Package ‘cartographer’

January 28, 2024

Title Turn Place Names into Map Data

Version 0.2.1

Description A tool for easily matching spatial data when you have a list of place/region names. You might have a data frame that came from a spreadsheet tracking some data by suburb or state. This package can convert it into a spatial data frame ready for plotting. The actual map data is provided by other packages (or your own code).

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

Imports cli (>= 3.6.0), dplyr (>= 1.1.0), rlang (>= 1.1.0), sf (>= 1.0.12)


Additional_repositories http://packages.ropensci.org

Depends R (>= 4.1)

LazyData true

Suggests ggplot2 (>= 3.4.2), knitr, maturaleza, maturaleza_hires, rmarkdown

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


VignetteBuilder knitr

NeedsCompilation yes

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 2024-01-28 11:40:02 UTC
Description

Cartographer is a framework for easily matching spatial data when you have a list of standardised place names. You might have a data frame that came from a spreadsheet tracking some data by suburb or state. This package can convert it into a spatial data frame ready for plotting. The actual map data is provided by other packages (or your own code) that register the data 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://cidm-ph.github.io/cartographer/
### add_geometry

**Convert input data frame into a spatial data frame**

**Description**

Convert input data frame into a spatial data frame.

**Usage**

```r
add_geometry(x, location, feature_type = NA, geom_name = "geometry")
```

**Arguments**

- `x`: Data frame with a feature name column.
- `location`: Feature names (tidy evaluation).
- `feature_type`: The registered map corresponding to values in `location`. If NA (the default), the type is guessed from the values in `location`.
- `geom_name`: Name for the new column to contain the geometry.

**Value**

A spatial data frame containing all of the columns from the input data frame.

**Examples**

```r
add_geometry(nc_type_example_2, county, feature_type = "sf.nc")
```

### feature_names

**List known feature names**

**Description**

This gives the list of feature names that are part of the specified map data. The list includes any aliases defined when the map was registered. Note that the location column matching is case insensitive (see Details below).

**Usage**

```r
feature_names(feature_type)
```

**Arguments**

- `feature_type`: Type of map feature. See `feature_types()` for a list of registered types.
Value

Character vector of feature names.

See Also

register_map() and resolve_feature_names()

Examples

head(feature_names("sf.nc"))

---

feature_types

List known feature types

Description

Each feature type corresponds to map data that has been registered.

Usage

feature_types()

Value

Character vector of registered feature types.

See Also

register_map()

Examples

feature_types()
map_outline

Retrieve a map outline registered with cartographer.

Description

Retrieve a map outline registered with cartographer.

Usage

map_outline(feature_type)

Arguments

feature_type Type of map feature. See feature_types() for a list of registered types.

Value

The map outline that was registered under feature_type. Note that the outline is optional, so this will return NULL if none was registered.

Examples

map_outline("sf.nc")

map_sf

Retrieve map data registered with cartographer.

Description

Retrieve map data registered with cartographer.

Usage

map_sf(feature_type)

Arguments

feature_type Type of map feature. See feature_types() for a list of registered types.

Value

The spatial data frame that was registered under feature_type.

Examples

map_sf("sf.nc")
map_sfc

Retrieve geometry of a single location.

Description

Retrieve geometry of a single location.

Usage

map_sfc(feature_names, feature_type)

Arguments

feature_names  Name of the feature(s) to retrieve. This must be an exact case-sensitive match, and aliases are not consulted.
feature_type  Type of map feature. See feature_types() for a list of registered types.

Value

The geometry as a sfc object.

Examples

map_sfc("Ashe", "sf.nc")
map_sfc(c("Craven", "Buncombe"), "sf.nc")

nc_type_example_1

Example datasets with a feature name column and random data

Description

This dataset contains random data compatible with the sf.nc example map data for illustrating cartographer’s features. nc_type_example_1 contains a deliberate error in the county name for a single row, whereas nc_type_example_2 contains correct data.

Usage

nc_type_example_1
nc_type_example_2

Format

Objects of class data.frame with 50 and 200 rows respectively, and 2 columns:

county  Feature names that match the NAME field of the nc dataset
type  Arbitrary categorical data
Description

This adds a new feature type that can then be used by all the geoms in this package. If registering from another package, this should occur in the `.onLoad()` hook in the package.

Usage

```r
register_map(
  feature_type,
  data,
  feature_column,
  aliases = NULL,
  outline = NULL,
  lazy = TRUE
)
```

Arguments

- `feature_type` Name of the type. If registering from within a package, the suggested format is "<package name>.<map name>" to avoid clashes between packages.
- `data` A simple feature data frame with the map data, or a function that returns a data frame. When `lazy` is `TRUE`, the value will not be evaluated until the data is first accessed.
- `feature_column` Name of the column of `data` that contains the feature names.
- `aliases` Optional named character vector or list that maps aliases to values that appear in the feature column. This allows abbreviations or alternative names to be supported.
- `outline` Optional `sf` geometry containing just the outline of the map, or a function returning such a geometry. When `lazy` is `TRUE`, the value will not be evaluated until the data is first accessed.
- `lazy` When `TRUE`, defer evaluation of `data` and `outline` until it is used.

Details

Registration supports delayed evaluation (lazy loading). This is particularly useful for larger datasets, so that they are not loaded into memory until they are accessed.

Value

No return value; this updates the global feature registry.

See Also

`vignette("registering_maps")`
Examples

```r
# register a map of the states of Italy from rnaturalearth using the
# Italian names, and providing an outline of the country
register_map(
  "italy",
  data = rnaturalearth::ne_states(country = "italy", returnclass = "sf"),
  feature_column = "name_it",
  outline = rnaturalearth::ne_countries(country = "italy", returnclass = "sf", scale = "large")
)
```

`resolve_feature_names`  
Canonicalise feature names accounting for aliases and character case

Description

Names are resolved by checking for the first match using:

1. case sensitive match, then
2. case sensitive match using aliases, then
3. case insensitive match, then
4. case insensitive match using aliases.

Usage

```r
resolve_feature_names(feature_names, feature_type, unmatched = "error")
```

Arguments

- `feature_names`: Character vector of feature names in the data.
- `feature_type`: Type of map feature. See `feature_types()` for a list of registered types.
- `unmatched`: Controls behaviour when `feature_names` contains values that do not match registered feature names. Possible values are "error" to throw an error or "pass" to return the original values unaltered.

Value

Character vector of the canonicalised names.

Examples

```r
resolve_feature_names(c("LEE", "ansoN"), feature_type = "sf.nc")
resolve_feature_names(c("LEE", "ansoNe"), feature_type = "sf.nc", unmatched = "pass")
```


**Description**

If `feature_type` is provided, this simply checks that the type has been registered. If it is `NA`, however, an attempt is made to guess the appropriate choice. This is done by comparing the example values provided as `feature_names` with the names of all registered map datasets. If there is an unambiguous match, that will be filled in.

**Usage**

```r
resolve.feature.type(feature_type, feature_names)
```

**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`.
- `feature_names`: Character vector of feature names in the data. This can be a subset of the values.

**Details**

Cartographer lazy loads map data for registered maps. In order to compare the example `feature_names` with registered maps, it might be necessary to force some of these datasets to load. To minimise the impact, any maps that have already been loaded are checked first. Other maps are then loaded one at a time until any matches are found. Consequently, the result returned by this function is not deterministic.

In the worst case where none of the registered maps matches, all of them will be loaded. This might take several seconds and occupy some memory, depending on which maps are registered. If a match is found, however, it will be found quickly on subsequent calls since the data will have already been loaded.

The best way to avoid these issues is to explicitly specify the `feature_type`.

**Value**

The resolved feature type as a scalar character.

**Examples**

```r
resolve.feature.type("sf.nc")
resolve.feature.type(NA, feature_names = c("ANSON", "Stanly"))
```
Index

* datasets
  nc_type_example_1, 6
add_geometry, 3
cartographer (cartographer-package), 2
cartographer-package, 2
feature_names, 3
feature_types, 4
feature_types(), 3, 5, 6, 8, 9
map_outline, 5
map_sf, 5
map_sfc, 6
nc_type_example_1, 6
nc_type_example_2 (nc_type_example_1), 6
register_map, 7
register_map(), 4
resolve_feature_names, 8
resolve_feature_names(), 4
resolve_feature_type, 9