# Package 'giscoR'

February 25, 2021

**Type** Package

**Title** Download Map Data from GISCO API - Eurostat

**Version** 0.2.3.9000

**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

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**Roxygen** list(markdown = TRUE)

**BugReports** https://github.com/dieghernan/giscoR/issues

**URL** https://dieghernan.github.io/giscoR/, https://github.com/dieghernan/giscoR

**Depends** R (>= 3.6.0)

**Imports** sf (>= 0.9),
lwgeom (>= 0.2-2),
countrycode (>= 1.2.0),
geojsonsf (>= 2.0)

**Suggests** cartography (>= 2.4),
tinytest

**LazyData** true

## R topics documented:

---

`giscoR-package` *Download geospatial data from GISCO API - Eurostat*

---

## Description

giscoR is a API package that helps to retrieve data from Eurostat - GISCO (the Geographic Information System of the COmmission)

## Details

giscoR package

| | |
|---|---|
| Package: | giscoR |
| Type: | Package |
| Version: | See sessionInfo() or DESCRIPTION file |
| Date: | 2020 |
| License: | GPL-3 |
| LazyLoad: | yes |

## Note

COPYRIGHT NOTICE

When data downloaded from this page 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 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.

## Author(s)

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

## Source

GISCO webpage

## References

See `citation("giscoR")`

## See Also

Useful links:

- https://dieghernan.github.io/giscoR/
- https://github.com/dieghernan/giscoR
- Report bugs at https://github.com/dieghernan/giscoR/issues

---

# 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. 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

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.

## Examples

```
en <- 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 = "countries",
  year = "2016",
  cache_dir = NULL,
  update_cache = FALSE,
  verbose = FALSE,
  resolution = "10",
  ext = "geojson",
  recursive = TRUE
)
```

## Arguments

| | |
|---|---|
| id_giscoR | Type of dataset to be downloaded. Values supported are:<br>• "coastallines"<br>• "communes"<br>• "countries"<br>• "lau"<br>• "nuts"<br>• "urban_audit" |
| year | Release year. See Details. |
| cache_dir | a path to a cache directory. The directory have to exist. The NULL (default) uses and creates /gisco directory in the temporary directory from tempdir. The directory can also be set with options(gisco_cache_dir = "path/to/dir"). |
| 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 | Display information. Useful for debugging, default if FALSE. |
| resolution | Resolution of the geospatial data. One of<br>• "60" (1:60million),<br>• "20" (1:20million)<br>• "10" (1:10million)<br>• "03" (1:3million) or<br>• "01" (1:1million). |
| ext | Extension of the file(s) to be downloaded. Available formats 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 gisco_get

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

This function helps building a personal shape library on cache_dir (or options(gisco_cache_dir = "path/to/dir"), if set by the user).

## Value

Silent function.

## Note

For downloading specific files use gisco_get functions.

## Source

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

## 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.

## Examples

```
gisco_check_access()
```

gisco_coastallines *World coastal lines* POLYGON *object*

## Description

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

## Format

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

- **FID**
- **COAS_ID**
- **geometry**: geometry field

## Source

https://gisco-services.ec.europa.eu/distribution/v2/coas/geojson/ COAS_RG_20M_2016_4326.geojson file.

## See Also

gisco_get_coastallines()

## Examples

```
library(sf)

coasts <- gisco_coastallines

plot(
  st_geometry(coasts),
  xlim = c(100, 120),
  ylim = c(-24, 24),
  col = "grey90",
  border = "deepskyblue4",
  lwd = 2
)
box()
title(
  main = "Coasts on Southeastern Asia",
  sub = gisco_attributions(),
  cex.sub = 0.7,
  line = 1
)
```

gisco_countries *World countries* POLYGON *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

https://gisco-services.ec.europa.eu/distribution/v2/countries/geojson/, CNTR_RG_20M_2016_4326.geojso file.

## See Also

gisco_get_countries()

## Examples

```
library(sf)

cntry <- gisco_countries
GBR <- subset(cntry, ISO3_CODE == "GBR")

plot(st_geometry(GBR), col = "red3", border = "blue4")
title(sub = gisco_attributions(), line = 1)
```

`gisco_countrycode` *Dataframe with different country code schemes and world regions*

## Description

A dataframe 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 dataset is extracted from **countrycode** package.

## Format

A data frame object with 249 rows and 12 variables:

- **CNTR_CODE**: Eurostat code of each country
- **iso2c**: ISO 3166-1 alpha-2 code of each country
- **ISO3_CODE**: 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 `http://cldr.unicode.org/translation/displaynames/country-names`
- **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 code UN (M49)
- **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 as per February 2021.

## Source

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

## See Also

countrycode::codelist, countrycode::countrycode-package

## Examples

```
data(gisco_countrycode)
```

## gisco_db *GISCO database*

### Description

Database with the list of files that the package can load.

### Format

A data frame

### Details

This dataframe is used to check the validity of the API calls.

### Source

GISCO API `datasets.json`.

### Examples

```
data(gisco_db)
```

## gisco_get *Get geospatial data from GISCO API*

### Description

Loads a simple feature (`sf`) object from GISCO API entry point or your local library.

### Usage

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

gisco_get_communes(
  year = "2016",
  epsg = "4326",
```

```
  cache = TRUE,
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  spatialtype = "RG",
  country = NULL
)

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
)

gisco_get_lau(
  year = "2016",
  epsg = "4326",
  cache = TRUE,
  update_cache = FALSE,
  cache_dir = NULL,
  verbose = FALSE,
  country = NULL,
  gisco_id = NULL
)

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"
)

gisco_get_urban_audit(
  year = "2020",
```

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

## Arguments

| | |
|---|---|
| year | Release year. See Details. |
| epsg | projection of the map: 4-digit EPSG code. One of:<br>• "4326" - WGS84<br>• "3035" - ETRS89 / ETRS-LAEA<br>• "3857" - Pseudo-Mercator |
| cache | a logical whether to do caching. Default is TRUE. |
| 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. The directory have to exist. The NULL (default) uses and creates /gisco directory in the temporary directory from tempdir. The directory can also be set with options(gisco_cache_dir = "path/to/dir"). |
| verbose | Display information. Useful for debugging, default if FALSE. |
| resolution | Resolution of the geospatial data. One of<br>• "60" (1:60million),<br>• "20" (1:20million)<br>• "10" (1:10million)<br>• "03" (1:3million) or<br>• "01" (1:1million). |
| spatialtype | Type of geometry to be returned:<br>• "RG": Regions - MULTIPOLYGON/POLYGON object.<br>• "LB": Labels - POINT object.<br>• "BN": Boundaries - LINESTRING object.<br>• "COASTL": coastlines - LINESTRING object.<br>• "INLAND": inland boundaries - LINESTRING object. |
| country | Optional. A character vector of country codes. See Details. |
| 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 Details and gisco_countrycode |
| gisco_id | Optional. A character vector of GISCO_ID LAU values. |
| nuts_id | Optional. A character vector of NUTS IDs. |

nuts_level — NUTS level. One of "0" (Country-level), "1","2" or "3". See https://ec.europa.eu/eurostat/web/nuts/background.

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

## Details

country only available on specific datasets. Some spatialtype options (as BN,COASTL,INLAND) may not present country-level identifies.

country 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("Turkey","US","FRA")) would not work.

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 set cache_dir = "path/to/dir" or options(gisco_cache_dir = "path/to/dir)".

For a complete list of files available check gisco_db.

### About world regions

Regions are defined as per the geographic regions defined by the UN (see M49 region codes. Note that under this scheme Cyprus is assigned to Asia. You may use region = "EU" to get the EU members (reference date: 2021).

### Release years available

gisco_get_coastallines: one of "2006","2010","2013" or "2016".

gisco_get_communes: one of "2001","2004","2006","2008","2010","2013" or "2016".

gisco_get_countries: one of "2001","2006","2010","2013","2016" or "2020".

gisco_get_lau: one of "2016","2017","2018" or "2019".

gisco_get_nuts: one of "2003","2006","2010","2013","2016" or "2021".

gisco_get_urban_audit: one of "2001","2004","2014","2018" or "2020".

## Value

A sf object specified by spatialtype. See Details.

## Note

Please check the download and usage provisions on gisco_attributions.

## Author(s)

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

## Source

GISCO API

## See Also

gisco_db, gisco_attributions, gisco_coastallines

gisco_countrycode, gisco_countries

gisco_nuts

## Examples

```
library(sf)

#################################
# Example - gisco_get_coastallines
#################################

coastlines <- gisco_get_coastallines()
plot(st_geometry(coastlines), col = "palegreen", border = "lightblue3")
title(
  main = "Coastal Lines",
  sub = gisco_attributions(),
  line = 1
)

##############################
# Example - gisco_get_countries
##############################

sf_world <- gisco_get_countries()
plot(st_geometry(sf_world), col = "seagreen2")
title(sub = gisco_attributions(), line = 1)


sf_africa <- gisco_get_countries(region = "Africa")
plot(st_geometry(sf_africa),
  col = c("springgreen4", "darkgoldenrod1", "red2")
)
title(sub = gisco_attributions(), line = 1)

sf_benelux <-
  gisco_get_countries(country = c("Belgium", "Netherlands", "Luxembourg"))
plot(st_geometry(sf_benelux),
  col = c("grey10", "orange", "deepskyblue2")
)
title(sub = gisco_attributions(), line = 1)

#########################
# Example - gisco_get_nuts
#########################

nuts1 <- gisco_get_nuts(
  resolution = "20",
  year = "2016",
  epsg = "4326",
```

```
  nuts_level = "1",
  country = "ITA"
)
nuts2 <- gisco_get_nuts(
  resolution = "20",
  year = "2016",
  epsg = "4326",
  nuts_level = "2",
  country = "ITA"
)
nuts3 <- gisco_get_nuts(
  resolution = "20",
  year = "2016",
  epsg = "4326",
  nuts_level = "3",
  country = "ITA"
)

plot(st_geometry(nuts3),
  border = "grey60",
  lty = 3
)

plot(st_geometry(nuts2),
  lwd = 2,
  border = "red2",
  add = TRUE
)

plot(st_geometry(nuts1),
  lwd = 3,
  border = "springgreen4",
  add = TRUE
)

box()
title(
  main = "NUTS Levels on Italy",
  sub = gisco_attributions(),
  cex.sub = 0.7,
  line = 1
)
legend(
  "topright",
  legend = c("NUTS 1", "NUTS 2", "NUTS 3"),
  col = c("springgreen4", "red2", "grey60"),
  lty = c(1, 1, 3),
  lwd = c(3, 2, 1),
  bty = "n",
  y.intersp = 2
)
```

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

## Description

Loads a simple feature (sf) object from GISCO API entry point or your local library.

## Usage

```
gisco_get_airports(year = "2013", country = NULL)

gisco_get_ports(year = "2013")
```

## Arguments

| | |
|---|---|
| year | Year of reference. |
| country | A list of countries, see gisco_get_countries |

## Details

year available:

- gisco_get_airports (2006,2013)
- gisco_get_ports (2009,2013)

Ports 2009 contains worldwide information, the rest of datasets refer to Europe. All shapefiles provided in EPSG:4326

## Value

A POINT object on EPSG:4326.

## Author(s)

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

## Source

GISCO API

## Examples

```
library(sf)

NL <- gisco_get_countries(country = "NL")
AirP_NL <- gisco_get_airports(country = "NL")

Ports <- gisco_get_ports()
# Intersect with NL
```

```
PortsNL <- st_intersection(Ports, NL)


plot(st_geometry(NL), col = "wheat")
plot(
  st_geometry(PortsNL),
  pch = 22,
  col = "forestgreen",
  add = TRUE,
  cex = 0.8
)

plot(
  st_geometry(AirP_NL),
  pch = 20,
  col = "steelblue",
  add = TRUE,
  cex = 1.2
)
legend(
  "topright",
  legend = c("Port", "Airport"),
  col = c("forestgreen", "steelblue"),
  cex = 0.9,
  bty = "n",
  pch = c(22, 20),
  pt.cex = c(1, 1.5),
  y.intersp = 2
)

title(
  main = "Transport Network on the Nethelands",
  sub = gisco_attributions(),
  line = 1,
  cex.sub = 0.7,
  font.sub = 3
)
```

---

`gisco_get_grid` *Get the grid cells covering the European land territory, 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 = "REGION",
  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,POINT` |
| `cache_dir, update_cache, verbose` | See gisco_get |

## Details

Files are distributed on EPSG:3035.

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

## Value

A `POLYGON/POINT` object.

## Note

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

## Author(s)

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

## Source

GISCO API Grids

## Examples

```
library(sf)

grid <- gisco_get_grid(resolution = 20)
grid$popdens <- grid$TOT_P_2011 / 20

breaks <-
  c(
    0,
    500,
    1000,
```

```
    2500,
    5000,
    10000,
    25000,
    50000,
    max(grid$popdens) + 1
  )

pal <- hcl.colors(length(breaks) - 2, palette = "inferno", alpha = 0.7)
pal <- c("black", pal)

opar <- par(no.readonly = TRUE)
par(mar = c(0, 0, 0, 0), bg = "grey2")
plot(
  grid[, "popdens"],
  pal = pal,
  key.pos = NULL,
  breaks = breaks,
  main = NA,
  xlim = c(2500000, 7000000),
  ylim = c(1500000, 5200000),
  border = NA
)
par(opar)
```

---

`gisco_get_healthcare` *Get the 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`, `update_cache`, `cache_dir`, `verbose`, `country`
: See gisco_get

## Details

Files are distributed on EPSG:4326. Link to metadata

## Value

A POINT object.

## Author(s)

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

## Source

GISCO Healthcare services

## See Also

gisco_get

---

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 shapefiles for each unit.

## Usage

```
gisco_get_units(
  id_giscoR = "nuts",
  unit = "ES4",
  mode = "sf",
  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, epsg, cache, update_cache, cache_dir, verbose, resolution | See gisco_get. |
| spatialtype | Type of geometry to be returned: "RG" for POLYGON and "LB" for POINT. |

## Details

The function can return a dataframe on mode = "df" or a sf object on mode = "sf"

In order to see the available unit ids with the required combination of what,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 what,year.

## Value

A sf object on mode = "sf" or a dataframe on mode = "df".

## Note

countries file would be renamed on your cache_dir to avoid naming conflicts with nuts datasets.

## Author(s)

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

## Source

GISCO API

## See Also

gisco_get

## Examples

```
## Not run:
library(sf)

if (gisco_check_access()) {
  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"
    )

  # Surrounding area
  NUTS1 <-
    gisco_get_units(
      id_giscoR = "nuts",
      unit = c("ES5"),
      resolution = "01"
    )

  # Plot
  plot(
    st_geometry(Provincia),
    col = "gray1",
    border = "grey50",
    lwd = 3
  )
  plot(st_geometry(NUTS1),
    border = "grey50",
    lwd = 3,
    add = TRUE
  )
  plot(
    st_geometry(VAL.sf),
    col = c("deeppink4", "brown2", "khaki1"),
    add = TRUE
  )
  box()
  title(
    "Urban Audit - Valencia (ES)",
    sub = gisco_attributions("es"),
    line = 1,
    cex.sub = 0.7
  )
}

## End(Not run)
```

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 11 variables:

- **id**: row ID
- **COAST_TYPE**: COAST_TYPE
- **MOUNT_TYPE**: MOUNT_TYPE
- **NAME_LATN**: Name on Latin characters
- **CNTR_CODE**: Eurostat Country code
- **FID**: FID
- **NUTS_ID**: NUTS identifier
- **NUTS_NAME**: NUTS name on local alphabet
- **LEVL_CODE**: NUTS level code (0,1,2,3)
- **URBN_TYPE**: URBN_TYPE
- **geometry**: geometry field

## Source

https://gisco-services.ec.europa.eu/distribution/v2/nuts/geojson/ NUTS_RG_20M_2016_4326.geojson file.

## See Also

gisco_get_nuts()

## Examples

```
library(sf)

nuts <- gisco_nuts

italy <- subset(nuts, CNTR_CODE == "IT" & LEVL_CODE == 3)

plot(st_geometry(italy), col = c("springgreen4", "ivory", "red2"))
title(
  sub = gisco_attributions(),
  line = 1,
  cex.sub = 0.7,
  font.sub = 3
)
```

tgs00026 *Disposable income of private households by NUTS 2 regions*

## Description

The disposable income of private households is the balance of primary income (operating surplus/mixed income plus compensation of employees plus property income received minus property income paid) and the redistribution of income in cash. These transactions comprise social contributions paid, social benefits in cash received, current taxes on income and wealth paid, as well as other current transfers. Disposable income does not include social transfers in kind coming from public administrations or non-profit institutions serving households.

## Format

data_frame:

- **geo**: NUTS2 identifier
- **time**: reference year (2007 to 2018)
- **values**: value in euros

## Source

https://ec.europa.eu/eurostat, extracted on 2020-10-27

## Examples

```
data(tgs00026)
```

# Index