Package 'ggmapcn'

October 10, 2025

Title Customizable China Map Visualizations

Version 0.2.0

Description A 'ggplot2' extension for visualizing China's map, offering customizable projections, boundary styles, and buffer zones for thematic maps.
Suitable for spatial data analysis and enhancing map visualization with flexible styling options.

License GPL-3

Encoding UTF-8

Depends R (>= 3.5.0), ggplot2 (>= 3.3.0)

Imports sf (>= 1.0.0), dplyr (>= 1.1.0), terra (>= 1.7), tidyterra (>= 0.6.0), curl (>= 5.0.0), rlang, digest, grid

URL https://rimagination.github.io/ggmapcn/

BugReports https://github.com/Rimagination/ggmapcn/issues

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.2

NeedsCompilation no

Author Liang Ren [aut, cre] (ORCID: <https://orcid.org/0000-0002-2360-7900>)

Maintainer Liang Ren <rl23@mails.tsinghua.edu.cn>

Repository CRAN

Date/Publication 2025-10-10 11:10:02 UTC

Contents

`ggmapcn-package` *ggmapcn: Customizable China Map Visualizations*

Description

A 'ggplot2' extension for visualizing China's map, offering customizable projections, boundary styles, buffer zones, and annotation layers such as compass and scale bars.

Details

Main features: - Projection-aware compass (‘annotation_compass‘) and scale bar (‘annotation_scalebar‘). - Built-in geodata management via ‘check_geodata‘. - Multiple compass styles (classic, rose, Sinan, guiding fish, etc.). - Easy integration with ggplot2 and sf workflows.

Author(s)

Maintainer: Liang Ren <rl23@mails.tsinghua.edu.cn> (ORCID)

See Also

Useful links:

- https://rimagination.github.io/ggmapcn/
- Report bugs at https://github.com/Rimagination/ggmapcn/issues

`annotation_compass` *Add a Spatially-Aware Compass*

Description

‘annotation_compass()‘ adds a compass (north arrow) to a ‘ggplot2‘ map. It can be aligned to **grid north** (top of the plot) or **true north** (geographic north). Styles can be supplied as a grob or a function returning a grob (e.g., ‘north_arrow_classic()‘, ‘compass_sinan()‘).

Usage

```
annotation_compass(
  mapping = NULL,
  data = NULL,
  ...,
  location = "bl",
  which_north = "grid",
  height = unit(1.5, "cm"),
  width = unit(1.5, "cm"),
  pad_x = unit(0.5, "cm"),
  pad_y = unit(0.5, "cm"),
  rotation = NULL,
  style = north_arrow_classic()
)
```

Arguments

`mapping, data`	Standard ggplot2 arguments (typically unused).
`...`	Additional parameters passed to the layer (rarely needed).
`location`	Character. One of ‘"tl"‘, ‘"tr"‘, ‘"bl"‘, ‘"br"‘ indicating top/bottom + left/right placement. Default: ‘"bl"‘.
`which_north`	Character. ‘"grid"‘ (default) or ‘"true"‘.
`height, width`	‘grid::unit‘. Compass box dimensions. Defaults: ‘1.5 cm‘.
`pad_x, pad_y`	‘grid::unit‘. Padding from panel edges. Defaults: ‘0.5 cm‘.
`rotation`	Numeric. Fixed rotation in degrees (counter-clockwise). When provided, it overrides ‘"grid"‘/‘"true"‘ logic.
`style`	A grob, ‘gList‘/‘gTree‘, or a function returning a grob (e.g., ‘north_arrow_classic()‘). Default: ‘north_arrow_classic()‘.

Details

* ‘"grid"‘ north: compass points straight up in the plotting space (no CRS needed). * ‘"true"‘ north: compass rotates toward geographic North Pole using the plot CRS. This requires a valid CRS available via ‘coord_sf()‘ or injected by setting ‘layer$geom_params$crs‘. * You can override any auto-rotation by providing ‘rotation‘ (degrees CCW). * The layer is annotation-like: it draws once per panel using the panel bounds.

Value

A ‘ggplot2‘ layer object.

See Also

[compass-styles]

Examples

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

base <- ggplot2::ggplot() +
  ggplot2::geom_sf(data = nc, fill = "grey90") +
  ggplot2::theme_minimal()

# Example 1: Grid north (no CRS required), bottom-left
base + annotation_compass()

# Example 2: Custom style & position (top-left)
base + annotation_compass(location = "tl", style = compass_sinan())

# Example 3: True north (requires a CRS)
base +
  ggplot2::coord_sf(crs = "+proj=lcc +lon_0=-100 +lat_1=33 +lat_2=45") +
  annotation_compass(which_north = "true")
```

`annotation_scalebar` *Add a Spatially-Aware Scale Bar*

Description

[annotation_scalebar()] adds a projection-aware scale bar to a [ggplot2::ggplot2-package] map. It detects the map's CRS and chooses a readable width and units automatically. Robust fallbacks prevent "zero-length unit" errors and allow the scale bar to render even when CRS information is limited.

Supported styles: - ‘"segment"‘ (minimal horizontal bar with ticks and labels) - ‘"ticks"‘ (baseline + vertical ticks) - ‘"bar"‘ (alternating black/white blocks)

Usage

```
annotation_scalebar(
  mapping = NULL,
  data = NULL,
  ...,
  location = "bl",
  style = "segment",
  fixed_width = NULL,
```

```
  crs_unit = NULL,
  crs = NULL,
  display_unit = NULL,
  unit_labels = NULL,
  width_hint = 0.25,
  unit_category = "metric",
  bar_cols = c("black", "white"),
  line_width = 1,
  height = grid::unit(0.25, "cm"),
  pad_x = grid::unit(0.25, "cm"),
  pad_y = grid::unit(0.25, "cm"),
  text_pad = grid::unit(0.15, "cm"),
  text_cex = 0.7,
  text_face = NULL,
  text_family = "",
  tick_height = 0.6,
  segments = NULL,
  label_show = "ends",
  minor_tick_height = 0.5,
  geographic_mode = c("approx_m", "degrees"),
  text_col = "black",
  line_col = "black"
)
```

Arguments

`mapping, data`	Standard ‘ggplot2‘ layer arguments (typically unused).
`...`	Additional parameters passed to the layer (rarely needed).
`location`	Character. One of ‘"bl"‘, ‘"br"‘, ‘"tr"‘, ‘"tl"‘. Placement relative to panel edges. Default: ‘"bl"‘.
`style`	Character. Scale bar style: ‘"segment"‘ (default), ‘"bar"‘, or ‘"ticks"‘.
`fixed_width`	Numeric. Force the bar width in *native CRS units* (e.g., meters). Overrides automatic width selection.
`crs_unit`	Character. Units of the CRS (e.g., ‘"m"‘, ‘"ft"‘, ‘"°"‘). Usually auto-detected; set only when auto-detection is not possible.
`crs`	[sf::st_crs] object or proj string. Fallback CRS if the plot does not provide one (e.g., when not using [ggplot2::coord_sf()]).
`display_unit`	Character. Force display units (e.g., ‘"m"‘, ‘"km"‘). Ignored when ‘geographic_mode = "degrees"‘.
`unit_labels`	Named character vector for i18n, e.g., ‘c(km = "Kilometers", m = "Meters", "°" = "°")‘.
`width_hint`	Numeric in (0, 1]. Target fraction of panel width used by the bar. Default: ‘0.25‘.
`unit_category`	Character: ‘"metric"‘ (default) or ‘"imperial"‘. Affects auto-promotion (m→km, ft→mi).

`bar_cols` Character(2). Colors for ‘"bar"‘ style alternating blocks. Default: ‘c("black", "white")‘.

`line_width` Numeric. Line thickness for outlines/ticks. Default: ‘1‘.

`height` [grid::unit]. Bar height. Default: ‘unit(0.25, "cm")‘.

`pad_x, pad_y` [grid::unit]. Padding from panel edges. Default: ‘unit(0.25, "cm")‘.

`text_pad` [grid::unit]. Gap between bar and labels. Default: ‘unit(0.15, "cm")‘.

`text_cex, text_face, text_family`
Font settings for labels. Defaults: ‘0.7‘, ‘NULL‘, ‘""‘.

`tick_height` Numeric in [0,1]. Relative height of interior ticks for ‘"ticks"‘ style. Default: ‘0.6‘.

`segments` Integer. For ‘"segment"‘ style, number of major divisions; if ‘NULL‘, an automatic, readable choice is used.

`label_show` Which ticks get labels: ‘"ends"‘ (default), ‘"all"‘, ‘"major"‘, numeric frequency (e.g., ‘2‘), or a numeric vector of indices (1-based).

`minor_tick_height`
Numeric in [0,1]. For ‘"segment"‘ style, minor ticks’ relative height. Default: ‘0‘.

`geographic_mode`
Character. For **geographic CRS** only: - ‘"approx_m"‘: approximate meters/kilometers (default; warns about approximation) - ‘"degrees"‘: display degrees directly (no metric conversion)

`text_col, line_col`
Colors for labels and outlines/ticks. Defaults: ‘"black"‘, ‘"black"‘.

Details

* If a **projected CRS** is in use (e.g., UTM/AEQD with meters), the scale bar is accurate in native units. * If a **geographic CRS** (EPSG:4326, degrees) is in use, distances vary with latitude. The ‘geographic_mode‘ parameter controls how to display the scale: - ‘"approx_m"‘ (default): approximate meters/kilometers using great-circle distance at the panel’s center latitude. A warning is issued. - ‘"degrees"‘: display raw degree units (e.g., ‘1°‘) without converting to meters. * You can also override the width with ‘fixed_width‘ (in native CRS units).

Value

A ‘ggplot2‘ layer object representing the scale bar.

Dependencies

Requires **ggplot2**, **sf**, and **grid**.

Examples

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

base_plot <- ggplot2::ggplot() +
  ggplot2::geom_sf(data = nc, fill = "grey90") +
```

```
  ggplot2::theme_minimal()

# Example 1: Projected CRS with a longer scale bar
base_plot + ggplot2::coord_sf(crs = 32617) +
  annotation_scalebar(location = "bl", width_hint = 0.5)

# Example 2: Ticks style, top-right (now correctly rendered)
base_plot + ggplot2::coord_sf(crs = 32617) +
  annotation_scalebar(location = "tr", style = "ticks")

# Example 3: Geographic CRS (EPSG:4326), approximate meters (warns)
base_plot + ggplot2::coord_sf(crs = 4326) +
  annotation_scalebar(location = "bl", geographic_mode = "approx_m")

# Example 4: Force a 100 km bar with red lines (now works correctly)
base_plot + ggplot2::coord_sf(crs = 32617) +
  annotation_scalebar(location = "bl", fixed_width = 100000, display_unit = "km",
                      line_col = "red")
```

---

basemap_dem *Elevation Map of China Layer for ggplot2*

Description

'basemap_dem' adds a digital elevation model (DEM) raster map of China as a layer to ggplot2. The function ensures the output map remains rectangular, regardless of the chosen projection. It supports displaying the DEM either within China's boundary or in a larger rectangular area around China. Users can provide their own DEM data using the 'data' parameter, or the default built-in DEM data will be used.

Usage

```
basemap_dem(
  data = NULL,
  crs = NULL,
  within_china = FALSE,
  maxcell = 1e+06,
  na.rm = FALSE,
  ...
)
```

Arguments

data	Optional. A 'terra' raster object for custom DEM data.
crs	Coordinate reference system (CRS) for the projection. Defaults to the CRS of the DEM data. Users can specify other CRS strings (e.g., '"EPSG:4326"' or custom projections).

within_china	Logical. If 'TRUE', displays only the DEM within China's boundary. If 'FALSE', displays the DEM for a larger rectangular area around China. Default is 'FALSE'.
maxcell	Maximum number of cells for rendering (to improve performance). Defaults to '1e6'.
na.rm	Logical. If 'TRUE', removes missing values. Default is 'FALSE'.
...	Additional parameters passed to 'geom_spatraster'.

Value

A 'ggplot' object containing the elevation map of China as a layer, which can be further customized or plotted.

See Also

geom_boundary_cn

Examples

```
# Before using the basemap_dem function, make sure the required data files are available.
# The required files are: "gebco_2024_China.tif" and "China_mask.gpkg".
# You can use check_geodata() to download them from GitHub if they are not available locally.

# Check and download the required data files if they are missing
check_geodata(files = c("gebco_2024_China.tif", "China_mask.gpkg"))

# Define the CRS for China (EPSG:4326 is a common global geographic coordinate system)
china_proj <- "+proj=aeqd +lat_0=35 +lon_0=105 +ellps=WGS84 +units=m +no_defs"

# Example 1: Display full rectangular area around China using built-in DEM data
ggplot() +
  basemap_dem(within_china = FALSE) +
  tidyterra::scale_fill_hypso_tint_c(
    palette = "gmt_globe",
    breaks = c(-10000, -5000, 0, 2000, 5000, 8000)
  ) +
  theme_minimal()

# Example 2: Display only China's DEM and boundaries using built-in DEM data
ggplot() +
  basemap_dem(crs = china_proj, within_china = TRUE) +
  geom_boundary_cn(crs = china_proj) +
  tidyterra::scale_fill_hypso_c(
    palette = "dem_print",
    breaks = c(0, 2000, 4000, 6000),
    limits = c(0, 7000)
  ) +
  labs(fill = "Elevation (m)") +
  theme_minimal()
```

basemap_vege *Vegetation Map of China Layer for ggplot2*

Description

Adds a vegetation raster map of China to a ggplot2 plot, with color-coded vegetation types.

Usage

```
basemap_vege(
  color_table = NULL,
  crs = NULL,
  maxcell = 1e+06,
  use_coltab = TRUE,
  na.rm = FALSE,
  ...
)
```

Arguments

color_table	A data frame containing vegetation types and their corresponding colors. It should have columns "code" (raster values), "type" (vegetation names), and "col" (hex color codes). If NULL, a default color table based on standard vegetation classifications for China is used.
crs	A character string specifying the coordinate reference system for the projection. If NULL, the default projection "+proj=aeqd +lat_0=35 +lon_0=105 +ellps=WGS84 +units=m +no_defs" is applied.
maxcell	An integer indicating the maximum number of cells for rendering to improve performance. Defaults to 1e6.
use_coltab	A logical value indicating whether to use the color table for raster values. Default is TRUE.
na.rm	A logical value indicating whether to remove missing values. Default is FALSE.
...	Additional parameters passed to ‘geom_spatraster‘.

Value

A ggplot2 layer object representing the vegetation map of China.

References

Zhang X, Sun S, Yong S, et al. (2007). *Vegetation map of the People’s Republic of China (1:1000000)*. Geology Publishing House, Beijing.

Examples

```
# Example1: Check and load the vegetation raster map

# Make sure the required raster data is available
check_geodata(files = c("vege_1km_projected.tif"))

# Once the data is checked or downloaded, add the vegetation raster to a ggplot
ggplot() +
  basemap_vege() +
  theme_minimal()

# Example2: Customize color table
custom_colors <- data.frame(
  code = 0:11,
  type = c(
    "Non-vegetated", "Needleleaf forest", "Needleleaf and broadleaf mixed forest",
    "Broadleaf forest", "Scrub", "Desert", "Steppe", "Grassland",
    "Meadow", "Swamp", "Alpine vegetation", "Cultivated vegetation"
  ),
  col = c(
    "#8D99B3", "#97B555", "#34BF36", "#9ACE30", "#2EC6C9", "#E5CE0E",
    "#5BB1ED", "#6494EF", "#7AB9CB", "#D97A80", "#B87701", "#FEB780"
  )
)
ggplot() +
  basemap_vege(color_table = custom_colors) +
  labs(fill = "Vegetation type group") +
  theme_minimal()
```

`check_geodata` *Check and Download Geospatial Data*

Description

Ensure required geodata files exist locally. The function searches and reuses existing files (when `overwrite = FALSE`) *before* attempting any network download, in the following order:

1. user-provided **local_dirs**
2. installed package **extdata** (even if not writable)
3. per-user cache **tools::R_user_dir("ggmapcn","data")**

If no valid local file is found (or `overwrite = TRUE`), the function downloads from mirrors *in order*. By default, a China-friendly CDN (jsDelivr) is tried first, then GitHub raw.

Robust features: multiple mirrors, atomic writes, resume, timeouts, retries, safe checksum checks, and correct `curl` progress callback.

Usage

```
check_geodata(
  files = NULL,
  overwrite = FALSE,
  quiet = FALSE,
  max_retries = 3,
  mirrors = NULL,
  use_checksum = TRUE,
  checksums = NULL,
  resume = TRUE,
  local_dirs = NULL
)
```

Arguments

files	Character vector of file names. If 'NULL', all known files are used.
overwrite	Logical. Force re-download even if a non-empty file exists. Default 'FALSE'.
quiet	Logical. Suppress progress and messages. Default 'FALSE'.
max_retries	Integer. Max retry attempts per (mirror, file). Default '3'.
mirrors	Character vector of base URLs (end with '/'). Tried in order. Default: jsDelivr first, then GitHub raw.
use_checksum	Logical. Verify SHA-256 when available. Default 'TRUE'.
checksums	Named character vector of SHA-256 digests (names = file names). If 'NULL', built-in defaults are used for known files; unknown files skip verification.
resume	Logical. Try HTTP range resume if a '.part' exists (only for writable dirs). Default 'TRUE'.
local_dirs	Character vector of directories to search *before* any download. If a matching non-empty file is found and `overwrite = FALSE`, it is returned immediately.

Value

Character vector of absolute file paths (NA for failures).

Examples

```
# Basic: ensure default files exist
check_geodata()

# Single file: reuse existing file if present (default overwrite = FALSE)
check_geodata(files = "boundary.rda")

# Force re-download a file (e.g., suspected corruption)
check_geodata(files = "boundary.rda", overwrite = TRUE)

# Search local folders first; skip download if a valid file is found there
check_geodata(
  files = c("boundary.rda", "world.rda"),
```

```
  local_dirs = c(getwd())  # add more directories if needed
)

# Provide your own mirror order (first tried wins)
check_geodata(
  files = "boundary.rda",
  mirrors = c(
    "https://cdn.jsdelivr.net/gh/Rimagination/ggmapcn-data@main/data/",
    "https://raw.githubusercontent.com/Rimagination/ggmapcn-data/main/data/"
  )
)
```

`coord_proj` *Coordinate System with Transformed Limits for Custom Projections*

Description

'coord_proj' is a wrapper around `ggplot2::coord_sf()`. It simplifies specifying map limits ('xlim', 'ylim') in longitude and latitude (WGS84 CRS) and automatically transforms them into the specified CRS for accurate projections.

This function extends the functionality of `coord_sf()` to seamlessly handle user-specified geographic boundaries in any projection, ensuring accurate mapping.

Usage

```
coord_proj(
  crs = NULL,
  xlim = NULL,
  ylim = NULL,
  expand = TRUE,
  default_crs = "EPSG:4326",
  ...
)
```

Arguments

`crs`	A character string specifying the coordinate reference system (CRS) for the projection (e.g., '"EPSG:4326"' or custom projections like '"+proj=merc"').
`xlim`	Longitude range (in degrees) to display, as a numeric vector of length 2.
`ylim`	Latitude range (in degrees) to display, as a numeric vector of length 2.
`expand`	Logical, whether to expand the plot limits. Default is 'TRUE'.
`default_crs`	A character string specifying the CRS of the input 'xlim' and 'ylim'. Default is '"EPSG:4326"'.
`...`	Additional arguments passed to `ggplot2::coord_sf()`.

Value

A ggplot2 `coord_sf` object with the transformed limits.

See Also

`ggplot2::coord_sf`, `geom_world`

Examples

```
# World map with default projection and limits
ggplot() +
  geom_world() +
  coord_proj(
    crs = "+proj=longlat +datum=WGS84",
    xlim = c(-180, 180),
    ylim = c(-90, 90),
    expand=FALSE
  ) +
  theme_minimal()

# Focused view with Azimuthal Equidistant projection
china_proj <- "+proj=aeqd +lat_0=35 +lon_0=105 +ellps=WGS84 +units=m +no_defs"
ggplot() +
  geom_world(fill = "lightblue") +
  coord_proj(
    crs = china_proj,
    xlim = c(60, 140),
    ylim = c(-10, 50)
  ) +
  theme_minimal()

# Display a small map of the South China Sea Islands with a custom projection
ggplot() +
  geom_boundary_cn() +
  theme_bw() +
  coord_proj(
    crs = china_proj,
    expand = FALSE,
    xlim = c(105, 123),
    ylim = c(2, 23)
  )
```

---

geom_boundary_cn *Plot Boundaries of China*

Description

Draw China's administrative boundaries and optional map decorations (compass and scale bar). Each boundary category (mainland, coastline, provinces, etc.) can be styled independently. The data are reprojected to the specified CRS before plotting.

Usage

```
geom_boundary_cn(
  crs = "+proj=aeqd +lat_0=35 +lon_0=105 +ellps=WGS84 +units=m +no_defs",
  compass = FALSE,
  scale = FALSE,
  mainland_color = "black",
  mainland_size = 0.5,
  mainland_linetype = "solid",
  coastline_color = "blue",
  coastline_size = 0.3,
  coastline_linetype = "solid",
  ten_segment_line_color = "black",
  ten_segment_line_size = 0.5,
  ten_segment_line_linetype = "solid",
  SAR_boundary_color = "grey40",
  SAR_boundary_size = 0.5,
  SAR_boundary_linetype = "dashed",
  undefined_boundary_color = "black",
  undefined_boundary_size = 0.5,
  undefined_boundary_linetype = "longdash",
  province_color = "transparent",
  province_size = 0.3,
  province_linetype = "solid",
  ...
)
```

Arguments

crs	Character or ‘sf::crs‘. Target coordinate reference system for plotting. Defaults to an azimuthal equidistant projection centered on China (‘+proj=aeqd +lat_0=35 +lon_0=105 +ellps=WGS84 +units=m +no_defs‘).
compass	Logical. If ‘TRUE‘, add a compass pointing to true north in the top-left corner. Default: ‘FALSE‘.
scale	Logical. If ‘TRUE‘, add a scale bar in the bottom-left corner. Default: ‘FALSE‘.
mainland_color	Character. Line color for the mainland boundary. Default: ‘"black"‘.
mainland_size	Numeric. Line width for the mainland boundary. Default: ‘0.5‘.
mainland_linetype	Character. Line type for the mainland boundary. Default: ‘"solid"‘.
coastline_color	Character. Line color for coastlines. Default: ‘"blue"‘.
coastline_size	Numeric. Line width for coastlines. Default: ‘0.3‘.
coastline_linetype	Character. Line type for coastlines. Default: ‘"solid"‘.
ten_segment_line_color	Character. Line color for the South China Sea ten-segment line. Default: ‘"black"‘.

`ten_segment_line_size`
: Numeric. Line width for the ten-segment line. Default: ‘0.5‘.

`ten_segment_line_linetype`
: Character. Line type for the ten-segment line. Default: ‘"solid"‘.

`SAR_boundary_color`
: Character. Line color for Hong Kong and Macau SAR boundaries. Default: ‘"grey40"‘.

`SAR_boundary_size`
: Numeric. Line width for SAR boundaries. Default: ‘0.5‘.

`SAR_boundary_linetype`
: Character. Line type for SAR boundaries. Default: ‘"dashed"‘.

`undefined_boundary_color`
: Character. Line color for undefined or disputed boundaries. Default: ‘"black"‘.

`undefined_boundary_size`
: Numeric. Line width for undefined boundaries. Default: ‘0.5‘.

`undefined_boundary_linetype`
: Character. Line type for undefined boundaries. Default: ‘"longdash"‘.

`province_color`
: Character. Line color for provincial boundaries. Default: ‘"transparent"‘.

`province_size`
: Numeric. Line width for provincial boundaries. Default: ‘0.3‘.

`province_linetype`
: Character. Line type for provincial boundaries. Default: ‘"solid"‘.

`...`
: Additional arguments passed to ‘ggplot2::geom_sf()‘ (e.g., ‘alpha‘).

Value

A list of ‘ggplot2‘ layers.

Examples

```
# Example 1: Basic China map
ggplot() +
  geom_boundary_cn() +
  theme_minimal()

# Example 2: Add compass and scale bar (easy mode)
ggplot() +
  geom_boundary_cn(compass = TRUE, scale = TRUE) +
  theme_minimal()

# Example 3: Custom styling
ggplot() +
  geom_boundary_cn(
    coastline_color = "steelblue",
    province_color = "grey70",
    province_linetype = "dashed"
  ) +
  theme_minimal()
```

```
# Example 4: Advanced usage with a custom projected CRS
Albers <-  "+proj=aea +lat_1=25 +lat_2=47 +lat_0=0 +lon_0=105 +x_0=0 +y_0=0 +
+ datum=WGS84 +units=m +no_defs"

ggplot() +
  geom_boundary_cn(crs = Albers) +
  annotation_compass(location = "tl", which_north = "true") +
  annotation_scalebar(location = "bl", fixed_width = 500000, display_unit = "km") +
  coord_sf(crs = Albers) +
  theme_minimal()
```

`geom_buffer_cn` *Plot Buffered Layers for China's Boundary*

Description

Creates a ggplot2 layer for displaying buffered areas around China's boundaries, including both the mainland boundary and the ten-segment line. Buffers with user-defined distances are generated around each boundary, providing flexibility in projection and appearance.

Usage

```
geom_buffer_cn(
  mainland_dist = 20000,
  ten_line_dist = NULL,
  crs = "+proj=aeqd +lat_0=35 +lon_0=105 +ellps=WGS84 +units=m +no_defs",
  color = NA,
  fill = "#D2D5EB",
  ...
)
```

Arguments

`mainland_dist`	Numeric. The buffer distance (in meters) for the mainland boundary.
`ten_line_dist`	Numeric. The buffer distance (in meters) for each segment of the ten-segment line. If not specified, it defaults to the same value as 'mainland_dist'.
`crs`	Character. The coordinate reference system (CRS) for the projection. Defaults to "+proj=aeqd +lat_0=35 +lon_0=105 +ellps=WGS84 +units=m +no_defs". Users can specify other CRS strings (e.g., "+proj=merc" for Mercator).
`color`	Character. The border color for the buffer area. Default is 'NA' (transparent).
`fill`	Character. The fill color for the buffer area. Default is '"#D2D5EB"'.
`...`	Additional parameters passed to 'geom_sf'.

Value

A ggplot2 layer displaying buffered areas around China's boundaries, with customizable buffer distances for the mainland boundary and the ten-segment line, using the specified projection.

Examples

```
# Plot buffers with specified distances for mainland and ten-segment line
ggplot() +
  geom_buffer_cn(
    mainland_dist = 10000,
    ten_line_dist = 5000
  ) +
  theme_minimal()
```

`geom_loc` *Visualize Spatial Point Data*

Description

'geom_loc' is a wrapper around `ggplot2::geom_sf()` designed for visualizing spatial point data. It supports both `sf` objects and tabular data frames with longitude and latitude columns, automatically transforming them into the specified coordinate reference system (CRS).

Usage

```
geom_loc(
  data,
  lon = NULL,
  lat = NULL,
  crs = "+proj=aeqd +lat_0=35 +lon_0=105 +ellps=WGS84 +units=m +no_defs",
  mapping = ggplot2::aes(),
  ...
)
```

Arguments

`data`	A data frame, tibble, or `sf` object containing spatial point data.
`lon`	A character string. The name of the longitude column in `data` (required if `data` is tabular).
`lat`	A character string. The name of the latitude column in `data` (required if `data` is tabular).
`crs`	A character string. The target coordinate reference system (CRS) for the data. Defaults to `"+proj=aeqd +lat_0=35 +lon_0=105 +ellps=WGS84 +units=m +no_defs"`.
`mapping`	Aesthetic mappings created by `ggplot2::aes()`, such as `color` or `size`.
`...`	Additional parameters passed to `ggplot2::geom_sf()`, such as `size`, `alpha`, or `color`.

Details

This function simplifies the process of visualizing spatial data in ggplot2 by automatically handling CRS transformations and providing an interface for both `sf` and tabular data. If the input is a tabular data frame, it will be converted to an `sf` object using the specified longitude and latitude columns.

See `ggplot2::geom_sf()` for details on additional parameters and aesthetics.

Value

A ggplot2 layer for visualizing spatial point data, either from an ‘sf‘ object or a tabular data frame with longitude and latitude columns, after transforming the data to the specified coordinate reference system (CRS).

See Also

`geom_boundary_cn`

Examples

```
# Generate a random dataset with latitude and longitude
set.seed(123)
data_sim <- data.frame(
  Longitude = runif(100, 80, 120),
  Latitude = runif(100, 28, 40),
  Category = sample(c("Type A", "Type B", "Type C"), 100, replace = TRUE)
)

# Visualize the data with China's boundaries
ggplot() +
  geom_boundary_cn() +
  geom_loc(
    data = data_sim, lon = "Longitude", lat = "Latitude",
    mapping = aes(color = Category), size = 1, alpha = 0.7
  ) +
  theme_minimal()
```

---

geom_mapcn *Plot China Map with Customizable Options*

Description

‘geom_mapcn()‘ plots China’s administrative boundaries with a simple, opinionated interface. It loads packaged map data when ‘data‘ is ‘NULL‘, removes the special row labeled ‘"Boundary Line"‘, supports optional attribute-based filtering, and can reproject to a user-specified CRS.

Usage

```
geom_mapcn(
  data = NULL,
  admin_level = "province",
  crs = "+proj=aeqd +lat_0=35 +lon_0=105 +ellps=WGS84 +units=m +no_defs",
  color = "black",
  fill = "white",
  linewidth = 0.5,
  filter_attribute = NULL,
  filter = NULL,
  mapping = NULL,
  ...
)
```

Arguments

data	An 'sf' object of geometries to draw. If 'NULL', the function loads the packaged dataset for the chosen 'admin_level'.
admin_level	Administrative level to plot. One of '"province"' (default), '"city"', or '"county"'. These correspond to packaged files 'China_sheng.rda', 'China_shi.rda', and 'China_xian.rda'.
crs	Coordinate Reference System to use for plotting. Defaults to an Azimuthal Equidistant projection centered on China: '"+proj=aeqd +lat_0=35 +lon_0=105 +ellps=WGS84 +units=m +no_defs"'. Accepts proj strings or EPSG codes (e.g., '"EPSG:4326"').
color	Border color. Default '"black"'.
fill	Fill color. Default '"white"'.
linewidth	Border line width. Default '0.5'. For older 'ggplot2' versions, use 'size' instead of 'linewidth'.
filter_attribute	Optional column name used to filter features (e.g., '"name_en"').
filter	Optional character vector of values to keep (e.g., 'c("Beijing","Shanghai")'). If supplied with 'filter_attribute', features are filtered accordingly. If the result is empty, an error is thrown.
mapping	Optional aesthetics mapping passed to 'geom_sf()'. Useful when you already have aesthetics to apply (e.g., fill).
...	Additional arguments forwarded to 'ggplot2::geom_sf()'.

Value

A 'ggplot2' layer.

Examples

```
# Basic provincial map
ggplot2::ggplot() +
```

```
  geom_mapcn() +
  ggplot2::theme_minimal()

# Filter by names stored in the data (e.g., English names)
ggplot2::ggplot() +
  geom_mapcn(filter_attribute = "name_en",
             filter = c("Beijing", "Shanghai"),
             fill = "red") +
  ggplot2::theme_minimal()

# Use a different projection
ggplot2::ggplot() +
  geom_mapcn(crs = "+proj=merc", linewidth = 0.7) +
  ggplot2::theme_minimal()
```

`geom_world` *Plot World Map with Customizable Options*

Description

A wrapper around [ggplot2::geom_sf()] for visualizing world maps with customizable options. This function allows for custom projections, filtering specific countries or regions, and detailed aesthetic customizations for borders and fills.

Usage

```
geom_world(
  data = NULL,
  crs = "+proj=longlat +datum=WGS84",
  color = "black",
  fill = "white",
  linewidth = 0.5,
  filter_attribute = "SOC",
  filter = NULL,
  ...
)
```

Arguments

`data`	An ‘sf‘ object containing world map data. If ‘NULL‘, the default world map data from the package will be loaded from a ‘.rda‘ file.
`crs`	A character string specifying the target coordinate reference system (CRS) for the map projection. Defaults to ‘"+proj=longlat +datum=WGS84"‘.
`color`	A character string specifying the border color for administrative boundaries. Default is ‘"black"‘.
`fill`	A character string specifying the fill color for administrative areas. Default is ‘"white"‘.

`linewidth` A numeric value specifying the line width for administrative boundaries. Default is '0.5'.

`filter_attribute` A character string specifying the column name used for filtering countries or regions. Default is '"SOC"', which refers to the ISO 3166-1 alpha-3 country code in the default dataset.

`filter` A character vector specifying the values to filter specific countries or regions. Default is 'NULL'.

`...` Additional parameters passed to [ggplot2::geom_sf()], such as 'size', 'alpha', or 'lty'.

Details

This function simplifies the process of creating world maps by combining the functionality of 'geom_sf' with user-friendly options for projections, filtering, and custom styling. Key features include: - **Custom projections**: Easily apply any CRS to the map. - **Filtering by attributes**: Quickly focus on specific countries or regions. - **Flexible aesthetics**: Customize fill, borders, transparency, and other visual properties.

Value

A 'ggplot2' layer for world map visualization.

See Also

[ggplot2::geom_sf()], [sf::st_transform()], [sf::st_read()]

Examples

```
# Plot the default world map
ggplot() +
  geom_world() +
  theme_minimal()

# Using Robinson projection with central meridian at 0°
ggplot() +
  geom_world(crs = "+proj=robin +lon_0=0 +x_0=0 +y_0=0 +datum=WGS84 +units=m") +
  theme_minimal()

# Filter specific countries (e.g., China and its neighbors)
china_neighbors <- c("CHN", "AFG", "BTN", "MMR", "LAO", "NPL", "PRK", "KOR",
                     "KAZ", "KGZ", "MNG", "IND", "BGD", "TJK", "PAK", "LKA", "VNM")
ggplot() +
  geom_world(filter = china_neighbors) +
  theme_minimal()

# Background map + Highlight specific region
ggplot() +
  geom_world(fill = "gray80", color = "gray50", alpha = 0.5) +
  geom_world(filter = c("CHN"), fill = "red", color = "black", linewidth = 1.5) +
  theme_minimal()
```

```
# Customize styles with transparency and bold borders
ggplot() +
  geom_world(fill = "lightblue", color = "darkblue", linewidth = 1, alpha = 0.8) +
  theme_void()
```

`north_arrow_classic` *Classic North Arrow Style (Minimal)*

Description

A collection of style constructors that return 'grid' grobs for use with 'annotation_compass(style = ...)'. These styles provide different visual appearances for a compass or north arrow drawn as an annotation.

Usage

```
north_arrow_classic(
  fill = c("white", "black"),
  line_col = "black",
  line_width = 2,
  text_col = "black",
  text_size = 12,
  text_face = "plain",
  text_family = ""
)

compass_sinan(
  line_col = "black",
  square_pad = 0.1,
  ring_outer = 0.35,
  ring_ratio = 0.65,
  labels = c("N", "E", "S", "W"),
  text_size = 12,
  text_face = "plain",
  text_family = "",
  text_col = "black",
  label_offset = 0.05,
  spoon_fill = "black",
  spoon_col = "black",
  spoon_scale = 0.8,
  inner_fill = "lightgrey",
  square_width = 2,
  outer_width = 2,
  inner_width = 1,
  spoon_width = 1
```

```
)

north_arrow_classic(
  fill = c("white", "black"),
  line_col = "black",
  line_width = 2,
  text_col = "black",
  text_size = 12,
  text_face = "plain",
  text_family = ""
)

north_arrow_solid(
  fill = "black",
  line_col = "black",
  line_width = 1,
  text_col = "black",
  text_size = 12,
  text_face = "plain",
  text_family = ""
)

compass_rose_simple(
  fill = c("white", "black"),
  line_col = "black",
  line_width = 1,
  sharpness = 0.7,
  text_col = "black",
  text_size = 12,
  text_face = "plain",
  text_family = ""
)

compass_rose_classic(
  fill = c("white", "black"),
  line_col = "black",
  line_width = 1.5,
  sharpness = 0.6,
  text_col = "black",
  text_size = 12,
  text_face = "plain",
  text_family = ""
)

compass_rose_circle(
  fill = "white",
  line_col = "black",
  line_width = 3,
```

```
  text_col = "black",
  text_size = 12,
  text_face = "plain",
  text_family = ""
)

compass_guiding_fish(
  size = 1,
  ring_ratio = 0.2,
  ring_width = 2,
  n_seg = 16,
  fish_col = "black",
  fish_shift = -0.03,
  text_col = "black",
  text_size = 12,
  text_face = "plain",
  text_family = ""
)

compass_sinan(
  line_col = "black",
  square_pad = 0.1,
  ring_outer = 0.35,
  ring_ratio = 0.65,
  labels = c("N", "E", "S", "W"),
  text_size = 12,
  text_face = "plain",
  text_family = "",
  text_col = "black",
  label_offset = 0.05,
  spoon_fill = "black",
  spoon_col = "black",
  spoon_scale = 0.8,
  inner_fill = "lightgrey",
  square_width = 2,
  outer_width = 2,
  inner_width = 1,
  spoon_width = 1
)
```

Arguments

`fill`	Fill color(s) for polygons. Vectorized for alternating fills in some styles.
`line_col`	Stroke color for outlines.
`line_width`	Stroke width for outlines (numeric).
`text_col`	Text color for labels.
`text_size`	Text font size for labels (points).

text_face	Text font face (e.g., "plain", "bold").
text_family	Text font family.
square_pad	Padding around the outer square (Sinan style), fraction of box side.
ring_outer	Outer ring radius (Sinan style), expressed in npc units (0..1).
ring_ratio	Inner/outer radius ratio for ringed styles (0 < value < 1).
labels	Character vector of cardinal labels, usually ‘c("N","E","S","W")‘.
label_offset	Label offset from the square edges (Sinan style), npc units.
spoon_fill	Fill color for spoon glyph (Sinan style).
spoon_col	Stroke color for spoon glyph (Sinan style).
spoon_scale	Scale factor for spoon glyph (Sinan style).
inner_fill	Fill color for inner disk (Sinan style).
square_width, outer_width, inner_width, spoon_width	Stroke widths for respective elements in Sinan style.
sharpness	Controls star-point sharpness in rose styles, numeric in [0, 1].
size	Global size scaler (used by some styles).
ring_width	Stroke width of ring outlines (numeric).
n_seg	Number of ring segments (integer).
fish_col	Fill color for fish shape (guiding fish style).
fish_shift	Vertical shift for fish shape (guiding fish style).

Details

Exported constructors documented under this topic:

- north_arrow_classic()
- north_arrow_solid()
- compass_rose_simple()
- compass_rose_classic()
- compass_rose_circle()
- compass_guiding_fish()
- compass_sinan()

Each constructor returns a grob ready to be passed to annotation_compass(style = ...). All styles include an "N" (or cardinal labels) to indicate north.

Value

A ‘grid‘ graphical object (grob).

See Also

[annotation_compass] for adding the compass to a ggplot.

Examples

```
# Standalone preview
grid::grid.newpage(); grid::grid.draw(north_arrow_classic())
grid::grid.newpage(); grid::grid.draw(north_arrow_solid())
grid::grid.newpage(); grid::grid.draw(compass_rose_simple())
grid::grid.newpage(); grid::grid.draw(compass_rose_classic())
grid::grid.newpage(); grid::grid.draw(compass_rose_circle())
grid::grid.newpage(); grid::grid.draw(compass_guiding_fish())
grid::grid.newpage(); grid::grid.draw(compass_sinan())

# Use in ggplot

if (requireNamespace("ggplot2", quietly = TRUE) &&
    requireNamespace("sf", quietly = TRUE)) {
  nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)
  p <- ggplot2::ggplot() +
    ggplot2::geom_sf(data = nc, fill = "grey90") +
    ggplot2::theme_minimal()

  p + annotation_compass(location = "tr", style = north_arrow_classic())
  p + annotation_compass(location = "bl", style = compass_sinan())
}
```

Index