Package ‘regions’

June 21, 2021

Title Processing Regional Statistics
Version 0.1.8
Date 2021-06-19
Description Validating sub-national statistical typologies, re-coding across standard typologies of sub-national statistics, and making valid aggregate level imputation, re-aggregation, re-weighting and projection down to lower hierarchical levels to create meaningful data panels and time series.
License GPL-3
Encoding UTF-8
Language en-US
URL https://regions.dataobservatory.eu/
BugReports https://github.com/rOpenGov/regions
LazyData true
RoxygenNote 7.1.1
Depends R (>= 2.10)
Imports dplyr, magrittr, countrycode, tidyselect, utils, purrr, rlang, glue, stats, tidyrr, readxl, stringr, assertthat, tibble, here
Suggests knitr, testthat, rmarkdown, covr, spelling, devtools, eurostat, ggplot2
VignetteBuilder knitr
NeedsCompilation no
Author Daniel Antal [aut, cre] (<https://orcid.org/0000-0001-7513-6760>), Kasia Kulma [ctb] (<https://orcid.org/0000-0002-2952-9720>), Istvan Zsoldos [ctb] (<https://orcid.org/0000-0001-5712-2103>), Leo Lahti [ctb] (<https://orcid.org/0000-0001-5537-637X>)
Maintainer Daniel Antal <daniel.antal@ceemid.eu>
Repository CRAN
Date/Publication 2021-06-21 11:20:01 UTC


**R topics documented:**

- all_valid_nuts_codes ........................................... 2
- australia_states .................................................. 3
- create_nuts_lau_2019 ............................................. 3
- daily_internet_users ............................................. 4
- get_country_code .................................................. 4
- google_nuts_matchtable .......................................... 5
- impute_down ....................................................... 6
- impute_down_nuts .................................................. 8
- mixed_nuts_example ............................................... 9
- nuts_changes ....................................................... 10
- nuts_exceptions ................................................... 11
- nuts_lau_2019 ...................................................... 11
- nuts_recoded ...................................................... 13
- recode_nuts ....................................................... 13
- regional_rd_personnel ........................................... 14
- regions ............................................................ 15
- validate_geo_code ............................................... 16
- validate_nuts_countries ........................................ 17
- validate_nuts_regions ........................................... 18
- validate_parameters ............................................. 19

---

**Index**

21

---

**Description**

A dataset containing all recognised geo codes in the EU NUTS correspondence tables. This is re-arranged from **nuts_changes**.

**Usage**

all_valid_nuts_codes

**Format**

A data frame with 3 variables:

- **geo**  
  NUTS geo identifier

- **typology**  
  country, NUTS1, NUTS2 or NUTS3

- **nuts**  
  The NUTS definition where the geo code can be found.

**Source**

https://ec.europa.eu/eurostat/web/nuts/history/
See Also

nuts_recoded, nuts_changes, nuts_exceptions

---

**australia_states**

*Australia: States And Territories*

---

**Description**

A dataset containing the states and territories of Australia.

**Usage**

australia_states

**Format**

A data frame with 8 rows and 3 variables:

- **country_code**  ISO 3166-1 country codes
- **geo_code**  subdivision codes within Australia (states and territories)
- **geo_name**  subdivision names within Australia (states and territories)

**Source**


---

**create_nuts_lau_2019**

*Create the nuts_lau_2019 correspondence table May be used to create similar historical correspondence tables.*

---

**Description**

Create the nuts_lau_2019 correspondence table May be used to create similar historical correspondence tables.

**Usage**

create_nuts_lau_2019()

**Value**

A data.frame which is also saved and can be retrieved with data(nuts_lau_2019). Use this function as a template to obtain historical correspondence tables.
**daily_internet_users**  
*Daily Internet Users*

### Description
A dataset containing the percentage of individuals who used the Internet on a daily basis in the European countries and regions.

### Usage
```r
daily_internet_users
```

### Format
A data frame with 3 variables:

- **geo**  
  National and sub-national geographical codes from Eurostat

- **time**  
  Time, coded as a numeric variable of the year, 2006-2019

- **values**  
  The numeric statistical values

### Details
The fresh version of this statistic can be obtained by
```r
eurostat::get_eurostat("isoc_r_iuse_i", time_format = "num")
```
and filtered for the `indic_is = "I_IDAY"` indicator and the `unit="PC_IND"` unit.

### Source
The eventual source of the data is the Eurostat table `isoc_r_iuse_i` [https://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=isoc_r_iuse_i&lang=en](https://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=isoc_r_iuse_i&lang=en)

---

**get_country_code**  
*Get Country Code Of Regions*

### Description
The function identifies the sub-national geographical identifiers from known typologies and returns the ISO 3166-1 alpha-2 country codes.

### Usage
```r
get_country_code(geo, typology = "NUTS")
```
**Arguments**

`geo`  
A character variable with geo codes.

`typology`  
Currently the following typologies are supported: "NUTS1", "NUTS2", "NUTS3" or "NUTS" for any of the NUTS typologies. The technical typology "NUTS0" can be used to translate Eurostat country codes to ISO 3166-1 alpha-2 country codes.

**Value**

The ISO 3166-1 alpha-2 codes of the countries as a character vector.

**See Also**

Other recode functions: `recode_nuts()`

**Examples**

```r
{  
get_country_code (c("EL", "GR", "DED", "HU102")) 
}
```

---

**google_nuts_matchtable**

*Google Mobility Report European Correspondence Table*

**Description**

A dataset containing the correspondence table between the EU NUTS 2016 typology and the typology used by Google in the Google Mobility Reports.

**Usage**

`google_nuts_matchtable`

**Format**

A data frame with 817 rows and 6 variables:

- **country_code**  ISO 3166-1 alpha2 code
- **google_region_level**  Hierarchical level in the Google Mobility Reports
- **google_region_name**  The name used by Google.
- **code_2016**  NUTS code in the 2016 definition
- **typology**  country, NUTS1, NUTS2 or NUTS3, nuts_level_3_lau, nuts_level_3_iso-3166-2
- **valid_2016**  Logical variable, if the coding is valid in NUTS2016
Details

In some cases only a full correspondence is not possible. In these cases we created pseudo-NUTS codes, which have a FALSE valid_2016 value. These pseudo-NUTS codes can help approximation for the underlying regions.

Pseudo-NUTS codes were used in Estonia, Italy, Portugal, Slovenia and in parts of Latvia.

In Latvia and Slovenia, the pseudo NUTS code is a combination of the the containing NUTS3 code and the municipality’s LAU code.

In Estonia, they are a combination of the NUTS3 code and the ISO-3166-2 LAU code (county level.) This is the case in most of Portugal and the United Kingdom, too. In these cases the pseudo-codes refer to a quasi-NUTS4 code, which are smaller than the containing NUTS3 region, therefore they should be aggregated.

A special case is ITD_IT-32, which is is a combination of two NUTS2 statistical regions, but it forms under the ISO-3166-2 ITD_IT-32 a single unit, the autonomous region of Trentino and South Tyrol. In this case, they should be disaggregated.

A similar solution is required for the United Kingdom.

Author(s)

Istvan Zsoldos, Daniel Antal

Source

https://ec.europa.eu/eurostat/web/nuts/history/

| impute_down | Imputing Data From Larger To Smaller Units |

Description

Imputing Data From Larger To Smaller Units

Usage

```r
impute_down(
  upstream_data = NULL,
  downstream_data = NULL,
  country_var = "country_code",
  regional_code = "geo_code",
  values_var = "values",
  time_var = NULL,
  upstream_method_var = NULL,
  downstream_method_var = NULL
)
```
impute_down

Arguments

**upstream_data**  
An upstream data frame to project on containing smaller geographical units, for example, country-level data.

**downstream_data**  
A downstream data frame containing the smaller level missing data observations. It must contain all the necessary structural information for imputation.

**country_var**  
The geographical ID of the upstream data, defaults to "country_code".

**regional_code**  
The geographical ID of the downstream data, defaults to "geo_code".

**values_var**  
The variable that contains the upstream data to be imputed to the downstream data, defaults to "values".

**time_var**  
The time component, if present, defaults to "year".

**upstream_method_var**  
The name of the variable that contains the potentially applied imputation methods. Defaults to NULL.

**downstream_method_var**  
The name of the variable that will contain the metadata of the potentially applied imputation methods. Defaults to NULL in which case a variable called 'method' will be created. If possible, avoid using upstream_data or downstream_data that contains a variable called 'method' for other purposes.

Value

The upstream data frame (containing data of a larger unit) and the downstream data (containing data of smaller sub-divisional units) are joined; whenever data is missing in the downstream sub-divisional column, it is imputed with the corresponding values from the upstream data frame. The 'method' metadata column explains if the actual downstream data or the imputed data can be found in the downstream value column.

See Also

Other impute functions: impute_down_nuts()

Examples

```r
{
  upstream <- data.frame ( country_code = rep( "AU", 3),
                           year = c(2018:2020),
                           my_var = c(10,12,11),
                           description = c("note1", NA_character_,
                                            "note3")
  )

  downstream <- australia_states

  impute_down ( upstream_data = upstream,
                downstream_data = downstream,
                country_var = "country_code",
                regional_code = "geo_code",
                values_var = "values",
                time_var = "year",
                upstream_method_var = NULL,
                downstream_method_var = NULL)
}
```
impute_down_nuts

Imputing Data From Larger To Smaller Units in the EU NUTS

Description

This is a special case of impute_down for the EU NUTS hierarchical typologies. All valid actual rows will be projected down to all smaller constituent typologies where data is missing.

Usage

impute_down_nuts(
  dat,
  geo_var = "geo",
  values_var = "values",
  method_var = NULL,
  nuts_year = 2016
)

Arguments

dat       A data frame with exactly two or three columns: geo for the geo codes of the units, values for the values, and optionally method for describing the data source.
geo_var   The variable that contains the geographical codes in the NUTS typologies, defaults to code "geo_var".
values_var The variable that contains the upstream data to be imputed to the downstream data, defaults to "values".
method_var The variable that contains the metadata on various processing information, defaults to NULL in which case it will be returned as 'method'.
nuts_year The year of the NUTS typology to use, it defaults to the currently valid 2016. Alternative values can be any of these: 1999, 2003, 2006, 2010, 2013 and the already announced and defined 2021. For example, use 2013 for NUTS2013 data.

Details

The more general function requires typology information from the higher and lower level typologies. This is not needed when the EU vocabulary is used, and the hierarchy can be established from the EU vocabularies.

Be mindful that while all possible imputations are made, imputations beyond one hierarchical level will result in very crude estimates.

The imputed dataset dat must refer to a single time unit, i.e. panel data is not supported.
mixed_nuts_example

Value
An augmented version of the dat imputed data frame with all possible projections to valid smaller units, i.e. NUTS0 = country values imputed to all missing NUTS1 units, NUTS1 values imputed to all missing NUTS2 units, NUTS2 values imputed to all missing NUTS3 units.

See Also
Other impute functions: impute_down()

Examples

data(mixed_nuts_example)
impute_down_nuts(mixed_nuts_example, nuts_year = 2016)

mixed_nuts_example Example Data Frame: Mixed EU Typologies.

Description
This data frame is a fictious example that contains in a small, easy-to-review example many potential typological problems. It is used to test imputation functions and to create examples with them.

Usage
mixed_nuts_example

Format
A data frame with 22 rows and 3 variables:

- geo  NUTS geo identifier, mixed from 4 typology levels.
- values Random numbers.
- method Descriptive metadata.

Source
https://ec.europa.eu/eurostat/web/nuts/history/

See Also
nuts_changes, all_valid_nuts_codes, impute_down_nuts
**Description**

A dataset containing the joined correspondence tables of the EU NUTS typologies.

**Usage**

`nuts_changes`

**Format**

A data frame with 3097 rows and 22 variables:

- **typology**  country, NUTS1, NUTS2 or NUTS3
- **start_year** The year when the code was first used
- **end_year** The year when the code was last used
- **code_1999** NUTS code in the 2003 definition
- **code_2003** NUTS code in the 2003 definition
- **code_2006** NUTS code in the 2006 definition
- **code_2010** NUTS code in the 2010 definition
- **code_2013** NUTS code in the 2013 definition
- **code_2016** NUTS code in the 2016 definition
- **code_2021** NUTS code in the 2021 definition
- **geo_name_2003** NUTS territorial name in the 2003 definition
- **geo_name_2006** NUTS territorial name in the 2006 definition
- **geo_name_2010** NUTS territorial name in the 2010 definition
- **geo_name_2013** NUTS territorial name in the 2013 definition
- **geo_name_2016** NUTS territorial name in the 2016 definition
- **geo_name_2021** NUTS territorial name in the 2021 definition
- **change_2003** Change described in the 2003 correspondence table
- **change_2006** Change described in the 2006 correspondence table
- **change_2010** Change described in the 2010 correspondence table
- **change_2013** Change described in the 2013 correspondence table
- **change_2016** Change described in the 2016 correspondence table
- **change_2021** Change described in the 2021 correspondence table

**Source**

[https://ec.europa.eu/eurostat/web/nuts/history/](https://ec.europa.eu/eurostat/web/nuts/history/)

**See Also**

`nuts_recoded, all_valid_nuts_codes`
**nuts_exceptions**

**NUTS Coding Exceptions**

**Description**

A dataset containing exceptions to the NUTS geographical codes.

**Usage**

```r
nuts_exceptions
```

**Format**

A data frame with 2 variables:

- `geo` National and sub-national geographical codes from Eurostat
- `typology` Short description of exception

**Details**

They contain non-EU regions that are consistent with NUTS, but not defined within the NUTS. The also contain European country codes that do not conform with NUTS.

**Source**

Eurostat NUTS history: [https://ec.europa.eu/eurostat/web/nuts/history/](https://ec.europa.eu/eurostat/web/nuts/history/)

**See Also**

`nuts_recoded`, `nuts_changes`, `all_valid_nuts_codes`

**nuts_lau_2019**

**European Union: NUTS And LAU Correspondence**

**Description**

A dataset containing the joined correspondence tables of the EU NUTS and local administration units (LAU) typologies.

**Usage**

```r
nuts_lau_2019
```
Format

A data frame with 99140 rows and 22 variables:

code_2016  NUTS3 code of the local administrative unit, 2016 definition
lau_code   Local Administrative Unit code
lau_name_national  LAU name, official in national language(s)
lau_name_latin   LAU name, official Latin alphabet version
name_change_last_year  Change in name in the year before?
population  Population
total_area_m2  Area in square meters
degurba  Degree of urbanization
degurba_change_last_year  Change in degree of urbanization?
coastal_area  Part of coastal area classification?
coastal_change_last_year  Change in coastal area classification
city_id  NUTS territorial name in the 2006 definition
city_id_change_last_year  NUTS territorial name in the 2010 definition
city_name  Name of the city
greater_city_id  Containing metro area ID, if applicable
greater_city_id_change_last_year  Change in metro area ID
greater_city_name  Name of containing greater city (metropolitan) area, if applicable
fua_id  FUA ID
fua_id_change_last_year  Change of FUA ID since last year
fua_name  Name in FUA database
country  NUTS country code with exceptions: EL for Greece, UK for United Kingdom
gisco_id  GISCO ID

Details

This is also the authoritative vocabulary for local administration, names, including city and metropolitan area names.

Source

https://ec.europa.eu/eurostat/web/nuts/local-administrative-units

See Also

nuts_recoded, all_valid_nuts_codes
**nuts_recoded**

*European Union: Recoded NUTS units 1995-2021.*

**Description**

Containing all recoded NUTS units from the European Union. This is re-arranged from `nuts_changes`.

**Usage**

`nuts_recoded`

**Format**

A data frame with 8 rows and 3 variables:

- **geo**: NUTS geo identifier
- **typology**: country, NUTS1, NUTS2 or NUTS3
- **nuts_year**: year of the NUTS definition or version
- **change_year**: when the geo code changed
- **iso2c**: Two character ISO standard country codes.

**Source**

[https://ec.europa.eu/eurostat/web/nuts/history/](https://ec.europa.eu/eurostat/web/nuts/history/)

**See Also**

`nuts_changes, all_valid_nuts_codes`

---

**recode_nuts**

*Recode Region Codes From Source To Target NUTS Typology*

**Description**

Validate your geo codes, pair them with the appropriate standard typology, look up potential causes of invalidity in the EU correspondence tables, and look up the appropriate geographical codes in the other (target) typology. For example, validate geo codes in the 'NUTS2016' typology and translate them to the now obsolete the 'NUTS2010' typology to join current data with historical data sets.

**Usage**

`recode_nuts(dat, geo_var = "geo", nuts_year = 2016)`
Arguments

- **dat**: A data frame with a 3-5 character `geo_var` variable to be validated.
- **geo_var**: Defaults to "geo". The variable that contains the 3-5 character geo codes to be validated.
- **nuts_year**: The year of the NUTS typology to use. You can select any valid NUTS definition, i.e. 1999, 2003, 2006, 2010, 2013, the currently used 2016 and the already announced and defined 2021. Defaults to the current typology in force, which is 2016.

Value

The original data frame with a `geo_var` column is extended with a `typology` column that states in which typology is the `geo_var` a valid code. For invalid codes, looks up potential reasons of invalidity and adds them to the `typology_change` column, and at last it adds a column of character vector containing the desired codes in the target typology, for example, in the NUTS2013 typology.

See Also

Other recode functions: `get_country_code()`

Examples

```r
{  
  foo <- data.frame (  
    geo = c("FR", "DEE32", "UKI3",  
      "HUI2", "DEE",  
      "FRK"),  
    values = runif(6, 0, 100),  
    stringsAsFactors = FALSE )  

  recode_nuts(foo, nuts_year = 2013)  
}
```

regional_rd_personnel  

**regional_rd_personnel**  

**R&D Personnel by NUTS 2 Regions**

Description

A subset of the Eurostat dataset R&D personnel and researchers by sector of performance, sex and NUTS 2 regions.

Usage

`regional_rd_personnel`
Format

A data frame with 956 observations of 7 variables:

- **geo**: National and sub-national geographical codes from Eurostat
- **time**: Time, coded as a numeric variable of the year, 2006-2019
- **values**: The numeric statistical values
- **unit**: Unit of measurement, contains only FTE
- **sex**: Sex of researchers, contains only both sexes as T
- **prof_pos**: Professional position, contains all R&D employees not only researchers
- **sectperf**: Sector of performance, filtered for all sectors as TOTAL

Details

Mapping Regional Data, Mapping Metadata Problem

The fresh version of this statistic can be obtained by `eurostat::get_eurostat_json(id = "rd_p_persreg", filters = list(sex = "T", prof_pos = "TOTAL", sectperf = "TOTAL", unit = "FTE"))`

Source


See Also

recode_nuts

---

regions: A package for working with regional statistics.

Description

The regions package provides four categories of functions: validate, recode, impute and aggregate.

_validate functions_

The validate functions validate the conformity of a typological (geographical) label with a certain typology. Currently the EU statistical NUTS typologies and countries are implemented.

_recode functions_

These functions correct the geo coding of sub-national statistics, or bring them to a consistent format.
**impute functions**

The impute functions impute data from one regional unit to a different level of regional unit, such as a country level data to a province / state level data. `impute_down` provides imputation functions from higher aggregation hierarchy levels to lower ones, for example from ISO-3166-1 to ISO-3166-2. `impute_down_nuts` provides the same functionality with the EU typologies, but with far less work, because they rely on the internal hierarchical structure of these metadata, for example, from NUTS1 to NUTS2.

**aggregate functions**

Aggregation function from lower hierarchy levels to higher ones, for example from NUTS3 to NUTS1 or from ISO-3166-2 to ISO-3166-1. Disaggregation functions from higher hierarchy levels to lower ones, for example from NUTS1 to NUTS2 or from ISO-3166-1 to ISO-3166-2.

---

**validate_geo_code**

 Validate Conformity with NUTS Geo Codes (vector)

**Description**

Validate that `geo` is conforming with the NUTS1, NUTS2, or NUTS3 typologies. While country codes are technically not part of the NUTS typologies, Eurostat de facto uses a NUTS0 typology to identify countries. This de facto typology has three exception which are handled by the `validate_nuts_countries` function.

**Usage**

```r
validate_geo_code(geo, nuts_year = 2016)
```

**Arguments**

- `geo` A vector of geographical code to validate.
- `nuts_year` A valid NUTS edition year.

**Details**

NUTS typologies have different versions, therefore the conformity is validated with one specific version, which can be any of these: 1999, 2003, 2006, 2010, 2013, the currently used 2016 and the already announced and defined 2021.

The NUTS typology was codified with the NUTS2003, and the pre-1999 NUTS typologies may confuse programmatic data processing, given that some NUTS1 regions were identified with country codes in smaller countries that had no NUTS1 divisions.

Currently the 2016 is used by Eurostat, but many datasets still contain 2013 and sometimes earlier metadata.

**Value**

A character list with the valid typology, or 'invalid' in the cases when the geo coding is not valid.
Examples

```r
my_reg_data <- data.frame (
    geo = c("BE1", "HU102", "FR1", 
            "DED", "FR7", "TR", "DED2", 
            "EL", "XK", "GB"),
    values = runif(10))

validate_geo_code(my_reg_data$geo)
```

validate_nuts_countries

Validate Conformity with NUTS Country Codes

Description

This function is mainly a wrapper around the well-known `countrycode` function, with three exceptions that are particular to the European Union statistical nomenclature.

- **EL**: Treated valid, because NUTS uses EL instead of GR for Greece since 2010.
- **UK**: Treated valid, because NUTS uses UK instead of GB for the United Kingdom.
- **XK**: XK is used for Kosovo, because Eurostat uses this code, too.

All ISO-3166-1 country codes are validated, and also the three exceptions.

Usage

```r
validate_nuts_countries(dat, geo_var = "geo")
```

Arguments

- **dat**: A data frame with a 2-character geo variable to be validated
- **geo_var**: Defaults to "geo". The variable that contains the 2 character geo codes to be validated.

Value

The original data frame extended with the column 'typology'. This column states 'country' for valid country typology coding, or appropriate label for invalid ISO-3166-alpha-2 and ISO-3166-alpha-3 codes.

See Also

Other validate functions: `validate_nuts_regions()`
validate_nuts_regions

Examples

```r
my_dat <- data.frame(
  geo = c("AL", "GR", "XK", "EL", "UK", "GB", "NLD", "ZZ" ),
  values = runif(8)
)

## NLD is an ISO 3-character code and is not validated.
validate_nuts_countries(my_dat)
```

validate_nuts_regions  Validate Conformity With NUTS Geo Codes

Description

Validate that geo_var is conforming with the NUTS1, NUTS2, or NUTS3 typologies. While country codes are technically not part of the NUTS typologies, Eurostat de facto uses a NUTS0 typology to identify countries. This de facto typology has three exception which are handled by the validate_nuts_countries function.

Usage

```r
validate_nuts_regions(dat, geo_var = "geo", nuts_year = 2016)
```

Arguments

- **dat**: A data frame with a 3-5 character geo_var variable to be validated.
- **geo_var**: Defaults to "geo". The variable that contains the 3-5 character geo codes to be validated.
- **nuts_year**: The year of the NUTS typology to use. Defaults to 2016. You can select any valid NUTS definition, i.e. 1999, 2003, 2006, 2010, 2013, the currently used 2016 and the already announced and defined 2021.

Details

NUTS typologies have different versions, therefore the conformity is validated with one specific versions, which can be any of these: 1999, 2003, 2006, 2010, 2013, the currently used 2016 and the already announced and defined 2021.

The NUTS typology was codified with the NUTS2003, and the pre-1999 NUTS typologies may confuse programmatic data processing, given that some NUTS1 regions were identified with country codes in smaller countries that had no NUTS1 divisions.

Currently the 2016 is used by Eurostat, but many datasets still contain 2013 and sometimes earlier metadata.
validate_parameters

Value

Returns the original data frame with a column that specifies the conformity with the NUTS definition of the year nuts_year.

See Also

Other validate functions: validate_nuts_countries()

Examples

```r
my_reg_data <- data.frame(
    geo = c("BE1", "HU102", "FR1",
            "DED", "FR7", "TR", "DED2",
            "EL", "XK", "GB"),
    values = runif(10))
validate_nuts_regions (my_reg_data)
validate_nuts_regions (my_reg_data, nuts_year = 2013)
validate_nuts_regions (my_reg_data, nuts_year = 2003)
```

---

validate_parameters  Assertion for Correct Function Calls

Description

Assertions are made to give early and precise error messages for wrong API call parameters.

Usage

```r
validate_parameters(typology = NULL, param = NULL, param_name = NULL)
```

Arguments

- **typology**: Currently the following typologies are supported: "NUTS1", "NUTS2", "NUTS3" or "NUTS" for any of the NUTS typologies. The technical typology "NUTS0" can be used to translate Eurostat country codes to ISO 3166-1 alpha-2 country codes.
- **param**: A parameter value that must not be NULL.
- **param_name**: The name of the parameter that must not have a value of NULL.
validate_parameters

Details

These assertions are called from various wrapper functions. However, you can also call this function
directly to make sure that you are adding (programmatically) the correct parameters to a call.
All validate_parameters parameters default to NULL. Asserts the correct parameter values for any
values that are not NULL.

Value

A boolean, logical variable if the parameter calls are valid.
Index

* datasets
  all_valid_nuts_codes, 2
  australia_states, 3
  daily_internet_users, 4
  google_nuts_matchtable, 5
  mixed_nuts_example, 9
  nuts_changes, 10
  nuts_exceptions, 11
  nuts_lau_2019, 11
  nuts_recoded, 13
  regional_rd_personnel, 14

* impute functions
  impute_down, 6
  impute_down_nuts, 8

* recode functions
  get_country_code, 4
  recode_nuts, 13

* validate functions
  validate_nuts_countries, 17
  validate_nuts_regions, 18

all_valid_nuts_codes, 2
australia_states, 3
countrycode, 17
create_nuts_lau_2019, 3
daily_internet_users, 4
get_country_code, 4, 14
google_nuts_matchtable, 5
impute_down, 6, 8, 9, 16
impute_down_nuts, 7, 8, 16
mixed_nuts_example, 9
nuts_changes, 2, 10, 13
nuts_exceptions, 11
nuts_lau_2019, 11
nuts_recoded, 13
recode_nuts, 5, 13
regional_rd_personnel, 14
regions, 15
validate_geo_code, 16
validate_nuts_countries, 16, 17, 18, 19
validate_nuts_regions, 17, 18
validate_parameters, 19, 20

21