Package: aussiemaps (via r-universe)

October 8, 2024

Title Maps of Australia

Version 0.2.2.0002

Description Granular maps of Australia using ABS boundaries from 2006-2021.

URL https://carlosyanez.github.io/aussiemaps/

BugReports https://github.com/carlosyanez/aussiemaps/issues

License CC BY-SA 4.0

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

StagedInstall FALSE

Imports arrow, dplyr, digest, fs, lubridate, lwgeom, nngeo, piggyback, rlang, rmapshaper, progressr, sf, smoothr, stringr, tibble, tidyr, tidyselect, units, utils, zip, lifecycle

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Repository https://carlosyanez.r-universe.dev

RemoteUrl https://github.com/carlosyanez/aussiemaps

RemoteRef HEAD

RemoteSha 31952ede8b3f37fea808ed9a0f951bbc5f2ba0ba

Contents

data_maps_delete 2
data_maps_info 2
find_maps_cache 3
flexible_left_join 3

geo_aggregate 4
get_cache_name 5
get_map 5
list_attributes 7
list_proportions 8
list_structure 8

Index **10**

`data_maps_delete` *Helper function to delete data*

Description

Helper function to delete data

Usage

```
data_maps_delete(...)
```

Arguments

...	fs::dir_info() parameters

Value

nothing

`data_maps_info` *Helper function to update/download data*

Description

Helper function to update/download data

Usage

```
data_maps_info(...)
```

Arguments

...	fs::dir_info() parameters

Value

nothing

find_maps_cache *Helper function to find cache folder*

Description

Helper function to find cache folder

Usage

```
find_maps_cache()
```

Value

nothing

flexible_left_join *'Flexible left join, accounting for different formatting in different datasets*

Description

This function's preforms for a left_join() between to datasets, accomdating for small differences in formatting across the key, for example a O'Connor vs. Oconnor (as in the Australian electoral division)

Usage

```
flexible_left_join(df1, df2, by)
```

Arguments

df1	first dataset (of sf data frame)
df2	second dataset
by	named vector with join key, as in left_join

Value

A data frame containing the aggregated data

`geo_aggregate` *Aggregate data to a new geography*

Description

Convert data, aggregating smaller geographic structures into larger ones. By default it uses area to apportion values when there is no one-to-one correspondence. Weighting table can be provided

Usage

```
geo_aggregate(
  original_data,
  values_col,
  original_geo,
  new_geo,
  grouping_col = NULL,
  year,
  proportions_manual = NULL
)
```

Arguments

`original_data`	A data frame of original data
`values_col`	The name of the column containing the values to be aggregated
`original_geo`	The name of the column containing the original geography
`new_geo`	The name of the column containing the new geography
`grouping_col`	The name of the column containing the grouping variables
`year`	The year of the data to be aggregated
`proportions_manual`	A data frame of manual proportions,it will override in-package proportions based on area.

Value

A data frame containing the aggregated data

get_cache_name *Get name for a cached map*

Description

Get name for a cached map

Usage

```
get_cache_name(
  year,
  simplification_factor,
  smoothing_threshold,
  new_crs,
  filter_table,
  aggregation
)
```

Arguments

year	year
simplification_factor	simplification_factor
smoothing_threshold	smoothing_threshold
new_crs	new_crs
filter_table	filter_table
aggregation	aggregation

Value

cache path

get_map *Get a map sf tibble*

Description

This function tibble with sf objects, for a particular year. It allows to filter the results using a geo structure names / codes, and results can be aggregated by those too. Optionally, this function stores the results in the cache for faster retrieval of large objects (e.g. when covering) a metropolitan area.

Usage

```
get_map(
  filter_table = NULL,
  filters = NULL,
  year,
  aggregation = NULL,
  simplification_factor = NULL,
  new_crs = NULL,
  fill_holes = TRUE,
  smoothing_threshold = 4,
  use_cache = FALSE,
  cache_file = NULL,
  cache_intermediates = TRUE,
  interstate_merge = TRUE,
  message_string = ""
)
```

Arguments

filter_table	A data frame containing the filter table, usually the output of list_structure().
filters	A list of filters to be used. Item names should name column names in list_structure(). Contents should be vectors with regular expressions.
year	A number indicating the year for which the map should be created.
aggregation	A vector containing the aggregation parameters,matching list_structure() column names .
simplification_factor	A number indicating the simplification factor.
new_crs	CRS value if transformation is needed.
fill_holes	whether to fill holes after merging parts
smoothing_threshold	A number indicating the smoothing threshold.
use_cache	A boolean indicating whether to use the cache.
cache_file	Optional a string indicating the friendly name of the cache file (f not provided, an arbitrary name will be created).
cache_intermediates	whether to cache state intermediate
interstate_merge	whether to consolidate object across state/territory lines (e.g. CED for external territories)
message_string	extra message string to add to any message (for tracking)

Value

A map object.

See Also

list_structure

Examples

```
## Not run:
 small case
 preston <- get_map(filters=list(SSC_NAME_2016=c("Preston")),
                     year=2016,
                    aggregation = c("SSC_NAME_2016"))
 big map of Sydney,cached
 sydney_area <- get_map(filter_table = greater_sydney,
                        year=2021,
                        aggregation = "GCCSA_NAME_2021",
                        use_cache = TRUE)


## End(Not run)
```

---

list_attributes *List geographic structures (or attributes)*

Description

This function finds all the available geographic structures dataframe with the structures for each year.

Usage

```
list_attributes()
```

Details

Get list of all geographic structures (or attributes)

Value

A tibble with the attributes for each year.

Examples

```
## Not run:
list_attributes()

## End(Not run)
```

list_proportions *Get list elements to attribute mapping, with area proportion*

Description

Get list elements to attribute mapping, with area proportion

Usage

```
list_proportions(attribute_name, ids = NULL)
```

Arguments

attribute_name	attribute name
ids	ids

Value

tibble with structure

list_structure *List Structure*

Description

produce table with geo structure

Usage

```
list_structure(year, filters = NULL)
```

Arguments

year	A character string of the year for which the structure is requested.
filters	A list with the attributes and values of the filters to be applied (e.g. list("CED_NAME_2021"=c("Wills","N

Value

A data frame with the structure requested.

See Also

get_map

Examples

```
## Not run:
list_structure("2021")
list_structure("2021", list("CED_NAME_2021"=c("Wills","Melbourne"))

## End(Not run)
```

Index

∗ **helpers**
    data_maps_delete, 2
    data_maps_info, 2
    find_maps_cache, 3
    get_cache_name, 5
∗ **lists**
    list_attributes, 7

data_maps_delete, 2
data_maps_info, 2

find_maps_cache, 3
flexible_left_join, 3

geo_aggregate, 4
get_cache_name, 5
get_map, 5, *8*

list_attributes, 7
list_proportions, 8
list_structure, *7*, 8