

Package ‘giscoR’

December 10, 2025

Type Package

Title Download Map Data from GISCO API - Eurostat

Version 1.0.0

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 (>= 4.1)

Imports cli, countrycode (>= 1.2.0), httr2, jsonlite, lifecycle,
rappdirs (>= 0.3.0), sf (>= 1.0.0), tibble, utils

Suggests dplyr, eurostat, ggplot2 (>= 3.5.0), 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 Eurostat's general copyright notice and licence policy applies (see file COPYRIGHTS). Moreover, there are specific rules that apply to administrative and statistical data, see
<<https://ec.europa.eu/eurostat/web/gisco/geodata>>.

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

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, cran,
ggplot2, gis, cran-r

NeedsCompilation no

Author Diego Hernangómez [aut, cre, cph] (ORCID:
<<https://orcid.org/0000-0001-8457-4658>>),
Eurostat [cph] (ROR: <<https://ror.org/033d3q980>>),
EuroGeographics [cph]

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

Repository CRAN

Date/Publication 2025-12-10 07:40:02 UTC

Contents

gisco_address_api	3
gisco_attributions	6
gisco_bulk_download	8
gisco_clear_cache	10
gisco_coastal_lines	11
gisco_countries_2024	12
gisco_countrycode	13
gisco_db	14
gisco_get_airports	15
gisco_get_cached_db	16
gisco_get_census	17
gisco_get_coastal_lines	19
gisco_get_communes	20
gisco_get_countries	23
gisco_get_education	25
gisco_get_grid	27
gisco_get_healthcare	29
gisco_get_lau	32
gisco_get_metadata	34
gisco_get_nuts	35
gisco_get_ports	38
gisco_get_postal_codes	40
gisco_get_unit	42
gisco_get_units	44
gisco_get_urban_audit	46
gisco_id_api	49
gisco_nuts_2024	52
gisco_set_cache_dir	53

Index

56

gisco_address_api	<i>GISCO Address API</i>
-------------------	--------------------------

Description

Functions to interact with 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 arguments of each endpoint.

Usage

```
gisco_address_api_search(  
  country = NULL,  
  province = NULL,  
  city = NULL,  
  road = NULL,  
  housenumber = NULL,  
  postcode = NULL,  
  verbose = FALSE  
)  
  
gisco_address_api_reverse(x, y, country = NULL, verbose = FALSE)  
  
gisco_address_api_bbox(  
  country = NULL,  
  province = NULL,  
  city = NULL,  
  road = NULL,  
  postcode = NULL,  
  verbose = FALSE  
)  
  
gisco_address_api_countries(verbose = FALSE)  
  
gisco_address_api_provinces(country = NULL, city = NULL, verbose = FALSE)  
  
gisco_address_api_cities(country = NULL, province = NULL, verbose = FALSE)  
  
gisco_address_api_roads(  
  country = NULL,  
  province = NULL,  
  city = NULL,  
  verbose = FALSE  
)
```

```

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

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

gisco_address_api_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_address_api_provinces(country = "LU")).
city	A city within a province. For a list of cities within a certain province use the cities endpoint (gisco_address_api_cities(province = "capellen")).
road	A road within a city.
houenumber	The house number or house name within a road or street.
postcode	Can be used in combination with the previous arguments.
verbose	logical. If TRUE displays informational messages.
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 countries.
/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 address has multiple house numbers.
/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 combinations of arguments are possible.
/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 arguments 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 a
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 [tibble](#) in most cases, except `gisco_address_api_search()`, `gisco_address_api_reverse()` and `gisco_address_api_bbox()`, that return a [sf](#) object.

Source

<https://gisco-services.ec.europa.eu/addressapi/docs/screen/home>.

See Also

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

Other API tools: [gisco_id_api](#)

Examples

```
# Cities in a region

gisco_address_api_cities(country = "PT", province = "LISBOA")

# Geocode and reverse geocode with sf objects
# Structured search
struct <- gisco_address_api_search(
  country = "ES", city = "BARCELONA",
  road = "GRACIA"
)

struct
```

```
# Reverse geocoding
reverse <- gisco_address_api_reverse(x = struct$X[1], y = struct$Y[1])

reverse
```

gisco_attributions *Attribution for administrative and statistical GISCO data*

Description

Get the legal text to be used for administrative and statistical data downloaded from GISCO, see section **Copyright information**.

For other datasets you may abide to the [Eurostat's general copyright notice and licence policy](#).

Usage

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

Arguments

lang	character. Language (two-letter ISO code). See countrycode::codelist and Details .
copyright	logical 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.

Copyright information

These provisions described in this section apply to administrative and statistical data, provided by the following functions:

Administrative units

- `gisco_get_communes()`
- `gisco_get_countries()`
- `gisco_get_postal_codes()`

Statistical units

- `gisco_get_census()`
- `gisco_get_coastal_lines()`
- `gisco_get_lau()`
- `gisco_get_nuts()`
- `gisco_get_urban_audit()`

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: © EuroGeographics for the administrative boundaries.
- FR: © EuroGeographics pour les limites administratives.
- DE: © EuroGeographics bezüglich 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.

Examples

```
gisco_attributions()

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

gisco_attributions(lang = "XXX")

# Get list of codes from countrycodes
library(dplyr)

countrycode::codelist |>
  select(country.name.en, iso2c)
```

gisco_bulk_download *GISCO API bulk download*

Description

Download zipped data from GISCO to the [cache_dir](#) and extract the relevant ones.

Usage

```
gisco_bulk_download(
  id = c("countries", "coastal_lines", "communes", "lau", "nuts", "urban_audit",
        "postal_codes"),
  year = 2016,
  cache_dir = NULL,
  update_cache = FALSE,
  verbose = FALSE,
  resolution = 10,
  ext = c("shp", "geojson", "svg", "json", "gdb"),
  recursive = deprecated(),
  ...
)
```

Arguments

id	character string or number. Type of dataset to be downloaded, see Details . Values supported are: • "countries" • "coastal_lines" • "communes" • "lau" • "nuts" • "urban_audit" • "postal_codes" This argument replaces the previous (deprecated) argument <code>id_giscoR</code> .
year	character string or number. Release year of the file, see Details .
cache_dir	character string. A path to a cache directory. See Caching strategies section in gisco_set_cache_dir() .
update_cache	logical. Should the cached file be refreshed?. Default is FALSE. When set to TRUE it would force a new download.
verbose	logical. If TRUE displays informational messages.
resolution	character string or number. Resolution of the geospatial data. One of: • "60": 1:60 million. • "20": 1:20 million.

	• "10": 1:10 million. • "03": 1:3 million. • "01": 1:1 million.
ext	Extension of the file(s) to be downloaded. Formats available are "shp", "geojson", "svg", "json", "gdb". See Details .
recursive	[Deprecated] recursive is no longer supported; this function will never perform recursive extraction of child .zip files. This is the case of "shp.zip" inside the top-level .zip, that won't be unzipped.
...	Ignored. The argument id_giscoR ([Deprecated]) would be captured via ... and re-directed to id with a warning .

Details

Some arguments only apply to a specific value of "id". For example "resolution" would be ignored for values "communes", "lau", "urban_audit" and "postal_codes".

See years available in the corresponding functions:

- [gisco_get_countries\(\)](#).
- [gisco_get_coastal_lines\(\)](#).
- [gisco_get_communes\(\)](#).
- [gisco_get_lau\(\)](#).
- [gisco_get_nuts\(\)](#).
- [gisco_get_urban_audit\(\)](#).
- [gisco_get_postal_codes\(\)](#).

The usual extensions used across **giscoR** are "gpkg" and "shp", however other formats are already available on GISCO. Note that after performing a bulk download you may need to adjust the default "ext" value in the corresponding function to connect it with the downloaded files (see **Examples**).

Value

A (invisible) character vector with the full path of the files extracted. See **Examples**.

Source

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

See Also

Additional utils for downloading datasets: [gisco_get_unit](#)

Examples

```
tmp <- file.path(tempdir(), "testexample")

dest_files <- gisco_bulk_download(
  id = "countries", resolution = 60,
  year = 2024, ext = "geojson",
  cache_dir = tmp
)
# Read one
library(sf)
read_sf(dest_files[1]) |> head()

# Now we can connect the function with the downloaded data like:

connect <- gisco_get_countries(
  resolution = 60,
  year = 2024, ext = "geojson",
  cache_dir = tmp, verbose = TRUE
)

# Message shows that file is already cached ;)

# Clean
unlink(tmp, force = TRUE)
```

gisco_clear_cache	<i>Clear your R cache dir</i>
-------------------	---

Description

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

- Deletes the **giscoR** config directory (`tools::R_user_dir("giscoR", "config")`).
- Deletes the `cache_dir` directory.
- Deletes the values on stored on `Sys.getenv("GISCO_CACHE_DIR")`.

Usage

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

Arguments

<code>config</code>	if TRUE, will delete the configuration folder of giscoR .
<code>cached_data</code>	If this is set to TRUE, it will delete your <code>cache_dir</code> and all its content.
<code>verbose</code>	logical. If TRUE displays informational messages.

Details

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

Value

Invisible. This function is called for its side effects.

See Also

`tools::R_user_dir()`

Other cache utilities: `gisco_set_cache_dir()`

Examples

```
# Don't run this! It would modify your current state
## Not run:
my_cache <- gisco_detect_cache_dir()

# Set an example cache
ex <- file.path(tempdir(), "example", "cache")
gisco_set_cache_dir(ex, verbose = FALSE)

# Restore initial cache
gisco_clear_cache(verbose = TRUE)

gisco_set_cache_dir(my_cache)
identical(my_cache, gisco_detect_cache_dir())

## End(Not run)
```

<code>gisco_coastal_lines</code>	<i>Coastal lines 2016 sf object</i>
----------------------------------	---

Description

This object contains the coastal lines of the world.

Format

A [sf](#) object with POLYGON geometries, resolution: 1:20 million and [EPSG:4326](#).

Source

[COAS_RG_20M_2016_4326.gpkg](#) file.

See Also[gisco_get_coastal_lines\(\)](#)Other datasets: [gisco_countries_2024](#), [gisco_countrycode](#), [gisco_db](#), [gisco_nuts_2024](#)**Examples**

```
library(sf)
data("gisco_coastal_lines")
gisco_coastal_lines
```

`gisco_countries_2024` *Countries 2024 sf object*

Description

This object contains the administrative boundaries at country level of the world.

Format

A [sf](#) object with MULTIPOLYGON geometries, resolution: 1:20 million and [EPSG:4326](#). with 263 rows and 12 variables:

CNTR_ID Country ID as per Eurostat.

CNTR_NAME Official country name on local language.

NAME_ENGL Country name in English.

NAME_FREN Country name in French.

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

SVRG_UN Sovereign status as per United Nations.

CAPT Capitol city.

EU_STAT European Union member.

EFTA_STAT EFTA member.

CC_STAT EU candidate member.

NAME_GERM Country name in German.

geometry geometry field.

Source

[CNTR_RG_20M_2024_4326.gpkg](#) file.

See Also[gisco_get_countries\(\)](#)Other datasets: [gisco_coastal_lines](#), [gisco_countrycode](#), [gisco_db](#), [gisco_nuts_2024](#)

Examples

```
data("gisco_countries_2024")
head(gisco_countries_2024)
```

gisco_countrycode	<i>Database with different country code schemes and world regions</i>
-------------------	---

Description

A [tibble](#) 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 Standard](#)). This database has been extracted from the [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.

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.

Source

[countrycode::codelist](#) v1.6.1.

See Also

[gisco_get_countries\(\)](#), [countrycode::codelist](#).
See also the [Unicode Common Locale Data Repository](#).
Other datasets: [gisco_coastal_lines](#), [gisco_countries_2024](#), [gisco_db](#), [gisco_nuts_2024](#)

Examples

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

gisco_db	<i>Cached GISCO database</i>
----------	------------------------------

Description

Database with the list of files in the GISCO API as of 2025-12-03.

Format

A [tibble](#) with 9,609 rows.

Details

This database is used to redirect the corresponding functions to the right API endpoints.
This version of the database would be used in case there is any problem on update. Please use [gisco_get_cached_db\(\)](#) with `update_cache = TRUE` to update the corresponding API endpoints.

Source

GISCO API datasets.json.

See Also

Other datasets: [gisco_coastal_lines](#), [gisco_countries_2024](#), [gisco_countrycode](#), [gisco_nuts_2024](#)
Other database utils: [gisco_get_cached_db\(\)](#), [gisco_get_metadata\(\)](#)

Examples

```
data("gisco_db")
gisco_db |>
  dplyr::glimpse()
```

gisco_get_airports	<i>Airports dataset</i>
--------------------	-------------------------

Description

This dataset includes the location of over 11,800 Pan European airports and heliports. The airports are identified using the International Civil Aviation Organisation (ICAO) airport codes.

Usage

```
gisco_get_airports(  
  year = c(2013, 2006),  
  country = NULL,  
  cache_dir = NULL,  
  update_cache = FALSE,  
  verbose = FALSE  
)
```

Arguments

year	character string or number. Release year of the file. One of 2013, 2006.
country	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. See also countrycode::countrycode() .
cache_dir	character string. A path to a cache directory. See Caching strategies section in gisco_set_cache_dir() .
update_cache	logical. Should the cached file be refreshed?. Default is FALSE. When set to TRUE it would force a new download.
verbose	logical. If TRUE displays informational messages.

Details

Dataset includes objects in [EPSG:4326](#).

Value

A [sf](#) object.

Source

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

Copyright: <https://ec.europa.eu/eurostat/web/gisco/geodata>.

See Also

Other transport networks datasets: [gisco_get_ports\(\)](#)

Examples

```
airp <- gisco_get_airports(year = 2013)
coast <- giscoR::gisco_coastal_lines

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

  ggplot(coast) +
    geom_sf(fill = "grey10", color = "grey20") +
    geom_sf(
      data = airp, color = "#00F0FF",
      size = 0.2, alpha = 0.25
    ) +
    theme_void() +
    theme(
      plot.background = element_rect(fill = "black"),
      text = element_text(color = "white"),
      panel.grid = element_blank(),
      plot.title = element_text(face = "bold", hjust = 0.5),
      plot.subtitle = element_text(face = "italic", hjust = 0.5)
    ) +
    labs(
      title = "Airports in Europe", subtitle = "Year 2013",
      caption = "Source: Eurostat, Airports 2013 dataset."
    ) +
    # Center in Europe: EPSG 3035
    coord_sf(
      crs = 3035,
      xlim = c(2377294, 7453440),
      ylim = c(1313597, 5628510)
    )
}
```

gisco_get_cached_db	<i>Retrieve and update the GISCO database in use by</i> Rhrefhttps://CRAN.R-project.org/package=giscoR
---------------------	--

Description

Returns an optionally updates the cached database with the endpoints of the GISCO API.

Usage

```
gisco_get_cached_db(update_cache = FALSE)
```

Arguments

update_cache	logical. On TRUE the cached database would be rebuilt with the most updated information of the GISCO API.
--------------	---

Details

The cached database is stored in the **giscoR** cache path, see `gisco_set_cache_dir()` for details. The cached database would be used in subsequent **R** sessions.

On new GISCO data releases, you can access the new updated data simply by refreshing the cached database without waiting for a new version of **giscoR**.

A static database `gisco_db` is shipped with the package. This database would be used in case there is any problem on update.

Value

A `tibble`.

Source

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

See Also

Other database utils: `gisco_db`, `gisco_get_metadata()`

Examples

```
gisco_get_cached_db() |>
  dplyr::glimpse()
```

<code>gisco_get_census</code>	<i>Census dataset</i>
-------------------------------	-----------------------

Description

This dataset shows pan European communal boundaries depicting the situation at the corresponding Census.

Usage

```
gisco_get_census(  
  year = 2011,  
  cache_dir = NULL,  
  update_cache = FALSE,  
  verbose = FALSE,  
  spatialtype = c("RG", "PT")  
)
```

Arguments

year	character string or number. Release year of the file. Currently only "2011" is provided.
cache_dir	character string. A path to a cache directory. See Caching strategies section in gisco_set_cache_dir() .
update_cache	logical. Should the cached file be refreshed?. Default is FALSE. When set to TRUE it would force a new download.
verbose	logical. If TRUE displays informational messages.
spatialtype	Type of geometry to be returned: • "PT": Points - POINT object. • "RG": Regions - MULTIPOLYGON/POLYGON object.

Value

A [sf](#) object.

Source

<https://ec.europa.eu/eurostat/web/gisco/geodata/statistical-units/census>.

Copyright: <https://ec.europa.eu/eurostat/web/gisco/geodata/statistical-units>.

See Also

See [gisco_id_api_census_grid\(\)](#) to download via GISCO ID service API.

Other statistical units datasets: [gisco_get_coastal_lines\(\)](#), [gisco_get_lau\(\)](#), [gisco_get_nuts\(\)](#), [gisco_get_urban_audit\(\)](#)

Examples

```
library(sf)

pts <- gisco_get_census(spatialtype = "PT")

pts
```

gisco_get_coastal_lines

Coastal lines dataset

Description

Downloads worldwide coastlines.

Usage

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

Arguments

year	character string or number. Release year of the file. One of "2016", "2013", "2010", "2006".
epsg	character string or number. Projection of the map: 4-digit EPSG code . One of: "4326": WGS84. "3035": ETRS89 / ETRS-LAEA. "3857": Pseudo-Mercator.
cache	logical. Whether to do caching. Default is TRUE. See Caching strategies section in gisco_set_cache_dir() .
update_cache	logical. Should the cached file be refreshed?. Default is FALSE. When set to TRUE it would force a new download.
cache_dir	character string. A path to a cache directory. See Caching strategies section in gisco_set_cache_dir() .
verbose	logical. If TRUE displays informational messages.
resolution	character string or number. Resolution of the geospatial data. One of: "60": 1:60 million. "20": 1:20 million. "10": 1:10 million. "03": 1:3 million. "01": 1:1 million.
ext	character. Extension of the file (default "gpkg"). One of "shp", "gpkg", "geojson".

Value

A [sf](#) object.

Note

Please check the download and usage provisions on [gisco_attributions\(\)](#).

Source

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

Copyright: <https://ec.europa.eu/eurostat/web/gisco/geodata/statistical-units>.

See Also

[gisco_coastal_lines](#).

See [gisco_bulk_download\(\)](#) to perform a bulk download of datasets.

Other statistical units datasets: [gisco_get_census\(\)](#), [gisco_get_lau\(\)](#), [gisco_get_nuts\(\)](#), [gisco_get_urban_audit\(\)](#)

Examples

```
coast <- gisco_get_coastal_lines()

library(ggplot2)

ggplot(coast) +
  geom_sf(color = "#1278AB", fill = "#FDFBEA") +
  # Zoom on Mediterranean Sea
  coord_sf(
    xlim = c(-4, 35),
    ylim = c(31, 45)
  ) +
  theme_minimal() +
  theme(
    panel.background = element_rect(fill = "#C7E7FB", color = NA),
    panel.border = element_rect(colour = "black", fill = NA)
  )
```

<code>gisco_get_communes</code>	<i>Communes dataset</i>
---------------------------------	-------------------------

Description

This dataset shows pan European administrative boundaries down to commune level. Communes are equivalent to Local Administrative Units, see [gisco_get_lau\(\)](#).

Usage

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

Arguments

year	character string or number. Release year of the file. One of "2016", "2013", "2010", "2008", "2006", "2004", "2001".
epsg	character string or number. Projection of the map: 4-digit EPSG code . One of: • "4326": WGS84. • "3035": ETRS89 / ETRS-LAEA. • "3857": Pseudo-Mercator.
cache	[Deprecated] . These functions always caches the result due to the size. See See Caching strategies section in gisco_set_cache_dir() .
update_cache	logical. Should the cached file be refreshed?. Default is FALSE. When set to TRUE it would force a new download.
cache_dir	character string. A path to a cache directory. See Caching strategies section in gisco_set_cache_dir() .
verbose	logical. If TRUE displays informational messages.
spatialtype	character string. Type of geometry to be returned. Options available are: • "RG": Regions - MULTIPOLYGON/POLYGON object. • "LB": Labels - POINT object. • "BN": Boundaries - LINESTRING object. Note that argument country would be only applied when spatialtype is "RG" or "LB".
country	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. See also countrycode::countrycode() .
ext	character. Extension of the file (default "shp"). One of "shp", "gpkg", "geojson".

Details

The Nomenclature of Territorial Units for Statistics (NUTS) and the LAU nomenclature are hierarchical classifications of statistical regions that together subdivide the EU economic territory into regions of five different levels (NUTS 1, 2 and 3 and LAU , respectively, moving from larger to smaller territorial units).

The dataset is based on EuroBoundaryMap from [EuroGeographics](#). Geographical extent covers the European Union 28, EFTA countries, and candidate countries. The scale of the dataset is 1:100 000.

The LAU classification is not covered by any legislative act.

Value

A [sf](#) object.

Note

Please check the download and usage provisions on [gisco_attributions\(\)](#).

Source

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

Copyright: <https://ec.europa.eu/eurostat/web/gisco/geodata/administrative-units>.

See Also

[gisco_get_lau\(\)](#).

See [gisco_bulk_download\(\)](#) to perform a bulk download of datasets.

Other administrative units datasets: [gisco_get_countries\(\)](#), [gisco_get_postal_codes\(\)](#)

Examples

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

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

  ggplot(ire_comm) +
    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_countries` *Countries dataset*

Description

This dataset contains the administrative boundaries at country level of the world. This dataset consists of 2 feature classes (regions, boundaries) per scale level and there are 5 different scale levels (1M, 3M, 10M, 20M and 60M).

Please note that this function gets data from the aggregated GISCO country file. If you prefer to download individual country files, please use [gisco_get_unit_country\(\)](#).

Usage

```
gisco_get_countries(
  year = 2024,
  epsg = 4326,
  cache = TRUE,
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  resolution = 20,
  spatialtype = "RG",
  country = NULL,
  region = NULL,
  ext = "gpkg"
)
```

Arguments

<code>year</code>	character string or number. Release year of the file. One of "2024", "2020", "2016", "2013", "2010", "2006", "2001".
<code>epsg</code>	character string or number. Projection of the map: 4-digit EPSG code . One of: "4326": WGS84. "3035": ETRS89 / ETRS-LAEA. "3857": Pseudo-Mercator.
<code>cache</code>	logical. Whether to do caching. Default is TRUE. See Caching strategies section in gisco_set_cache_dir() .
<code>update_cache</code>	logical. Should the cached file be refreshed?. Default is FALSE. When set to TRUE it would force a new download.
<code>cache_dir</code>	character string. A path to a cache directory. See Caching strategies section in gisco_set_cache_dir() .
<code>verbose</code>	logical. If TRUE displays informational messages.
<code>resolution</code>	character string or number. Resolution of the geospatial data. One of: "60": 1:60 million.

	• "20": 1:20 million. • "10": 1:10 million. • "03": 1:3 million. • "01": 1:1 million.
spatialtype	character string. Type of geometry to be returned. Options available are: • "RG": Regions - MULTIPOLYGON/POLYGON object. • "LB": Labels - POINT object. • "BN": Boundaries - LINESTRING object. • "COASTL": coastlines - LINESTRING object. • "INLAND": inland boundaries - LINESTRING object. Note that arguments country and region would be only applied when spatialtype is "RG" or "LB".
country	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. 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 World Regions and gisco_countrycode .
ext	character. Extension of the file (default "gpkg"). One of "shp", "gpkg", "geojson".

Value

A [sf](#) object.

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.

Note

Please check the download and usage provisions on [gisco_attributions\(\)](#).

Source

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

Copyright: <https://ec.europa.eu/eurostat/web/gisco/geodata/administrative-units>.

See Also

[gisco_countrycode](#), [gisco_countries_2024](#), [gisco_get_metadata\(\)](#), [countrycode::countrycode\(\)](#).

See [gisco_bulk_download\(\)](#) to perform a bulk download of datasets.

See [gisco_get_unit_country\(\)](#) to download single files.

See [gisco_id_api_country\(\)](#) to download via GISCO ID service API.

Other administrative units datasets: [gisco_get_communes\(\)](#), [gisco_get_postal_codes\(\)](#)

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()

```

<code>gisco_get_education</code>	<i>Education services in Europe</i>
----------------------------------	-------------------------------------

Description

This dataset is an integration of Member States official data on the location of education services. Additional information on these services is included when available (see **Details**).

Usage

```

gisco_get_education(
  year = c(2023, 2020),
  cache = TRUE,
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  country = NULL
)

```

Arguments

<code>year</code>	character string or number. Release year of the file. One of 2023, 2020.
<code>cache</code>	logical. Whether to do caching. Default is TRUE. See Caching strategies section in gisco_set_cache_dir() .
<code>update_cache</code>	logical. Should the cached file be refreshed?. Default is FALSE. When set to TRUE it would force a new download.
<code>cache_dir</code>	character string. A path to a cache directory. See Caching strategies section in gisco_set_cache_dir() .
<code>verbose</code>	logical. If TRUE displays informational messages.
<code>country</code>	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. See also countrycode::countrycode() .

Details

Files are distributed [EPSG:4326](#).

Brief description of each attribute:

Attribute	Description
id	The education service identifier. This identifier is based on national identification codes, if it exists.
name	The name of the education institution.
site_name	The name of the specific site or branch of an education institution.
lat	Latitude (WGS 84).
lon	Longitude (WGS 84).
street	Street name.
house_number	House number.
postcode	Postcode.
address	Address information when the different components of the address are not separated in the source.
city	City name (sometimes refers to a region or municipality).
cntr_id	Country code (2 letters, ISO 3166-1 alpha-2).
levels	Education levels represented by a single integer or range (ISCED 2011).
max_students	Measure of capacity by maximum number of students.
enrollment	Measure of capacity by number of enrolled students.
fields	Academic disciplines the institution specializes in (ISCED-F 2013).
facility_type	Type of institution in reference to ownership and operation e.g. Catholic, International, etc.
public_private	The public or private status of the education service.
tel	Telephone number.
email	Email address.
url	URL link to the institution's website.
ref_date	The reference date (DD/MM/YYYY) the data refers to. The dataset represents the reality as it was at this date.
geo_qual	Geolocation quality indicator: 1=Good, 2=Medium, 3=Low, 4=From source, -1=Unknown, -2=Not geocoded.
comments	Some additional information on the education service.

Value

A [sf](#) object.

Source

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

There are no specific download rules for the datasets shown below. However, please refer to [the general copyright notice](#) and licence provisions, which must be complied with. Permission to download and use these data is subject to these rules being accepted.

The data are extracted from official national registers. They may contain inconsistencies, inaccuracies and gaps, due to the heterogeneity of the input national data.

See Also

Other basic services datasets: [gisco_get_healthcare\(\)](#)

Examples

```

edu_austria <- gisco_get_education(country = "Austria", year = 2023)

# Plot if downloaded
if (!is.null(edu_austria)) {
  austria_nuts3 <- gisco_get_nuts(country = "Austria", nuts_level = 3)

  library(ggplot2)
  ggplot(austria_nuts3) +
    geom_sf(fill = "grey10", color = "grey60") +
    geom_sf(
      data = edu_austria, aes(color = rev(public_private)),
      alpha = 0.25
    ) +
    theme_void() +
    theme(
      plot.background = element_rect(fill = "black"),
      text = element_text(color = "white"),
      panel.grid = element_blank(),
      plot.title = element_text(face = "bold", hjust = 0.5),
      plot.subtitle = element_text(face = "italic", hjust = 0.5)
    ) +
    labs(
      title = "Education", subtitle = "Austria (2023)",
      caption = "Source: Eurostat, Education 2023 dataset.",
      color = "Type"
    ) +
    coord_sf(crs = 3035)
}

```

gisco_get_grid

Grid dataset

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 = c(100, 50, 20, 10, 5, 2, 1),
  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	character string. A path to a cache directory. See Caching strategies section in gisco_set_cache_dir() .
update_cache	logical. Should the cached file be refreshed?. Default is FALSE. When set to TRUE it would force a new download.
verbose	logical. If TRUE displays informational messages.

Details

Files are distributed on [EPSG:3035](#).

The file sizes range is from 428Kb (resolution = 100) to 1.7Gb resolution = 1.

Value

A [sf](#) object.

Source

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

There are specific downloading provisions, please see <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",
    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")
  )
}

```

Description

The dataset contains information on main healthcare services considered to be 'hospitals' by Member States. The definition varies slightly from country to country, but roughly includes the following:

"Hospitals' comprises licensed establishments primarily engaged in providing medical, diagnostic, and treatment services that include physician, nursing, and other health services to in-patients and the specialised accommodation services required by inpatients."

Usage

```
gisco_get_healthcare(
  year = c(2023, 2020),
  cache = TRUE,
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  country = NULL
)
```

Arguments

year	character string or number. Release year of the file. One of 2023, 2020.
cache	logical. Whether to do caching. Default is TRUE. See Caching strategies section in gisco_set_cache_dir() .
update_cache	logical. Should the cached file be refreshed?. Default is FALSE. When set to TRUE it would force a new download.
cache_dir	character string. A path to a cache directory. See Caching strategies section in gisco_set_cache_dir() .
verbose	logical. If TRUE displays informational messages.
country	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. See also countrycode::countrycode() .

Details

Files are distributed [EPSG:4326](#).

Brief description of each attribute:

Attribute	Description
id	The healthcare service identifier. This identifier is based on national identification codes, if it exists.
hospital_name	The name of the healthcare institution.
site_name	The name of the specific site or branch of a healthcare institution.
lat	Latitude (WGS 84).
lon	Longitude (WGS 84).
street	Street name.
house_number	House number.

postcode	Postcode.
address	Address information when the different components of the address are not separated in the source.
city	City name (sometimes refers to a region or municipality).
cntr_id	Country code (2 letters, ISO 3166-1 alpha-2).
emergency	'yes/no' for whether the healthcare site provides emergency medical services.
cap_beds	Measure of capacity by number of beds (most common).
cap_prac	Measure of capacity by number of practitioners.
cap_rooms	Measure of capacity by number of rooms.
facility_type	Type of healthcare service (e.g., psychiatric hospital), based on national classification.
public_private	'public/private' status of the healthcare service.
list_specs	List of specialties recognized in the EU and EEA according to the 2005 EU Directive (Annex V).
tel	Telephone number.
email	Email address.
url	URL link to the institution's website.
ref_date	The date (DD/MM/YYYY) the data refers to (reference date).
pub_date	The publication date of the dataset by Eurostat (DD/MM/YYYY).
geo_qual	Geolocation quality indicator: 1=Good, 2=Medium, 3=Low, 4=From source, -1=Unknown, -2=Not geocoded
comments	Additional information on the healthcare service.

Value

A `sf` object.

Source

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

There are no specific download rules for the datasets shown below. However, please refer to [the general copyright notice](#) and licence provisions, which must be complied with. Permission to download and use these data is subject to these rules being accepted.

The data are extracted from official national registers. They may contain inconsistencies, inaccuracies and gaps, due to the heterogeneity of the input national data.

See Also

Other basic services datasets: `gisco_get_education()`

Examples

```
health_benelux <- gisco_get_healthcare(
  country = c("BE", "NL", "LU"),
  year = 2023
)

# Plot if downloaded
if (!is.null(health_benelux)) {
  benelux <- gisco_get_countries(country = c("BE", "NL", "LU"))

  library(ggplot2)
```

```

ggplot(benelux) +
  geom_sf(fill = "grey10", color = "grey20") +
  geom_sf(
    data = health_benelux, color = "red",
    size = 0.2, alpha = 0.25
  ) +
  theme_void() +
  theme(
    plot.background = element_rect(fill = "black"),
    text = element_text(color = "white"),
    panel.grid = element_blank(),
    plot.title = element_text(face = "bold", hjust = 0.5),
    plot.subtitle = element_text(face = "italic", hjust = 0.5)
  ) +
  labs(
    title = "Healthcare services", subtitle = "Benelux (2023)",
    caption = "Source: Eurostat, Healthcare 2023 dataset."
  ) +
  coord_sf(crs = 3035)
}

```

gisco_get_lau

Local Administrative Units (LAU) dataset

Description

This dataset shows pan European administrative boundaries down to commune level. Local Administrative units are equivalent to Communes, see [gisco_get_communes\(\)](#).

Usage

```

gisco_get_lau(
  year = 2024,
  epsg = 4326,
  cache = deprecated(),
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  country = NULL,
  gisco_id = NULL,
  ext = "gpkg"
)

```

Arguments

year	character string or number. Release year of the file. One of "2024", "2023", "2022", "2021", "2020", "2019", "2018", "2017", "2016", "2015", "2014", "2013", "2012", "2011".
------	--

epsg	character string or number. Projection of the map: 4-digit EPSG code . One of: • "4326": WGS84. • "3035": ETRS89 / ETRS-LAEA. • "3857": Pseudo-Mercator.
cache	[Deprecated] . These functions always caches the result due to the size. See See Caching strategies section in gisco_set_cache_dir() .
update_cache	logical. Should the cached file be refreshed?. Default is FALSE. When set to TRUE it would force a new download.
cache_dir	character string. A path to a cache directory. See Caching strategies section in gisco_set_cache_dir() .
verbose	logical. If TRUE displays informational messages.
country	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. See also countrycode::countrycode() .
gisco_id	Optional. A character vector of GISCO_ID LAU values.
ext	character. Extension of the file (default "gpkg"). One of "shp", "gpkg", "geojson".

Details

The Nomenclature of Territorial Units for Statistics (NUTS) and the LAU nomenclature are hierarchical classifications of statistical regions that together subdivide the EU economic territory into regions of five different levels (NUTS 1, 2 and 3 and LAU , respectively, moving from larger to smaller territorial units).

The LAU classification is not covered by any legislative act. Geographical extent covers the European Union, EFTA countries, and candidate countries. The scale of the dataset is 1:100 000.

The data contains the National Statistical agency LAU code which can be joined to LAU lists as well as a field GISCO_ID which is a unique identifier consisting of the Country code and LAU code.

Total resident population figures (31 December) have also been added ins some versions based on the associated LAU lists

Value

A [sf](#) object.

Note

Please check the download and usage provisions on [gisco_attributions\(\)](#).

Source

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

Copyright: <https://ec.europa.eu/eurostat/web/gisco/geodata/statistical-units>.

See Also

[gisco_get_communes\(\)](#).

See [gisco_bulk_download\(\)](#) to perform a bulk download of datasets.

See [gisco_id_api_lau\(\)](#) to download via GISCO ID service API.

Other statistical units datasets: [gisco_get_census\(\)](#), [gisco_get_coastal_lines\(\)](#), [gisco_get_nuts\(\)](#), [gisco_get_urban_audit\(\)](#)

Examples

```
## Not run:

lu_lau <- gisco_get_lau(year = 2024, country = "Luxembourg")

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

  ggplot(lu_lau) +
    geom_sf(aes(fill = POP_DENS_2024)) +
    labs(
      title = "Population Density in Luxembourg",
      subtitle = "Year 2024",
      caption = gisco_attributions()
    ) +
    scale_fill_viridis_b(
      option = "cividis",
      label = \"(x) prettyNum(x, big.mark = \",\")
    ) +
    theme_void() +
    labs(fill = "pop/km2")
}

## End(Not run)
```

<code>gisco_get_metadata</code>	<i>Get metadata</i>
---------------------------------	---------------------

Description

Get a table with the names and ids of administrative of statistical units.

Usage

```
gisco_get_metadata(
  id = c("nuts", "countries", "urban_audit"),
  year = 2024,
  verbose = FALSE
)
```

Arguments

id	character string. Select the unit type to be downloaded. Accepted values are "nuts", "countries" or "urban_audit".
year	character string or number. Release year of the metadata.
verbose	logical. If TRUE displays informational messages.

Value

A [tibble](#).

Source

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

See Also

[gisco_get_nuts\(\)](#), [gisco_get_countries\(\)](#), [gisco_get_urban_audit\(\)](#).

Other database utils: [gisco_db](#), [gisco_get_cached_db\(\)](#)

Examples

```
cities <- gisco_get_metadata(id = "urban_audit", year = 2020)

cities
```

gisco_get_nuts

Territorial units for statistics (NUTS) dataset

Description

The GISCO statistical unit dataset represents the NUTS (nomenclature of territorial units for statistics) and statistical regions by means of multipart polygon, polyline and point topology. The NUTS geographical information is completed by attribute tables and a set of cartographic help lines to better visualise multipart polygonal regions.

The NUTS are a hierarchical system divided into 3 levels:

- NUTS 1: major socio-economic regions
- NUTS 2: basic regions for the application of regional policies
- NUTS 3: small regions for specific diagnoses.

Also, there is a NUTS 0 level, which usually corresponds to the national boundaries.

Please note that this function gets data from the aggregated GISCO NUTS file, that contains data of all the countries at the requested NUTS level(s). If you prefer to download individual NUTS files, please use [gisco_get_unit_nuts\(\)](#).

Usage

```
gisco_get_nuts(
  year = 2024,
  epsg = 4326,
  cache = TRUE,
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  resolution = 20,
  spatialtype = "RG",
  country = NULL,
  nuts_id = NULL,
  nuts_level = c("all", "0", "1", "2", "3"),
  ext = "gpkg"
)
```

Arguments

year	character string or number. Release year of the file. One of "2024", "2021", "2016", "2013", "2010", "2006", "2003".
epsg	character string or number. Projection of the map: 4-digit EPSG code . One of: • "4326": WGS84. • "3035": ETRS89 / ETRS-LAEA. • "3857": Pseudo-Mercator.
cache	logical. Whether to do caching. Default is TRUE. See Caching strategies section in gisco_set_cache_dir() .
update_cache	logical. Should the cached file be refreshed?. Default is FALSE. When set to TRUE it would force a new download.
cache_dir	character string. A path to a cache directory. See Caching strategies section in gisco_set_cache_dir() .
verbose	logical. If TRUE displays informational messages.
resolution	character string or number. Resolution of the geospatial data. One of: • "60": 1:60 million. • "20": 1:20 million. • "10": 1:10 million. • "03": 1:3 million. • "01": 1:1 million.
spatialtype	character string. Type of geometry to be returned. Options available are: • "RG": Regions - MULTIPOLYGON/POLYGON object. • "LB": Labels - POINT object. • "BN": Boundaries - LINESTRING object.

Note that arguments country, nuts_level and nuts_id would be only applied when spatialtype is "RG" or "LB".

country	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. See also countrycode::countrycode() .
nuts_id	Optional. A character vector of NUTS IDs.
nuts_level	character string. NUTS level. One of 0, 1, 2, 3 or all for all levels.
ext	character. Extension of the file (default "gpkg"). One of "shp", "gpkg", "geojson".

Details

The NUTS nomenclature is a hierarchical classification of statistical regions and subdivides the EU economic territory into regions of three different levels (NUTS 1, 2 and 3, moving respectively from larger to smaller territorial units). NUTS 1 is the most aggregated level. An additional Country level (NUTS 0) is also available for countries where the nation at statistical level does not coincide with the administrative boundaries.

The NUTS classification has been officially established through Commission Delegated Regulation 2019/1755. A non-official NUTS-like classification has been defined for the EFTA countries, candidate countries and potential candidates based on a bilateral agreement between Eurostat and the respective statistical agencies.

An introduction to the NUTS classification is available here: <https://ec.europa.eu/eurostat/web/nuts/overview>.

Value

A [sf](#) object.

Note

Please check the download and usage provisions on [gisco_attributions\(\)](#).

Source

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

Copyright: <https://ec.europa.eu/eurostat/web/gisco/geodata/administrative-units>.

See Also

[gisco_nuts_2024](#), [eurostat::get_eurostat_geospatial\(\)](#).

See [gisco_bulk_download\(\)](#) to perform a bulk download of datasets.

See [gisco_get_unit_nuts\(\)](#) to download single files.

See [gisco_id_api_nuts\(\)](#) to download via GISCO ID service API.

Other statistical units datasets: [gisco_get_census\(\)](#), [gisco_get_coastal_lines\(\)](#), [gisco_get_lau\(\)](#), [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_ports

Ports dataset

Description

This dataset includes the location of over 2,440 Pan European ports. The ports are identified following the UN LOCODE list.

Usage

```

gisco_get_ports(
  year = c(2013, 2009),
  country = NULL,
  cache_dir = NULL,
  update_cache = FALSE,
  verbose = FALSE
)

```

Arguments

year	character string or number. Release year of the file. One of 2013, 2009.
country	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. See also countrycode::countrycode() .
cache_dir	character string. A path to a cache directory. See Caching strategies section in gisco_set_cache_dir() .
update_cache	logical. Should the cached file be refreshed?. Default is FALSE. When set to TRUE it would force a new download.
verbose	logical. If TRUE displays informational messages.

Details

Dataset includes objects in [EPSG:4326](#).

[gisco_get_ports\(\)](#) adds a new field CNTR_ISO2 to the original data identifying the country of the port.

Value

A [sf](#) object.

Source

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

Copyright: <https://ec.europa.eu/eurostat/web/gisco/geodata>.

See Also

Other transport networks datasets: [gisco_get_airports\(\)](#)

Examples

```
library(sf)

ports <- gisco_get_ports(2013)
coast <- giscoR::gisco_coastal_lines

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

  ggplot(coast) +
    geom_sf(fill = "grey10", color = "grey20") +
    geom_sf(
      data = ports, color = "#6bb857",
      size = 0.2, alpha = 0.25
    ) +
    theme_void() +
    theme(
      plot.background = element_rect(fill = "black"),
```

```

    text = element_text(color = "white"),
    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 = "Source: Eurostat, Ports 2013 dataset."
  ) +
  coord_sf(crs = "ESRI:54030")
}

```

gisco_get_postal_codes

Postal codes dataset

Description

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.

Usage

```

gisco_get_postal_codes(
  year = 2024,
  country = NULL,
  cache_dir = NULL,
  update_cache = FALSE,
  verbose = FALSE,
  ext = "gpkg"
)

```

Arguments

year	character string or number. Release year of the file. One of "2024", "2020".
country	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. See also countrycode::countrycode() .
cache_dir	character string. A path to a cache directory. See Caching strategies section in gisco_set_cache_dir() .
update_cache	logical. Should the cached file be refreshed?. Default is FALSE. When set to TRUE it would force a new download.
verbose	logical. If TRUE displays informational messages.
ext	character. Extension of the file (default "gpkg"). One of "shp", "gpkg", "geojson".

Value

A `sf` object.

Copyright

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

© European Union - GISCO, 2024, postal code point dataset, Licence CC-BY-SA 4.0.

Note

Please check the download and usage provisions on [gisco_attributions\(\)](#).

Source

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

Copyright: <https://ec.europa.eu/eurostat/web/gisco/geodata/administrative-units>.

See Also

See [gisco_bulk_download\(\)](#) to perform a bulk download of datasets.

Other administrative units datasets: [gisco_get_communes\(\)](#), [gisco_get_countries\(\)](#)

Examples

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

pc_bel <- gisco_get_postal_codes(country = "BE")

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

  ggplot(pc_bel) +
    geom_sf(color = "gold") +
    theme_bw() +
    labs(
      title = "Postcodes of Belgium",
      subtitle = "2024",
      caption = paste("\u00a9 European Union - GISCO, 2024,",
        "postal code point dataset",
        "Licence CC-BY-SA 4.0",
        sep = "\n"
      )
    )
}

## End(Not run)
```

gisco_get_unit*GISCO API single download*

Description

Download datasets of single spatial units from GISCO to the `cache_dir`.

Unlike `gisco_get_countries()`, `gisco_get_nuts()` or `gisco_get_urban_audit()` (that download a full dataset and applies filters), these functions download a single per unit, reducing the time of downloading and reading into your **R** session.

Usage

```
gisco_get_unit_country(  
  unit = "ES",  
  year = 2024,  
  epsg = c(4326, 3857, 3035),  
  cache = TRUE,  
  update_cache = FALSE,  
  cache_dir = NULL,  
  verbose = FALSE,  
  resolution = c(1, 3, 10, 20, 60),  
  spatialtype = c("RG", "LB")  
)
```

```
gisco_get_unit_nuts(  
  unit = "ES416",  
  year = 2024,  
  epsg = c(4326, 3857, 3035),  
  cache = TRUE,  
  update_cache = FALSE,  
  cache_dir = NULL,  
  verbose = FALSE,  
  resolution = c(1, 3, 10, 20, 60),  
  spatialtype = c("RG", "LB")  
)
```

```
gisco_get_unit_urban_audit(  
  unit = "ES001F",  
  year = 2021,  
  epsg = c(4326, 3857, 3035),  
  cache = TRUE,  
  update_cache = FALSE,  
  cache_dir = NULL,  
  verbose = FALSE,  
  spatialtype = c("RG", "LB")  
)
```

Arguments

unit	character vector of unit ids to be downloaded. See Details .
year	character string or number. Release year of the file.
epsg	character string or number. Projection of the map: 4-digit EPSG code . One of: • "4326": WGS84. • "3035": ETRS89 / ETRS-LAEA. • "3857": Pseudo-Mercator.
cache	logical. Whether to do caching. Default is TRUE. See Caching strategies section in gisco_set_cache_dir() .
update_cache	logical. Should the cached file be refreshed?. Default is FALSE. When set to TRUE it would force a new download.
cache_dir	character string. A path to a cache directory. See Caching strategies section in gisco_set_cache_dir() .
verbose	logical. If TRUE displays informational messages.
resolution	character string or number. Resolution of the geospatial data. One of: • "60": 1:60 million. • "20": 1:20 million. • "10": 1:10 million. • "03": 1:3 million. • "01": 1:1 million.
spatialtype	character string. Type of geometry to be returned. Options available are: • "RG": Regions - MULTIPOLYGON/POLYGON object. • "LB": Labels - POINT object.

Details

Check the available unit ids with the required combination of arguments with [gisco_get_metadata\(\)](#).

Value

A [sf](#) object.

Note

Please check the download and usage provisions on [gisco_attributions\(\)](#).

Source

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

All the source files are .geojson files.

See Also

[gisco_get_metadata\(\)](#), [gisco_get_countries\(\)](#), [gisco_get_nuts\(\)](#), [gisco_get_urban_audit\(\)](#).

See [gisco_id_api](#) to download via GISCO ID service API.

Additional utils for downloading datasets: [gisco_bulk_download\(\)](#)

Examples

```
# Get metadata
cities <- gisco_get_metadata("urban_audit", 2020)

# Valencia, Spain
valencia <- cities[grep("Valencia", cities$URAU_NAME), ]
valencia
library(dplyr)
# Now get the sf objects and order by AREA_SQM
valencia_sf <- gisco_get_unit_urban_audit(
  unit = valencia$URAU_CODE,
  year = "2020",
) |>
  arrange(desc(AREA_SQM))
# Plot
library(ggplot2)

ggplot(valencia_sf) +
  geom_sf(aes(fill = URAU_CATG)) +
  scale_fill_viridis_d() +
  labs(
    title = "Valencia",
    subtitle = "Urban Audit 2020",
    fill = "Category"
  )
```

gisco_get_units

Get geospatial units data from GISCO API

Description**[Deprecated]**

This function is deprecated. Use:

- [gisco_get_metadata\(\)](#) (equivalent to mode = "df").
- [?gisco_get_unit](#) functions (equivalent to mode = "sf")

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.
mode	Controls the output of the function. Possible values are "sf" or "df". See Value .
year	character string or number. Release year of the file.
epsg	character string or number. Projection of the map: 4-digit EPSG code . One of: • "4326": WGS84. • "3035": ETRS89 / ETRS-LAEA. • "3857": Pseudo-Mercator.
cache	logical. Whether to do caching. Default is TRUE. See Caching strategies section in gisco_set_cache_dir() .
update_cache	logical. Should the cached file be refreshed?. Default is FALSE. When set to TRUE it would force a new download.
cache_dir	character string. A path to a cache directory. See Caching strategies section in gisco_set_cache_dir() .
verbose	logical. If TRUE displays informational messages.
resolution	character string or number. Resolution of the geospatial data. One of: • "60": 1:60 million. • "20": 1:20 million. • "10": 1:10 million. • "03": 1:3 million. • "01": 1:1 million.
spatialtype	character string. Type of geometry to be returned. Options available are: • "RG": Regions - MULTIPOLYGON/POLYGON object. • "LB": Labels - POINT object.

Value

A [sf](#) object on mode = "sf" or a [tibble](#) on mode = "df".

Note

Please check the download and usage provisions on [gisco_attributions\(\)](#).

Source

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

All the source files are .geojson files.

See Also

[gisco_get_metadata\(\)](#), [?gisco_get_unit](#) functions.

Examples

```
# mode df
gisco_get_units("nuts", mode = "df", year = 2016)
# ->
gisco_get_metadata("nuts", year = 2016)

# mode sf for NUTS
gisco_get_units("nuts", unit = "ES111", mode = "sf", year = 2016)
# ->
gisco_get_unit_nuts(unit = "ES111", year = 2016)
```

`gisco_get_urban_audit` *Urban Audit dataset*

Description

The dataset contains the boundaries of cities ("CITIES"), greater cities ("GREATER_CITIES") and functional urban areas ("FUA") as defined according to the EC-OECD city definition. This is used for the Eurostat Urban Audit data collection.

Please note that this function gets data from the aggregated GISCO Urban Audit file. If you prefer to download individual urban audit files, please use [gisco_get_unit_urban_audit\(\)](#).

Usage

```
gisco_get_urban_audit(
  year = 2021,
  epsg = 4326,
  cache = TRUE,
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  spatialtype = c("RG", "LB"),
  country = NULL,
  level = c("all", "CITIES", "FUA", "GREATER_CITIES", "CITY", "KERN", "LUZ"),
  ext = "gpkg"
)
```

Arguments

year	character string or number. Release year of the file. One of "2021", "2020", "2018", "2014", "2004", "2001".
epsg	character string or number. Projection of the map: 4-digit EPSG code . One of: • "4326": WGS84. • "3035": ETRS89 / ETRS-LAEA. • "3857": Pseudo-Mercator.
cache	logical. Whether to do caching. Default is TRUE. See Caching strategies section in gisco_set_cache_dir() .
update_cache	logical. Should the cached file be refreshed?. Default is FALSE. When set to TRUE it would force a new download.
cache_dir	character string. A path to a cache directory. See Caching strategies section in gisco_set_cache_dir() .
verbose	logical. If TRUE displays informational messages.
spatialtype	character string. Type of geometry to be returned. Options available are: • "RG": Regions - MULTIPOLYGON/POLYGON object. • "LB": Labels - POINT object.
country	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. See also countrycode::countrycode() .
level	character string. Level of Urban Audit. Possible values "all" (the default), that would download the full dataset or "CITIES", "FUA", and (for versions prior to year = 2020) "GREATER_CITIES", "CITY", "KERN" or "LUZ".
ext	character. Extension of the file (default "gpkg"). One of "shp", "gpkg", "geojson".

Details

See more in [Eurostat - Statistics Explained](#).

The cities are defined at several conceptual levels:

- The core city ("CITIES"), using an administrative definition.
- The Functional Urban Area/Large Urban Zone ("FUA"), approximating the functional urban region. The coverage is the EU plus Iceland, Norway and Switzerland . The dataset includes polygon features, point features and a related attribute table which can be joined on the URAU code field.

The "URAU_CATG" field defines the Urban Audit category:

- "C" = City.
- "F" = Functional Urban Area Service Type.

Value

A [sf](#) object.

Note

Please check the download and usage provisions on [gisco_attributions\(\)](#).

Source

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

Copyright: <https://ec.europa.eu/eurostat/web/gisco/geodata/administrative-units>.

See Also

See [gisco_bulk_download\(\)](#) to perform a bulk download of datasets.

See [gisco_get_unit_urban_audit\(\)](#) to download single files.

Other statistical units datasets: [gisco_get_census\(\)](#), [gisco_get_coastal_lines\(\)](#), [gisco_get_lau\(\)](#), [gisco_get_nuts\(\)](#)

Examples

```
cities <- gisco_get_urban_audit(year = 2021, level = "CITIES")

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

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

gisco_id_api

GISCO ID service API

Description

Functions to interact with the **GISCO ID service API**, which returns attributes and, optionally, geometry for different datasets at specified longitude and latitude coordinates.

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

Usage

```
gisco_id_api_geonames(  
  x = NULL,  
  y = NULL,  
  xmin = NULL,  
  ymin = NULL,  
  xmax = NULL,  
  ymax = NULL,  
  verbose = FALSE  
)  
  
gisco_id_api_nuts(  
  x = NULL,  
  y = NULL,  
  year = 2024,  
  epsg = c(4326, 4258, 3035),  
  verbose = FALSE,  
  nuts_id = NULL,  
  nuts_level = NULL,  
  geometry = TRUE  
)  
  
gisco_id_api_lau(  
  x,  
  y,  
  year = 2024,  
  epsg = c(4326, 4258, 3035),  
  verbose = FALSE,  
  geometry = TRUE  
)  
  
gisco_id_api_country(  
  x,  
  y,  
  year = 2024,  
  epsg = c(4326, 4258, 3035),
```

```

    verbose = FALSE,
    geometry = TRUE
  )

  gisco_id_api_river_basin(
    x,
    y,
    year = 2019,
    epsg = c(4326, 4258, 3035),
    verbose = FALSE,
    geometry = TRUE
  )

  gisco_id_api_biogeo_region(
    x,
    y,
    year = 2016,
    epsg = c(4326, 4258, 3035),
    verbose = FALSE,
    geometry = TRUE
  )

  gisco_id_api_census_grid(
    x,
    y,
    year = 2021,
    epsg = c(4326, 4258, 3035),
    verbose = FALSE,
    geometry = TRUE
  )

```

Arguments

x, y	character string or numeric. x and y coordinates (as longitude and latitude) to be identified.
xmin, ymin, xmax, ymax	character string or numeric. Bounding box coordinates to identify all geonames within the box.
verbose	logical. If TRUE displays informational messages.
year	character string or numeric. Year of the dataset, see Details .
epsg	character string or numeric. EPSG code for the coordinate reference system.
nuts_id	character. NUTS ID code.
nuts_level	character string. NUTS level. One of 0, 1, 2 or 3.
geometry	logical. Whether to return geometry. On TRUE a sf object would be returned. On FALSE a tibble would be returned.

Details

The available endpoints are:

- `gisco_id_api_geonames()`: Get geographic placenames either from x/y coordinates or a bounding box.
- `gisco_id_api_nuts()`: Returns NUTS regions from either a specified longitude and latitude (x,y) or id. Accepted year are "2024", "2021", "2016", "2013", "2010", "2006".
- `gisco_id_api_lau()`: Returns the id and - optionally - geometry for Large Urban Areas (LAU) at specified longitude and latitude (x,y). Accepted year are "2024", "2023", "2022", "2021", "2020", "2019", "2018", "2017", "2016", "2015", "2014", "2013", "2012", "2011".
- `gisco_id_api_country()`: Returns the id and - optionally - geometry for countries at specified longitude and latitude (x,y). Accepted year are "2024", "2020", "2016", "2013", "2010", "2006".
- `gisco_id_api_river_basin()`: Returns the id and - optionally - geometry for river basins at specified longitude and latitude (x,y), based on the Water Framework Directive (WFD) reference spatial data sets. Accepted year is "2019".
- `gisco_id_api_biogeo_region()`: Returns the id and - optionally - geometry for biogeo regions at specified longitude and latitude (x,y). The biogeographical regions dataset contains the official delineations used in the Habitats Directive (92/43/EEC) and for the EMERALD Network. Accepted year is "2016".
- `gisco_id_api_census_grid()`: Returns the id and - optionally - geometry for census grid cells at specified longitude and latitude (x,y). Accepted year is "2021".

Value

A [tibble](#) or a [sf](#) object.

Source

<https://gisco-services.ec.europa.eu/id/api-docs/>.

See Also

[gisco_get_nuts\(\)](#), [gisco_get_lau\(\)](#), [gisco_get_countries\(\)](#), [gisco_get_census\(\)](#).

Other API tools: [gisco_address_api](#)

Examples

```
gisco_id_api_geonames(x = -2.5, y = 43.06)

lau <- gisco_id_api_lau(x = -2.5, y = 43.06)
nuts3 <- gisco_id_api_nuts(x = -2.5, y = 43.06, nuts_level = 3)

if (all(!is.null(lau), !is.null(nuts3))) {
  library(ggplot2)

  ggplot(nuts3) +
```

```

geom_sf(fill = "lightblue", color = "black") +
geom_sf(data = lau, fill = "orange", color = "red") +
labs(
  title = "NUTS3 and LAU boundaries",
  subtitle = "Arrasate, Basque Country, Spain",
  caption = "Source: GISCO ID service API"
)
}

```

gisco_nuts_2024	NUTS 2024 sf object
-----------------	-------------------------------------

Description

This dataset represents the regions for levels 0, 1, 2 and 3 of the Nomenclature of Territorial Units for Statistics (NUTS) for 2024.

Format

A [sf](#) object with MULTIPOLYGON geometries, resolution: 1:20 million and [EPSG:4326](#). with 263 rows and 10 variables:

NUTS_ID NUTS identifier.

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

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**.

URBN_TYPE Urban Type, see **Details**.

COAST_TYPE Coast Type, see **Details**.

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_2024_4326.gpkg](#) file.

See Also

[gisco_get_nuts\(\)](#)

Other datasets: [gisco_coastal_lines](#), [gisco_countries_2024](#), [gisco_countrycode](#), [gisco_db](#)

Examples

```
data("gisco_nuts_2024")
head(gisco_nuts_2024)
```

<code>gisco_set_cache_dir</code>	<i>Set your cache dir</i>	Rhrefhttps://CRAN.R-project.org/package=giscoR
----------------------------------	---------------------------	--

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\(\)](#).

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 <code>base::tempdir()</code>).
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 argument is set to FALSE automatically.
verbose	logical. If TRUE displays informational messages.

Details

By default, when no cache cache_dir is set the package uses a folder inside `base::tempdir()` (so files are temporary and are removed when the **R** session ends). To persist a cache across **R** sessions, use `gisco_set_cache_dir(cache_dir, install = TRUE)` which writes the chosen path to a small configuration file under `tools::R_user_dir("giscoR", "config")`.

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.

Caching strategies

Some files can be read from its online source without caching using the option `cache = FALSE`. Otherwise the source file would be downloaded to your computer. **giscoR** implements the following caching options:

- For occasional use, rely on the default `tempdir()`-based cache (no install).
- Modify the cache for a single session setting `gisco_set_cache_dir(cache_dir = "a/path/here")`.
- For reproducible workflows, install a persistent cache with `gisco_set_cache_dir(cache_dir = "a/path/here, install = TRUE)` that would be kept across **R** sessions.
- For caching specific files, use the `cache_dir` argument in the corresponding function. See example in `gisco_get_nuts()`.

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

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

Note

In **giscoR** $\geq 1.0.0$ the location of the configuration file has moved from `rappdirs::user_config_dir("giscoR", "R")` to `tools::R_user_dir("giscoR", "config")`. We have implemented a functionality that would migrate previous configuration files from one location to another with a message. This message would appear only once informing of the migration.

See Also[tools::R_user_dir\(\)](#)Other cache utilities: [gisco_clear_cache\(\)](#)**Examples**

```
# Don't run this! It would modify your current state
## Not run:
my_cache <- gisco_detect_cache_dir()

# Set an example cache
ex <- file.path(tempdir(), "example", "cachenew")
gisco_set_cache_dir(ex)

gisco_detect_cache_dir()

# Restore initial cache
gisco_set_cache_dir(my_cache)
identical(my_cache, gisco_detect_cache_dir())

## End(Not run)

gisco_detect_cache_dir()
```

Index

- * **API tools**
 - gisco_address_api, 3
 - gisco_id_api, 49
- * **admin**
 - gisco_get_communes, 20
 - gisco_get_countries, 23
 - gisco_get_postal_codes, 40
- * **cache utilities**
 - gisco_clear_cache, 10
 - gisco_set_cache_dir, 53
- * **database**
 - gisco_db, 14
 - gisco_get_cached_db, 16
 - gisco_get_metadata, 34
- * **datasets**
 - gisco_coastal_lines, 11
 - gisco_countries_2024, 12
 - gisco_countrycode, 13
 - gisco_db, 14
 - gisco_nuts_2024, 52
- * **deprecated functions**
 - gisco_get_units, 44
- * **extra**
 - gisco_bulk_download, 8
 - gisco_get_unit, 42
- * **grids**
 - gisco_get_grid, 27
- * **misc**
 - gisco_attributions, 6
- * **services**
 - gisco_get_education, 25
 - gisco_get_healthcare, 29
- * **stats**
 - gisco_get_census, 17
 - gisco_get_coastal_lines, 19
 - gisco_get_lau, 32
 - gisco_get_nuts, 35
 - gisco_get_urban_audit, 46
- * **transport**
 - gisco_get_airports, 15
 - gisco_get_ports, 38
 - ?gisco_get_unit, 44, 46
- base::tempdir(), 54
- cache_dir, 8, 42
- countrycode::codelist, 6, 13, 14
- countrycode::countrycode(), 15, 21, 24, 25, 30, 33, 37, 39, 40, 47
- eurostat::get_eurostat_geospatial(), 37
- gisco_address_api, 3, 51
- gisco_address_api_bbox
 - (gisco_address_api), 3
- gisco_address_api_cities
 - (gisco_address_api), 3
- gisco_address_api_copyright
 - (gisco_address_api), 3
- gisco_address_api_countries
 - (gisco_address_api), 3
- gisco_address_api_housenumbers
 - (gisco_address_api), 3
- gisco_address_api_postcodes
 - (gisco_address_api), 3
- gisco_address_api_provinces
 - (gisco_address_api), 3
- gisco_address_api_reverse
 - (gisco_address_api), 3
- gisco_address_api_roads
 - (gisco_address_api), 3
- gisco_address_api_search
 - (gisco_address_api), 3
- gisco_addressapi (gisco_address_api), 3
- gisco_addressapi_bbox
 - (gisco_address_api), 3
- gisco_addressapi_cities
 - (gisco_address_api), 3

- `gisco_addressapi_copyright`
(`gisco_address_api`), 3
- `gisco_addressapi_countries`
(`gisco_address_api`), 3
- `gisco_addressapi_housenumbers`
(`gisco_address_api`), 3
- `gisco_addressapi_postcodes`
(`gisco_address_api`), 3
- `gisco_addressapi_provinces`
(`gisco_address_api`), 3
- `gisco_addressapi_reverse`
(`gisco_address_api`), 3
- `gisco_addressapi_roads`
(`gisco_address_api`), 3
- `gisco_addressapi_search`
(`gisco_address_api`), 3
- `gisco_attributions`, 6
- `gisco_attributions()`, 20, 22, 24, 33, 37, 41, 43, 46, 48
- `gisco_bulk_download`, 8, 44
- `gisco_bulk_download()`, 20, 22, 24, 34, 37, 41, 48
- `gisco_clear_cache`, 10, 55
- `gisco_coastal_lines`, 11, 12, 14, 20, 53
- `gisco_countries_2024`, 12, 12, 14, 24, 53
- `gisco_countrycode`, 12, 13, 14, 24, 53
- `gisco_db`, 12, 14, 14, 17, 35, 53
- `gisco_detect_cache_dir`
(`gisco_set_cache_dir`), 53
- `gisco_detect_cache_dir()`, 53, 54
- `gisco_get` (`gisco_get_countries`), 23
- `gisco_get_airports`, 15, 39
- `gisco_get_cached_db`, 14, 16, 35
- `gisco_get_cached_db()`, 14
- `gisco_get_census`, 17, 20, 34, 37, 48
- `gisco_get_census()`, 7, 51
- `gisco_get_coastal_lines`, 18, 19, 34, 37, 48
- `gisco_get_coastal_lines()`, 7, 9, 12
- `gisco_get_coastallines`
(`gisco_get_coastal_lines`), 19
- `gisco_get_communes`, 20, 24, 41
- `gisco_get_communes()`, 7, 9, 32, 34
- `gisco_get_countries`, 22, 23, 41
- `gisco_get_countries()`, 7, 9, 12, 14, 35, 42, 44, 51
- `gisco_get_education`, 25, 31
- `gisco_get_grid`, 27
- `gisco_get_healthcare`, 26, 29
- `gisco_get_lau`, 18, 20, 32, 37, 48
- `gisco_get_lau()`, 7, 9, 20, 22, 51
- `gisco_get_metadata`, 14, 17, 34
- `gisco_get_metadata()`, 24, 43, 44, 46
- `gisco_get_nuts`, 18, 20, 34, 35, 48
- `gisco_get_nuts()`, 7, 9, 35, 42, 44, 51, 53, 54
- `gisco_get_ports`, 15, 38
- `gisco_get_ports()`, 39
- `gisco_get_postal_codes`, 22, 24, 40
- `gisco_get_postal_codes()`, 7, 9
- `gisco_get_postalcodes`
(`gisco_get_postal_codes`), 40
- `gisco_get_unit`, 9, 42
- `gisco_get_unit_country`
(`gisco_get_unit`), 42
- `gisco_get_unit_country()`, 23, 24
- `gisco_get_unit_nuts` (`gisco_get_unit`), 42
- `gisco_get_unit_nuts()`, 35, 37
- `gisco_get_unit_urban_audit`
(`gisco_get_unit`), 42
- `gisco_get_unit_urban_audit()`, 46, 48
- `gisco_get_units`, 44
- `gisco_get_urban_audit`, 18, 20, 34, 37, 46
- `gisco_get_urban_audit()`, 7, 9, 35, 42, 44
- `gisco_id_api`, 5, 44, 49
- `gisco_id_api_biogeo_region`
(`gisco_id_api`), 49
- `gisco_id_api_census_grid`
(`gisco_id_api`), 49
- `gisco_id_api_census_grid()`, 18
- `gisco_id_api_country` (`gisco_id_api`), 49
- `gisco_id_api_country()`, 24
- `gisco_id_api_geonames` (`gisco_id_api`), 49
- `gisco_id_api_lau` (`gisco_id_api`), 49
- `gisco_id_api_lau()`, 34
- `gisco_id_api_nuts` (`gisco_id_api`), 49
- `gisco_id_api_nuts()`, 37
- `gisco_id_api_river_basin`
(`gisco_id_api`), 49
- `gisco_nuts_2024`, 12, 14, 37, 52
- `gisco_set_cache_dir`, 11, 53
- `gisco_set_cache_dir()`, 8, 15, 17–19, 21, 23, 25, 28, 30, 33, 36, 39, 40, 43, 45, 47
- `sf`, 5, 11, 12, 15, 18, 20, 22, 24, 26, 28, 31, 33, 37, 39, 41, 43, 46, 48, 50–52

`tempdir()`, [54](#)

`tibble`, [5](#), [13](#), [14](#), [17](#), [35](#), [46](#), [50](#), [51](#)

`tools::R_user_dir()`, [11](#), [55](#)

warning, [9](#)