Package: cartographr (via r-universe)

November 1, 2024

Title Crafting Print-Ready Maps and Layered Visualizations

Version 0.2.2

Description Simplifying the creation of print-ready maps, this package offers a user-friendly interface derived from 'ggplot2' for handling OpenStreetMap data. It streamlines the map-making process, allowing users to focus on the story their maps tell. Transforming raw geospatial data into informative visualizations is made easy with simple features 'sf' geometries. Whether for urban planning, environmental studies, or impactful public presentations, this tool facilitates straightforward and effective map creation. Enhance the dissemination of spatial information with high-quality, narrative-driven visualizations!

License GPL (>= 3)

URL https://da-wi.github.io/cartographr/, https://github.com/da-wi/cartographr

BugReports https://github.com/da-wi/cartographr/issues

Depends R (>= 3.6)

Imports cli, crayon, osmdata, ggplot2, grid, showtext, sysfonts, sf, utils

Suggests dplyr, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

LazyDataCompression xz

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Config/testthat/edition 3

Repository https://da-wi.r-universe.dev

RemoteUrl https://github.com/da-wi/cartographr

RemoteRef HEAD

RemoteSha ed2ca853efa6ba1ba947f01a8383ae0fab4d9912

Contents

`crime` *Manhattan crime dataset*

Description

This dataset encompasses all reported felony, misdemeanor, and violation crimes as recorded by the New York City Police Department (NYPD) for the borough of Manhattan, starting from the year 2004. The data is sourced from the public domain and is available for analysis and research purposes. It provides a comprehensive overview of crime patterns and can be utilized for developing crime prevention strategies, conducting sociological research, and enhancing public awareness.

Usage

```
crime
```

Format

A data frame with columns representing various attributes of crimes such as type, location, date, and time.

Source

New York City Police Department (NYPD) https://catalog.data.gov/

crop *Crop a preprocessed map*

Description

This function crops an OpenStreetMap (OSM) object that has been preprocessed. It supports different types of geometric boundaries such as rectangles, circles, and hexagons, or a custom boundary provided as an 'sf' object.

Usage

```
crop(osm, boundary = "rect")
```

Arguments

osm	A preprocessed OSM object to which the crop will be applied.
boundary	The type of geometric boundary to apply to the OSM data. Can be "rect" for a rectangular boundary, "circle" for a circular boundary, "hex" for a hexagonal boundary, or an 'sf' object for a custom boundary. Default is "rect".

Details

If the OSM object has not been preprocessed, the function will call `preprocess_map()` to preprocess the data before applying the cropping. The type of preprocessing applied is stored in the OSM object's metadata.

Value

The OSM object with the specified geometric crop applied.

Examples

```
data("osm")
# Apply a circular crop
osm_circle_cropped <- osm |> crop(boundary = "circle")
```

get_osmdata *Retrieve OpenStreetMap Data*

Description

This function retrieves OpenStreetMap (OSM) data based on geographic coordinates or a bounding box. It allows for the specification of distances and aspect ratios to define the area of interest. The function can return data in simple features (sf) format and has options to operate quietly.

Usage

```
get_osmdata(
  lat = NULL,
  lon = NULL,
  x_distance = NULL,
  y_distance = NULL,
  aspect_ratio = NULL,
  bbox = NULL,
  sf = NULL,
  quiet = TRUE,
  keep = TRUE
)
```

Arguments

lat	Latitude of the center point (optional if bbox or sf is provided).
lon	Longitude of the center point (optional if bbox or sf is provided).
x_distance	Distance in the x-direction from the center point (optional).
y_distance	Distance in the y-direction from the center point (optional).
aspect_ratio	Aspect ratio of the x and y distances (optional).
bbox	A bounding box to define the area of interest (optional if lat, lon, and distances are provided).
sf	An sf object to define the area of interest (optional if bbox is provided).
quiet	Logical flag to suppress progress messages.
keep	Logical flag if additional OSM data should be kept.

Details

The function performs checks to ensure the correct combination of parameters is provided. It calculates the bounding box if not provided and retrieves various OSM features within the specified area.

Value

A list containing various elements of the OSM data, including street networks, buildings, water bodies, green areas, beaches, parking areas, railways, and the bounding box of the retrieved area.

Exported Features

The following table lists the OSM features that are retrieved by the function:

Feature Type	OSM Tags
highway	motorway, motorway_link, trunk, trunk_link, primary, secondary, tertiary, unclassified, residential, living_street, street_lamp, pedestrian, track, path, steps
water	*
building	*

natural	beach, water, strait, bay, island, wood
amenity	parking
man_made	pier
railway	rail
place	sea, ocean
boundary	maritime
waterway	stream
landuse	forest, farmland, grass, orchard, allotments, recreation_ground, vineyard, cemetery, meadow
leisure	swimming_pool, pitch, nature_reserve, garden, park
natural	bay, island, wood

Note: * all tags are retrieved

Examples

```
osm_data <- get_osmdata(lat=44.568611, lon=15.331389, x_distance=100)
```

`get_palette` *Create a palette for maps*

Description

This function creates a color theme to be used with `plot_map()`. It can accept a predefined palette name or a custom palette provided as a named list.

Usage

```
get_palette(palette)
```

Arguments

`palette`	The color palette to use. Can be one of "alphabet", "arctic", "autumn", "autumn-muted", "bw", "desert", "evening", "gray", "iberia", "imhof", "lines", "metropolitan", "midnight", "minimal", "swiss", "tropical", or a named list for a custom palette. If a named list is provided, it should contain color hex codes for each map element. If `NULL` or an unrecognized name is provided, the function will throw an error.

Details

The color moods for the predefined palettes are described as follows:

- **Alphabet**: A modern palette with a straightforward aesthetic.
- **Arctic**: A palette that reflects the clear and bright qualities of Arctic landscapes.
- **Autumn**: A palette with the warm and varied hues typical of the fall season.

- **BW**: A classic black and white palette with a hint of warmth for a traditional feel.
- **Evening**: A palette that embodies the quiet and contemplative nature of dusk.
- **Gray**: A balanced palette that provides a composed and refined look.
- **Iberia**: A palette that reflects the warm and diverse tones associated with the Iberian landscape.
- **Imhof**: A palette with natural and subdued tones, inspired by the work of cartographer Eduard Imhof.
- **Lines (BW)**: A contrasting black and white palette for a clear and defined appearance.
- **Metropolitan**: A palette with understated tones that suggest urban sophistication.
- **Midnight**: A palette that conveys the depth and tranquility of the night.
- **Minimal**: A palette focused on minimalism, utilizing primarily whites and light grays.
- **Serene**: A palette that embodies peace and simplicity, utilizing a soft color scheme with gentle contrasts.
- **Swiss**: A palette that emphasizes cleanliness and precision, reminiscent of Swiss design.
- **Tropical**: A lively palette with the bright and bold colors characteristic of tropical areas.

In addition, you can customize other settings:

- `border_color`: The color of the borders, set to a dark shade "#121212".
- `border_width`: The width of the borders, set to a very fine line of 0.001 units.
- `linewidth_buildings`: The line width for building outlines, set to 0.05 units.
- `linewidth_motorway`, `linewidth_trunk`, `linewidth_primary`, `linewidth_secondary`, `linewidth_tertiary`, `linewidth_unclassified`, `linewidth_residential`: The line widths for various types of roads, ranging from 6 units for motorways to 1 unit for pedestrian paths.
- `size_streetlamp`: The size representation for streetlamps, set to 0.2 units.
- `hatch_*`: A series of settings for hatching patterns, which can be applied to water, buildings, and green spaces. These include toggles for hatching (`hatch_water`, `hatch_buildings`, `hatch_green`), the number of points or lines (`hatch_*_npoints`, `hatch_*_nlines`), the type of hatching pattern (`hatch_*_type`), the size of the hatching elements (`hatch_*_size`), and the transparency level (`hatch_*_alpha`).

These settings allow for a high degree of customization when creating maps, providing users with the ability to fine-tune the appearance of their map elements according to their specific needs or preferences.

Value

A list containing color settings for the map elements.

Examples

```
# Use predefined palette
get_palette("imhof")

# Custom palette creation using a named list for a simple black and white palette
custom_palette <- list(
```

```
  palette_building = c("#000000", "#FFFFFF", "#CCCCCC"),
  water = "#000000",
  green = "#FFFFFF",
  beach = "#000000",
  parking = "#FFFFFF",
  street = "#000000",
  background = "#CCCCCC",
  railway = "#000000",
  hatch_water = TRUE,
  linewidth_buildings = 0.05,
  linewidth_motorway = 6,
  linewidth_primary = 4,
  linewidth_secondary = 4,
  linewidth_tertiary=3,
  linewidth_unclassified = 3,
  linewidth_residential = 3,
  linewidth_pedestrian = 1,
  linewidth_service = 1,
  linewidth_living_street = 1,
  size_hatch = 1,
  alpha_hatch = 0.1,
  size_streetlamp = 0.2
)

get_palette(custom_palette)
```

`osm` *OSM SoHo New York simple features*

Description

This dataset contains the Simple Features (sf) representation for Soho, New York. It includes various geographical and spatial attributes relevant to the area.

Usage

```
osm
```

Format

An named list of objects of class `sf` with rows and columns corresponding to the features and their attributes.

Source

OpenStreetMap

Examples

```
data("osm")
```

plot_map *Plot a map with custom palette*

Description

This function takes an 'osmdata' (osm) object and a palette name, preprocesses the map data if not already done, and plots the map using 'ggplot2' with the specified color palette.

Usage

```
plot_map(...)
```

Arguments

`...`	Variable argument list: • `osm`: A list retrieved from osmdata containing map data. • `palette`: A character string specifying the name of the palette to use. The default is "imhof". Additional arguments are passed on to the preprocessing and plotting functions.

Value

A `ggplot` object representing the map with the chosen palette.

Examples

```
data("osm")
my_map <- osm |> plot_map(palette = 'gray')
```

preprocess_map *Preprocess OSM Data*

Description

This function preprocesses OpenStreetMap (OSM) data for further analysis and visualization. It filters and organizes data related to streets, railways, buildings, water bodies, green areas, beaches, and parking areas.

Usage

```
preprocess_map(osm)
```

Arguments

`osm`	A list containing OSM data elements.

Details

The function performs the following steps:

- Filters streets based on specified highway types.
- Filters railway lines.
- Filters building polygons and multipolygons.
- Filters water bodies and sea areas using multiple criteria.
- Filters green areas based on land use and natural features.
- Filters beach areas based on natural features.
- Filters parking areas based on amenities, highways, and man-made features.
- Combines multiple polygons into a single multipolygon for water, buildings, and green areas.

It returns the original OSM list with additional elements for each category of data and combined multipolygons for easy plotting.

Value

A list with preprocessed OSM data elements, including streets, railways, buildings, water bodies, green areas, beaches, and parking areas, each as separate list elements. Also includes a combined multipolygon for water, buildings, and green areas for plotting.

Examples

```
data("osm")
preprocessed_osm <- osm |> preprocess_map()
```

---

print_config *Print configuration details*

Description

This function prints the configuration details stored in the current environment. It iterates through the variables in the environment and displays their names and values.

Usage

```
print_config()
```

Value

This function does not return any value; it only prints the configuration details.

Examples

```
print_config()
```

save_map *Save a map to a file*

Description

This function saves a ggplot object to a file using the specified filename. It checks for the orientation setting and warns if the scale factor has changed after the plot was created.

Usage

```
save_map(plot, filename, device = "pdf")
```

Arguments

plot	A ggplot object representing the map to be saved.
filename	A character string specifying the path and name of the file to save the plot to.
device	The output device defaulting to pdf

Value

The function saves the plot to a file and does not return anything.

Examples

```
data("osm")
my_map <- osm |> plot_map()
filename <- tempfile(fileext =  ".pdf")
save_map(my_map, filename)
unlink(filename)
```

set_attribution *Set or get attribution setting*

Description

This function sets a new value for the attribution setting in the cartographr environment or retrieves the current setting if no argument is provided.

Usage

```
set_attribution(attribution = NULL)
```

Arguments

attribution	A logical value to set the acknowledgments setting. If NULL, the current setting is returned. The default is NULL.

Value

If `attribution` is `NULL`, returns the current acknowledgments setting. If `attribution` is a logical value, the function will set the acknowledgments setting to that value and return invisibly.

Examples

```
# To get the current acknowledgments setting
set_attribution()

# To set the acknowledgments setting to TRUE
set_attribution(TRUE)
```

`set_output_size` *Set output size for maps*

Description

This function sets the output size for cartographic displays. It allows the user to specify a standard paper size or custom dimensions. If no size is specified, it returns the current output size.

Usage

```
set_output_size(size = NULL, orientation = "portrait")
```

Arguments

`size`	A character string specifying the standard paper size or a numeric vector with custom dimensions (width, height). The standard sizes can be one of "A0", "A1", "A2", "A3", "A4", "A5", "A6", "small_poster", "medium_poster", or "large_poster". If size is NULL, the current output size is returned.
`orientation`	The orientation of the output

Value

If size is NULL, returns the current output size as a numeric vector. If a size is specified, the function sets the output size..

Examples

```
set_output_size("A3") # Sets the output size to A3 dimensions
set_output_size(c(300, 200)) # Sets a custom output size
```

soho_boundary *SoHo Boundary Simple Features Vector*

Description

This dataset represents the Simple Features vector for the MN24 region of New York, specifically covering the SoHo neighborhood. It includes spatial boundaries and other relevant geographical attributes.

Usage

```
soho_boundary
```

Format

An object of class `sf` (inherits from `data.frame`), representing the SoHo boundary with its spatial attributes.

Source

https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html

Examples

```
data(soho_boundary)
```

theme_infomap *Apply infomap theme with specified font*

Description

This function applies a custom theme for information maps, allowing the user to specify a font from a predefined list. It sets various ggplot2 theme elements such as the title, subtitle, caption, and legend to use the specified font and adjusts their appearance based on a scale factor.

Usage

```
theme_infomap(font = "Poppins")

theme_infomap_anton()

theme_infomap_poppins()

theme_infomap_cinzel()

theme_infomap_barlow()
```

Arguments

font	A character string specifying the font to use for the theme elements. The default is "Poppins". Only "Poppins", "Anton", "Cinzel", and "Barlow" are valid options.

Details

`theme_infomap_poppins()`, `theme_infomap_anton()` are aliases to `theme_infomap("Poppins")`, etc.

Value

A 'ggplot2' theme object with the information map theme settings applied.

Examples

```
data("osm")
my_map <- osm |> plot_map() +
 theme_infomap_poppins()
```

---

theme_poster *Create a poster theme*

Description

This function generates a 'ggplot2' theme that resembles a poster style. It is designed to be used with 'ggplot2' plots to provide a clean and bold aesthetic suitable for poster visuals.

Usage

```
theme_poster(font = "Poppins")

theme_poster_poppins()

theme_poster_anton()

theme_poster_cinzel()

theme_poster_barlow()
```

Arguments

font	The font family to be used for text elements in the plot. The default font is set to "Poppins".

Details

`theme_poster_poppins()`, `theme_poster_anton()` are aliases to `theme_poster("Poppins")`, etc.

Value

A 'ggplot2' theme object that can be added to 'ggplot2' plotting calls.

Examples

```
data("osm")
my_map <- osm |> plot_map() +
 theme_poster()
```

Index