Package 'giscoR'

May 29, 2024

Type Package

Title Download Map Data from GISCO API - Eurostat

Version 0.5.0

Description Tools to download data from the GISCO (Geographic Information System of the Commission) Eurostat database <https://ec.europa.eu/eurostat/web/gisco>. Global and European map data available. This package is in no way officially related to or endorsed by Eurostat.

License GPL-3

URL https://ropengov.github.io/giscoR/, https://github.com/rOpenGov/giscoR

BugReports https://github.com/rOpenGov/giscoR/issues

Depends R (>= 3.6.0)

Imports countrycode (>= 1.2.0), geojsonsf (>= 2.0.0), jsonlite, rappdirs (>= 0.3.0), sf (>= 0.9.0), utils

Suggests dplyr, eurostat, ggplot2 (>= 3.5.0), httr2, knitr, lwgeom (>= 0.2-2), rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/Needs/website ropengov/rogtemplate, ragg, reactable, styler, devtools, remotes, geometries, rapidjsonr, sfheaders, jsonify

Config/testthat/edition 3

Config/testthat/parallel true

Copyright General Copyright © European Union, 1995 - today. See file COPYRIGHTS for specific provisions.

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

X-schema.org-applicationCategory cartography

X-schema.org-isPartOf http://ropengov.org/

X-schema.org-keywords ropengov, r, spatial, api-wrapper, rstats, r-package, eurostat, gisco, thematic-maps, eurostat-data

NeedsCompilation no

Author Diego Hernangómez [aut, cre, cph] (<https://orcid.org/0000-0001-8457-4658>, rOpenGov), EuroGeographics [cph] (for the administrative boundaries.), Vincent Arel-Bundock [cph] (<https://orcid.org/0000-0003-2042-7063>, for the gisco_countrycode dataset.)

Maintainer Diego Hernangómez <diego.hernangomezherrero@gmail.com>

Repository CRAN

Date/Publication 2024-05-29 20:50:06 UTC

R topics documented:

`gisco_addressapi` *GISCO Address API*

Description

Access the GISCO Address API, that allows to carry out both geocoding and reverse geocoding using a pan-european address database.

Each endpoint available is implemented through a specific function, see **Details**.

The API supports fuzzy searching (also referred to as approximate string matching) for all parameters of each endpoint.

Usage

```
gisco_addressapi_search(
  country = NULL,
  province = NULL,
  city = NULL,
  road = NULL,
  housenumber = NULL,
  postcode = NULL,
  verbose = FALSE
)

gisco_addressapi_reverse(x, y, country = NULL, verbose = FALSE)

gisco_addressapi_bbox(
  country = NULL,
  province = NULL,
  city = NULL,
  road = NULL,
  postcode = NULL,
  verbose = FALSE
)

gisco_addressapi_countries(verbose = FALSE)

gisco_addressapi_provinces(country = NULL, city = NULL, verbose = FALSE)

gisco_addressapi_cities(country = NULL, province = NULL, verbose = FALSE)

gisco_addressapi_roads(
  country = NULL,
  province = NULL,
  city = NULL,
  verbose = FALSE
)

gisco_addressapi_housenumbers(
  country = NULL,
  province = NULL,
  city = NULL,
  road = NULL,
```

```
  postcode = NULL,
  verbose = FALSE
)

gisco_addressapi_postcodes(
  country = NULL,
  province = NULL,
  city = NULL,
  verbose = FALSE
)

gisco_addressapi_copyright(verbose = FALSE)
```

Arguments

country	Country code (country = "LU").
province	A province within a country. For a list of provinces within a certain country use the provinces endpoint (gisco_addressapi_provinces(country = "LU")).
city	A city within a province. For a list of cities within a certain province use the cities endpoint (gisco_addressapi_cities(province = "capellen")).
road	A road within a city.
housenumber	The house number or house name within a road or street.
postcode	Can be used in combination with the previous parameters.
verbose	Logical, displays information. Useful for debugging, default is FALSE.
x, y	x and y coordinates (as longitude and latitude) to be converted into a human-readable address.

Details

Brief description of the API endpoints (source GISCO Address API \> Endpoints:

Endpoint	Description
/countries	Returns all country codes that are compatible with the address API. Check the coverage map for available c
/provinces	Returns all provinces within the specified country. Can also be used to get the province of a specified city.
/cities	Returns all cities within a specified province or country.
/roads	Returns all roads or streets within a specified city.
/housenumbers	Returns all house numbers or names within the specified road. It is possible that in certain countries an add
/postcodes	Returns all postcodes within the specified address component (Country or Province or City).
/search	The search endpoint allows structured queries to the address database. Please note that various combinatior
/reverse	The API's reverse theme allows you to specify x and y coordinates in order to retrieve a structured address.
/bbox	Returns a WKT bounding box for an address component depending on the parameters specified.
/copyright	Returns the copyright text for each available country in the Address API.

The resulting object may present the following variables:

Property name	Description

LD	Refers to "Locator Designator" and represents the house number part of the address
TF	Refers to "Thoroughfare" and represents the street or road part of the address
L0	Refers to Level 0 of the API administrative levels. Values are country codes consisting of 2 characters.
L1	Refers to Level 1 of the API administrative levels. Values are province names. Please note that "province" is
L2	Refers to Level 2 of the API administrative levels. Values are town or city names. Please note that "city" is a
PC	Postal Code
N0	Refers to "NUTS 0"
N1	Refers to "NUTS 1"
N2	Refers to "NUTS 2"
N3	Refers to "NUTS 3"
X and Y	Refers to the x and y coordinates of the address point
OL	Refers to the address' Open Location Code

Value

A `data.frame` object in most cases, except `gisco_addressapi_search()`, `gisco_addressapi_reverse()` and `gisco_addressapi_bbox()`, that return a `sf` object.

See Also

See the docs: https://gisco-services.ec.europa.eu/addressapi/docs/screen/home.

Examples

```
library(dplyr)

# Cities in a region

gisco_addressapi_cities(country = "PT", province = "LISBOA") %>%
  as_tibble()



# Geocode and reverse geocode with sf objects
library(ggplot2)

# Structured search

struct <- gisco_addressapi_search(
  country = "ES", city = "BARCELONA",
  road = "GRACIA"
)

struct %>%
  ggplot() +
  geom_sf(aes(color = PC)) +
  labs(color = "POSTAL CODES")
```

```
# Reverse geocoding

reverse <- gisco_addressapi_reverse(x = struct$X[1], y = struct$Y[1])

glimpse(reverse)
```

`gisco_attributions` *Attribution when publishing GISCO data*

Description

Get the legal text to be used along with the data downloaded with this package.

Usage

```
gisco_attributions(lang = "en", copyright = FALSE)
```

Arguments

`lang`	Language (two-letter ISO code). See https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes and **Details**.
`copyright`	Boolean `TRUE/FALSE`. Whether to display the copyright notice or not on the console.

Details

Current languages supported are:

- `"en"`: English.
- `"da"`: Danish.
- `"de"`: German.
- `"es"`: Spanish.
- `"fi"`: Finish.
- `"fr"`: French.
- `"no"`: Norwegian.
- `"sv"`: Swedish.

Please consider contributing if you spot any mistake or want to add a new language.

Value

A string with the attribution to be used.

Note

COPYRIGHT NOTICE

When data downloaded from GISCO is used in any printed or electronic publication, in addition to any other provisions applicable to the whole Eurostat website, data source will have to be acknowledged in the legend of the map and in the introductory page of the publication with the following copyright notice:

- EN: (C) EuroGeographics for the administrative boundaries.
- FR: (C) EuroGeographics pour les limites administratives.
- DE: (C) EuroGeographics bezuglich der Verwaltungsgrenzen.

For publications in languages other than English, French or German, the translation of the copyright notice in the language of the publication shall be used.

If you intend to use the data commercially, please contact EuroGeographics for information regarding their licence agreements.

See Also

Other helper: `gisco_check_access()`

Examples

```
gisco_attributions()

gisco_attributions(lang = "es", copyright = TRUE)

gisco_attributions(lang = "XXX")
```

---

gisco_bulk_download *Bulk download from GISCO API*

Description

Downloads zipped data from GISCO and extract them on the `cache_dir` folder.

Usage

```
gisco_bulk_download(
  id_giscoR = c("countries", "coastallines", "communes", "lau", "nuts", "urban_audit"),
  year = "2016",
  cache_dir = NULL,
  update_cache = FALSE,
  verbose = FALSE,
  resolution = "10",
  ext = c("geojson", "shp", "svg", "json", "gdb"),
  recursive = TRUE
)
```

Arguments

id_giscoR	Type of dataset to be downloaded. Values supported are: • "coastallines". • "communes". • "countries". • "lau". • "nuts". • "urban_audit".
year	Release year of the file. See **Details**.
cache_dir	A path to a cache directory. See **About caching**.
update_cache	A logical whether to update cache. Default is FALSE. When set to TRUE it would force a fresh download of the source .geojson file.
verbose	Logical, displays information. Useful for debugging, default is FALSE.
resolution	Resolution of the geospatial data. One of • "60": 1:60million • "20": 1:20million • "10": 1:10million • "03": 1:3million • "01": 1:1million
ext	Extension of the file(s) to be downloaded. Formats available are "geojson", "shp", "svg", "json", "gdb". See **Details**.
recursive	Tries to unzip recursively the zip files (if any) included in the initial bulk download (case of ext = "shp").

Details

See the years available in the corresponding functions:

- gisco_get_coastallines().
- gisco_get_communes().
- gisco_get_countries().
- gisco_get_lau().
- gisco_get_nuts().
- gisco_get_urban_audit().

The usual extension used across **giscoR** is "geojson", however other formats are already available on GISCO.

Value

Silent function.

About caching

You can set your cache_dir with gisco_set_cache_dir().

Sometimes cached files may be corrupt. On that case, try re-downloading the data setting update_cache = TRUE.

If you experience any problem on download, try to download the corresponding .geojson file by any other method and save it on your cache_dir. Use the option verbose = TRUE for debugging the API query.

For a complete list of files available check gisco_db.

Source

https://gisco-services.ec.europa.eu/distribution/v2/

See Also

Other political: gisco_get_coastallines(), gisco_get_countries(), gisco_get_lau(), gisco_get_nuts(), gisco_get_postalcodes(), gisco_get_units(), gisco_get_urban_audit()

Examples

```
## Not run:

# Countries 2016 - It would take some time
gisco_bulk_download(id_giscoR = "countries", resolution = "60")

## End(Not run)
```

---

gisco_check_access *Check access to GISCO API*

Description

Check if **R** has access to resources at https://gisco-services.ec.europa.eu/distribution/v2/.

Usage

```
gisco_check_access()
```

Value

a logical.

See Also

Other helper: gisco_attributions()

Examples

```
gisco_check_access()
```

`gisco_clear_cache` *Clear your* R*hrefhttps://CRAN.R-project.org/package=giscoR***giscoR** *cache dir*

Description

Use this function with caution. This function would clear your cached data and configuration, specifically:

- Deletes the **giscoR** config directory (`rappdirs::user_config_dir("giscoR", "R")`).
- Deletes the `cache_dir` directory.
- Deletes the values on stored on `Sys.getenv("GISCO_CACHE_DIR")` and `options(gisco_cache_dir)`.

Usage

```
gisco_clear_cache(config = FALSE, cached_data = TRUE, verbose = FALSE)
```

Arguments

`config`	if `TRUE`, will delete the configuration folder of **giscoR**.
`cached_data`	If this is set to `TRUE`, it will delete your `cache_dir` and all its content.
`verbose`	Logical, displays information. Useful for debugging, default is `FALSE`.

Details

This is an overkill function that is intended to reset your status as it you would never have installed and/or used **giscoR**.

Value

Invisible. This function is called for its side effects.

See Also

`rappdirs::user_config_dir()`

Other cache utilities: `gisco_set_cache_dir()`

Examples

```
# Don't run this! It would modify your current state
## Not run:
gisco_clear_cache(verbose = TRUE)

Sys.getenv("GISCO_CACHE_DIR")

# Set new cache on a temp dir
newcache <- file.path(tempdir(), "giscoR", "pkgdown")

newcache

gisco_set_cache_dir(newcache)

Sys.getenv("GISCO_CACHE_DIR")

## End(Not run)
```

---

gisco_coastallines *World coastal lines* POLYGON *object*

Description

A sf object as provided by GISCO (2016 version).

Format

A POLYGON sf object (resolution: 1:20million, EPSG:4326) with 3 variables:

COAS_ID Coast ID.

FID FID.

geometry geometry field.

Source

COAS_RG_20M_2016_4326.geojson file.

See Also

gisco_get_coastallines()

Other dataset: gisco_countries, gisco_countrycode, gisco_db, gisco_nuts

Examples

```
data("gisco_coastallines")
head(gisco_coastallines)
```

gisco_countries *World countries* POLYGON sf *object*

Description

A sf object including all countries as provided by GISCO (2016 version).

Format

A MULTIPOLYGON data frame (resolution: 1:20million, EPSG:4326) object with 257 rows and 7 variables:

id row ID.

CNTR_NAME Official country name on local language.

ISO3_CODE ISO 3166-1 alpha-3 code of each country, as provided by GISCO.

CNTR_ID Country ID.

NAME_ENGL Country name in English.

FID FID.

geometry geometry field.

Source

CNTR_RG_20M_2016_4326.geojson file.

See Also

gisco_get_countries()

Other dataset: gisco_coastallines, gisco_countrycode, gisco_db, gisco_nuts

Examples

```
data("gisco_countries")
head(gisco_countries)
```

`gisco_countrycode` *Data frame with different country code schemes and world regions*

Description

A data frame containing conversions between different country code schemes (Eurostat/ISO2 and 3) as well as geographic regions as provided by the World Bank and the UN (M49). This data set is extracted from **countrycode** package.

Format

A data frame object with 249 rows and 13 variables:

ISO3_CODE Eurostat code of each country.

CNTR_CODE ISO 3166-1 alpha-2 code of each country.

iso2c ISO 3166-1 alpha-3 code of each country.

iso.name.en ISO English short name.

cldr.short.en English short name as provided by the Unicode Common Locale Data Repository.

continent As provided by the World Bank.

un.region.code Numeric region code UN (M49).

un.region.name Region name UN (M49).

un.regionintermediate.code Numeric intermediate Region.

un.regionintermediate.name Intermediate Region name UN (M49).

un.regionsub.code Numeric sub-region code UN (M49).

un.regionsub.name Sub-Region name UN (M49).

eu Logical indicating if the country belongs to the European Union.

Source

countrycode::codelist **v1.2.0**.

See Also

`gisco_get_countries()` and countrycode::codelist, included in **countrycode**.

See also the Unicode Common Locale Data Repository.

Other dataset: `gisco_coastallines`, `gisco_countries`, `gisco_db`, `gisco_nuts`

Examples

```
data("gisco_countrycode")
dplyr::glimpse(gisco_countrycode)
```

`gisco_get_airports` *Get location of airports and ports from GISCO API*

Description

Loads a `sf` object from GISCO API or your local library.

Usage

```
gisco_get_airports(
  year = "2013",
  country = NULL,
  cache_dir = NULL,
  update_cache = FALSE,
  verbose = FALSE
)

gisco_get_ports(
  year = "2013",
  country = NULL,
  cache_dir = NULL,
  update_cache = FALSE,
  verbose = FALSE
)
```

Arguments

`year`	Year of reference. Only year available right now is `"2013"`.
`country`	Optional. A character vector of country codes. It could be either a vector of country names, a vector of ISO3 country codes or a vector of Eurostat country codes. Mixed types (as `c("Italy","ES","FRA")`) would not work. See also `countrycode::countrycode()`.
`cache_dir`	A path to a cache directory. See **About caching**.
`update_cache`	A logical whether to update cache. Default is `FALSE`. When set to `TRUE` it would force a fresh download of the source `.geojson` file.
`verbose`	Logical, displays information. Useful for debugging, default is `FALSE`.

Details

`gisco_get_airports()` refer to Europe. All shapefiles provided in EPSG:4326.

`gisco_get_ports()` adds a new field `CNTR_ISO2` to the original data identifying the country of the port. Worldwide information available. The port codes are aligned with UN/LOCODE standard.

Value

A `POINT` object on EPSG:4326.

About caching

You can set your `cache_dir` with `gisco_set_cache_dir()`.

Sometimes cached files may be corrupt. On that case, try re-downloading the data setting `update_cache = TRUE`.

If you experience any problem on download, try to download the corresponding `.geojson` file by any other method and save it on your `cache_dir`. Use the option `verbose = TRUE` for debugging the API query.

For a complete list of files available check gisco_db.

Source

https://ec.europa.eu/eurostat/web/gisco/geodata/transport-networks

See Also

Other infrastructure: `gisco_get_education()`, `gisco_get_healthcare()`

Examples

```
library(sf)

Greece <- gisco_get_countries(country = "EL", resolution = "1")
AirP_GC <- gisco_get_airports(country = "EL")
AirP_GC <- st_transform(AirP_GC, st_crs(Greece))

library(ggplot2)


ggplot(Greece) +
  geom_sf(fill = "grey80") +
  geom_sf(data = AirP_GC, color = "blue") +
  labs(
    title = "Airports on Greece",
    shape = NULL,
    color = NULL,
    caption = gisco_attributions()
  )


##############################
#         Plot ports         #
##############################

ports <- gisco_get_ports()
coast <- giscoR::gisco_coastallines

# To Equal Earth projection :)
```

```
library(sf)
coast <- st_transform(coast, 8857)
ports <- st_transform(ports, st_crs(coast))


ggplot(coast) +
  geom_sf(fill = "#F6E1B9", color = "#0978AB") +
  geom_sf(data = ports, fill = "red", shape = 21) +
  theme_void() +
  theme(
    panel.background = element_rect(fill = "#C6ECFF"),
    panel.grid = element_blank(),
    plot.title = element_text(face = "bold", hjust = 0.5),
    plot.subtitle = element_text(face = "italic", hjust = 0.5)
  ) +
  labs(
    title = "Ports Worldwide", subtitle = "Year 2013",
    caption = "(c) European Union, 1995 - today"
  )
```

`gisco_get_coastallines`

Get GISCO coastlines `sf` *polygons*

Description

Downloads worldwide coastlines

Usage

```
gisco_get_coastallines(
  year = "2016",
  epsg = "4326",
  cache = TRUE,
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  resolution = "20"
)
```

Arguments

`year`	Release year. One of `"2006"`, `"2010"`, `"2013"` or `"2016"`.
`epsg`	projection of the map: 4-digit EPSG code. One of: • `"4258"`: ETRS89 • `"4326"`: WGS84

- "3035": ETRS89 / ETRS-LAEA
- "3857": Pseudo-Mercator

cache — A logical whether to do caching. Default is TRUE. See **About caching**.

update_cache — A logical whether to update cache. Default is FALSE. When set to TRUE it would force a fresh download of the source .geojson file.

cache_dir — A path to a cache directory. See **About caching**.

verbose — Logical, displays information. Useful for debugging, default is FALSE.

resolution — Resolution of the geospatial data. One of

- "60": 1:60million
- "20": 1:20million
- "10": 1:10million
- "03": 1:3million
- "01": 1:1million

Value

A sf POLYGON object.

About caching

You can set your cache_dir with gisco_set_cache_dir().

Sometimes cached files may be corrupt. On that case, try re-downloading the data setting update_cache = TRUE.

If you experience any problem on download, try to download the corresponding .geojson file by any other method and save it on your cache_dir. Use the option verbose = TRUE for debugging the API query.

For a complete list of files available check gisco_db.

Note

Please check the download and usage provisions on gisco_attributions().

Source

https://gisco-services.ec.europa.eu/distribution/v2/

See Also

gisco_coastallines

Other political: gisco_bulk_download(), gisco_get_countries(), gisco_get_lau(), gisco_get_nuts(), gisco_get_postalcodes(), gisco_get_units(), gisco_get_urban_audit()

Examples

```
coast <- gisco_get_coastallines()

library(ggplot2)

ggplot(coast) +
  geom_sf(color = "#1278AB", fill = "#FDFBEA") +
  # Zoom on Caribe
  coord_sf(
    xlim = c(-99, -49),
    ylim = c(4, 30)
  ) +
  theme_minimal() +
  theme(
    panel.background = element_rect(fill = "#C7E7FB", color = NA),
    panel.border = element_rect(colour = "black", fill = NA)
  )
```

---

gisco_get_countries *Get GISCO world country sf polygons, points and lines*

Description

Returns world country polygons, lines and points at a specified scale, as provided by GISCO. Also, specific areas as Gibraltar or Antarctica are presented separately. The definition of country used on GISCO correspond roughly with territories with an official ISO-3166 code.

Usage

```
gisco_get_countries(
  year = "2016",
  epsg = "4326",
  cache = TRUE,
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  resolution = "20",
  spatialtype = "RG",
  country = NULL,
  region = NULL
)
```

Arguments

year	Release year of the file. One of "2001", "2006", "2010", "2013", "2016" or "2020".

epsg — projection of the map: 4-digit EPSG code. One of:

- "4258": ETRS89
- "4326": WGS84
- "3035": ETRS89 / ETRS-LAEA
- "3857": Pseudo-Mercator

cache — A logical whether to do caching. Default is TRUE. See **About caching**.

update_cache — A logical whether to update cache. Default is FALSE. When set to TRUE it would force a fresh download of the source .geojson file.

cache_dir — A path to a cache directory. See **About caching**.

verbose — Logical, displays information. Useful for debugging, default is FALSE.

resolution — Resolution of the geospatial data. One of

- "60": 1:60million
- "20": 1:20million
- "10": 1:10million
- "03": 1:3million
- "01": 1:1million

spatialtype — Type of geometry to be returned:

- "BN": Boundaries - LINESTRING object.
- "COASTL": coastlines - LINESTRING object.
- "INLAND": inland boundaries - LINESTRING object.
- "LB": Labels - POINT object.
- "RG": Regions - MULTIPOLYGON/POLYGON object.

Note that parameters country and region would be only applied when spatialtype is "BN" or "RG".

country — Optional. A character vector of country codes. It could be either a vector of country names, a vector of ISO3 country codes or a vector of Eurostat country codes. Mixed types (as c("Italy","ES","FRA")) would not work. See also countrycode::countrycode().

region — Optional. A character vector of UN M49 region codes or European Union membership. Possible values are "Africa", "Americas", "Asia", "Europe", "Oceania" or "EU" for countries belonging to the European Union (as per 2021). See **About world regions** and gisco_countrycode.

Value

A sf object specified by spatialtype.

About caching

You can set your cache_dir with gisco_set_cache_dir().

Sometimes cached files may be corrupt. On that case, try re-downloading the data setting update_cache = TRUE.

If you experience any problem on download, try to download the corresponding `.geojson` file by any other method and save it on your `cache_dir`. Use the option `verbose = TRUE` for debugging the API query.

For a complete list of files available check gisco_db.

World Regions

Regions are defined as per the geographic regions defined by the UN (see https://unstats.un.org/unsd/methodology/m49/. Under this scheme Cyprus is assigned to Asia. You may use `region = "EU"` to get the EU members (reference date: 2021).

Note

Please check the download and usage provisions on `gisco_attributions()`.

Source

https://gisco-services.ec.europa.eu/distribution/v2/

See Also

`gisco_countrycode()`, gisco_countries, `countrycode::countrycode()`

Other political: `gisco_bulk_download()`, `gisco_get_coastallines()`, `gisco_get_lau()`, `gisco_get_nuts()`, `gisco_get_postalcodes()`, `gisco_get_units()`, `gisco_get_urban_audit()`

Examples

```
cntries <- gisco_get_countries()

library(ggplot2)
ggplot(cntries) +
  geom_sf()

# Get a region

africa <- gisco_get_countries(region = "Africa")
ggplot(africa) +
  geom_sf(fill = "#078930", col = "white") +
  theme_minimal()
```

gisco_get_education *Get locations of education services in Europe*

Description

The dataset contains information on main education services by Member States.

Usage

```
gisco_get_education(
  cache = TRUE,
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  country = NULL
)
```

Arguments

cache	A logical whether to do caching. Default is TRUE. See **About caching**.
update_cache	A logical whether to update cache. Default is FALSE. When set to TRUE it would force a fresh download of the source .geojson file.
cache_dir	A path to a cache directory. See **About caching**.
verbose	Logical, displays information. Useful for debugging, default is FALSE.
country	Optional. A character vector of country codes. It could be either a vector of country names, a vector of ISO3 country codes or a vector of Eurostat country codes. Mixed types (as c("Italy","ES","FRA")) would not work. See also countrycode::countrycode().

Details

Files are distributed on EPSG:4326. Metadata available on https://gisco-services.ec.europa.eu/pub/education/metadata.pdf.

Value

A POINT sf object.

About caching

You can set your cache_dir with gisco_set_cache_dir().

Sometimes cached files may be corrupt. On that case, try re-downloading the data setting update_cache = TRUE.

If you experience any problem on download, try to download the corresponding .geojson file by any other method and save it on your cache_dir. Use the option verbose = TRUE for debugging the API query.

For a complete list of files available check gisco_db.

Author(s)

dieghernan, https://github.com/dieghernan/

Source

https://ec.europa.eu/eurostat/web/gisco/geodata/basic-services

See Also

gisco_get_countries()

Other infrastructure: gisco_get_airports(), gisco_get_healthcare()

Examples

```
edu_BEL <- gisco_get_education(country = "Belgium")

# Plot if downloaded
if (nrow(edu_BEL) > 3) {
  library(ggplot2)
  ggplot(edu_BEL) +
    geom_sf(shape = 21, size = 0.15)
}
```

---

gisco_get_grid *Get grid cells covering covering Europe for various resolutions*

Description

These datasets contain grid cells covering the European land territory, for various resolutions from 1km to 100km. Base statistics such as population figures are provided for these cells.

Usage

```
gisco_get_grid(
  resolution = "20",
  spatialtype = c("REGION", "POINT"),
  cache_dir = NULL,
  update_cache = FALSE,
  verbose = FALSE
)
```

Arguments

resolution	Resolution of the grid cells on kms. Available values are "1", "2", "5", "10", "20", "50", "100". See **Details**.
spatialtype	Select one of "REGION" or "POINT".
cache_dir	A path to a cache directory. See **About caching**.
update_cache	A logical whether to update cache. Default is FALSE. When set to TRUE it would force a fresh download of the source .geojson file.
verbose	Logical, displays information. Useful for debugging, default is FALSE.

Details

Files are distributed on EPSG:3035.

The file sizes range is from 428Kb (resolution = "100") to 1.7Gb resolution = "1". For resolutions 1km and 2km you would need to confirm the download.

Value

A POLYGON/POINT sf object.

About caching

You can set your cache_dir with gisco_set_cache_dir().

Sometimes cached files may be corrupt. On that case, try re-downloading the data setting update_cache = TRUE.

If you experience any problem on download, try to download the corresponding .geojson file by any other method and save it on your cache_dir. Use the option verbose = TRUE for debugging the API query.

For a complete list of files available check gisco_db.

Note

There are specific downloading provisions, please see https://ec.europa.eu/eurostat/web/gisco/geodata/grids

Author(s)

dieghernan, https://github.com/dieghernan/

Source

https://ec.europa.eu/eurostat/web/gisco/geodata/grids

Examples

```
grid <- gisco_get_grid(resolution = 20)

# If downloaded correctly proceed

if (!is.null(grid)) {
  library(dplyr)

  grid <- grid %>%
    mutate(popdens = TOT_P_2021 / 20)

  breaks <- c(0, 0.1, 100, 500, 1000, 5000, 10000, Inf)

  # Cut groups
  grid <- grid %>%
    mutate(popdens_cut = cut(popdens,
      breaks = breaks,
      include.lowest = TRUE
    ))

  cut_labs <- prettyNum(breaks, big.mark = " ")[-1]
  cut_labs[1] <- "0"
  cut_labs[7] <- "> 10 000"

  pal <- c("black", hcl.colors(length(breaks) - 2,
    palette = "Spectral",
    alpha = 0.9
  ))

  library(ggplot2)

  ggplot(grid) +
    geom_sf(aes(fill = popdens_cut), color = NA, linewidth = 0) +
    coord_sf(
      xlim = c(2500000, 7000000),
      ylim = c(1500000, 5200000)
    ) +
    scale_fill_manual(
      values = pal, na.value = "black",
      name = "people per sq. kilometer",
      labels = cut_labs,
      guide = guide_legend(
        direction = "horizontal",
        nrow = 1
      )
    ) +
    theme_void() +
    labs(
      title = "Population density in Europe (2021)",
      subtitle = "Grid: 20 km.",
```

```
      caption = gisco_attributions()
    ) +
    theme(
      text = element_text(colour = "white"),
      plot.background = element_rect(fill = "grey2"),
      plot.title = element_text(hjust = 0.5),
      plot.subtitle = element_text(hjust = 0.5, face = "bold"),
      plot.caption = element_text(
        color = "grey60", hjust = 0.5, vjust = 0,
        margin = margin(t = 5, b = 10)
      ),
      legend.position = "bottom",
      legend.title.position = "top",
      legend.text.position = "bottom",
      legend.key.height = unit(0.5, "lines"),
      legend.key.width = unit(1, "lines")
    )
}
```

`gisco_get_healthcare` *Get locations of healthcare services in Europe*

Description

The dataset contains information on main healthcare services considered to be 'hospitals' by Member States.

Usage

```
gisco_get_healthcare(
  cache = TRUE,
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  country = NULL
)
```

Arguments

`cache`	A logical whether to do caching. Default is `TRUE`. See **About caching**.
`update_cache`	A logical whether to update cache. Default is `FALSE`. When set to `TRUE` it would force a fresh download of the source `.geojson` file.
`cache_dir`	A path to a cache directory. See **About caching**.
`verbose`	Logical, displays information. Useful for debugging, default is `FALSE`.

country — Optional. A character vector of country codes. It could be either a vector of country names, a vector of ISO3 country codes or a vector of Eurostat country codes. Mixed types (as `c("Italy","ES","FRA")`) would not work. See also `countrycode::countrycode()`.

Details

Files are distributed on EPSG:4326. Metadata available on https://gisco-services.ec.europa.eu/pub/healthcare/metadata.pdf.

Value

A `POINT` `sf` object.

About caching

You can set your `cache_dir` with `gisco_set_cache_dir()`.

Sometimes cached files may be corrupt. On that case, try re-downloading the data setting `update_cache = TRUE`.

If you experience any problem on download, try to download the corresponding `.geojson` file by any other method and save it on your `cache_dir`. Use the option `verbose = TRUE` for debugging the API query.

For a complete list of files available check gisco_db.

Author(s)

dieghernan, https://github.com/dieghernan/

Source

https://ec.europa.eu/eurostat/web/gisco/geodata/basic-services

See Also

`gisco_get_countries()`

Other infrastructure: `gisco_get_airports()`, `gisco_get_education()`

Examples

```
health_BEL <- gisco_get_healthcare(country = "Belgium")

# Plot if downloaded
if (nrow(health_BEL) > 3) {
  library(ggplot2)
  ggplot(health_BEL) +
    geom_sf(aes(color = emergency))
}
```

gisco_get_lau *Get GISCO urban areas sf polygons, points and lines*

Description

gisco_get_communes() and gisco_get_lau() download shapes of Local Urban Areas, that correspond roughly with towns and cities.

Usage

```
gisco_get_communes(
  year = "2016",
  epsg = "4326",
  cache = TRUE,
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  spatialtype = "RG",
  country = NULL
)

gisco_get_lau(
  year = "2021",
  epsg = "4326",
  cache = TRUE,
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  country = NULL,
  gisco_id = NULL
)
```

Arguments

year — Release year of the file:

- For gisco_get_communes() one of "2001", "2004", "2006", "2008", "2010", "2013" or "2016".
- For gisco_get_lau() one of "2011", "2012", "2013", "2014", "2015", "2016", "2017", "2018", "2019", "2020" or "2021".

epsg — projection of the map: 4-digit EPSG code. One of:

- "4258": ETRS89
- "4326": WGS84
- "3035": ETRS89 / ETRS-LAEA

- "3857": Pseudo-Mercator

cache — A logical whether to do caching. Default is TRUE. See **About caching**.

update_cache — A logical whether to update cache. Default is FALSE. When set to TRUE it would force a fresh download of the source .geojson file.

cache_dir — A path to a cache directory. See **About caching**.

verbose — Logical, displays information. Useful for debugging, default is FALSE.

spatialtype — Type of geometry to be returned:

- "BN": Boundaries - LINESTRING object.
- "COASTL": coastlines - LINESTRING object.
- "INLAND": inland boundaries - LINESTRING object.
- "LB": Labels - POINT object.
- "RG": Regions - MULTIPOLYGON/POLYGON object.

Note that parameters country and region would be only applied when spatialtype is "BN" or "RG".

country — Optional. A character vector of country codes. It could be either a vector of country names, a vector of ISO3 country codes or a vector of Eurostat country codes. Mixed types (as c("Italy","ES","FRA")) would not work. See also countrycode::countrycode().

gisco_id — Optional. A character vector of GISCO_ID LAU values.

Value

A sf object specified by spatialtype. In the case of gisco_get_lau(), a POLYGON object.

About caching

You can set your cache_dir with gisco_set_cache_dir().

Sometimes cached files may be corrupt. On that case, try re-downloading the data setting update_cache = TRUE.

If you experience any problem on download, try to download the corresponding .geojson file by any other method and save it on your cache_dir. Use the option verbose = TRUE for debugging the API query.

For a complete list of files available check gisco_db.

Note

Please check the download and usage provisions on gisco_attributions().

See Also

Other political: gisco_bulk_download(), gisco_get_coastallines(), gisco_get_countries(), gisco_get_nuts(), gisco_get_postalcodes(), gisco_get_units(), gisco_get_urban_audit()

Examples

```
ire_lau <- gisco_get_communes(spatialtype = "LB", country = "Ireland")

if (!is.null(ire_lau)) {
  library(ggplot2)

  ggplot(ire_lau) +
    geom_sf(shape = 21, col = "#009A44", size = 0.5) +
    labs(
      title = "Communes in Ireland",
      subtitle = "Year 2016",
      caption = gisco_attributions()
    ) +
    theme_void() +
    theme(text = element_text(
      colour = "#009A44",
      family = "serif", face = "bold"
    ))
}
```

`gisco_get_nuts` *Get GISCO NUTS* `sf` *polygons, points and lines*

Description

Returns NUTS regions polygons, lines and points at a specified scale, as provided by GISCO.

NUTS are provided at three different levels:

- `"0"`: Country level
- `"1"`: Groups of states/regions
- `"2"`: States/regions
- `"3"`: Counties/provinces/districts

Note that NUTS-level definition may vary across countries. See also https://ec.europa.eu/eurostat/web/gisco/geodata//statistical-units/territorial-units-statistics.

Usage

```
gisco_get_nuts(
  year = "2016",
  epsg = "4326",
  cache = TRUE,
  update_cache = FALSE,
```

```
  cache_dir = NULL,
  verbose = FALSE,
  resolution = "20",
  spatialtype = "RG",
  country = NULL,
  nuts_id = NULL,
  nuts_level = "all"
)
```

Arguments

year	Release year of the file. One of "2003", "2006", "2010", "2013", "2016" or "2021".
epsg	projection of the map: 4-digit EPSG code. One of: • "4258": ETRS89 • "4326": WGS84 • "3035": ETRS89 / ETRS-LAEA • "3857": Pseudo-Mercator
cache	A logical whether to do caching. Default is TRUE. See **About caching**.
update_cache	A logical whether to update cache. Default is FALSE. When set to TRUE it would force a fresh download of the source .geojson file.
cache_dir	A path to a cache directory. See **About caching**.
verbose	Logical, displays information. Useful for debugging, default is FALSE.
resolution	Resolution of the geospatial data. One of • "60": 1:60million • "20": 1:20million • "10": 1:10million • "03": 1:3million • "01": 1:1million
spatialtype	Type of geometry to be returned: • "BN": Boundaries - LINESTRING object. • "LB": Labels - POINT object. • "RG": Regions - MULTIPOLYGON/POLYGON object. **Note that** parameters country, nuts_level and nuts_id would be only applied when spatialtype is "BN" or "RG".
country	Optional. A character vector of country codes. It could be either a vector of country names, a vector of ISO3 country codes or a vector of Eurostat country codes. Mixed types (as c("Italy","ES","FRA")) would not work. See also countrycode::countrycode().
nuts_id	Optional. A character vector of NUTS IDs.
nuts_level	NUTS level. One of "0", "1", "2" or "3". See **Description**.

Value

A sf object specified by spatialtype. The resulting sf object would present an additional column geo (equal to NUTS_ID) for improving compatibility with **eurostat** package. See eurostat::get_eurostat_geospatial())

See also gisco_nuts to understand the columns and values provided.

About caching

You can set your cache_dir with gisco_set_cache_dir().

Sometimes cached files may be corrupt. On that case, try re-downloading the data setting update_cache = TRUE.

If you experience any problem on download, try to download the corresponding .geojson file by any other method and save it on your cache_dir. Use the option verbose = TRUE for debugging the API query.

For a complete list of files available check gisco_db.

Source

https://gisco-services.ec.europa.eu/distribution/v2/

See Also

gisco_nuts, gisco_get_countries(), eurostat::get_eurostat_geospatial()

Other political: gisco_bulk_download(), gisco_get_coastallines(), gisco_get_countries(), gisco_get_lau(), gisco_get_postalcodes(), gisco_get_units(), gisco_get_urban_audit()

Examples

```
nuts2 <- gisco_get_nuts(nuts_level = 2)

library(ggplot2)

ggplot(nuts2) +
  geom_sf() +
  # ETRS89 / ETRS-LAEA
  coord_sf(
    crs = 3035, xlim = c(2377294, 7453440),
    ylim = c(1313597, 5628510)
  ) +
  labs(title = "NUTS-2 levels")

# NUTS-3 for Germany
germany_nuts3 <- gisco_get_nuts(nuts_level = 3, country = "Germany")

ggplot(germany_nuts3) +
  geom_sf() +
  labs(
    title = "NUTS-3 levels",
    subtitle = "Germany",
    caption = gisco_attributions()
```

```
  )


# Select specific regions
select_nuts <- gisco_get_nuts(nuts_id = c("ES2", "FRJ", "FRL", "ITC"))

ggplot(select_nuts) +
  geom_sf(aes(fill = CNTR_CODE)) +
  scale_fill_viridis_d()
```

`gisco_get_postalcodes` *Get postal code points from GISCO*

Description

Get postal codes points of the EU, EFTA and candidate countries.

Usage

```
gisco_get_postalcodes(
  year = "2020",
  country = NULL,
  cache_dir = NULL,
  update_cache = FALSE,
  verbose = FALSE
)
```

Arguments

`year`	Year of reference. Currently only `"2020"` is available.
`country`	Optional. A character vector of country codes. It could be either a vector of country names, a vector of ISO3 country codes or a vector of Eurostat country codes. Mixed types (as `c("Italy","ES","FRA")`) would not work. See also `countrycode::countrycode()`.
`cache_dir`	A path to a cache directory. See **About caching**.
`update_cache`	A logical whether to update cache. Default is `FALSE`. When set to `TRUE` it would force a fresh download of the source `.geojson` file.
`verbose`	Logical, displays information. Useful for debugging, default is `FALSE`.

Details

The postal code point dataset shows the location of postal codes, NUTS codes and the Degree of Urbanisation classification across the EU, EFTA and candidate countries from a variety of sources. Its primary purpose is to create correspondence tables for the NUTS classification (EC) 1059/2003 as part of the Tercet Regulation (EU) 2017/2391

Value

A POINT sf object on EPSG:4326.

Copyright

The dataset is released under the CC-BY-SA-4.0 licence and requires the following attribution whenever used:

(c) European Union - GISCO, 2021, postal code point dataset, Licence CC-BY-SA 4.0 available at https://ec.europa.eu/eurostat/web/gisco/geodata//administrative-units/postal-codes.

Shapefiles provided in ETRS89 (EPSG:4258).

About caching

You can set your cache_dir with gisco_set_cache_dir().

Sometimes cached files may be corrupt. On that case, try re-downloading the data setting update_cache = TRUE.

If you experience any problem on download, try to download the corresponding .geojson file by any other method and save it on your cache_dir. Use the option verbose = TRUE for debugging the API query.

For a complete list of files available check gisco_db.

Source

https://ec.europa.eu/eurostat/web/gisco/geodata//administrative-units/postal-codes.

See Also

Other political: gisco_bulk_download(), gisco_get_coastallines(), gisco_get_countries(), gisco_get_lau(), gisco_get_nuts(), gisco_get_units(), gisco_get_urban_audit()

Examples

```
# Heavy-weight download!
## Not run:

pc_bel <- gisco_get_postalcodes(country = "BE")

if (!is.null(pc_bel)) {
  library(ggplot2)

  ggplot(pc_bel) +
    geom_sf(color = "gold") +
    theme_bw() +
    labs(
      title = "Postcodes of Belgium",
      subtitle = "2020",
      caption = paste("(c) European Union - GISCO, 2021,",
```

```
        "postal code point dataset",
        "Licence CC-BY-SA 4.0",
        sep = "\n"
      )
    )
}

## End(Not run)
```

`gisco_get_units` *Get geospatial units data from GISCO API*

Description

Download individual shapefiles of units. Unlike `gisco_get_countries()`, `gisco_get_nuts()` or `gisco_get_urban_audit()`, that downloads a full dataset and applies filters, `gisco_get_units()` downloads a single shapefile for each unit.

Usage

```
gisco_get_units(
  id_giscoR = c("nuts", "countries", "urban_audit"),
  unit = "ES4",
  mode = c("sf", "df"),
  year = "2016",
  epsg = "4326",
  cache = TRUE,
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  resolution = "20",
  spatialtype = "RG"
)
```

Arguments

`id_giscoR`	Select the `unit` type to be downloaded. Accepted values are `"nuts"`, `"countries"` or `"urban_audit"`.
`unit`	Unit ID to be downloaded. See **Details**.
`mode`	Controls the output of the function. Possible values are `"sf"` or `"df"`. See **Value** and **Details**.
`year`	Release year of the file. One of `"2001"`, `"2006"`, `"2010"`, `"2013"`, `"2016"` or `"2020"`.
`epsg`	projection of the map: 4-digit EPSG code. One of: • `"4258"`: ETRS89

- "4326": WGS84
- "3035": ETRS89 / ETRS-LAEA
- "3857": Pseudo-Mercator

cache	A logical whether to do caching. Default is TRUE. See **About caching**.
update_cache	A logical whether to update cache. Default is FALSE. When set to TRUE it would force a fresh download of the source .geojson file.
cache_dir	A path to a cache directory. See **About caching**.
verbose	Logical, displays information. Useful for debugging, default is FALSE.
resolution	Resolution of the geospatial data. One of • "60": 1:60million • "20": 1:20million • "10": 1:10million • "03": 1:3million • "01": 1:1million
spatialtype	Type of geometry to be returned: "RG", for POLYGON and "LB" for POINT.

Details

The function can return a data frame on mode = "df" or a sf object on mode = "sf".

In order to see the available unit ids with the required combination of spatialtype, year, first run the function on "df" mode. Once that you get the data frame you can select the required ids on the unit parameter.

On mode = "df" the only relevant parameters are spatialtype, year.

Value

A sf object on mode = "sf" or a data frame on mode = "df".

About caching

You can set your cache_dir with gisco_set_cache_dir().

Sometimes cached files may be corrupt. On that case, try re-downloading the data setting update_cache = TRUE.

If you experience any problem on download, try to download the corresponding .geojson file by any other method and save it on your cache_dir. Use the option verbose = TRUE for debugging the API query.

For a complete list of files available check gisco_db.

Note

Country-level files would be renamed on your cache_dir to avoid naming conflicts with NUTS-0 datasets.

Please check the download and usage provisions on gisco_attributions().

Author(s)

dieghernan, https://github.com/dieghernan/

Source

https://gisco-services.ec.europa.eu/distribution/v2/

See Also

gisco_get_countries()

Other political: gisco_bulk_download(), gisco_get_coastallines(), gisco_get_countries(), gisco_get_lau(), gisco_get_nuts(), gisco_get_postalcodes(), gisco_get_urban_audit()

Examples

```
cities <- gisco_get_units(
  id_giscoR = "urban_audit",
  mode = "df",
  year = "2020"
)
VAL <- cities[grep("Valencia", cities$URAU_NAME), ]
#   Order from big to small
VAL <- VAL[order(as.double(VAL$AREA_SQM), decreasing = TRUE), ]

VAL.sf <- gisco_get_units(
  id_giscoR = "urban_audit",
  year = "2020",
  unit = VAL$URAU_CODE
)
# Provincia
Provincia <-
  gisco_get_units(
    id_giscoR = "nuts",
    unit = c("ES523"),
    resolution = "01"
  )

# Reorder
VAL.sf$URAU_CATG <- factor(VAL.sf$URAU_CATG, levels = c("F", "K", "C"))

# Plot
library(ggplot2)

ggplot(Provincia) +
  geom_sf(fill = "gray1") +
  geom_sf(data = VAL.sf, aes(fill = URAU_CATG)) +
  scale_fill_viridis_d() +
  labs(
    title = "Valencia",
```

```
  subtitle = "Urban Audit",
  fill = "Urban Audit\ncategory"
)
```

`gisco_get_urban_audit` *Get GISCO greater cities and metropolitan areas* `sf` *objects*

Description

Returns polygons and points corresponding to cities, greater cities and metropolitan areas included on the Urban Audit report of Eurostat.

Usage

```
gisco_get_urban_audit(
  year = "2021",
  epsg = "4326",
  cache = TRUE,
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  spatialtype = "RG",
  country = NULL,
  level = NULL
)
```

Arguments

`year`	Release year of the file. One of `"2001"`, `"2004"`, `"2014"`, `"2018"`, `"2020"` or `"2021"`.
`epsg`	projection of the map: 4-digit EPSG code. One of: • `"4258"`: ETRS89 • `"4326"`: WGS84 • `"3035"`: ETRS89 / ETRS-LAEA • `"3857"`: Pseudo-Mercator
`cache`	A logical whether to do caching. Default is `TRUE`. See **About caching**.
`update_cache`	A logical whether to update cache. Default is `FALSE`. When set to `TRUE` it would force a fresh download of the source `.geojson` file.
`cache_dir`	A path to a cache directory. See **About caching**.
`verbose`	Logical, displays information. Useful for debugging, default is `FALSE`.
`spatialtype`	Type of geometry to be returned: • `"LB"`: Labels - `POINT` object.

- "RG": Regions - MULTIPOLYGON/POLYGON object.

country — Optional. A character vector of country codes. It could be either a vector of country names, a vector of ISO3 country codes or a vector of Eurostat country codes. Mixed types (as c("Italy","ES","FRA")) would not work. See also countrycode::countrycode().

level — Level of Urban Audit. Possible values are "CITIES", "FUA", "GREATER_CITIES" or NULL, that would download the full dataset.

Value

A sf object specified by spatialtype.

About caching

You can set your cache_dir with gisco_set_cache_dir().

Sometimes cached files may be corrupt. On that case, try re-downloading the data setting update_cache = TRUE.

If you experience any problem on download, try to download the corresponding .geojson file by any other method and save it on your cache_dir. Use the option verbose = TRUE for debugging the API query.

For a complete list of files available check gisco_db.

Note

Please check the download and usage provisions on gisco_attributions().

Source

https://gisco-services.ec.europa.eu/distribution/v2/

See Also

gisco_get_communes(), gisco_get_lau()

Other political: gisco_bulk_download(), gisco_get_coastallines(), gisco_get_countries(), gisco_get_lau(), gisco_get_nuts(), gisco_get_postalcodes(), gisco_get_units()

Examples

```
cities <- gisco_get_urban_audit(year = "2020", level = "CITIES")

if (!is.null(cities)) {
  bcn <- cities[cities$URAU_NAME == "Barcelona", ]

  library(ggplot2)
  ggplot(bcn) +
    geom_sf()
}
```

gisco_nuts *All NUTS* POLYGON *object*

Description

A sf object including all NUTS levels as provided by GISCO (2016 version).

Format

A POLYGON data frame (resolution: 1:20million, EPSG:4326) object with 2,016 rows and 11 variables:

NUTS_ID NUTS identifier.

LEVL_CODE NUTS level code (0,1,2,3).

URBN_TYPE Urban Type, see **Details**.

CNTR_CODE Eurostat Country code.

NAME_LATN NUTS name on Latin characters.

NUTS_NAME NUTS name on local alphabet.

MOUNT_TYPE Mount Type, see **Details**.

COAST_TYPE Coast Type, see **Details**.

FID FID.

geo Same as NUTS_ID, provided for compatibility with **eurostat**.

geometry geometry field.

Details

MOUNT_TYPE: Mountain typology:

- 1: More than 50 % of the surface is covered by topographic mountain areas.
- 2: More than 50 % of the regional population lives in topographic mountain areas.
- 3: More than 50 % of the surface is covered by topographic mountain areas and where more than 50 % of the regional population lives in these mountain areas.
- 4: Non-mountain region / other regions.
- 0: No classification provided.

URBN_TYPE: Urban-rural typology:

- 1: Predominantly urban region.
- 2: Intermediate region.
- 3: Predominantly rural region.

- `0`: No classification provided.

COAST_TYPE: Coastal typology:

- `1`: Coastal (on coast).
- `2`: Coastal (less than 50% of population living within 50 km. of the coastline).
- `3`: Non-coastal region.
- `0`: No classification provided.

Source

NUTS_RG_20M_2016_4326.geojson file.

See Also

`gisco_get_nuts()`

Other dataset: `gisco_coastallines`, `gisco_countries`, `gisco_countrycode`, `gisco_db`

Examples

```
data("gisco_nuts")
head(gisco_nuts)
```

`gisco_set_cache_dir` *Set your* R*hrefhttps://CRAN.R-project.org/package=giscoR***giscoR** *cache dir*

Description

This function will store your `cache_dir` path on your local machine and would load it for future sessions. Type `Sys.getenv("GISCO_CACHE_DIR")` to find your cached path or use `gisco_detect_cache_dir()`.

Alternatively, you can store the `cache_dir` manually with the following options:

- Run `Sys.setenv(GISCO_CACHE_DIR = "cache_dir")`. You would need to run this command on each session (Similar to `install = FALSE`).
- Write this line on your `.Renviron` file: `GISCO_CACHE_DIR = "value_for_cache_dir"` (same behavior than `install = TRUE`). This would store your `cache_dir` permanently. See also `usethis::edit_r_environ()`.

Usage

```
gisco_set_cache_dir(
  cache_dir,
  overwrite = FALSE,
  install = FALSE,
  verbose = TRUE
)

gisco_detect_cache_dir(...)
```

Arguments

cache_dir	A path to a cache directory. On missing value the function would store the cached files on a temporary dir (See base::tempdir()).
overwrite	If this is set to TRUE, it will overwrite an existing GISCO_CACHE_DIR that you already have in local machine.
install	If TRUE, will install the key in your local machine for use in future sessions. Defaults to FALSE. If cache_dir is FALSE this parameter is set to FALSE automatically.
verbose	Logical, displays information. Useful for debugging, default is FALSE.
...	Ignored

Value

gisco_set_cache_dir() returns an (invisible) character with the path to your cache_dir, but it is mainly called for its side effect.

gisco_detect_cache_dir() returns the path to the cache_dir used in this session.

See Also

rappdirs::user_config_dir()

Other cache utilities: gisco_clear_cache()

Examples

```
# Don't run this! It would modify your current state
## Not run:
gisco_set_cache_dir(verbose = TRUE)

## End(Not run)

Sys.getenv("GISCO_CACHE_DIR")

gisco_detect_cache_dir()
```

Index