# Package 'ggautomap'

May 24, 2023

**Type** Package

**Title** Create Maps from a Column of Place Names

**Version** 0.3.2

**Description** Mapping tools that convert place names to coordinates on the fly. These 'ggplot2' extensions make maps from a data frame where one of the columns contains place names, without having to directly work with the underlying geospatial data and tools. The corresponding map data must be registered with 'cartographer' either by the user or by another package.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Depends** R (>= 4.1),

**Imports** cartographer (>= 0.2), cli (>= 3.4.0), dplyr (>= 1.0.0), ggmapinset (>= 0.3), ggplot2 (>= 3.4.2), packcircles (>= 0.3.4), rlang (>= 1.0.0), sf (>= 1.0), tidyr (>= 1.2.0), vctrs (>= 0.4.0)

**Collate** 'aaa.R' 'stat_automap.R' 'stat_automap_coords.R' 'choropleth.R' 'configure_inset.R' 'coord_automap.R' 'crs.R' 'geom_boundaries.R' 'geom_centroids.R' 'geoscatter.R' 'ggautomap-package.R' 'position_circle_repel.R'

**URL** https://github.com/cidm-ph/ggautomap, https://cidm-ph.github.io/ggautomap/

**BugReports** https://github.com/cidm-ph/ggautomap/issues

**Suggests** ggrepel, knitr, nswgeo, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Carl Suster [aut, cre] (<https://orcid.org/0000-0001-7021-9380>), Western Sydney Local Health District, NSW Health [cph]

**Maintainer** Carl Suster <Carl.Suster@health.nsw.gov.au>

**Repository** CRAN

**Date/Publication** 2023-05-24 09:00:02 UTC

# R topics documented:



---

`ggautomap-package` *Create Maps From a Column of Place Names*

---

## Description

Mapping geometries based on `{ggplot2}` for turning a spreadsheet into maps. This package provides ggplot geoms that make maps from a data frame where one of the columns contains place names. The map data must be registered with `{cartographer}`.

## Author(s)

**Maintainer**: Carl Suster <Carl.Suster@health.nsw.gov.au> (ORCID)

Other contributors:

- Western Sydney Local Health District, NSW Health [copyright holder]

## See Also

Useful links:

- https://github.com/cidm-ph/ggautomap
- https://cidm-ph.github.io/ggautomap/
- Report bugs at https://github.com/cidm-ph/ggautomap/issues

configure_inset *Configure transformations underpinning a map inset*

## Description

This specialises `ggmapinset::configure_inset()` to allow the `centre` to be specified as a location. The centroid of that location is used as the inset's centre.

## Usage

```
configure_inset(
  centre = NULL,
  scale = NULL,
  translation = NULL,
  radius = NULL,
  units = "km",
  feature_type = NA
)
```

## Arguments

| | |
|---|---|
| `centre` | Coordinates of the inset centre. Can instead be the name of a geographic feature if `feature_type` is also provided. |
| `scale` | Zoom scale: values larger than one will make the circle bigger. |
| `translation` | Translate (shift) the inset. This can be an `st_point` or simply a vector of length 2 containing the x and y offsets respectively. |
| `radius` | Radius of the inset circle. |
| `units` | Base length unit (e.g. `"km"` or `"mi"`). See `ggmapinset::configure_inset()` for supported values. |
| `feature_type` | Type of map feature. See `feature_types()` for a list of registered types. If `NA`, the type is guessed based on the values in `feature_names`. |

## Value

An inset configuration object.

## See Also

ggmapinset::configure_inset

## Examples

```
cfg <- configure_inset(
  centre = "Yancey",
  feature_type = "sf.nc",
  scale = 2,
```

```
  translation = c(70, -180),
  radius = 50,
  units = "mi"
)
```

---

coord_automap *Specify an inset configuration for the whole plot*

---

## Description

This allows a default inset configuration to be provided to avoid having to repeat it for each layer. Any layer that is inset-aware can use this as the default configuration if none is specifically provided to that layer. This coord also expands the axis limits to include the inset area.

## Usage

```
coord_automap(feature_type = NA, inset = NULL, ...)
```

## Arguments

| | |
|---|---|
| feature_type | Type of map feature. See feature_types() for a list of registered types. If NA, the type is guessed based on the values in feature_names. |
| inset | Inset configuration; see configure_inset(). |
| ... | Arguments passed to ggmapinset::coord_sf_inset() |

## Value

A ggplot coordinate

## Examples

```
library(ggplot2)
library(cartographer)

ggplot(nc_type_example_2, aes(location = county)) +
  geom_choropleth(aes(colour = type), size = 0.5) +
  geom_sf_label_inset(aes(label = county), stat = "automap_coords", size = 3) +
  coord_automap(feature_type = "sf.nc")
```

## crs_eqc *Coordinate reference system for spatial computations*

### Description

`crs_eqc()` gives a CRS that can be used for e.g. computing centroids or distances. It is an equidistant cylindrical system that by does not distort latitudes near `latitude`. The CRS is in units of kilometres by default.

### Usage

```
crs_eqc(latitude = 0, units = "km")
```

### Arguments

| | |
|---|---|
| `latitude` | The latitude of true scale (the `proj` parameter `lat_ts`). This is the latitude where the scale is not distorted by the projection. |
| `units` | Base length unit (e.g. `"km"` or `"mi"`). See `ggmapinset::configure_inset()` for supported values. |

### Value

CRS object from `sf`.

### Examples

```
# Sydney, Australia has a latitude of 33.87 S so this CRS will be suitable
# for computations close to there:
crs_eqc(latitude = -33.87)
```

## geom_boundaries *Map feature boundaries*

### Description

Retrieves the full map data from `{cartographer}` and plots the boundaries. As well as the chosen feature boundaries, the outline of the map is drawn separately if one has been registered with the map data, with the possibility to override its aesthetics.

## Usage

```
geom_boundaries(
  mapping = ggplot2::aes(),
  data = NULL,
  stat = "sf_inset",
  position = "identity",
  ...,
  feature_type = NULL,
  inset = NA,
  map_base = "normal",
  map_inset = "auto",
  na.rm = FALSE,
  outline.aes = list(colour = "#666666"),
  show.legend = NA,
  inherit.aes = FALSE
)
```

## Arguments

`mapping`, `stat`, `position`, `na.rm`, `show.legend`, `inherit.aes`, `...`
: See `ggplot2::geom_sf()`.

`data`
: Ignored (this geometry always uses the registered geographic data).

`feature_type`
: Type of map feature. See `feature_types()` for a list of registered types. If `NA`, the type is guessed based on the values in `feature_names`.

`inset`
: Inset configuration; see `configure_inset()`. If `NA` (the default), this is inherited from the coord (see `coord_sf_inset()`).

`map_base`
: Controls the layer with the base map. Possible values are `"normal"` to create a layer as though the inset were not specified, `"clip"` to create a layer with the inset viewport cut out, and `"none"` to prevent the insertion of a layer for the base map.

`map_inset`
: Controls the layer with the inset map. Possible values are `"auto"` to choose the behaviour based on whether `inset` is specified, `"normal"` to create a layer with the viewport cut out and transformed, and `"none"` to prevent the insertion of a layer for the viewport map.

`outline.aes`
: A list to override the aesthetics for the outline of the map. This has no effect if the map wasn't registered with a separate outline.

## Value

A ggplot layer.

## Examples

```
library(ggplot2)

ggplot() +
  geom_boundaries(feature_type = "sf.nc")
```

geom_centroids *Geographic centroid of locations*

## Description

Assigns each point a longitude and latitude corresponding to the geographic centre of its administrative area. This means that all points in the same area will overlap. The default `position` uses `position_circle_repel()` to repel the points outwards with an amount controllable with its `scale` parameter.

## Usage

```
geom_centroids(
  mapping = ggplot2::aes(),
  data = NULL,
  stat = "automap_coords",
  position = "circle_repel_sf",
  ...,
  fun.geometry = NULL,
  feature_type = NA,
  inset = NA,
  map_base = "clip",
  map_inset = "auto",
  na.rm = TRUE,
  show.legend = "point",
  inherit.aes = TRUE
)
```

## Arguments

`mapping, data, stat, position, na.rm, show.legend, inherit.aes, ...`
: See `ggplot2::stat_sf_coordinates()`.

`fun.geometry`
: A function that takes a `sfc` object and returns a `sfc_POINT` with the same length as the input. If `NULL`, `function(x) sf::st_point_on_surface(sf::st_zm(x))` will be used. Note that the function may warn about the incorrectness of the result if the data is not projected, but you can ignore this except when you really care about the exact locations.

`feature_type`
: Type of map feature. See `feature_types()` for a list of registered types. If `NA`, the type is guessed based on the values in `feature_names`.

`inset`
: Inset configuration; see `configure_inset()`. If `NA` (the default), this is inherited from the coord (see `coord_sf_inset()`).

`map_base`
: Controls the layer with the base map. Possible values are `"normal"` to create a layer as though the inset were not specified, `"clip"` to create a layer with the inset viewport cut out, and `"none"` to prevent the insertion of a layer for the base map.

`map_inset` Controls the layer with the inset map. Possible values are `"auto"` to choose the behaviour based on whether `inset` is specified, `"normal"` to create a layer with the viewport cut out and transformed, and `"none"` to prevent the insertion of a layer for the viewport map.

## Value

A ggplot layer.

## Aesthetics

The `location` aesthetic is required. `geom_centroids()` understands the same aesthetics as `ggplot2::geom_point()`.

## Examples

```
library(ggplot2)

cartographer::nc_type_example_2 |>
  head(n = 100) |>
  ggplot(aes(location = county)) +
  geom_boundaries(feature_type = "sf.nc") +
 geom_centroids(aes(colour = type), position = position_circle_repel_sf(scale = 6), size = 0.5) +
  coord_automap(feature_type = "sf.nc")
```

---

# geom_choropleth *Associate regions with counts*

---

## Description

Counts the number of occurrences of each location, then by default maps the count to the fill aesthetic. If your data has only one row per location and some other field that you'd like to map to aesthetics, use `geom_sf()` or `geom_sf_inset()` with `stat = "automap"` instead.

## Usage

```
geom_choropleth(
  mapping = ggplot2::aes(),
  data = NULL,
  stat = "choropleth",
  position = "identity",
  ...,
  feature_type = NA,
  inset = NA,
  map_base = "normal",
  map_inset = "auto",
  na.rm = TRUE,
  show.legend = NA,
  inherit.aes = TRUE
```

```
)

stat_choropleth(
  mapping = NULL,
  data = NULL,
  geom = "sf",
  position = "identity",
  ...,
  feature_type = NA,
  na.rm = TRUE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

## Arguments

| | |
|---|---|
| `mapping, data, stat, geom, position, na.rm, show.legend, inherit.aes, ...` | See `ggplot2::geom_sf()`. |
| `feature_type` | Type of map feature. See `feature_types()` for a list of registered types. If `NA`, the type is guessed based on the values in `feature_names`. |
| `inset` | Inset configuration; see `configure_inset()`. If `NA` (the default), this is inherited from the coord (see `coord_sf_inset()`). |
| `map_base` | Controls the layer with the base map. Possible values are `"normal"` to create a layer as though the inset were not specified, `"clip"` to create a layer with the inset viewport cut out, and `"none"` to prevent the insertion of a layer for the base map. |
| `map_inset` | Controls the layer with the inset map. Possible values are `"auto"` to choose the behaviour based on whether `inset` is specified, `"normal"` to create a layer with the viewport cut out and transformed, and `"none"` to prevent the insertion of a layer for the viewport map. |

## Details

Note that choropleths have a tendency to be misleading by emphasising geographically larger areas.

## Value

A ggplot layer.

## Aesthetics

The `location` aesthetic is required. `geom_choropleth()` understands the same aesthetics as `ggplot2::geom_sf()`.

## Computed variables

**count** rows matching the region

**geometry** `sf` geometry column

**...** limits as computed by `ggplot2::stat_sf()`

## Examples

```
library(ggplot2)

cartographer::nc_type_example_2 |>
  ggplot(aes(location = county)) +
  geom_choropleth() +
  geom_boundaries(feature_type = "sf.nc") +
  scale_fill_steps(low = "#e6f9ff", high = "#00394d") +
  coord_automap(feature_type = "sf.nc")
```

---

# geom_geoscatter *Place points randomly or in a grid within locations*

---

## Description

Each row of data is drawn as a single point inside the geographic area. This has similar strengths to a standard scatter plot, but has the potential to be misleading by implying that there is significance to the exact placement of the points.

## Usage

```
geom_geoscatter(
  mapping = ggplot2::aes(),
  data = NULL,
  stat = "geoscatter",
  position = "identity",
  ...,
  feature_type = NA,
  sample_type = "random",
  inset = NA,
  map_base = "clip",
  map_inset = "auto",
  na.rm = TRUE,
  show.legend = "point",
  inherit.aes = TRUE
)

stat_geoscatter(
  mapping = NULL,
  data = NULL,
  geom = "sf_inset",
  position = "identity",
  ...,
  feature_type = NA,
  sample_type = "random",
  show.legend = NA,
  inherit.aes = TRUE
)
```

## Arguments

`mapping, data, stat, geom, position, na.rm, show.legend, inherit.aes, ...`
: See `ggplot2::geom_sf()`.

`feature_type`
: Type of map feature. See `feature_types()` for a list of registered types. If `NA`, the type is guessed based on the values in `feature_names`.

`sample_type`
: sampling type (see the `type` argument of `sf::st_sample()`). `"random"` will place points randomly inside the boundaries, whereas `"regular"` and `"hexagonal"` will evenly space points, leaving a small margin close to the boundaries.

`inset`
: Inset configuration; see `configure_inset()`. If `NA` (the default), this is inherited from the coord (see `coord_sf_inset()`).

`map_base`
: Controls the layer with the base map. Possible values are `"normal"` to create a layer as though the inset were not specified, `"clip"` to create a layer with the inset viewport cut out, and `"none"` to prevent the insertion of a layer for the base map.

`map_inset`
: Controls the layer with the inset map. Possible values are `"auto"` to choose the behaviour based on whether `inset` is specified, `"normal"` to create a layer with the viewport cut out and transformed, and `"none"` to prevent the insertion of a layer for the viewport map.

## Value

A ggplot layer.

## Aesthetics

The `location` aesthetic is required. `geom_geoscatter()` understands the same aesthetics as `ggplot2::geom_point()`.

## Computed variables

**x** longitude

**y** latitude

## Examples

```
library(ggplot2)

cartographer::nc_type_example_2 |>
  ggplot(aes(location = county)) +
  geom_boundaries(feature_type = "sf.nc") +
  geom_geoscatter(aes(colour = type), size = 0.5) +
  coord_automap(feature_type = "sf.nc")
```

`position_circle_repel` *Pack overlapping points into a circle*

## Description

This position looks for any points with identical `x` and `y` positions and packs them in a circle around the original point. The `_sf` version applies the position adjustment in projected coordinates.

## Usage

```
position_circle_repel(scale = 1/4)

position_circle_repel_sf(scale = 10)
```

## Arguments

| | |
|---|---|
| `scale` | Scale of packing around the central point. This is in data units, so for the `_sf` variant it will depend on the units specified by the coordinate reference system. |

## Details

Note that extreme choices of `scale` may cause errors.

The `scale` parameter can instead be specified as an aesthetic for geoms that support it (`geom_centroids()`). This allows different locations to have different scales, which is especially useful when combined with map insets.

## Value

A ggplot position object.

## Examples

```
library(ggplot2)

points <- data.frame(
  x = c(rep(1, 10), 1:3),
  y = c(rep(2, 10), 3:5),
  s = 0.05
)
ggplot(points, aes(x, y)) +
  geom_point(size = 3, colour = "red") +
  geom_point(position = position_circle_repel(0.05), size = 3, alpha = 0.5)

cartographer::nc_type_example_2 |>
  dplyr::filter(!county %in% c("HENDERSON", "GASTON", "LINCOLN")) |>
  ggplot(aes(location = county)) +
  geom_boundaries(feature_type = "sf.nc") +
 geom_centroids(aes(colour = type), position = position_circle_repel_sf(scale = 4), size = 0.2) +
  coord_automap(feature_type = "sf.nc")
```

stat_automap *Attach spatial data with 'cartographer'*

## Description

Use cartographer to attach a spatial column to the data based on place names in another column. The result can then be used by ggplot2::geom_sf() or ggmapinset::geom_sf_inset().

## Usage

```
stat_automap(
  mapping = NULL,
  data = NULL,
  geom = "sf",
  position = "identity",
  ...,
  feature_type = NA,
  na.rm = TRUE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

## Arguments

| | |
|---|---|
| mapping, data, geom, position, na.rm, show.legend, inherit.aes, ... | See ggplot2::geom_sf(). |
| feature_type | Type of map feature. See feature_types() for a list of registered types. If NA, the type is guessed based on the values in feature_names. |

## Value

A ggplot layer

## Computed variables

**geometry** sf geometry column

**...** limits as computed by ggplot2::stat_sf()

## Examples

```
library(ggplot2)

events <- data.frame(
  county = c("Mecklenburg", "Carteret", "Moore", "Caldwell"),
  proportion_A = c(0.1, 0.8, 0.0, 0.6)
)

ggplot(events, aes(location = county)) +
```

```
  geom_sf(aes(fill = proportion_A), stat = "automap")

ggplot(events, aes(location = county)) +
  stat_automap(aes(fill = proportion_A)) +
  coord_automap(feature_type = "sf.nc")
```

---

`stat_automap_coords` *Attach coordinates with 'cartographer'*

---

## Description

Use `cartographer` to attach a spatial column to the data based on place names in another column. The spatial data is then reduced to coordinates in the same way as `stat_sf_coordinates()`.

## Usage

```
stat_automap_coords(
  mapping = NULL,
  data = NULL,
  geom = "sf_inset",
  position = "identity",
  ...,
  feature_type = NA,
  na.rm = TRUE,
  inset = NA,
  fun.geometry = NULL,
  show.legend = NA,
  inherit.aes = TRUE
)
```

## Arguments

| | |
|---|---|
| `mapping, data, geom, position, na.rm, show.legend, inherit.aes, ...` | See `ggplot2::stat_sf_coordinates()`. |
| `feature_type` | Type of map feature. See `feature_types()` for a list of registered types. If `NA`, the type is guessed based on the values in `feature_names`. |
| `inset` | Inset configuration; see `configure_inset()`. If `NA` (the default), this is inherited from the coord (see `coord_sf_inset()`). |
| `fun.geometry` | A function that takes a `sfc` object and returns a `sfc_POINT` with the same length as the input. If `NULL`, `function(x) sf::st_point_on_surface(sf::st_zm(x))` will be used. Note that the function may warn about the incorrectness of the result if the data is not projected, but you can ignore this except when you really care about the exact locations. |

## Value

A plot layer

## Computed variables

**geometry** `sf` geometry column representing the points

**x** X dimension of the simple feature

**y** Y dimension of the simple feature

**x_inset** X dimension of the simple feature after inset transformation

**y_inset** Y dimension of the simple feature after inset transformation

**inside_inset** logical indicating points inside the inset viewport

**inset_scale** 1 for points outside the inset, otherwise the configured inset scale parameter

## See Also

`ggmapinset::stat_sf_coordinates_inset()`

## Examples

```
library(ggplot2)

events <- data.frame(
  county = c("Mecklenburg", "Carteret", "Moore", "Caldwell"),
  proportion_A = c(0.1, 0.8, 0.0, 0.6)
)

ggplot(events, aes(location = county)) +
  geom_sf(aes(fill = proportion_A), stat = "automap") +
  geom_label(aes(label = county), stat = "automap_coords") +
  coord_automap(feature_type = "sf.nc")
```

# Index