contains a **ggproto** object), add x and basemap object to `ggplot2::ggplot()`.
`palette`	Name of palette as a string. Must be on the form packagename::palettename.
`aesthetics`	Aesthetic to map to groupname_col. Defaults to "fill"; also supports "color" or c("fill", "color").
`...`	Additional parameters passed to `layer_location_data()`

Details

Scales are applied a palette and aesthetic are provided and basemap is set to `TRUE`.

`layer_icon` *Use ggsvg to add a layer with icons at feature locations*

Description

Use the `ggsvg::geom_point_svg()` function to plot icons using the centroids from the input simple feature object to set the icon location.

Usage

```
layer_icon(
  data = NULL,
  iconname_col = "icon",
  icon = NULL,
  px = NULL,
  source = NULL,
  svg = NULL,
  crs = getOption("maplayer.crs", default = 3857),
  ...
)
```

```
geom_sf_icon(
  data = NULL,
  iconname_col = "icon",
  icon = NULL,
  px = NULL,
  source = NULL,
  svg = NULL,
  crs = getOption("maplayer.crs", default = 3857),
  ...
)
```

Arguments

`data`	A `sf` object. Any objects with polygon geometry are converted to points using `sf::st_centroid()`.
`iconname_col`	The column name in the input data to use as the icon name. If the name matches multiple icons, the first match from `map_icons` is used. You may provide a px or source value to select a different match if needed but, in that case, all icons must use the same px or source value. Note that the icon column should not be mapped with `ggplot2::aes()`.
`icon`	Icon name. Default `NULL`. If `icon` is provided, `iconname_col` is ignored. See `map_icons$name` for supported options.
`px`	Icon size in pixels. See `map_icons$px` for supported options. Optional but may be necessary to differentiate icons with duplicate names.
`source`	Icon source. See `map_icons$repo` for supported options. Optional but may be required to differentiate icons with duplicate names.
`svg`	Optional. Custom file path or URL with SVG to pass to `svg` parameter for `ggsvg::geom_point_svg()`. If `icon` is provided, `svg` is ignored.
`crs`	Coordinate reference system; defaults to `NULL`.
`...`	Arguments passed on to `ggsvg::geom_point_svg` `defaults` Advanced option. A named list of default values for new aesthetics. In general this is not necessary when using `css()` aesthetics, as a default value will be determined based upon the CSS property e.g. `stroke` property will have a default value of "black" Set ‘options(GGSVG_DEBUG = TRUE)‘ for some verbose debugging which will cause ‘ggsvg‘ to output (to the console) the final SVG for each and every element in the plot.

See Also

`ggsvg::geom_point_svg()` map_icons

Examples

```
nc <- getdata::get_location(type = system.file("shape/nc.shp", package = "sf"), crs = 3857)

basemap <-
```

```
  ggplot2::ggplot() +
  ggplot2::theme_void() +
  layer_location_data(data = nc)

# icon can be set by name matching a name from map_icons
basemap +
  layer_icon(data = nc, icon = "point-start", size = 8)

# layer_icon can also use a column from the sf object
nc$icon <- rep(c("1", "2", "3", "4"), nrow(nc) / 4)

basemap +
  layer_icon(data = nc, iconname_col = "icon", size = 6)
```

`layer_image_path` *Use ggpath to create a layer with images at locations*

Description

Use the `ggpath::geom_from_path()` function with `sfext::read_sf_exif()` and `stat = "sf_coordinates"` to create a layer showing images at locations.

Usage

```
layer_image_path(
  data = NULL,
  path = NULL,
  path_col = "path",
  width = 0.1,
  crs = getOption("maplayer.crs", 3857),
  segment_params = NULL,
  neatline = FALSE,
  basemap = FALSE,
  ...
)
```

Arguments

`data`	A `sf` object with a column containing file paths that has the same name as the path_col argument. Optional if "path" is provided for `sfext::read_sf_exif()`. Required if path is `NULL`. If path is provided, data is ignored.
`path`	A path to folder or file.
`path_col`	Column name with file paths from data. Defaults to "path" (path name used by data returned from `sfext::read_sf_exif()`)
`width`	Width of the image in npc (Normalised Parent Coordinates) passed to `ggpath::geom_from_path()`; defaults to 0.1.

`crs`	The coordinate reference system (CRS) into which all data should be projected before plotting. If not specified, will use the CRS defined in the first sf layer of the plot.
`segment_params`	Not implemented: parameters to define segments connecting images to the location.
`neatline`	A logical object, `CoordSf` object, or a list containing a `CoordSf` object (typically from `layer_neatline()`) added to layer by `set_neatline()`. • If logical and `TRUE`, add a neatline layer using data, crs and any additional parameters passed to ... If logical and `FALSE`, return x as is. • If object from `layer_neatline()`, add it as is.
`basemap`	Either a logical vector or ggplot object. If **logical** and `TRUE`, add x to `ggplot2::ggplot()`. If `FALSE`, return x as is. If a **ggplot**, add x to basemap object. If a **ggproto** object (or list that contains a **ggproto** object), add x and basemap object to `ggplot2::ggplot()`.
`...`	Additional parameters to pass to `exiftoolr::exif_read()`

`layer_inset` *Use patchwork to create a map with an inset context map or figpatch to stamp an inset image*

Description

`layer_inset()` is useful when you want to add an inset to a plot.

Usage

```
layer_inset(
  map = NULL,
  inset = NULL,
  position = "bottomright",
  scale = 1,
  nudge_x = 0,
  nudge_y = 0,
  align_to = "full",
  ...
)

make_inset_map(
  map = NULL,
  inset = NULL,
  location = NULL,
  context = NULL,
  position = "bottomright",
  scale = 1,
  nudge_x = 0,
  nudge_y = 0,
  align_to = "full",
  ...
)

stamp_inset_img(
  path,
  plot = NULL,
  img_margin = ggplot2::margin(0, 0, 0, 0),
  position = "bottomright",
  scale = 1,
  nudge_x = 0,
  nudge_y = 0,
  align_to = "full",
  ...
)
```

Arguments

`inset`	plot or map created with `ggplot2()` passed to p argument of `patchwork::inset_element()`. If both location and context are provided to `make_inset_map()`, inset is optional and any provided value is replaced with a new layer created by `layer_location_context()`.
`position`	inset map position, Default: 'bottomright'. position, nudge_x, and nudge_y are used to set the left, bottom, top, and right parameters for `patchwork::inset_element()`.
`scale`	scale of inset map, defaults to 1.
`nudge_x, nudge_y`	nudge x and/or y position of inset map, Default: 0.
`align_to`	Specifies what `left`, `bottom`, etc should be relative to. Either `'panel'` (default), `'plot'`, or `'full'`.
`...`	Arguments passed on to `patchwork::inset_element` `p` A grob, ggplot, patchwork, formula, raster, or nativeRaster object to add as an inset `left,bottom,right,top` numerics or units giving the location of the outer bounds. If given as numerics they will be converted to npc units. `on_top` Logical. Should the inset be placed on top of the other plot or below (but above the background)? `clip` Logical. Should clipping be performed on the inset? `ignore_tag` Logical. Should autotagging ignore the inset?
`location`	A location passed to `layer_location_context()`. This can be a sf object, a ggplot layer, or a formula or function. If it is a formula or function, it is applied to the context data is passed to the location function and the results used as the data for the location layer.
`context`	A `sf` object for context area or a ggplot layer representing the context.
`path`	image path passed to `figpatch::fig()` for `stamp_inset_img()`

`plot, map`	plot or map created with `ggplot2()`
`img_margin`	margin around image for `stamp_inset_img()` created by `ggplot2::margin()`. Defaults to no margin.

Details

`make_inset_map()` is useful for creating an inset map just using the location with fewer options for customization. In that case, the ... parameters are passed to `layer_location_context()` instead of `patchwork::inset_element()`

`stamp_inset_img()` is useful for applying a logo to a map. The ... parameters are passed to `figpatch::fig()`

Note, currently, plots created with `layer_inset()` do not work with `map_ggsave_ext()` using the `single_file = TRUE` parameter.

Value

ggplot2 map with inset map added using patchwork

`layer_labelled` *Label simple feature objects in a location*

Description

Label markers, streets, areas, or other simple feature objects using any of the following geoms: "text", "sf_text", "label", "sf_label", "textsf", "labelsf", "text_repel", or "label_repel".

Usage

```
layer_labelled(
  data,
  location = NULL,
  geom = "text",
  fn = NULL,
  label_col = NULL,
  mapping = NULL,
  union = FALSE,
  clip = NULL,
  dist = NULL,
  diag_ratio = NULL,
  unit = NULL,
  drop_shadow = FALSE,
  shadow_params = NULL,
  ...
)
```

Arguments

`data`	Data to use for labels (must be an sf object or a data frame that can be converted to an sf object using `sfext::as_sf()`)
`location`	Location to label (if not specified the data is assumed to conver the whole location);
`geom`	A geom to use "text", "label", "textsf", "labelsf", "text_repel", or "label_repel" or a geom function (passed to layer_fn).
`fn`	Function to apply to data before creating labels; can be used in the creation of the label_col.
`label_col`	Label column name
`mapping`	Aesthetic mapping, Default: `NULL`
`union`	If TRUE, group by label_col and union geometry, Default: `FALSE`
`clip`	Character string describing the part of the area to clip or remove. Options include c("top", "right", "bottom", "left", "topright", "bottomright", "bottomleft", "topleft"). If NULL, the area is not clipped and a full edge can be returned.
`dist`	Numeric. Distance to use for the edge. Default NULL meters. Use negative values for an inside edge or positive numbers for an outside edge.
`diag_ratio`	Alternate way to define edge distance.
`unit`	Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"
`drop_shadow`	If `TRUE`, use `ggfx::with_shadow()` to add a drop shadow to the label layer with shadow_params. Defaults to `FALSE`.
`shadow_params`	Parameters passed to `ggfx::with_shadow()` if `drop_shadow = TRUE`. Defaults to `list(x_offset = 5, y_offset = 5, sigma = 0.5)`.
`...`	Arguments passed on to `layer_location_data`

`smooth_params` Optional. Logical or a list of parameters passed to `smoothr::smooth()`. If `TRUE`, apply `smoothr::smooth()` to location data using default parameters. smooth_params is ignored if data is `NULL` (inheriting data from ggplot).

`asp` Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, `get_asp()` returns the same value without modification.

`pkg,package` Name of the package to search for data.

`fileext,filetype` File extension or type to use if passing parameters to `sfext::read_sf_download()` or `sfext::read_sf_pkg()` (required for extdata and cached data).

`crop` If `TRUE`, x is cropped to y using `sf::st_crop()`.

`trim` If `TRUE`, x is trimmed to y with `st_trim()`.

`crs` Coordinate reference system to return.

`basemap` Either a logical vector or ggplot object.
If **logical** and `TRUE`, add x to `ggplot2::ggplot()`. If `FALSE`, return x as is.
If a **ggplot**, add x to basemap object.
If a **ggproto** object (or list that contains a **ggproto** object), add x and basemap object to `ggplot2::ggplot()`.

Details

Note: Unlike in some getdata package functions, fn is applied to all of the data; not a subset of the data based on location. For this function, dist and unit are both used by `sfext::st_clip()` (not by `layer_location_data()`)

The function also overrides the label aesthetics to hide the colored letters that would otherwise appear when using a theme legend.

See Also

`sfext::st_clip()`,`layer_location_data()`

`layer_location` *Layer a location border into a ggplot2 map*

Description

Helper function to make a ggplot2 layer from data returned by `get_location`

Usage

```
layer_location(
  mapping = ggplot2::aes(),
  data = NULL,
  type = NULL,
  name = NULL,
  id = NULL,
  location = NULL,
  name_col = "name",
  id_col = "id",
  index = NULL,
  label = NULL,
  label_geom = NULL,
  label_col = name_col,
  union = FALSE,
  crs = getOption("maplayer.crs", default = 3857),
  color = "gray40",
  linewidth = 0.5,
  linetype = "dashed",
  fill = NA,
  alpha = 1,
  dist = NULL,
  diag_ratio = NULL,
  unit = NULL,
  asp = NULL,
  mask = FALSE,
  neatline = FALSE,
  smooth_params = NULL,
  shadow_params = NULL,
  ...
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	Data for location to show.
`type`	Type of location to return. Type can be an sf object, e.g. a data frame with multiple neighborhoods or a character string that can be passed to `get_location_data()`. If index is provided, character can also be a character string to match the name of a list.
`name`	Location name to return.
`id`	Location id to return. id is coerced to character or numeric to match the class of the id_col for type.
`location`	An address, bounding box (bbox), or simple feature (`sf`) object passed to `sf::st_filter()`. Any valid address or addresses are geocoded with `tidygeocoder::geo()`, converted to a simple feature object, and then used as a spatial filter. bbox objects are converted using `sfext::sf_bbox_to_sf()`. Multiple addresses are supported.
`name_col`	Column name in type with name values, Default: 'name' Required if name provided.
`id_col`	Column name in type with id values, Default: 'id'. Required if id is provided.
`index`	Optional list used to match type to data, Default: `NULL`
`label`	label type (e.g. "text", "label")
`label_geom`	Optional character string or function with geom to use for labelling location layer. Passed to geom parameter of `layer_labelled()`
`label_col`	Column name or id for a column with the text or labels to pass to any text geom.
`union`	If `TRUE`, the location geometry is unioned with `sf::st_union()` and the names are combined into a single value. Default: `FALSE`.
`crs`	Coordinate reference system to return; defaults to `NULL` which returns data using the same coordinate reference system as the provided type of location.
`color`	Color for location; defaults to "black".
`linewidth`	Line width for location; defaults to 0.5.
`linetype`	Line type for location; defaults to "dashed".
`fill`	Fill for location; defaults to "NA".
`alpha`	mask alpha/transparency; defaults to 0.5
`dist`	buffer distance in units. Optional.

`diag_ratio`	ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will be used. Ignored when `dist` is provided.
`unit`	unit to adjust location by dist or diag_ratio; defaults to "meter"
`asp`	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, `get_asp()` returns the same value without modification.
`mask`	A `sf`, `sfc`, or bbox object to define the mask area. `diag_ratio`, `dist`, and `asp` parameters are ignored if a `mask` is provided. defaults to `NULL`
`neatline`	A logical object, `CoordSf` object, or a list containing a `CoordSf` object (typically from `layer_neatline()`) added to layer by `set_neatline()`. • If logical and `TRUE`, add a neatline layer using data, crs and any additional parameters passed to ... If logical and `FALSE`, return x as is. • If object from `layer_neatline()`, add it as is.
`smooth_params`	Optional. Logical or a list of parameters passed to `smoothr::smooth()`. If `TRUE`, apply `smoothr::smooth()` to location data using default parameters. smooth_params is ignored if data is `NULL` (inheriting data from ggplot).
`shadow_params`	Optional. Logical or a list of parameters passed to `ggfx::with_shadow()`. If `TRUE`, apply `ggfx::with_shadow()` to the layer using default parameters. shadow_params is ignored if layer_fn is provided.
`...`	Additional parameters passed to get_location if data is `NULL`.

Value

list of ggplot2 geoms

See Also

`ggplot2::CoordSf()`

`layer_location_context`

Create a layer showing a location and related context

Description

Create a ggplot2 layer for a location in a provided context. This function is useful for making inset locator maps in combination with the make_inset_map function.

Usage

```
layer_location_context(
  data = NULL,
  location = NULL,
  fill = "gray70",
  color = "black",
  context = NULL,
  context_params = list(fill = "white", color = "black", alpha = 1, ...),
  crs = getOption("maplayer.crs", default = 3857),
  mid_layer = NULL,
  neatline = TRUE,
  basemap = FALSE,
  ...
)
```

Arguments

`data, location`	A data and location can be a either `sf` object or a ggplot layer. Either data or location is required and, if data and location are both provided, location is ignored. If data is `NULL` and location is a formula or function, the context data is passed to the location function and the results used as the data for the location layer.
`fill, color`	Fill and color fixed aesthetics for location data.
`context`	A `sf` object for context area or a ggplot layer representing the context.
`context_params`	A list with parameters for context layer; defaults to `list(fill = "white", color = "black", alpha = 1, ...)`.
`crs`	Coordinate reference system to return.
`mid_layer`	A ggplot2 layer to insert between the context layer and the location layer. Optional.
`neatline`	A logical object, `CoordSf` object, or a list containing a `CoordSf` object (typically from `layer_neatline()`) added to layer. • If logical and `TRUE`, add a neatline layer using data from the context layer with `color = NA` and `bgcolor = "none"`. • If object from `layer_neatline()`, add it as is.
`basemap`	Either a logical vector or ggplot object. If **logical** and `TRUE`, add x to `ggplot2::ggplot()`. If `FALSE`, return x as is. If a **ggplot**, add x to basemap object. If a **ggproto** object (or list that contains a **ggproto** object), add x and basemap object to `ggplot2::ggplot()`.
`...`	Additional parameters passed to `layer_location_data()` if data or location is an `sf` object.

`layer_location_data` *Layer location data into a ggplot2 map*

Description

Helper function to make a ggplot2 layer from data returned by `get_location_data()`. For text geoms, the required aesthetic mapping is set based on the name_col but values passed to mapping take precedence.

Usage

```
layer_location_data(
  mapping = NULL,
  data = NULL,
  geom = "sf",
  location = NULL,
  dist = getOption("maplayer.dist"),
  diag_ratio = getOption("maplayer.diag_ratio"),
  unit = getOption("maplayer.unit", default = "meter"),
  asp = getOption("maplayer.asp"),
  package = getOption("maplayer.data_package"),
  pkg = getOption("maplayer.data_package"),
  fileext = getOption("maplayer.data_fileext", "gpkg"),
  filetype = NULL,
  fn = NULL,
  layer_fn = NULL,
  crop = TRUE,
  trim = FALSE,
  from_crs = getOption("maplayer.from_crs", 4326),
  crs = getOption("maplayer.crs", 3857),
  label_col = "name",
  smooth_params = NULL,
  shadow_params = NULL,
  basemap = FALSE,
  ...
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	Character string (e.g. url, file path, or name of data from package) for a spatial data or a `sf`, `sfc`, or bbox object with geometry overlapping the location. If data is `NULL`, all unnamed parameters are passed to `sfext::read_sf_ext()` with a bbox based on location. If data is not `NULL` and not a data.frame, url, file path, or bbox, conversion to a sf object will still always be attempted with `sfext::as_sf()`.
`geom`	A character string indicating which ggplot2 geom to use, Default: 'sf'. Options include "sf" (`ggplot2::geom_sf()`), "icon" (`layer_icon()`), "markers" (`layer_markers()`), "sf_text" (`ggplot2::geom_sf_text()`), and "sf_label" (`ggplot2::geom_sf_labe` See details for a full list.
`location`	sf object. If multiple areas are provided, they are unioned into a single sf object using `sf::st_union()`
`dist`	buffer distance in units. Optional.
`diag_ratio`	ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will

	be used. Ignored when `dist` is provided.
`unit`	unit to adjust location by dist or diag_ratio; defaults to "meter"
`asp`	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, `get_asp()` returns the same value without modification.
`pkg, package`	Name of the package to search for data.
`fileext, filetype`	File extension or type to use if passing parameters to `sfext::read_sf_download()` or `sfext::read_sf_pkg()` (required for extdata and cached data).
`fn`	Function to apply to data after filtering by location but before returning from function.
`layer_fn`	ggplot2 geom or custom function using lambda syntax. Use for passing custom mapping functions to layer_location_data beyond the supported geom options.
`crop`	If `TRUE`, x is cropped to y using `sf::st_crop()`.
`trim`	If `TRUE`, x is trimmed to y with `st_trim()`.
`from_crs`	Coordinate reference system used to match the location CRS to the source data.
`crs`	Coordinate reference system to return.
`label_col`	Column name or id for a column with the text or labels to pass to any text geom.
`smooth_params`	Optional. Logical or a list of parameters passed to `smoothr::smooth()`. If `TRUE`, apply `smoothr::smooth()` to location data using default parameters. smooth_params is ignored if data is `NULL` (inheriting data from ggplot).
`shadow_params`	Optional. Logical or a list of parameters passed to `ggfx::with_shadow()`. If `TRUE`, apply `ggfx::with_shadow()` to the layer using default parameters. shadow_params is ignored if layer_fn is provided.
`basemap`	Either a logical vector or ggplot object. If **logical** and `TRUE`, add x to `ggplot2::ggplot()`. If `FALSE`, return x as is. If a **ggplot**, add x to basemap object. If a **ggproto** object (or list that contains a **ggproto** object), add x and basemap object to `ggplot2::ggplot()`.
`...`	Additional parameters passed to selected geom or layer_fn

Using the geom parameter

This function provides a convenient option for filtering data by location while calling a different layer function from ggplot2, maplayer, or a different package.

Options for the geom parameter from ggplot2:

- "sf" (ggplot2::geom_sf - default)
- "sf_text" or "text (ggplot2::geom_sf_text)
- "sf_label" or "label (ggplot2::geom_sf_label)

Options for the geom parameter included in this package include:

- "location" (layer_location)

- "context" or "location_context" (layer_location_context)
- "icon" (layer_icon),
- "mapbox" (layer_mapbox)
- "markers" (layer_markers)
- "numbers" or "numbered" (layer_numbers)
- "mark" or "marked" (layer_marked)

Options for the geom parameter from other suggested packages include:

- "textsf" (geomtextpath::geom_textsf)
- "labelsf" (geomtextpath::geom_labelsf)
- "text_repel" (ggrepel::geom_text_repel)
- "label_repel" (ggrepel::geom_label_repel)
- "sf_pattern" or "pattern" (ggpattern::geom_sf_pattern)

`stat = "sf_coordinates"` is automatically added to the parameters for both ggrepel functions. `label = .data[[name_col]]` is automatically added to all provided geoms where label is a required parameter.

Using the layer_fn parameter

layer_fn can be a purrr-style lamba function (converted with `rlang::as_function()`) or a function.

See Also

Other layer: `layer_frame()`, `layer_neatline()`, `layer_scaled()`

`layer_mapbox` *Use mapboxapi to make a Mapbox static map layer*

Description

Use mapboxapi to make a Mapbox static map layer

Usage

```
layer_mapbox(
  data = NULL,
  dist = NULL,
  diag_ratio = NULL,
  unit = "meter",
  asp = NULL,
  style_url = "mapbox://styles/mapbox/satellite-streets-v11",
  style_id = NULL,
  username = NULL,
  basemap = FALSE,
  scale = 0.75,
  scaling_factor = "1x",
  attribution = TRUE,
  logo = TRUE,
  access_token = NULL,
  neatline = TRUE,
  color = "black",
  bgcolor = "white",
  linewidth = 0.5,
  linetype = "solid",
  expand = TRUE,
  hide_grid = TRUE,
  label_axes = "----",
  ...
)
```

Arguments

`data`	A `sf`, `sfc`, or bbox object.
`dist`	buffer distance in units. Optional.
`diag_ratio`	ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will be used. Ignored when `dist` is provided.
`unit`	Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"
`asp`	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, `get_asp()` returns the same value without modification.
`style_url`	Map style url used to fill style_id and username parameters, Default: "mapbox://styles/mapbox/satellite-streets-v11"
`style_id`	A style ID (required if style_url is `NULL`).
`username`	A Mapbox username (required if `style_url` = `NULL`).
`basemap`	If FALSE, create a standalone layer; if `TRUE`, the layer is preceded by `ggplot2::ggplot()` to allow use as a basemap, Default: `TRUE`
`scale`	ratio to scale the output image; `scale = 1` will return the largest possible image. defaults to 0.5
`scaling_factor`	The scaling factor of the tiles; either `"1x"` (the default) or `"2x"`
`attribution`	Controls whether there is attribution on the image. Defaults to `TRUE`. If `FALSE`, the watermarked attribution is removed from the image. You still have a legal responsibility to attribute maps that use OpenStreetMap data, which includes most maps from Mapbox. If you specify `attribution = FALSE`, you are legally required to include proper attribution elsewhere on the webpage or document.
`logo`	Controls whether there is a Mapbox logo on the image. Defaults to `TRUE`.

`access_token`	A Mapbox access token; which can be set with mb_access_token.
`neatline`	If `TRUE`, add a neatline matching the provided data, Default: `TRUE`
`color`	Color of panel border, Default: 'black'
`bgcolor`	Fill color of panel background; defaults to "white". If "none", panel background is set to `ggplot2::element_blank()`
`linewidth`	Line width of panel border, Default: 0.5
`linetype`	Line type of panel border, Default: 'solid'
`expand`	If `TRUE`, the default, adds a small expansion factor to the limits to ensure that data and axes don't overlap. If `FALSE`, limits are taken exactly from the data or `xlim`/`ylim`.
`hide_grid`	If `TRUE`, hide grid lines. Default: `TRUE`
`label_axes`	A description of which axes to label passed to `ggplot2::coord_sf()`; defaults to '—-' which hides all axes labels.
`...`	Additional parameter passed to mapboxapi::layer_static_mapbox

See Also

`mapboxapi::layer_static_mapbox()`

`layer_marked` *Use ggforce to create an annotation layer using simple feature data*

Description

Use `ggforce::geom_mark_rect()`, `ggforce::geom_mark_circle()`, `ggforce::geom_mark_ellipse()`, or `ggforce::geom_mark_hull()` to annotate simple feature objects. This function modifies the geometry and sets the stat to "st_coordinates" to make it easy to use these annotation geoms with simple feature objects.

Usage

```
layer_marked(
  data,
  fn = NULL,
  mapping = NULL,
  label_col = NULL,
  desc_col = NULL,
  geom = NULL,
  center = FALSE,
  font_family = NULL,
  font_face = c("bold", "plain"),
  font_color = NULL,
  expand = ggplot2::unit(5, "mm"),
  radius = expand,
  stat = "sf_coordinates",
  drop_shadow = FALSE,
  shadow_params = list(x_offset = 5, y_offset = 5, sigma = 0.5, ...),
  ...
)
```

Arguments

`data`	A `sf`, `sfc`, or bbox object that can be converted with sfext::as_sf
`fn`	Function to apply to data before passing to geom, typically a predicate or filter to define area for annotation. A filter can also be passed to any of the {ggforce} functions using the filter aesthetic. Default: NULL
`mapping`	Aesthetic mapping to pass to geom, Default: `NULL`
`label_col`	Label column name
`desc_col`	Column name to use for description. Defaults to `NULL`.
`geom`	geom to use for layer (options include "rect", "circle", "ellipse", or "hull"), Default: `NULL`
`center`	If `FALSE`, use sfext::st_cast_ext MULTIPOLYGON and POLYGON data to POINT; If `TRUE`, use sfext::st_center use centroid as the feature geometry. Defaults to FALSE.
`font_family`, `font_face`, `font_color`	Parameters passed to label.family, label.fontface, and label.colour. If `NULL`, values are set to match ggplot2::geom_label defaults. Defaults to `NULL`.
`expand`	A numeric or unit vector of length one, specifying the expansion amount. Negative values will result in contraction instead. If the value is given as a numeric it will be understood as a proportion of the plot area width.
`radius`	As expand but specifying the corner radius.
`stat`	stat to pass to ggforce function; defaults to "sf_coordinates"
`drop_shadow`	If `TRUE`, use `ggfx::with_shadow()` to add a drop shadow to the label layer with shadow_params. Defaults to `FALSE`.
`shadow_params`	Parameters passed to `ggfx::with_shadow()` if `drop_shadow = TRUE`. Defaults to `list(x_offset = 5, y_offset = 5, sigma = 0.5)`.
`...`	Additional parameters passed to `ggforce::geom_mark_rect()`, `ggforce::geom_mark_circle()`, `ggforce::geom_mark_ellipse()`, or `ggforce::geom_mark_hull()` or on to `ggplot2::layer()`

Geometry conversion for MULTIPOLYGON or POLYGON data

If cast is `FALSE` and the data geometry type is MULTIPOLYGON or POLYGON, the annotation uses a centroid for the annotation geometry. If cast is `TRUE`, the data is converted to POINT geometry using `sfext::st_cast_ext()` and the modified geometry passed on to selected geom.

Setting label and description

Labels and descriptions can be included in the aesthetic mapping for the layer consistent with the standard documented approaches for all four functions.

Labels and descriptions also can be set in two non-standard ways:

- Setting label_col and/or or desc_col to character strings with the column names for labels and/or descriptions
- Setting label_col and/or desc_col with the value of the label/description

If the first approach is used, label_col and desc_col can also can be created or modified by a function provided to the fn parameter. Otherwise, the columns must be present in the data to work. If the second approach is used, the length and order of vector provided to label_col and/or desc_col must match that length and order of the data (after the fn is applied).

See Also

- `ggforce::geom_mark_rect()`
- `ggforce::geom_mark_circle()`
- `ggforce::geom_mark_ellipse()`
- `ggforce::geom_mark_hull()`

Examples

```
## Not run:
if (interactive()) {
  # Mark the 12th Council District on a Baltimore neighborhood basemap
  ggplot::ggplot() +
    layer_location_data(
      data = "neighborhoods",
      package = "mapbaltimore"
    ) +
    layer_marked(
      data = getdata::get_location(
        type = "council district",
        name = "District 12",
        package = "mapbaltimore"
      ),
      geom = "hull",
      color = "red",
      size = 1.5
    )
}

## End(Not run)
```

layer_markers *Create a ggplot2 layer with map markers or numbered markers*

Description

If make is TRUE, groupname_col, group_meta, crs, and fn is all passed on to make_markers.

Usage

```
layer_markers(
  data,
  mapping = NULL,
  geom = "sf",
  make = FALSE,
  groupname_col = NULL,
  group_meta = NULL,
  crs = getOption("maplayer.crs", default = 3857),
  number = FALSE,
  num_by_group = FALSE,
  num_style = NULL,
  num_start = 1,
  suffix = NULL,
  sort = "dist_xmin_ymax",
  desc = FALSE,
  fn = NULL,
  ...
)

layer_numbers(
  data,
  mapping = NULL,
  geom = "label",
  make = FALSE,
  groupname_col = NULL,
  style = "roundrect",
  size = 5,
  sort = "dist_xmin_ymax",
  num_by_group = FALSE,
  num_style = NULL,
  num_start = 1,
  suffix = NULL,
  desc = FALSE,
  fn = NULL,
  crs = getOption("maplayer.crs", default = 3857),
  label.size = 0,
  label.padding = ggplot2::unit(size/10, "lines"),
  label.r = label.padding * 1.5,
  hjust = 0.5,
  vjust = 0.5,
  ...
)

make_markers(
  data,
  groupname_col = NULL,
  group_meta = NULL,
  join = sf::st_intersects,
  geo = FALSE,
  coords = c("lon", "lat"),
  address = "address",
  point = TRUE,
  crs = NULL,
  fn = NULL,
  ...
)
```

Arguments

`data`	Character string (e.g. url, file path, or name of data from package) for a spatial data or a `sf`, `sfc`, or bbox object with geometry overlapping the location. If data is `NULL`, all unnamed parameters are passed to `sfext::read_sf_ext()` with a bbox based on location. If data is not `NULL` and not a data.frame, url, file path, or bbox, conversion to a sf object will still always be attempted with `sfext::as_sf()`.
`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`geom`	A character string indicating which ggplot2 geom to use, Default: 'sf'. Options include "sf" (`ggplot2::geom_sf()`), "icon" (`layer_icon()`), "markers" (`layer_markers()`), "sf_text" (`ggplot2::geom_sf_text()`), and "sf_label" (`ggplot2::geom_sf_label` See details for a full list.
`make`	If `TRUE`, pass data to make_markers.
`groupname_col`	Group column name, used to join group metadata if group_meta is a non-spatial data frame; Default: `NULL`
`group_meta`	Group metadata as a data frame or sf object that intersect with markers (using join function); Default: `NULL`
`crs`	Coordinate reference system for markers, Default: `NULL`
`number`	If `TRUE`, number markers using `layer_markers()` (not currently supported)
`num_by_group`	If `TRUE`, numbers are added by group based on groupname_col.
`num_style`	Style of enumeration, either "arabic", "alph", "Alph", "roman", "Roman".
`num_start`	Starting number; defaults to 1.
`suffix`	Character to appended to "number" column. (e.g. "." for "1." or ":" for "1:"). Can also be a character vector with the same length as the number column.

`sort`	Sort column name, Default: "dist_xmin_ymax".
`desc`	If `TRUE`, sort descending; default `FALSE`.
`fn`	Function to apply to data before results; gives warning if data is grouped; Default: `NULL`
`...`	Additional parameters passed to `get_location_data()` when using `make = TRUE` to pass data to make_markers
`style`	Marker style; defaults to `NULL` for `layer_markers()` (supports "facet"); defaults to "roundrect" for `layer_markers()` when `numbered = TRUE` (default is only supported option at present).
`size`	Marker size, Default: 5
`label.size`	Size of label border, in mm.
`label.padding`	Amount of padding around label. Defaults to 0.25 lines.
`label.r`	Radius of rounded corners. Defaults to 0.15 lines.
`hjust, vjust`	Horizontal and vertical justification.
`join`	Spatial relation function to combine data with group_meta, passed to `sf::st_join()`. Defaults to `sf::st_intersects()`.
`geo`	If `FALSE`, pass data to `getdata::get_location_data()` with `geo = TRUE` parameter.
`coords`	Coordinate columns for input data.frame or output sf object (if geometry is 'centroid' or 'point') Default: c("lon", "lat").
`address`	Address column name passed to `tidygeocoder::geocode()` or tidygeocoder::geo
`point`	If `TRUE` and data does not have POINT or MULTIPOINT geometry, convert to POINT data using `sf::st_centroid()`.

Value

ggplot2 layers

Examples

```
nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))
nc <- sf::st_transform(nc, 3857)

basemap <-
  ggplot() +
  layer_location_data(
    data = nc,
    fill = NA
  )

basemap +
  layer_markers(
    data = nc[1:10],
    mapping = aes(size = AREA),
    make = TRUE
  )
```

```
large_nc <-
  getdata::get_location_data(
    data = nc,
    fn = ~ dplyr::filter(.x, AREA > 0.2)
  )

large_nc$number <- 1
large_nc$dist <- 2

basemap +
  layer_numbers(
    data = large_nc,
    mapping = aes(fill = NAME),
    sort = "dist_xmax_ymin",
    num_style = "Roman",
    geom = "label",
    size = 3
    ) +
  guides(fill = "none")
```

`layer_mask` *Create a mask layer based on a simple feature object*

Description

Returns a mask for an area or areas as a simple feature object. `neatline = TRUE` only works for this layer if data is passed directly; not inherited.

Usage

```
layer_mask(
  data = NULL,
  dist = NULL,
  diag_ratio = NULL,
  unit = NULL,
  asp = NULL,
  crs = getOption("maplayer.crs", default = 3857),
  fill = "white",
  color = NA,
  alpha = 0.5,
  mask = NULL,
  neatline = FALSE,
  expand = TRUE,
  ...
)

set_mask(x = NULL, mask = TRUE, data = NULL, crs = NULL, ...)
```

Arguments

`data`	`sf`, `sfc`, or `bbox` object. If dist, diag_ratio, and/or asp are provided, data is adjusted to set the boundaries of the mask. If data is not provided, `mask` is required. If data is `NA`, mask is continuous otherwise the data is erased from the mask area.
`dist`	buffer distance in units. Optional.
`diag_ratio`	ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will be used. Ignored when `dist` is provided.
`unit`	Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"
`asp`	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, `get_asp()` returns the same value without modification.
`crs`	Coordinate reference system of bounding box to return; defaults to `NULL` which maintains the crs of the input object.
`fill`	mask fill color; defaults to "white"
`color`	mask edge color; defaults to `NA`
`alpha`	mask alpha/transparency; defaults to 0.5
`mask`	A `sf`, `sfc`, or `bbox` object to define the mask area. `diag_ratio`, `dist`, and `asp` parameters are ignored if a `mask` is provided. defaults to `NULL`
`neatline`	A logical object, `CoordSf` object, or a list containing a `CoordSf` object (typically from `layer_neatline()`) added to layer by `set_neatline()`. • If logical and `TRUE`, add a neatline layer using data, crs and any additional parameters passed to ... If logical and `FALSE`, return x as is. • If object from `layer_neatline()`, add it as is.
`expand`	If `TRUE`, the default, adds a small expansion factor to the limits to ensure that data and axes don't overlap. If `FALSE`, limits are taken exactly from the data or `xlim`/`ylim`.
`...`	Additional parameters to pass to `ggplot2::geom_sf()`
`x`	A `sf`, `sfc`, `bbox`, `sfg`, `Raster`, `Spatial`, `Extent`, `numeric`, or `character` object (a place name passed to `osmdata::getbb()`). See `as_bbox()` for more details.

Value

`ggplot2::geom_sf()` function.

layer_neatline *Set map limits to a bounding box with a buffer and set aspect ratio*

Description

Set limits for a map to the bounding box of a feature using `ggplot2::coord_sf()`. Optionally, adjust the x size by applying a buffer and/or adjust the aspect ratio of the limiting bounding box to match a set aspect ratio.

Usage

```
layer_neatline(
  data = NULL,
  dist = getOption("maplayer.dist"),
  diag_ratio = getOption("maplayer.diag_ratio"),
  unit = getOption("maplayer.unit", default = "meter"),
  asp = getOption("maplayer.asp"),
  crs = getOption("maplayer.crs"),
  nudge = getOption("maplayer.nudge"),
  color = "black",
  linewidth = 0.5,
  linetype = "solid",
  bgcolor = "white",
  expand = TRUE,
  hide_grid = TRUE,
  label_axes = "----",
  axis.title = NULL,
  axis.text = NULL,
  axis.ticks = NULL,
  axis.ticks.length = ggplot2::unit(x = 0, units = "mm"),
  axis.line = NULL,
  panel.grid = NULL,
  panel.grid.major = NULL,
  panel.grid.minor = NULL,
  panel.border = NULL,
  panel.background = NULL,
  plot.background = NULL,
  plot.margin = NULL,
  default_plot_margin = ggplot2::margin(1, 1, 1, 1),
  xlim = NULL,
  ylim = NULL,
  ...
)

theme_grid(
  hide_grid = TRUE,
  grid = FALSE,
  panel.grid = NULL,
  panel.grid.major = NULL,
  panel.grid.minor = NULL
)

theme_sf_axis(
  label_axes = "----",
  axis.title = NULL,
  axis.text = NULL,
  axis.text.x = NULL,
  axis.text.y = NULL,
  axis.ticks = NULL,
  axis.ticks.length = ggplot2::unit(x = 0, units = "mm"),
  axis.line = NULL,
  ...
)

set_neatline(x = NULL, neatline = TRUE, data = NULL, crs = NULL, ...)
```

Arguments

`data`	A `sf`, `sfc`, or bbox class object.
`dist`	buffer distance in units. Optional.
`diag_ratio`	ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will be used. Ignored when `dist` is provided.
`unit`	Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"
`asp`	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, `get_asp()` returns the same value without modification.
`crs`	The coordinate reference system (CRS) into which all data should be projected before plotting. If not specified, will use the CRS defined in the first sf layer of the plot.
`nudge`	Passed as to parameter `st_nudge()` when not `NULL`. A numeric vector, a `sf` object, or any other object that can be converted to a simple feature collection with `as_sfc()`..
`color`	Color of panel border, Default: 'black'
`linewidth`	Line width of panel border, Default: 0.5
`linetype`	Line type of panel border, Default: 'solid'
`bgcolor`	Fill color of panel background; defaults to "white". If "none", panel background is set to `ggplot2::element_blank()`
`expand`	If `TRUE`, the default, adds a small expansion factor to the limits to ensure that data and axes don't overlap. If `FALSE`, limits are taken exactly from the data or `xlim`/`ylim`.

`hide_grid`	If `TRUE`, hide grid lines. Default: `TRUE`
`label_axes`	A description of which axes to label passed to `ggplot2::coord_sf()`; defaults to '—-' which hides all axes labels.
`axis.title`, `axis.text`, `axis.ticks`, `axis.ticks.length`, `axis.line`	Theme elements passed as is if label_axes is anything other than "—-".
`panel.grid`, `panel.grid.major`, `panel.grid.minor`	Passed as is if hide_grid is FALSE.
`panel.border`, `panel.background`, `plot.background`, `plot.margin`	panel.border is used as is if not NULL or `ggplot2::element_blank()` if it is NULL unless color is NA or "none". panel.background and plot.background are used as is or `ggplot2::element_blank()` if bg color is NA or "none". plot.margin is set to `ggplot2::margin(1, 1, 1, 1)` if `NULL` or `ggplot2::margin(0, 0, 0, 0)` if expand is `FALSE`.
`default_plot_margin`	Defaults to `ggplot2::margin(1, 1, 1, 1)`. Ignored if `expand = FALSE`
`xlim`, `ylim`	Limits for the x and y axes. These limits are specified in the units of the default CRS. By default, this means projected coordinates (`default_crs = NULL`). How limit specifications translate into the exact region shown on the plot can be confusing when non-linear or rotated coordinate systems are used as the default crs. First, different methods can be preferable under different conditions. See parameter `lims_method` for details. Second, specifying limits along only one direction can affect the automatically generated limits along the other direction. Therefore, it is best to always specify limits for both x and y. Third, specifying limits via position scales or `xlim()`/`ylim()` is strongly discouraged, as it can result in data points being dropped from the plot even though they would be visible in the final plot region.
`...`	Additional parameters passed to `ggplot2::coord_sf()`.
`grid`	If `grid` is `TRUE` and `hide_grid` is `FALSE`, grid will be included in the theme. Otherwise, suppress the grid.
`axis.text.x`, `axis.text.y`	Passed to `ggplot2::theme()`
`x`	For `set_neatline()`, a `ggplot` class object, `ggproto` class object or list of `ggproto` objects to combine with neatline layer.
`neatline`	A logical object, `CoordSf` object, or a list containing a `CoordSf` object (typically from `layer_neatline()`) added to layer by `set_neatline()`. • If logical and `TRUE`, add a neatline layer using data, crs and any additional parameters passed to ... If logical and `FALSE`, return x as is. • If object from `layer_neatline()`, add it as is.

Value

List of ggplot2::coord_sf and ggplot2::theme calls.

See Also

`ggplot2::CoordSf()`

Other layer: `layer_frame()`, `layer_location_data()`, `layer_scaled()`

Examples

```
library(ggplot2)

nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))

ggplot() +
  layer_location_data(data = nc) +
 layer_neatline(data = nc[1, ], asp = 1, color = "red", linewidth = 1, linetype = "dashed")

ggplot() +
  layer_location_data(data = nc) +
  layer_neatline(data = nc[1, ], dist = 20, unit = "mi", color = "none")
```

---

layer_repel *Use ggrepel to create text annotations based on simple features*

Description

Use ggrepel::geom_label_repel or ggrepel::geom_text_repel with `ggplot2::stat_sf_coordinates()` to create a layer of textual annotations repelled from simple feature locations.

Usage

```
layer_repel(
  mapping = aes(),
  data = NULL,
  label_col = "name",
  geom = c("text", "label"),
  location_lims = NULL,
  xlim = c(NA, NA),
  ylim = c(NA, NA),
  ...
)

geom_sf_label_repel(mapping = aes(), data = NULL, label_col = "name", ...)

geom_sf_text_repel(mapping = aes(), data = NULL, label_col = "name", ...)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes` or `aes_`. If specified and `inherit.aes = TRUE` (the default), is combined with the default mapping at the top level of the plot. You only need to supply `mapping` if there isn't a mapping defined for the plot.
`data`	A data frame. If specified, overrides the default data frame defined at the top level of the plot.

`label_col`	Column name to use for label aesthetic mapping. Optional if label is provided to mapping; required otherwise.
`geom`	Character vector with geom to use, "text" for `ggrepel::geom_text_repel()` or "label" for `ggrepel::geom_label_repel()`.
`location_lims`	A `sf`, `sfc`, or `bbox` object to use in setting xlim and ylim values if no xlim and ylim value area provided. Using this parameter constrains labels to the bounding box of `location_lims`.
`xlim, ylim`	Limits for the x and y axes. Text labels will be constrained to these limits. By default, text labels are constrained to the entire plot area.
`...`	Arguments passed on to `ggrepel::geom_label_repel`

`position` Position adjustment, either as a string, or the result of a call to a position adjustment function.

`parse` If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath

`box.padding` Amount of padding around bounding box, as unit or number. Defaults to 0.25. (Default unit is lines, but other units can be specified by passing `unit(x, "units")`).

`label.padding` Amount of padding around label, as unit or number. Defaults to 0.25. (Default unit is lines, but other units can be specified by passing `unit(x, "units")`).

`point.padding` Amount of padding around labeled point, as unit or number. Defaults to 0. (Default unit is lines, but other units can be specified by passing `unit(x, "units")`).

`label.r` Radius of rounded corners, as unit or number. Defaults to 0.15. (Default unit is lines, but other units can be specified by passing `unit(x, "units")`).

`label.size` Size of label border, in mm.

`min.segment.length` Skip drawing segments shorter than this, as unit or number. Defaults to 0.5. (Default unit is lines, but other units can be specified by passing `unit(x, "units")`).

`arrow` specification for arrow heads, as created by `arrow`

`force` Force of repulsion between overlapping text labels. Defaults to 1.

`force_pull` Force of attraction between a text label and its corresponding data point. Defaults to 1.

`max.time` Maximum number of seconds to try to resolve overlaps. Defaults to 0.5.

`max.iter` Maximum number of iterations to try to resolve overlaps. Defaults to 10000.

`max.overlaps` Exclude text labels when they overlap too many other things. For each text label, we count how many other text labels or other data points it overlaps, and exclude the text label if it has too many overlaps. Defaults to 10.

`nudge_x,nudge_y` Horizontal and vertical adjustments to nudge the starting position of each text label. The units for `nudge_x` and `nudge_y` are the same as for the data units on the x-axis and y-axis.

`xlim,ylim` Limits for the x and y axes. Text labels will be constrained to these limits. By default, text labels are constrained to the entire plot area.

`na.rm` If `FALSE` (the default), removes missing values with a warning. If `TRUE` silently removes missing values.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes.

`direction` "both", "x", or "y" – direction in which to adjust position of labels

`seed` Random seed passed to `set.seed`. Defaults to `NA`, which means that `set.seed` will not be called.

`verbose` If `TRUE`, some diagnostics of the repel algorithm are printed

`inherit.aes` If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders`.

`layer_scaled` *Create a ggplot2 layer scaled to a paper and orientation for a location*

Description

Uses layer_neatline, sfext::standard_scales, and sfext::convert_dist_scale.

Usage

```
layer_scaled(
  data = NULL,
  dist = NULL,
  diag_ratio = NULL,
  unit = NULL,
  asp = NULL,
  crs = getOption("maplayer.crs", default = 3857),
  scale = NULL,
  paper = NULL,
  orientation = NULL,
  clip = FALSE
)
```

Arguments

`data`	A `sf`, `sfc`, or bbox class object.
`dist`	distance to convert. If paper is provided, dist is optional and paper width and height are used as dist.
`diag_ratio`	ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will be used. Ignored when `dist` is provided.

unit	Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"
asp	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, `get_asp()` returns the same value without modification.
crs	Coordinate reference system of bounding box to return; defaults to `NULL` which maintains the crs of the input object.
scale	Scale name from `standard_scales[["scale"]]`.
paper	Paper, Default: 'letter'.
orientation	Orientation "portrait", "landscape", or "square", Default: 'portrait'.
clip	If `TRUE`, create scaled layer even if the data is cut off; defaults to `FALSE`.

See Also

Other layer: `layer_frame()`, `layer_location_data()`, `layer_neatline()`

`make_location_map` *Make a ggplot map using* `layer_location_data()`

Description

Location is used as the data parameter of layer_location_data so this function is primarily appropriate for the layer_mapbox (`geom = "mapbox"`).

Usage

```
make_location_map(
  location = NULL,
  dist = NULL,
  diag_ratio = NULL,
  unit = NULL,
  asp = NULL,
  data = NULL,
  crs = NULL,
  paper = NULL,
  width = NULL,
  height = NULL,
  units = "in",
  orientation = NULL,
  geom = "sf",
  basemap = TRUE,
  bg_layer = NULL,
  layer = NULL,
  fg_layer = NULL,
  addon = NULL,
  neatline = FALSE,
  labs_ext_params = list(...),
  save = FALSE,
  ggsave_params = list(dpi = 300, ...),
  ...,
  env = caller_env(),
  call = caller_env()
)

make_social_map(
  location,
  dist = NULL,
  diag_ratio = NULL,
  unit = NULL,
  asp = NULL,
  crs = 3857,
  image = NULL,
  platform = NULL,
  format = NULL,
  orientation = NULL,
  basemap = TRUE,
  geom = "mapbox",
  save = FALSE,
  ggsave_params = list(fileext = "jpeg", dpi = 72, ...),
  ...
)

make_image_map(
  image_path,
  location = NULL,
  dist = NULL,
  diag_ratio = NULL,
  unit = NULL,
  asp = NULL,
  data = NULL,
  crs = 3857,
  paper = "Letter",
  orientation = NULL,
  geom = "mapbox",
  style_url = NULL,
  basemap = TRUE,
  bg_layer = NULL,
  fg_layer = NULL,
  save = FALSE,
  ggsave_params = list(dpi = 300, ...),
  image_geom = "label",
  groupname_col = NULL,
  group_meta = NULL,
  number = FALSE,
  num_by_group = FALSE,
  num_style = NULL,
  num_start = 1,
  suffix = NULL,
  sort = "dist_xmin_ymax",
  desc = FALSE,
  ...
)

make_layer_map(
  bg_layer = NULL,
  layer = NULL,
  fg_layer = NULL,
  addon = NULL,
  basemap = NULL,
  neatline = NULL,
  labs_ext_params = NULL,
  save = FALSE,
  ggsave_params = list(width = 5, height = 4, unit = "in", dpi = 300),
  env = caller_env(),
  call = caller_env()
)
```

Arguments

`location`	A sf object passed to `layer_location_data()` with data if layer is `NULL`. location is ignored if layer is provided. If data is `NULL`, location is passed as data to facilitate using this function with geom = `"mapbox"` where data is used to define the map area. Defaults to `NULL`
`dist`	buffer distance in units. Optional.
`diag_ratio`	ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will be used. Ignored when `dist` is provided.
`unit`	Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"
`asp`	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, `get_asp()` returns the same value without modification.
`data`	Character string (e.g. url, file path, or name of data from package) for a spatial data or a `sf`, `sfc`, or bbox object with geometry overlapping the location. If data is `NULL`, all unnamed parameters are passed to `sfext::read_sf_ext()` with a bbox based on location. If data is not `NULL` and not a data.frame, url, file path, or bbox, conversion to a sf object will still always be attempted with `sfext::as_sf()`.

`crs`	Coordinate reference system of bounding box to return; defaults to `NULL` which maintains the crs of the input object.
`paper`	Paper matching name from `paper_sizes` (e.g. "letter"). Not case sensitive.
`width`	Page width in "in", "px" or "mm" units. Default: `NULL`
`height`	Page height in "in", "px" or "mm" units. Default: `NULL`
`units`	Units to convert page dimensions to using `convert_unit_type()`.
`orientation`	Page orientation, Default: `NULL`. Supported options are "portrait", "landscape", or "square".
`geom`	A character string indicating which ggplot2 geom to use, Default: 'sf'. Options include "sf" (`ggplot2::geom_sf()`), "icon" (`layer_icon()`), "markers" (`layer_markers()`), "sf_text" (`ggplot2::geom_sf_text()`), and "sf_label" (`ggplot2::geom_sf_label` See details for a full list.
`basemap`	Either a logical vector or ggplot object. If **logical** and `TRUE`, add x to `ggplot2::ggplot()`. If `FALSE`, return x as is. If a **ggplot**, add x to basemap object. If a **ggproto** object (or list that contains a **ggproto** object), add x and basemap object to `ggplot2::ggplot()`.
`bg_layer`, `fg_layer`, `addon`	A ggplot2 layer or a list of ggproto objects (e.g. scales, labels, etc.) to add to the background or foreground of the primary map layer defined by `"geom"` and other parameters. If the geom creates an opaque layer or layer is an opaque layer (e.g. a layer produced by `layer_mapbox()`) that covers the full map extent, the bg_layer will not be visible.
`layer`	A ggplot2 layer or a list of ggproto objects. If layer is provided, all parameters passed to `layer_location_data()` (including data, location, dist, diag_ratio, unit, asp, crs, and geom) will be ignored. In this case, the function simply stacks the bg_layer, layer, and fg_layer objects then applies the basemap and neatline (using the `set_basemap()` and `set_neatline()` helper functions.)
`neatline`	A logical object, `CoordSf` object, or a list containing a `CoordSf` object (typically from `layer_neatline()`) added to layer by `set_neatline()`. • If logical and `TRUE`, add a neatline layer using data, crs and any additional parameters passed to ... If logical and `FALSE`, return x as is. • If object from `layer_neatline()`, add it as is.
`labs_ext_params`	Optional parameters passed to `labs_ext()`.
`save`	If `TRUE`, save file with ggsave_ext using `ggsave_params`. Defaults to `FALSE`.
`ggsave_params`	List of parameters passed to `papersize::ggsave_ext()`.
`...`	Additional parameters passed to `layer_location_data()` for `make_location_map()` or `make_social_map()` or to `layer_markers()` for `make_image_map()`.
`env`	Environment for evaluation of `labs_ext()` if labs_ext_params is supplied.
`call`	The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

`image`	Image name passed to name parameter of `get_social_size()`.
`platform`	Social media platform, "Instagram", "Facebook", or "Twitter", Default: `NULL`
`format`	Image format, "post", "story", or "cover", Default: `NULL`
`image_path`	path to location of images for `make_image_map()`
`style_url`	A Mapbox style url; defaults to `NULL`.
`image_geom`	For `make_image_map()`, geom to use with layer_markers to mark the location of images (based on EXIF metadata).
`groupname_col`	Group column name, used to join group metadata if group_meta is a non-spatial data frame; Default: `NULL`
`group_meta`	Group metadata as a data frame or sf object that intersect with markers (using join function); Default: `NULL`
`number`	If `TRUE`, number markers using `layer_markers()` (not currently supported)
`num_by_group`	If `TRUE`, numbers are added by group based on groupname_col.
`num_style`	Style of enumeration, either "arabic", "alph", "Alph", "roman", "Roman".
`num_start`	Starting number; defaults to 1.
`suffix`	Character to appended to "number" column. (e.g. "." for "1." or ":" for "1:"). Can also be a character vector with the same length as the number column.
`sort`	Sort column name, Default: "dist_xmin_ymax".
`desc`	If `TRUE`, sort descending; default `FALSE`.

Details

Using `make_image_map()`:

`make_image_map()` wraps `sfext::read_sf_exif()` and `make_location_map()`. It is designed for making simple maps of photos in combination with reference tables.

Examples

```
nc <- sf::read_sf(system.file("shape/nc.shp", package = "sf"))

make_location_map(
  location = nc
)

name <- "North Carolina"

make_location_map(
  location = nc,
  labs_ext_params = list(
    title = "Map of {name}"
  )
)

make_location_map(
  data = nc,
  location = nc[2, ],
  dist = 2,
  unit = "mi",
  crop = FALSE,
  addon = ggplot2::theme_minimal(),
  labs_ext_params = list(
    title = "Map of {nc[2, ]$NAME} County"
  )
)

make_location_map(
  basemap = ggplot(data = nc[2,]),
  fg_layer = geom_sf_text(aes(label = NAME)),
  data = nc,
  location = nc[2, ],
  mapping = aes(fill = NAME),
  addon = guides(fill = "none"),
  dist = 2,
  unit = "mi",
  crop = FALSE,
  neatline = TRUE,
  labs_ext_params = list(
    title = "Map of {nc[2, ]$NAME} and surrounding {name} counties"
  )
)
```

make_mapbox_map *Make a map using layer_mapbox*

Description

Wraps `make_layer_map()` and passes layer created with `layer_mapbox()` to basemap and the neatline parameters to `layer_neatline()` (using the same data as the Mapbox background layer). The neatline parameters are only used if neatline is `NULL`.

Usage

```
make_mapbox_map(
  data = NULL,
  dist = NULL,
  diag_ratio = NULL,
  unit = "meter",
  asp = NULL,
  style_url = "mapbox://styles/mapbox/satellite-streets-v11",
  style_id = NULL,
  username = NULL,
  basemap = TRUE,
  scale = 0.75,
  scaling_factor = "1x",
  attribution = TRUE,
  logo = TRUE,
  access_token = NULL,
  neatline = NULL,
  color = "black",
  bgcolor = "white",
  linewidth = 0.5,
  linetype = "solid",
  expand = TRUE,
  hide_grid = TRUE,
  label_axes = "----",
  width = NULL,
  height = NULL,
  units = NULL,
  orientation = NULL,
  location = NULL,
  ...
)
```

Arguments

`data`	A `sf`, `sfc`, or bbox object.
`dist`	buffer distance in units. Optional.
`diag_ratio`	ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will be used. Ignored when `dist` is provided.
`unit`	Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"
`asp`	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, `get_asp()` returns the same value without modification.
`style_url`	Map style url used to fill style_id and username parameters, Default: "mapbox://styles/mapbox/satellite-streets-v11"
`style_id`	A style ID (required if style_url is `NULL`).
`username`	A Mapbox username (required if `style_url` = `NULL`).
`basemap`	If FALSE, create a standalone layer; if `TRUE`, the layer is preceded by `ggplot2::ggplot()` to allow use as a basemap, Default: `TRUE`
`scale`	ratio to scale the output image; `scale` = 1 will return the largest possible image. defaults to 0.5
`scaling_factor`	The scaling factor of the tiles; either `"1x"` (the default) or `"2x"`
`attribution`	Controls whether there is attribution on the image. Defaults to `TRUE`. If `FALSE`, the watermarked attribution is removed from the image. You still have a legal responsibility to attribute maps that use OpenStreetMap data, which includes most maps from Mapbox. If you specify `attribution` = `FALSE`, you are legally required to include proper attribution elsewhere on the webpage or document.

`logo`	Controls whether there is a Mapbox logo on the image. Defaults to `TRUE`.
`access_token`	A Mapbox access token; which can be set with mb_access_token.
`neatline`	If `TRUE`, add a neatline matching the provided data, Default: `TRUE`
`color`	Color of panel border, Default: 'black'
`bgcolor`	Fill color of panel background; defaults to "white". If "none", panel background is set to `ggplot2::element_blank()`
`linewidth`	Line width of panel border, Default: 0.5
`linetype`	Line type of panel border, Default: 'solid'
`expand`	If `TRUE`, the default, adds a small expansion factor to the limits to ensure that data and axes don't overlap. If `FALSE`, limits are taken exactly from the data or `xlim`/`ylim`.
`hide_grid`	If `TRUE`, hide grid lines. Default: `TRUE`
`label_axes`	A description of which axes to label passed to `ggplot2::coord_sf()`; defaults to '—-' which hides all axes labels.
`width`, `height`	Page width and height. Both are required, if asp is `NULL`. Default to `NULL`.
`units`	Units for width and height. Required unless units is included in dims. Passed to `as_unit_type()` to validate.
`orientation`	Page orientation, Default: `NULL`. Supported options are "portrait", "landscape", or "square". If width and height suggest a portrait orientation when orientation = "landscape", the dimensions are reversed so the page dimensions match the provided orientation.
`location`	If `location` is provided and `data` is `NULL`, `location` is used in place of `data`.
`...`	Arguments passed on to `make_layer_map`

`labs_ext_params` Optional parameters passed to `labs_ext()`.

`ggsave_params` List of parameters passed to `papersize::ggsave_ext()`.

`layer` A ggplot2 layer or a list of ggproto objects. If layer is provided, all parameters passed to `layer_location_data()` (including data, location, dist, diag_ratio, unit, asp, crs, and geom) will be ignored. In this case, the function simply stacks the bg_layer, layer, and fg_layer objects then applies the basemap and neatline (using the `set_basemap()` and `set_neatline()` helper functions.)

`bg_layer,fg_layer,addon` A ggplot2 layer or a list of ggproto objects (e.g. scales, labels, etc.) to add to the background or foreground of the primary map layer defined by `"geom"` and other parameters. If the geom creates an opaque layer or layer is an opaque layer (e.g. a layer produced by `layer_mapbox()`) that covers the full map extent, the bg_layer will not be visible.

`save` If `TRUE`, save file with ggsave_ext using `ggsave_params`. Defaults to `FALSE`.

`env` Environment for evaluation of `labs_ext()` if labs_ext_params is supplied.

`call` The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

`map_icons` *Map icons*

Description

An index of map icons from seven sources:

- mapbox/maki
- ideditor/temaki
- manifestinteractive/weather-underground-icons
- openstreetmap/map-icons
- openstreetmap/lane-icons
- Esri/calcite-point-symbols
- NPMap Symbol Library

Most of these icon sources use open licenses. Maki, Temaki, and the OSM lane icons all use a CC0 license. The Weather Underground Icons use an MIT license. The OSM map icons use an unspecified PD style license. The Calcite icons are available under the Esri Master License Agreement (MLA). The NPMap Symbol Library is created by the National Park Service so is assumed to be a Public Domain work in the U.S..

This index was last updated on 2024-04-08 and may not include any icons that have been added since that date.

Usage

```
map_icons
```

Format

A tibble with 5,036 rows and 5 variables:

`name` Icon name

`url` Icon URL on GitHub repo

`style` Icon style (blank if not applicable)

`size` Icon width/height (pixels)

`repo` GitHub repository for icon collection

Details

The name column is not unique so a px or source may be required when using `layer_icon()`.

scale_group_data *Create discrete fill and color scales for grouped data*

Description

Designed for use with layer_group_data. group_data_pal generates palettes that are passed to ggplot2::scale_fill_manual and ggplot2::scale_color_manual.

Usage

```
scale_group_data(
  ...,
  data,
  col = NULL,
  palette = NULL,
  n = NULL,
  direction = 1,
  na.value = "grey50",
  drop = FALSE,
  limits = NULL,
  aesthetics = "fill"
)

group_data_pal(
  data,
  palette = NULL,
  col = NULL,
  n = NULL,
  direction = 1,
  pkg = NULL
)

get_group_data_pal_scale(data, col = NULL, palette = NULL, ...)
```

Arguments

`...`	Arguments passed on to `discrete_scale` `limits` One of: • `NULL` to use the default scale values • A character vector that defines possible values of the scale and their order • A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang lambda function notation. `drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`na.translate` Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify `na.translate = FALSE`.

`name` The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.

`labels` One of:

- `NULL` for no labels
- `waiver()` for the default labels computed by the transformation object
- A character vector giving labels (must be same length as `breaks`)
- An expression vector (must be the same length as breaks). See ?plotmath for details.
- A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.

`guide` A function used to create a guide or its name. See `guides()` for more information.

`call` The `call` used to construct the scale for reporting messages.

`super` The super class to use for the constructed scale

`data`	Data to use when generating scale or palette.
`col`	Grouping column found in data to use in generating scale or palette; defaults to `NULL`.
`palette`	Name of palette as a string. Must be on the form packagename::palettename.
`n`	Number of colors desired. If omitted, returns complete palette.
`direction`	Either `1` or `-1`. If `-1` the palette will be reversed.
`na.value`	The aesthetic value to use for missing (`NA`) values
`drop`	Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.
`limits`	One of: • `NULL` to use the default scale values • A character vector that defines possible values of the scale and their order • A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang lambda function notation.
`aesthetics`	Character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with. This can be useful, for example, to apply colour settings to the `colour` and `fill` aesthetics at the same time, via `aesthetics = c("colour", "fill")`.
`pkg`	Package name to use as prefix for palette name when selecting a palette with `paletteer::paletteer_d()`

See Also

`scales::viridis_pal()` `paletteer::paletteer_d()`

set_basemap *Create a base map by adding the object*

Description

Add a basemap to a ggplot2 layer.

Usage

```
set_basemap(x, basemap = FALSE, call = caller_env())

make_basemap(x, basemap = FALSE, call = caller_env())
```

Arguments

x	A ggproto object or list of ggproto objects.
basemap	Either a logical vector or ggplot object. If **logical** and `TRUE`, add x to `ggplot2::ggplot()`. If `FALSE`, return x as is. If a **ggplot**, add x to basemap object. If a **ggproto** object (or list that contains a **ggproto** object), add x and basemap object to `ggplot2::ggplot()`.
call	The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

theme_ext *Modify the text, margins, or legend for a ggplot theme*

Description

Helper functions for modifying a ggplot theme. The "replace" and "update" options for the method parameter are not currently working; keeping method = NULL or method = "set" is recommended.

Usage

```
theme_text(
  font_family = NULL,
  color = "black",
  geom_text = TRUE,
  hjust = NULL,
  vjust = NULL,
  method = NULL,
  ...
)
```

```
theme_margin(
  margin = "standard",
  paper = NULL,
  orientation = NULL,
  dist = NULL,
  unit = "in",
  block_width = NULL,
  header = 0,
  footer = 0,
  fill = NA,
  color = NA,
  linewidth = 0,
  size = NULL,
  method = NULL,
  ...
)

theme_legend(
  position = NULL,
  justification = NULL,
  margin = 8,
  unit = "pt",
  inset = TRUE,
  nudge_inset = 0.05,
  bgcolor = "white",
  title = list(face = "bold", align = 0),
  method = NULL,
  ...
)
```

Arguments

`font_family`	Font family, Default: 'Helvetica' If `NULL`, font_family is pulled from current set theme which is helpful for resetting all text families to the theme.
`color`	Color for text elements (passed to `ggplot2::element_text()` by theme_text), `plot.background` (passed to `ggplot2::element_rect()` by theme_margin). Default: `NA`.
`geom_text`	If `TRUE`, update text family for `ggplot2::geom_text()`, `ggplot2::geom_sf_text()`, `ggplot2::geom_label()`, and `ggplot2::geom_sf_label()` to match `font_family` and color. If `FALSE`, make no changes to the theme. Default: `TRUE`.
`hjust`, `vjust`	Horizontal and vertical justification.
`method`	Method with name of the ggplot2 geom function to use for modifying theme ("set", "update", or "replace"); defaults to `NULL`.
`...`	Additional parameters passed to `ggplot2::theme()`.
`margin`	Margin distance, a margin style supported by `get_margin()` or a margin object; defaults to 10.

`paper`	Paper, Default: 'letter'.
`orientation`	Page orientation, Default: `NULL`. Supported options are "portrait", "landscape", or "square".
`dist`	Margin distance (single value used to all sides), Default: `NULL`
`unit`	Legend margin units; defaults to 'pt'.
`block_width`	Plot or map width in units. If `paper` and `block_width` are provided, margins are half the distance between the two evenly distributed. This sets the margin distance for height as well as width so does not work well with header and footers and should be improved in the future.
`header, footer`	Header and footer height in units; defaults to 0. Please note: headers and footers are not currently supported for "px" units.
`fill`	Fill for `plot.background` theme element passed to `ggplot2::element_rect()` Default: `NA`.
`linewidth, size`	Passed to linewidth of `ggplot2::element_rect()` to define the plot.background theme element. linewidth is ignored if size is provided.
`position`	Legend position ("left","top", "right", "bottom") or a two-element numeric vector to set position using Normalized Parent Coordinates ("npc"); defaults NULL
`justification`	If `NULL`, justification is set to "center"; defaults to `NULL`. Use justification to set legend position if `inset = FALSE`. Supports "topleft", "bottomleft", "topright", or "bottomright" values.
`inset`	If `TRUE` and position is "topleft", "bottomleft", "topright", or "bottomright", place the legend in an inset position; defaults to `TRUE`.
`nudge_inset`	Position adjustment in "npc" units to use for inset legends. Defaults to 0.05.
`bgcolor`	Fill color for legend background; defaults to 'white'.
`title`	Attributes to use for legend.title text (e.g. face and align).

See Also

- `ggplot2::theme()`
- `ggplot2::margin()`
- `ggplot2::theme_get()`
- `ggplot2::update_geom_defaults()`

Index