Package 'ggmapcn'

November 2, 2026

Title Customizable China and Global Map Visualizations

Version 0.3.1

Description Provides 'ggplot2' layers and annotations for thematic mapping of China and the world. Uses administrative division data from the 'Tianditu' geospatial information service and separate layers for land borders, coastlines, and maritime boundaries. Supports province-, city-, and county-level maps, map projections, graticules, scale bars, north arrows, boundary buffers, and elevation and vegetation raster layers. The package provides default map datasets. Larger data files are hosted in a companion repository <https://github.com/Rimagination/ggmapcn-data> and retrieved as needed.

License GPL-3

Encoding UTF-8

Depends R (>= 4.3.0), ggplot2 (>= 3.5.0)

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 7.3.3

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 2026-11-02 07:50:02 UTC

Contents

ggmapcn-package . . . 2
annotation_compass . . . 3
annotation_graticule . . . 5
annotation_scalebar . . . 7
basemap_dem . . . 10
basemap_vege . . . 12
check_geodata . . . 13
coord_proj . . . 15
geom_boundary_cn . . . 17
geom_buffer_cn . . . 19
geom_loc . . . 21
geom_mapcn . . . 22
geom_world . . . 24
north_arrow_classic . . . 27

Index **33**

`ggmapcn-package` *Customizable China and Global Map Visualizations*

Description

Provides layers and annotations for drawing maps of China and the world with **ggplot2**. Supports map projections, administrative boundaries, graticules, scale bars, north arrows, buffer zones, and raster overlays. China administrative division data are derived from the *Tianditu* geospatial information service, with separate layers for land borders, coastlines, and maritime boundaries.

Map layers

- **World maps**: `geom_world()` draws a global basemap with countries, coastlines, boundaries, and optional ocean fill.
- **China maps**: `geom_mapcn()` and `geom_boundary_cn()` provide provincial, prefecture-level maps and coastlines.
- **Annotation tools**:
  - `annotation_graticule()` — global graticules with projection-aware labels.
  - `annotation_scalebar()` — scale bar with automatic units and CRS detection.
  - `annotation_compass()` — north arrow with several styles.
- **Projection helper**:
  - `coord_proj()` — specify geographic `xlim`/`ylim` in degrees and automatically transform to any projection.

Geospatial data

The package provides default boundary, world map, elevation, and vegetation datasets for direct use by its mapping functions. Some files are distributed in the installed package; larger files are hosted in the companion ggmapcn-data repository. `check_geodata()` reuses local copies or retrieves missing default files. It returns `NA` for files it cannot retrieve. Mapping functions require the corresponding datasets to be available before plotting.

Integration

Mapping and annotation functions return layers or lists of layers for **ggplot2**. `check_geodata()` returns file paths, and `coord_proj()` returns a coordinate system.

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 ggplot map. It can align to **grid north** (top of the panel) or **true north** (geographic north). Styles are provided as grobs or functions returning grobs (for example `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 layer arguments (typically unused).
...	Additional parameters passed to the layer (rarely needed).
location	Character; one of "tl", "tr", "bl", "br", indicating top/bottom and 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 supplied, it overrides "grid" / "true" behavior.
style	A grob, gList / gTree, or a function returning a grob (for example north_arrow_classic()). Default: north_arrow_classic().

Details

- "grid" north: the compass points straight up in plotting space (no CRS required).
- "true" north: the compass rotates toward the geographic North Pole using the plot CRS. This requires a valid CRS supplied by coord_sf() or injected via layer$geom_params$crs.
- A fixed rotation (degrees counter-clockwise) always overrides the automatic "grid" / "true" logic.
- The layer is annotation-like: it draws once per panel based on 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 <- ggplot() +
  geom_sf(data = nc, fill = "grey90") +
  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 +
  coord_sf(crs = "+proj=lcc +lon_0=-100 +lat_1=33 +lat_2=45") +
```

```
annotation_compass(location = "br", which_north = "true")
```

`annotation_graticule` *Global graticule annotation for ggplot2 maps*

Description

Draw global latitude-longitude graticules with degree labels as annotation layers for `ggplot2` maps. Graticules are constructed in geographic coordinates (EPSG:4326) over a user-defined window (given by `xlim`/`ylim`, default: the full globe), optionally split at the antimeridian according to the target CRS, and then transformed into the map CRS. Regional maps usually do not need this function and can rely on the default `coord_sf()` axes.

Usage

```
annotation_graticule(
  xlim = NULL,
  ylim = NULL,
  crs = "+proj=longlat +datum=WGS84",
  lon_step = 60,
  lat_step = 30,
  line_color = "grey70",
  line_width = 0.3,
  line_type = "dashed",
  label_color = "grey30",
  label_size = 3,
  label_offset = 5,
  label_offset_lon = NULL,
  label_offset_lat = NULL,
  sides = c("left", "bottom"),
  ...,
  line_alpha = 1,
  label_family = "",
  label_alpha = 1,
  label_angle = 0,
  label_format = NULL
)
```

Arguments

`xlim`	Numeric vector of length 2 giving the longitude range in degrees as `c(xmin, xmax)` in longitude-latitude (WGS84, EPSG:4326). Longitudes are interpreted in `[-180, 180]`. If both `xlim` and `ylim` are `NULL` (default), the full globe `(-180, 180)` is used.

`ylim` Numeric vector of length 2 giving the latitude range in degrees as `c(ymin, ymax)` in longitude-latitude (WGS84, EPSG:4326). Latitudes are interpreted in `[-90, 90]`. If both `xlim` and `ylim` are `NULL` (default), the full globe `(-90, 90)` is used.

`crs` Target coordinate reference system for the graticule, given as a PROJ string or `sf::crs` object. This should match the CRS used in your map layers and `coord_sf()`. The default is a WGS84 longitude- latitude definition.

`lon_step` Spacing in degrees between meridians. Default is `60`.

`lat_step` Spacing in degrees between parallels. Default is `30`.

`line_color` Line colour for graticule lines. Default is `"grey70"`.

`line_width` Line width for graticule lines. Default is `0.3`.

`line_type` Line type for graticule lines. Default is `"dashed"`.

`label_color` Text colour for labels. Default is `"grey30"`.

`label_size` Text size for labels in **millimeters (mm)**, passed to `ggplot2::geom_text()`. Default is `3`. **Note:** This unit differs from the point unit (pt) used in `theme()`. To match a specific font size in points (e.g., 8 pt), use `label_size = 8 / ggplot2::.pt`.

`label_offset` Common offset applied to all labels, in the units of the target CRS. For geographic CRSs (degrees), this is interpreted as degrees (default `5`). For projected CRSs (e.g. metres), you typically need a much larger value (e.g. `3e5` for Robinson or azimuthal projections).

`label_offset_lon` Optional offset applied only to longitude labels. If supplied, this overrides `label_offset` for longitude labels.

`label_offset_lat` Optional offset applied only to latitude labels. If supplied, this overrides `label_offset` for latitude labels.

`sides` Character vector indicating on which sides labels should be drawn. Any combination of `"bottom"`, `"top"`, `"left"`, `"right"`. Default is `c("left", "bottom")`.

`...` Additional arguments forwarded to `ggplot2::geom_sf()` for the graticule line layer.

`line_alpha` Line transparency, range `[0, 1]`. Default is `1` (opaque). If both `line_alpha` and `alpha` in `...` are supplied, `line_alpha` takes precedence.

`label_family` Font family for labels (e.g., "sans", "serif", "mono"). Default is `""` (system default).

`label_alpha` Text transparency, range `[0, 1]`. Default is `1`.

`label_angle` Text rotation angle in degrees. Default is `0`.

`label_format` Optional function to format the degree labels. If `NULL` (default), standard degree labels with N/S/E/W suffixes are used. If provided, it should be a function accepting a numeric vector and returning a character vector.

Details

Graticules are always generated in WGS84 longitude-latitude (EPSG:4326). When a non-zero central meridian (`lon_0`) is detected in the target CRS, meridians and parallels can be split at the antimeridian via `sf::st_break_antimeridian()` before being transformed, which avoids unexpected line wrapping in projections centred away from 0 degrees.

Latitude labels at +/-90 degrees are always omitted. When drawing a full-globe longitude-latitude map with a 0 degree central meridian (that is, when `xlim` and `ylim` are both `NULL` and the CRS is geographic with `lon_0 = 0`), longitude labels at +/-180 degrees are omitted (the corresponding graticule lines may still be drawn).

Value

A list of two `ggplot2` layers: a `geom_sf()` layer for graticule lines and a `geom_text()` layer for the labels.

Examples

```
library(ggplot2)

# Draw a graticule in geographic coordinates
ggplot() +
  annotation_graticule(
    lon_step     = 60,
    lat_step     = 30,
    label_offset = 5
  ) +
  coord_sf(crs = "+proj=longlat +datum=WGS84") +
  theme_void()
```

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

Description

`annotation_scalebar()` adds a projection-aware scale bar to a ggplot map. It reads the map's CRS (from `coord_sf()` or from the `crs` argument), chooses a readable width and units, and uses robust fallbacks so that the scale bar still draws even when CRS information is limited.

Supported styles:

- `"segment"` – minimal horizontal bar with ticks and labels (default)
- `"ticks"` – baseline with 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. Bar width in *native CRS units* (for example, meters). Overrides automatic width selection.
`crs_unit`	Character. CRS units (for example `"m"`, `"ft"`, `"°"`). Usually auto-detected; set only when auto-detection is not possible.
`crs`	An sf::st_crs object or a PROJ string. Fallback CRS when the plot does not provide one (for example, when not using `coord_sf()`).
`display_unit`	Character. Display units for labels (for example `"m"`, `"km"`). Ignored when `geographic_mode = "degrees"`.

`unit_labels` Named character vector for custom unit labels, e.g. `c(km = "Kilometers", m = "Meters", "°" = "°")`.

`width_hint` Numeric in (0, 1]. Target fraction of the panel width used by the bar. Default: `0.25`.

`unit_category` Character. `"metric"` (default) or `"imperial"`. Affects automatic promotion of units (m → km, ft → mi).

`bar_cols` Character vector of length two. Fill colours for `"bar"` style blocks. Default: `c("black", "white")`.

`line_width` Numeric. Line width for outlines and 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 the bar and text 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"`, a numeric frequency (for example `2`), or a numeric vector of 1-based indices.

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

`geographic_mode`
Character, for **geographic CRS** only:

- `"approx_m"`: approximate meters/kilometers (default; warns about approximation).
- `"degrees"`: show raw degrees (no metric conversion).

`text_col, line_col`
Colours for text labels and outlines/ticks. Defaults: `"black"`, `"black"`.

Details

- With a **projected CRS** (for example UTM or AEQD in meters), the scale bar is measured in native map units and is as accurate as the projection.
- With a **geographic CRS** (EPSG:4326, degrees), distance depends on latitude. The `geographic_mode` argument controls how degrees are handled:
  - `"approx_m"` (default): approximate meters/kilometers using a great-circle distance at the panel's mid-latitude (a warning is issued).
  - `"degrees"`: display raw degree units (for example `1°`) without converting.
- You can override the automatically chosen width with `fixed_width`, which is interpreted in native CRS units.

Value

A ggplot2 layer representing a scale bar.

Examples

```
nc <- sf::st_read(system.file("shape/nc.shp", package = "sf"), quiet = TRUE)[1:3, ]

base_plot <- ggplot() +
  geom_sf(data = nc, fill = "grey90") +
  theme_minimal()

# Projected CRS with a scale bar
base_plot +
  coord_sf(crs = 32617) +
  annotation_scalebar(location = "bl", width_hint = 0.5)
```

---

basemap_dem *Elevation Map of China Layer for ggplot2*

Description

`basemap_dem()` adds a digital elevation model (DEM) raster layer to a **ggplot2** plot. It supports displaying elevation within China's boundary or in a rectangular area around China. Users can supply a `terra` raster through `data`. Otherwise, `check_geodata()` locates or retrieves the default elevation dataset from the package's companion data repository.

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.

Details

The default elevation layer is the China subset of GEBCO_2024. The dataset is located or retrieved as gebco_2024_China.tif by check_geodata().

Value

A layer produced by tidyterra::geom_spatraster() that can be added to a **ggplot2** plot.

References

GEBCO Compilation Group (2024). GEBCO 2024 Grid. NERC EDS British Oceanographic Data Centre NOC. https://www.gebco.net/.

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 the rectangular area around China using the default 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 the default 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.

Details

The default dataset is a 1 km raster representation of *Vegetation Map of the People's Republic of China 1:1000000* (The Editorial Committee of Vegetation Map of China, Chinese Academy of Sciences, 2007). Its default colors follow Su et al. (2020). check_geodata() locates or retrieves vege_1km_projected.tif.

Value

A ggplot2 layer object representing the vegetation map of China.

References

The Editorial Committee of Vegetation Map of China, Chinese Academy of Sciences (2007). *Vegetation Map of the People's Republic of China 1:1000000*. Geological Publishing House, Beijing.

Su YJ, Guo QH, Hu TY, et al. (2020). An updated vegetation map of China (1:1000000). *Science Bulletin*, 65(13), 1125–1136.

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 retrieve required geodata files*

Description

Ensures that default geospatial data files provided by **ggmapcn** are available locally. Existing files are reused when `overwrite = FALSE`; missing files are downloaded from remote mirrors when possible. If all mirrors fail (for example, due to network restrictions), the function returns `NA` for the affected files.

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 processed.
`overwrite`	Logical; if `TRUE`, forces re-download even when a non-empty file already exists. The refreshed file is saved in the user cache, without changing the installed package.
`quiet`	Logical; if `TRUE`, suppresses progress output and messages.
`max_retries`	Integer; number of retry attempts per file and mirror.
`mirrors`	Character vector of base URLs ending with `/`. If `NULL`, package defaults are used.
`use_checksum`	Logical; if `TRUE`, verifies SHA-256 checksums when available.
`checksums`	Optional named character vector of SHA-256 digests. If `NULL`, defaults derived from `known_files()` are used.
`resume`	Logical; whether to attempt HTTP range resume for partially downloaded `.part` files.
`local_dirs`	Character vector of directories to search prior to any download attempt.

Details

Default datasets are provided through the installed package and its companion data repository. Larger files are hosted in that repository to keep the installation small. `check_geodata()` locates or retrieves these files using the following priority:

1. user-specified `local_dirs`
2. a refreshed copy in the per-user cache, if explicitly requested with `overwrite = TRUE` for the current bundled file

3. the package `extdata` directory
4. the per-user cache, followed by the legacy user data directory

Downloads are saved under `tools::R_user_dir("ggmapcn", "cache")`; the installed package directory is only read. A refreshed copy of a bundled file is used until the corresponding bundled file changes in a package update.

High-level mapping functions such as `geom_mapcn()` and `geom_world()` call `check_geodata()` internally, so most users do not need to invoke it directly. However, running it explicitly can be useful to pre-fetch or verify required files.

On networks that cannot reliably access `cdn.jsdelivr.net` or `raw.githubusercontent.com`, downloads may time out and the corresponding entries in the returned vector will be `NA`. In such cases, users may manually download the required files from the data repository and place them into a directory supplied through `local_dirs` or the user cache directory so that downloads are skipped.

Note: recent versions of `geom_world()` use the following world datasets: `world_countries.rda`, `world_coastlines.rda`, and `world_boundaries.rda`. The legacy `world.rda` file is no longer used.

Value

A character vector of absolute file paths. Any file that cannot be obtained is returned as `NA`.

Examples

```
# Retrieve only the datasets used by geom_world()
check_geodata(c(
  "world_countries.rda",
  "world_coastlines.rda",
  "world_boundaries.rda"
))

# China administrative boundaries
check_geodata(c("China_sheng.rda", "China_shi.rda", "China_xian.rda"))

# Reuse files manually placed in the working directory
check_geodata("world_countries.rda", local_dirs = getwd())
```

---

coord_proj *Coordinate System with Geographic Limits Automatically Transformed to a Projection*

Description

`coord_proj()` extends `ggplot2::coord_sf()` by allowing users to specify map limits (`xlim`, `ylim`) in geographic coordinates (longitude/latitude, WGS84). These limits are automatically transformed into the target projected CRS, ensuring that maps display the intended region correctly under any projection.

Usage

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

Arguments

crs	Character string or object specifying the output coordinate reference system (e.g., "EPSG:3857", "+proj=robin", or an sf::crs object). **Required**.
xlim	Numeric vector of length 2. Longitude limits in degrees (WGS84).
ylim	Numeric vector of length 2. Latitude limits in degrees (WGS84).
expand	Logical. Passed to ggplot2::coord_sf(). Default is TRUE.
default_crs	Character or object. The CRS of the input xlim and ylim. Default is "EPSG:4326" (WGS84).
...	Additional arguments passed to ggplot2::coord_sf().

Details

This wrapper is particularly useful because ggplot2::coord_sf() interprets xlim and ylim as *projected* coordinates (in the units of the target CRS). Passing longitude/latitude directly to coord_sf() results in incorrect map extents unless the output CRS is also WGS84.

coord_proj() provides a safe, projection-aware workflow that calculates the bounding box in WGS84, transforms it to the target CRS, and passes the new limits to coord_sf().

Value

A CoordSf object (specifically a result of coord_sf()) with automatically transformed limits.

See Also

- ggplot2::coord_sf() for the underlying function.
- geom_world() for the basemap layer.

Examples

```
library(ggplot2)


# Example 1: China (AEQD projection) with geographic limits
china_proj <- "+proj=aeqd +lat_0=35 +lon_0=105 +ellps=WGS84 +units=m +no_defs"

ggplot() +
```

```
  geom_world(crs = china_proj) +
  coord_proj(
    crs = china_proj,
    xlim = c(60, 140),
    ylim = c(-10, 50)
  ) +
  theme_minimal()

# Example 2: Zooming into a specific region
# Even though the map is projected (Robinson), we specify limits in Lat/Lon
crs_robin <- "+proj=robin +lon_0=0 +datum=WGS84"

ggplot() +
  geom_world(crs = crs_robin) +
  coord_proj(
    crs = crs_robin,
    xlim = c(-20, 50), # Focus on Africa/Europe
    ylim = c(-40, 40)
  ) +
  theme_minimal()
```

---

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 boundary 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 = 0.2,
  mainland_linetype = "solid",
  coastline_color = "blue",
  coastline_size = 0.1,
  coastline_linetype = "solid",
  ten_segment_line_color = "black",
  ten_segment_line_size = 0.2,
  ten_segment_line_linetype = "solid",
  SAR_boundary_color = "grey40",
  SAR_boundary_size = 0.1,
```

```
  SAR_boundary_linetype = "dashed",
  undefined_boundary_color = "black",
  undefined_boundary_size = 0.2,
  undefined_boundary_linetype = "dotdash",
  province_color = "transparent",
  province_size = 0.1,
  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: 0.2.
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: 0.1.
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: 0.2.
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: 0.1.
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: 0.2.
undefined_boundary_linetype	Character. Line type for undefined boundaries. Default: "dotdash".

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

province_size Numeric. Line width for provincial boundaries. Default: 0.1.

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. If the boundary dataset cannot be obtained, an empty list is returned.

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)
albers_cn <- "+proj=aea +lat_1=25 +lat_2=47 +lat_0=0 +lon_0=105 +datum=WGS84 +units=m +no_defs"

ggplot() +
  geom_boundary_cn(crs = albers_cn) +
  annotation_compass(location = "tl", which_north = "true") +
  annotation_scalebar(location = "bl", fixed_width = 500000, display_unit = "km") +
  coord_sf(crs = albers_cn) +
  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() draws China's administrative units with a simple, opinionated interface. When data is NULL, it loads packaged map data for the requested administrative level, optionally applies attribute-based filtering, removes internal clipping rows, and reprojects the layer to the target CRS.

In typical use, geom_mapcn() is combined with geom_boundary_cn() to draw coastlines, national borders, and other boundary features on top of the administrative polygons.

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 = 0.1,
  filter_attribute = NULL,
  filter = NULL,
  mapping = NULL,
  ...
)
```

Arguments

data	An sf object with geometries to draw. If NULL, the function loads the packaged dataset corresponding to admin_level.
admin_level	Administrative level to plot. One of "province" (default), "city", or "county". These map to packaged files China_sheng.rda, China_shi.rda, and China_xian.rda, respectively.
crs	Coordinate reference system used 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 for polygons. Default "black".
fill	Fill color for polygons. Default "white".
linewidth	Border line width. Default 0.1. For older ggplot2 versions, use size instead of linewidth.
filter_attribute	Optional name of an attribute column used to filter features (e.g., "name" for English provincial names).
filter	Optional character vector of values to keep in filter_attribute (e.g., c("Beijing", "Shanghai")). If supplied together with filter_attribute, features are subsetted accordingly. If no features remain after filtering, an error is thrown.
mapping	Optional aesthetics mapping passed to ggplot2::geom_sf(). This is useful when additional aesthetics (e.g., fill) should be mapped from columns in data.
...	Additional arguments forwarded to ggplot2::geom_sf().

Details

If data is NULL, geom_mapcn() selects one of the packaged datasets:

- admin_level = "province" → China_sheng.rda
- admin_level = "city" → China_shi.rda
- admin_level = "county" → China_xian.rda

The file is ensured to exist locally via check_geodata(), which may reuse an existing copy in the package extdata directory or user cache, or download it from the external repository when necessary. The .rda file is then loaded from the resolved path, and the main sf object is extracted.

A special row labelled "Boundary Line" (used for technical clipping) is removed automatically when present. Attribute-based filtering can be applied using filter_attribute and filter before reprojecting to the target CRS. In the current provincial dataset, name contains English names, name_cn contains Chinese names, and gb contains the source administrative codes. City- and county-level datasets may use different attribute names.

Value

A ggplot2 layer that can be added to a plot.

Examples

```
# Plot two provinces using the English name field
ggplot() +
  geom_mapcn(
    filter_attribute = "name",
    filter = c("Beijing", "Shanghai"),
    fill = "tomato"
  ) +
  theme_minimal()
```

---

geom_world *Convenient Global Basemap Layer for ggplot2*

Description

geom_world() draws a styled global basemap using country polygons, coastlines, and administrative boundary data. It automatically handles antimeridian splitting and CRS transformation, and supports optional country filtering for focused maps. The function uses the package's default world datasets; check_geodata() reuses local copies or retrieves missing files when possible.

Usage

```
geom_world(
  crs = 4326,
  filter_attribute = "SOC",
  filter = NULL,
  show_ocean = TRUE,
  show_admin_boundaries = TRUE,
  show_frame = FALSE,
  ocean_fill = "#c7e8fb",
  frame_color = "black",
  frame_size = 0.2,
  frame_linetype = "solid",
  country_fill = "grey90",
  country_boundary_color = "transparent",
  country_boundary_size = 0.1,
  country_boundary_linetype = "solid",
  coastline_color = "#26ace7",
  coastline_size = 0.1,
  coastline_linetype = "solid",
  international_boundary_color = "grey20",
  international_boundary_size = 0.1,
  international_boundary_linetype = "solid",
  regional_boundary_color = "grey20",
  regional_boundary_size = 0.1,
  regional_boundary_linetype = "dashed",
```

```
  undefined_boundary_color = "grey20",
  undefined_boundary_size = 0.1,
  undefined_boundary_linetype = "longdash",
  military_boundary_color = "grey20",
  military_boundary_size = 0.1,
  military_boundary_linetype = "dotted",
  ...
)
```

Arguments

`crs`	Coordinate reference system for the basemap. Accepts a numeric EPSG code, a PROJ string, or an `sf::st_crs()` object. The default is `4326` (WGS84).
`filter_attribute`	Name of the column in the `countries` dataset used for filtering. Default `"SOC"`.
`filter`	Character vector specifying which values of `filter_attribute` to retain. If `NULL` (default), no filtering is applied. When non-`NULL`, only the selected countries are drawn, and the ocean, coastlines, administrative boundaries, and frame are omitted.
`show_ocean`	Logical; draw an ocean background polygon. Default `TRUE`. Ignored when `filter` is not `NULL`.
`show_admin_boundaries`	Logical; draw administrative and political boundaries (international, regional, undefined/disputed, and military demarcation lines). Default `TRUE`. Ignored when `filter` is not `NULL`.
`show_frame`	Logical; draw an outer frame following the projected outline of the world. Default `FALSE`. Ignored when `filter` is not `NULL`.
`ocean_fill`	Fill color for the ocean polygon. Default `"#c7e8fb"`.
`frame_color`	Color of the outer frame line. Default `"grey20"`.
`frame_size`	Line width of the outer frame. Default `0.1`.
`frame_linetype`	Line type of the outer frame. Default `"solid"`.
`country_fill`	Fill color for country polygons. Default `"grey90"`.
`country_boundary_color`	Color of country boundary outlines. Default `"transparent"`.
`country_boundary_size`	Width of country boundary outlines. Default `0.1`.
`country_boundary_linetype`	Line type of country boundaries. Default `"solid"`.
`coastline_color`	Color of the coastline layer. Default `"#26ace7"`.
`coastline_size`	Line width of coastlines. Default `0.1`.
`coastline_linetype`	Line type of coastlines. Default `"solid"`.
`international_boundary_color`	Color for international boundary lines. Default `"grey20"`.

`international_boundary_size`
: Width for international boundaries. Default `0.1`.

`international_boundary_linetype`
: Line type for international boundaries. Default `"solid"`.

`regional_boundary_color`
: Color for regional boundaries (e.g. states). Default `"grey20"`.

`regional_boundary_size`
: Width for regional boundaries. Default `0.1`.

`regional_boundary_linetype`
: Line type for regional boundaries. Default `"dashed"`.

`undefined_boundary_color`
: Color for undefined or disputed boundaries. Default `"grey20"`.

`undefined_boundary_size`
: Width for undefined boundaries. Default `0.1`.

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

`military_boundary_color`
: Color for military demarcation lines. Default `"grey20"`.

`military_boundary_size`
: Width for military demarcation lines. Default `0.1`.

`military_boundary_linetype`
: Line type for military demarcation lines. Default `"dotted"`.

`...`
: Additional arguments passed to `ggplot2::geom_sf()` for the country polygons layer.

Details

This function supersedes early development versions that required users to supply their own map data.

The current implementation:

- Uses world map data located or retrieved by `check_geodata()`.
- Exposes dedicated arguments for ocean fill, coastlines, and administrative boundaries.
- Builds a projection-aware global outline for the ocean/frame layer. For **geographic CRSs** (including those with a shifted central meridian, e.g., `+lon_0=150`), it creates a seamless rectangular bounding box directly in the target CRS to avoid topological splitting artifacts (vertical lines). For **projected CRSs** (e.g., Robinson, Mollweide), it computes the convex hull of the projected graticule.

Value

A list of **ggplot2** layers representing the world map (or a filtered subset), ready to be added to a ggplot.

Examples

```
library(ggplot2)

# 1. Simple World Map (WGS84)
ggplot() +
  geom_world() +
  theme_void()

# 2. Pacific-Centered View (Shifted LongLat)
crs_longlat_150 <- "+proj=longlat +datum=WGS84 +lon_0=150"
ggplot() +
  geom_world(crs = crs_longlat_150, show_frame = TRUE, show_ocean = FALSE) +
  theme_void()

# 3. Robinson Projection (Projected CRS)
crs_robin <- "+proj=robin +lon_0=0 +datum=WGS84"
ggplot() +
  geom_world(crs = crs_robin, show_frame = TRUE) +
  theme_void()

# 4. Without administrative boundaries
ggplot() +
  geom_world(show_admin_boundaries = FALSE) +
  theme_minimal()

# 5. Highlighting specific countries (China)
ggplot() +
  geom_world(country_fill = "grey95") +
  geom_world(
    filter_attribute = "SOC",
    filter = "CHN",
    country_fill = "red",
    country_boundary_color = "black"
  ) +
  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 plot.

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

∗ **ggplot2.utils**
    geom_loc, 21
∗ **package**
    ggmapcn-package, 2

annotation_compass, 3
annotation_compass(), *31*
annotation_graticule, 5
annotation_scalebar, 7

basemap_dem, 10
basemap_vege, 12

check_geodata, 13
Compass styles, *4*
compass-styles *(*north_arrow_classic*)*, 27
compass_guiding_fish
        *(*north_arrow_classic*)*, 27
compass_rose_circle
        *(*north_arrow_classic*)*, 27
compass_rose_classic
        *(*north_arrow_classic*)*, 27
compass_rose_simple
        *(*north_arrow_classic*)*, 27
compass_sinan *(*north_arrow_classic*)*, 27
coord_proj, 15

geom_boundary_cn, *11*, 17, *22*
geom_buffer_cn, 19
geom_loc, 21
geom_mapcn, 22
geom_world, 24
geom_world(), *16*
ggmapcn *(*ggmapcn-package*)*, 2
ggmapcn-package, 2
ggplot2::aes(), *21*
ggplot2::coord_sf(), *15, 16*
ggplot2::geom_sf(), *21*, *26*
grid::unit, *9*

north_arrow_classic, 27
north_arrow_solid
        *(*north_arrow_classic*)*, 27

sf::st_crs, *8*
sf::st_crs(), *25*

tidyterra::geom_spatraster(), *11*