Package 'countries'

May 8, 2026

Type Package

Title Deal with Country Data in an Easy Way

Version 1.2.2

Description Wrangle country data more effectively and quickly.
This package contains functions to easily identify and convert country names, download country information, merge country data from different sources, and make quick world maps.

License GPL-3

Encoding UTF-8

RoxygenNote 7.2.3

Imports stringdist, tidyr, stringr, dplyr, knitr, fastmatch, lubridate, utils, methods, stats, httr, jsonlite, ggplot2, viridis, grDevices

Suggests data.table, rmarkdown, testthat, curl

Depends R (>= 2.10)

VignetteBuilder knitr

URL https://fbellelli.github.io/countries/, https://github.com/fbellelli/countries

BugReports https://github.com/fbellelli/countries/issues

Config/testthat/edition 3

LazyData true

NeedsCompilation no

Author Francesco Saverio Bellelli [aut, cre, cph] (Website: https://fbellelli.com/)

Maintainer Francesco Saverio Bellelli <fsabellelli@gmail.com>

Repository CRAN

Date/Publication 2025-06-14 11:00:02 UTC

Contents

`auto_melt` *Automatic pivoting of country and year columns to a long format*

Description

When at least 3 country names or years are found in the column names, the function will automatically transform the table from a wide to a long format by pivoting the country/year columns. This is equivalent to applying `tidyr::pivot_longer()` or `data.table::melt()` on the columns with years or countries as names. The function is able to detect years also when they are preceded by a prefix.

Usage

```
auto_melt(
  x,
  names_to = "pivoted_colnames",
  values_to = "pivoted_data",
  verbose = TRUE,
  pivoting_info = FALSE
)
```

Arguments

x	A data.frame object to check and pivot country or year columns.
names_to	String indicating how the column holding the name of the pivoted columns should be called in the output table. Default is "pivoted_colnames"
values_to	String indicating how the column containing the values of the pivoted columns should be called in the output table. Default is "pivoted_data"
verbose	Logical value. If set to TRUE (the default), a message will be displayed on the console indicating which columns are being pivoted. If set to FALSE, the messages are turned off.
pivoting_info	Logical value indicating whether to return the list of names of the column that have been pivoted. Default is FALSE. If set to TRUE, the output will be a list instead of simple data.frame. Teh list will contain 1) the pivoted table, 2) the list of pivoted columns.

Value

A table transformed into a "long" format by pivoting country or year columns. If year columns are found, a numeric column called "year_pivoted_colnames" is added isolating the years extracted from the table header's.

See Also

auto_merge, find_countrycol,find_timecol

Examples

```
# example data
example <- data.frame(Date = c("01.01.2019", "01.02.2019", "01.03.2019"),
                      Japan = 1:3,
                      Norway = 2:4,
                      Germany = 3:5,
                      US = 4:6)
example2 <- data.frame(Sector = c("Agriculture", "Mining", "Forestry"),
                       X2000 = 1:3,
                       X2001 = 2:4,
                       X2002 = 3:5,
                       X2003 = 4:6)

# examples pivotting countries and years from column names
auto_melt(example)
auto_melt(example2)
```

`auto_merge` *Simplified merging supporting different country nomenclatures and date formats*

Description

The aim of this function is to simplify country data merging for quick analyses. Compared to a normal merge function `auto_merge()`:

- Is able to perform the merging of multiple data tables at once.
- Supports automatic detection of columns to merge.
- It is able to handle different country naming conventions and date formats. For example, it will be able to recognise that "Italy" and "ITA" refer to the same country and will merge the two entries across tables.
- It detects if data is in a wide format with country names or years in the column names and will automatically pivot the data.

Usage

```
auto_merge(
  ...,
  by = NULL,
  country_to = "ISO3",
  inner_join = FALSE,
  merging_info = FALSE,
  verbose = TRUE,
  auto_melt = TRUE
)
```

Arguments

`...`	Data to be merged. Inputs need to be data frames or coercible to data frame. Tables can also be provided into a single list e.g. `tab1, tab2, tab3` or `list(tab1, tab2, tab3)`.
`by`	A list or a vector indicating the columns to be used for merging the data. *If not provided, the function will try to automatically detect columns to be merged*. For more information, refer to the details sections.
`country_to`	Nomenclature to which country names should be converted to in the output. Default is `ISO3`. For a description of possible options, refer to the table in the vignette Dealing with country names.
`inner_join`	Logical value indicating whether to perform an inner join. The default is `FALSE`, which results in a full join of the provided tables.
`merging_info`	Logical value. If `TRUE`, the function will output a list containing the merged data and information generated during the merging process, such as information on columns that have been merged or the conversion table used for country names. The default is `FALSE`, which results into a single merged table being returned.

verbose — Logical value indicating whether to print status messages on the console. Default is TRUE.

auto_melt — Logical value indicating whether to automatically pivot country names or years present in the column names. Default is TRUE. When at least 3 country names or years are found in the column names, the function will automatically transform the table from a wide to a long format by pivoting the country/year columns.

Details

Automatic detection of columns to merge. The automatic detection process starts by first identifying the key of each table, i.e. a set of variables identifying the entries in the table. This process is optimised for common formats of country data. The function will then try to match key columns across tables based on their values. Columns containing country names and time information are identified and are processed to take into account different nomenclatures and time formats. This automatic process works for the most common dataset structures, but it is not foolproof. Therefore, we always advise to check the columns that are being merged by setting `verbose = TRUE` and reading the printout. Moreover, users should be aware that this automatic detection process can increase the overall merging time considerably. This can be especially long for tables containing many columns or when a large number of tables is being merged.

Formatting of `by` argument If an argument is provided to `by`, it needs to be either 1) a list of column names, or 2) a vector of regular expressions. The format requirements are the following:

1. In case a **list** is passed, each element of the list must be a vector of length equal to the number of tables being merged (i.e., if 3 tables are being merged, the list needs to contain all vectors of length 3). The vectors should contain the names of columns to be merged in each table, `NA` can be inserted for tables that do not contain the variable, and names should be ordered in the same order of the tables that are being merged (i.e. the first column name should be present in the first table being merged). The name of the merged columns can be modified by assigning a name to the elements of the list. For example, `list("countries"=c("Nation",NA,"COUNTRY"), "sector"=c("Industry","industry",NA))` is requesting to merge the columns `tab1$Nation` and `tab3$COUNTRY`, and the columns `tab1$Industry` and `tab2$industry`. These two merged columns will be named `"countries"` and `"sector"` in the output, as requested by the user.
2. In case a **vector** is passed, each element is interpreted as a regular expression to be used for matching the columns to be merged. For example, the same order provided in the list example could be written as `c("countries"="Nation|COUNTRY", "sector"="[Ii]ndustry")`. This will merge the first column in each table whose name matches the pattern described by the regular expression and will name the two resulting columns as `"countries"` and `"sector"` respectively.

Value

If `merging_info = FALSE` a single merged table is returned. If `merging_info = TRUE`, a list object is returned, containing the merged table (`merged_table`), a table summarising which columns have been merged (`info_merged_columns`), a table summarising the conversion of country names (`info_country_names`), a table summarising the conversion of time columns to a common format (`info_time_formats`), a list of all the columns that have been pivoted when wide tables with country or years in column names were detected (`pivoted_columns`), a list recapitulating the inputs passed to the function (`call`).

See Also

country_name, find_keycol

Examples

```
# sample data
tab1 <- data.frame(Industry = c(1, 1, 2, 2), Nation = c("ITA", "FRA", "ITA", "FRA"), tot = runif(4))
tab2 <- data.frame(industry = 1:4, rate = runif(1:4))
tab3 <- data.frame(COUNTRY = c("United States", "France", "India"), national_avg = runif(3))

# examples of merging orders
auto_merge(tab1, tab2, tab3)
auto_merge(list(tab1, tab2, tab3))
auto_merge(tab1, tab2, tab3, by = c("countries"="Nation|COUNTRY", "sector"="[Ii]ndustry"))
auto_merge(tab1, tab2, tab3, country_to = "UN_fr")
```

---

check_countries_api *Check if the connection to Countries REST API is working*

Description

Check if the connection to REST Countries API is working. The function checks if the user has an internet connection and if any answer is returned from the Countries REST API.

Usage

```
check_countries_api(warnings = TRUE, timeout = 4)
```

Arguments

warnings	Logical value indicating whether to output a warning when there is no connection. Default is TRUE.
timeout	Numeric value giving the timeout in seconds for attempting connection to the API. Default is 4 second.

Value

Returns a logical value: TRUE if there is a connection, FALSE if there is no connection.

See Also

list_fields, country_info

Examples

```
check_countries_api()
```

check_wide_format *Check for wide country data formats*

Description

The function looks for country names or year information in the column names. This function is designed for simple panel country data, in which countries' time series are arranged side by side on columns or stacked on rows. The function will only return year/country column names if at least 3 country/year column names are detected.

Usage

```
check_wide_format(x, adjacency = TRUE)
```

Arguments

x	A dataframe
adjacency	Logical value indicating whether column names containing country or year information need to be adjacent to each other. Default is TRUE

Value

Returns a data.frame identifying the columns names that contain country or year information.

See Also

find_keycol, find_countrycol, find_timecol

Examples

```
example <- data.frame(Year=2000:2010, China=0:10, US=10:20, Vietnam=30:40)
check_wide_format(x=example)
```

country_info *Get information about countries*

Description

This function is an interface for REST Countries API. It allows to request and download information about countries, such as: currency, capital city, language spoken, flag, neighbouring countries, and much more. **NOTE:** Internet access is needed to download information from the API. At times the API may be unstable or slow to respond.

Usage

```
country_info(
  countries = NULL,
  fields = NULL,
  fuzzy_match = TRUE,
  match_info = FALSE,
  collapse = TRUE,
  base_url = "restcountries.com:8080/v3.1/"
)
```

Arguments

countries	A vector of countries for which we wish to download information. The function also supports fuzzy matching capabilities to facilitate querying. Information is only returned for the 249 countries in the ISO standard 3166.
fields	Character vector indicating the fields to query. A description of the accepted fields can be found here. Alternatively, a list of accepted field names can be obtained with the function list_fields().
fuzzy_match	Logical value indicating whether to allow fuzzy matching of country names. Default is TRUE.
match_info	Logical value indicating whether to return information on country names matched to each input in countries. If TRUE, two additional columns will be added to the output (matched_country and is_country). Default is FALSE.
collapse	Logical value indicating whether to collapse multiple columns relating to a same field together. Default is TRUE. For some specific fields (currencies, languages, names), multiple columns will be returned. This happens because countries can take multiple values for these fields. For example, country_info("Switzerland", "languages", collapse = FALSE) will return 4 columns for the field languages. When collapse = TRUE, these four columns will be collapsed into one string, with values separated by semicolons.
base_url	Base URL used to construct the API calls. The default is "restcountries.com:8080/v3.1/".

Value

Returns the requested information about the countries in a table. The rows of the table correspond to entries in countries, columns correspond to requested fields.

See Also

list_fields, check_countries_api

Examples

```
# Run examples only if a connection to the API is available:
if (check_countries_api(warnings = FALSE)){

# The example below queries information on the currency used in Brazil, US and France:
info <- country_info(countries = "Brazil", fields = "capital")
```

```
# data for multiple countries can be requested
info <- country_info(countries = c("Brazil", "USA", "FR"), fields = "capital")

#' # Data can be returned for all countries by leaving - countries - empty
info <- country_info(fields = "capital")

# All available fields can be requested by leaving fields empty
info <- country_info(countries = c("Brazil", "USA", "FR"))

# All information for all countries can be downloaded by leaving both arguments empty
info <- country_info()

}
```

`country_name` *Convert and translate country names*

Description

This function recognises and converts country names to different nomenclatures and languages using a fuzzy matching algorithm. `country_name()` can identify countries even when they are provided in mixed formats or in different languages. It is robust to small misspellings and recognises many alternative country names and old nomenclatures.

Usage

```
country_name(
  x,
  to = "ISO3",
  fuzzy_match = TRUE,
  verbose = FALSE,
  simplify = TRUE,
  poor_matches = FALSE,
  na_fill = FALSE,
  custom_table = NULL
)
```

Arguments

`x`	A vector of country names
`to`	A string containing the desired naming conventions to which `x` should be converted to (e.g. `"ISO3"`, `"name_en"`, `"UN_fr"`, ...). For a list of all possible values click here or refer to the vignette on country names `vignette("dealing_with_names")`. Default is `"ISO3"`, which converts to the 249 3-letter country codes currently in ISO standard 3166.
`fuzzy_match`	Logical value indicating whether fuzzy matching of country names should be allowed (`TRUE`), or only exact matches are allowed (`FALSE`). Default is `TRUE`.

verbose	Logical value indicating whether the function should print to the console a full report. Default is FALSE.
simplify	Logical value. If set to TRUE the function will return a vector of converted names. If set to FALSE, the function will return a list object containing the converted vector and additional details on the country matching process. Default is TRUE.
poor_matches	Logical value. If set to FALSE (the default), the function will return NA in case of poor matching. If set to TRUE, the function will always return the closest matching country name, even if the match is poor.
na_fill	Logical value. If set to TRUE, any NA in the output names will be filled with the original country name supplied in x. The default is FALSE (no filling). In general, NAs are produced if: 1) the country is not present in the nomenclature requested in to (e.g. country_name("Abkhazia", to = "ISO3")), 2) the input country name is NA, 3) No exact match is found and the user sets the option fuzzy_match = FALSE, 4) When the fuzzy match algorithm does not find a good match and the user sets the option poor_match = FALSE. The na_fill argument gives the option to replace the resulting NA with the original value in x.
custom_table	Custom conversion table to be used. This needs to be a data.frame object. Default is NULL.

Value

Returns a vector of converted country names. If multiple nomenclatures are passed to the argument to, the vectors are arranged in a data frame. If simplify=FALSE, the function will return a list object.

See Also

is_country, match_table, find_countrycol

Examples

```
#Convert country names to a single nomenclatures: (e.g. 3-letters ISO code)
country_name(x=c("UK","Estados Unidos","Zaire","C#te d^ivoire"), to= "ISO3")

#When multiple arguments are provided to the - to - argument, a data frame is returned:
country_name(x=c("UK","Estados Unidos","Zaire","C#te d^ivoire"), to= c("UN_en","UN_fr","ISO3"))

#This function can also be used to translate country names: (e.g. translating all to Chinese)
country_name(x=c("UK","Estados Unidos","Zaire","C#te d^ivoire"), to= "name_zh")
```

---

country_reference_list

Conversion table

Description

A table containing country names in different naming conventions

Usage

```
country_reference_list
```

Format

A data frame with columns corresponding to different country naming conventions.

simple Reference name for the geographic unit. The names in this column contain only ASCII characters. This nomenclature is available for all countries.

ISO3 3-letter country codes as defined in ISO standard `3166-1 alpha-3`. This nomenclature is available for the territories in the standard (currently 249 territories).

ISO2 2-letter country codes as defined in ISO standard `3166-1 alpha-2`. This nomenclature is available for the territories in the standard (currently 249 territories).

ISO_code Numeric country codes as defined in ISO standard `3166-1 numeric`. This country code is the same as the UN's country number (M49 standard). This nomenclature is available for the territories in the ISO standard (currently 249 countries).

UN_ar Official UN name in **Arabic**. This nomenclature is only available for countries in the M49 standard (currently 249 territories).

UN_zh Official UN name in **Chinese**. This nomenclature is only available for countries in the M49 standard (currently 249 territories).

UN_en Official UN name in **English**. This nomenclature is only available for countries in the M49 standard (currently 249 territories).

UN_fr Official UN name in **French**. This nomenclature is only available for countries in the M49 standard (currently 249 territories).

UN_es Official UN name in **Spanish**. This nomenclature is only available for countries in the M49 standard (currently 249 territories).

UN_ru Official UN name in **Russian**. This nomenclature is only available for countries in the M49 standard (currently 249 territories).

WTO_en Official WTO name in **English**. This nomenclature is only available for WTO members and observers (currently 189 entities).

WTO_fr Official WTO name in **French**. This nomenclature is only available for WTO members and observers (currently 189 entities).

WTO_es Official WTO name in **Spanish**. This nomenclature is only available for WTO members and observers (currently 189 entities).

name_ar Translation of ISO country names in **Arabic**. (currently 249 territories)

name_bg Translation of ISO country names in **Bulgarian**. (currently 249 territories)

name_cs Translation of ISO country names in **Czech**. (currently 249 territories)

name_da Translation of ISO country names in **Danish**. (currently 249 territories)

name_de Translation of ISO country names in **German**. (currently 249 territories)

name_el Translation of ISO country names in **Greek**. (currently 249 territories)

name_en Translation of ISO country names in **English**. (currently 249 territories)

name_es Translation of ISO country names in **Spanish**. (currently 249 territories)

name_et Translation of ISO country names in **Estonian**. (currently 249 territories)

name_eu Translation of ISO country names in **Basque**. (currently 249 territories)

name_fi Translation of ISO country names in **Finnish**. (currently 249 territories)

name_fr Translation of ISO country names in **French**. (currently 249 territories)

name_hu Translation of ISO country names in **Hungarian**. (currently 249 territories)

name_it Translation of ISO country names in **Italian**. (currently 249 territories)

name_ja Translation of ISO country names in **Japanese**. (currently 249 territories)

name_ko Translation of ISO country names in **Korean**. (currently 249 territories)

name_lt Translation of ISO country names in **Lithuanian**. (currently 249 territories)

name_nl Translation of ISO country names in **Dutch**. (currently 249 territories)

name_no Translation of ISO country names in **Norwegian**. (currently 249 territories)

name_pl Translation of ISO country names in **Polish**. (currently 249 territories)

name_pt Translation of ISO country names in **Portuguese**. (currently 249 territories)

name_ro Translation of ISO country names in **Romanian**. (currently 249 territories)

name_ru Translation of ISO country names in **Russian**. (currently 249 territories)

name_sk Translation of ISO country names in **Slovak**. (currently 249 territories)

name_sv Translation of ISO country names in **Swedish**. (currently 249 territories)

name_th Translation of ISO country names in **Thai**. (currently 249 territories)

name_uk Translation of ISO country names in **Ukranian**. (currently 249 territories)

name_zh Translation of ISO country names in **simplified Chinese**. (currently 249 territories)

name_zh-tw Translation of ISO country names in **traditional Chinese**. (currently 249 territories)

GTAP GTAP country and region codes.

Name0 Other variants of the country name included to improve the matching process

Name1 Other variants of the country name included to improve the matching process

Name2 Other variants of the country name included to improve the matching process

Name3 Other variants of the country name included to improve the matching process

Name4 Other variants of the country name included to improve the matching process

Name5 Other variants of the country name included to improve the matching process

Name6 Other variants of the country name included to improve the matching process

Name7 Other variants of the country name included to improve the matching process

Name8 Other variants of the country name included to improve the matching process

Name9 Other variants of the country name included to improve the matching process

Name10 Other variants of the country name included to improve the matching process

Name11 Other variants of the country name included to improve the matching process

Name12 Other variants of the country name included to improve the matching process

Name13 Other variants of the country name included to improve the matching process

Name14 Other variants of the country name included to improve the matching process

Name15 Other variants of the country name included to improve the matching process

Name16 Other variants of the country name included to improve the matching process

Name17 Other variants of the country name included to improve the matching process

Name18 Other variants of the country name included to improve the matching process

Name19 Other variants of the country name included to improve the matching process

---

`country_reference_list_long` *Conversion table in long format*

Description

A table containing country names in different naming conventions

Usage

```
country_reference_list_long
```

Format

A data frame with three columns providing information on country naming conventions. This table is a long-format version of "country_reference_list".

ID Numeric value that uniquely identifies entity. This corresponds to the row number in table "country_reference_list".

nomenclature Country naming convention (e.g. UN english, ISO 3-digit code, etc.).

name Country names

`find_countrycol` *Finds columns containing country names*

Description

This function takes a data frame as argument and returns the column name (or index) of all columns containing country names. It can be used to automate the search of country columns in data frames. For the purpose of this function, a country is any of the 249 territories designated in the ISO standard `3166`. On large datasets a random sample is used for evaluating the columns.

Usage

```
find_countrycol(
  x,
  return_index = FALSE,
  allow_NA = TRUE,
  min_share = 0.6,
  sample_size = 1000
)
```

Arguments

x	A data frame object
return_index	A logical value indicating whether the function should return the index of country columns instead of the column names. Default is FALSE, column names are returned.
allow_NA	Logical value indicating whether columns containing NA values are to be considered as country columns. Default is allow_NA=FALSE, the function will not return country column containing NA values.
min_share	A value between 0 and 1 indicating the minimum share of country names in columns that are returned. A value of 0 will return any column containing a country name. A value of 1 will return only columns whose entries are all country names. Default is 0.6, i.e. at least 60 percent of the column entries need to be country names.
sample_size	Either NA or a numeric value indicating the sample size used for evaluating columns. Default is 1000. If NA is passed, the function will evaluate the full table. The minimum accepted value is 100 (i.e. 100 randomly sampled rows are used to evaluate the columns). This parameter can be tuned to speed up computation on long datasets. Taking a sample could result in inexact identification of key columns, accuracy improves with larger samples.

Value

Returns a vector of country names (return_index=FALSE) or column indices (return_index=TRUE) of columns containing country names.

See Also

is_country, country_name, find_keycol, find_timecol

Examples

```
find_countrycol(x=data.frame(a=c("Brésil","Tonga","FRA"), b=c(1,2,3)))
```

---

find_keycol *Find a set of columns that uniquely identifies table entries*

Description

This function takes a data frame as argument and returns the column names (or indices) of a set of columns that uniquely identify the table entries (i.e. table key). It can be used to automate the search of table keys. Since the function was designed for country data, it will first search for columns containing country names and dates/years. These columns will be given priority in the search for keys. Next, the function prioritises left-most columns in the table. For time efficiency, the function does not test all possible combination of columns, it just tests the most likely combinations. The function will look for the most common country data formats (e.g. cross-sectional, time-series, panel data, dyadic, etc.) and searches for up to 2 additional key columns beyond country and time columns.

Usage

```
find_keycol(
  x,
  return_index = FALSE,
  search_only = NA,
  sample_size = 1000,
  allow_NA = FALSE
)
```

Arguments

x	A data frame object
return_index	A logical value indicating whether the function should return the index of country columns instead of the column names. Default is FALSE, column names are returned.
search_only	This parameter can be used to restrict the search of table keys to a subset of columns. The default is NA, which will result in the entire table being searched. Alternatively, users may restrict the search by providing a vector containing the name or the numeric index of columns to check. For example, search could be restricted to the first ten columns by passing 1:10. This could be useful in speeding up the search in wide tables.
sample_size	Either NA or a numeric value indicating the sample size used for evaluating columns. Default is 1000. If NA is passed, the function will evaluate the full table. The minimum accepted value is 100 (i.e. 100 randomly sampled rows are used to evaluate the columns). This parameter can be tuned to speed up computation on long datasets. Taking a sample could result in inexact identification of key columns, accuracy improves with larger samples.
allow_NA	Logical value indicating whether to allow key columns to have NA values. Default is allow_NA=FALSE. If set to TRUE, NA is considered as a distinct value.

Value

Returns a vector of column names (or indices) that uniquely identify the entries in the table. If no key is found, the function will return NULL. The output is a named vector indicating whether the identified key columns contain country names ("country"), year and dates ("time"), or other type of information ("other").

See Also

find_timecol, find_countrycol, is_keycol

Examples

```
example <-data.frame(nation=rep(c("FRA","ALB","JOR"),3),
                     year=c(rep(2000,3),rep(2005,3),rep(2010,3)),
                     var=runif(9))
find_keycol(x=example)
```

`find_timecol` *Finds columns containing date and year data*

Description

This function takes a data frame as argument and returns the column names (or indices) of all columns containing dates and the most likely column containing year information, if any. It can be used to automate the search of date and year columns in data frames.

Usage

```
find_timecol(x, return_index = FALSE, allow_NA = TRUE, sample_size = 1000)
```

Arguments

`x`	A data frame object
`return_index`	A logical value indicating whether the function should return the index of time columns instead of the column names. Default is `FALSE`, column names are returned.
`allow_NA`	Logical value indicating whether to allow time columns to contain `NA` values. Default is `allow_NA=FALSE`, the function will not return time column containing `NA` values.
`sample_size`	Either `NA` or a numeric value indicating the sample size used for evaluating columns. Default is `1000`. If `NA` is passed, the function will evaluate the full table. The minimum accepted value is `100` (i.e. 100 randomly sampled rows are used to evaluate the columns). This parameter can be tuned to speed up computation on long datasets. Taking a sample could result in inexact identification of key columns, accuracy improves with larger samples.

Value

Returns a vector of names (`return_index=FALSE`) or indices (`return_index=TRUE`) of columns containing date or year information. Only the most likely year column is returned.

See Also

is_date, find_countrycol

Examples

```
find_timecol(x=data.frame(a=1970:2020, year=1970:2020, b=rep("2020-01-01",51),c=sample(1:1000,51)))
```

`is_country` *Tests whether a string is a country name*

Description

This function checks whether the string is a country name. It supports different languages and naming conventions. The function returns `TRUE` if it relates to one of the 249 countries currently in the ISO standard `3166`. Alternatively, the argument `check_for` allows to narrow down the test to a subset of countries. Fuzzy matching can be used to allow a small margin of error in the string.

Usage

```
is_country(x, check_for = NULL, fuzzy_match = FALSE)
```

Arguments

`x`	A character vector to be tested (also supports UN/ISO country codes)
`check_for`	A vector of country names to narrow down testing. The function will return `TRUE` only if the string relates to a country in this vector. Default is NULL.
`fuzzy_match`	A logical value indicating whether to tolerate small discrepancies in the country name matching. The default and fastest option is `FALSE`.

Value

Returns a logical vector indicating whether the string is a country name

See Also

match_table, country_name, find_countrycol

Examples

```
#Detect strings that are country names
is_country(x=c("ITA","Estados Unidos","Estado Unidos","bungalow","dog",542), fuzzy_match=FALSE)
is_country(x=c("ITA","Estados Unidos","Estado Unidos","bungalow","dog",542), fuzzy_match=TRUE)
#Checking for a specific subset of countries
is_country(x=c("Ceylon","LKA","Indonesia","Inde"), check_for=c("India","Sri Lanka"))
```

`is_date` *Test whether the input is a date*

Description

This function checks if a value is a date by attempting to convert it to a date format. The user can specify which date formats should be tested with the argument `formats`.

Usage

```
is_date(
  x,
  formats = c("%Y-%m-%d", "%y-%m-%d", "%m-%d-%Y", "%m-%d-%y", "%d-%m-%Y",
    "%d-%m-%y", "%Y/%m/%d", "%y/%m/%d", "%m/%d/%Y", "%m/%d/%y",
    "%d/%m/%Y", "%d/%m/%y", "%Y.%m.%d", "%y.%m.%d", "%m.%d.%Y",
    "%m.%d.%y", "%d.%m.%Y", "%d.%m.%y", "%d %b %Y", "%d %B %Y",
    "%b %d %Y", "%B %d %Y", "%b %d, %Y", "%B %d, %Y", "%d%b%Y",
    "%d%B%Y", "%Y%B%d", "%Y%b%d", "%b %Y", "%B %Y", "%b %y", "%B %y",
    "%m-%Y", "%Y-%m", "%m/%Y", "%Y/%m")
)
```

Arguments

`x`	A vector of values to be tested
`formats`	Date formats to be checked for (expressed in R date notation).

Value

Returns a logical vector indicating whether the values can be converted to any of the date formats provided. Notice that unless specified, the default allowed formats do not include simple year numbers (e.g. 2022 or 1993) because number vectors could wrongly be identified as dates. Also, notice that testing `NA` values will return `FALSE`.

See Also

find_timecol, find_keycol, is_country

Examples

```
is_date(c("2020-01-01","test",2020,"March 2030"))
```

`is_keycol` *Test whether a set of column could be a data frame key*

Description

This function takes a data frame and a vector of column names as argument and returns a logical value indicating whether the indicated columns uniquely identify entries in the data frame. If the output is `TRUE`, the indicated columns could be the keys of the table.

Usage

```
is_keycol(x, cols, allow_NA = FALSE, verbose = TRUE)
```

Arguments

`x`	A data frame object
`cols`	A vector of column names or indices to be tested.
`allow_NA`	Logical value indicating whether to allow key columns to have `NA` values. Default is `allow_NA=FALSE`, the function will return `FALSE` if any `NA` value is present in `colnames`.
`verbose`	Logical value indicating whether messages should be printed on the console. Default is `TRUE`.

Value

Returns a logical value. If `TRUE`, the columns indicated in `colnames` uniquely identify the entries in `x`.

See Also

find_keycol, find_countrycol, find_timecol

Examples

```
is_keycol(data.frame(a=1:10,b=sample(c("a","b","c"),10, replace=TRUE)), cols="a")
is_keycol(data.frame(a=1:10,b=sample(c("a","b","c"),10, replace=TRUE)), cols="b")
is_keycol(
data.frame(a=c(1:5,1:5),
b=sample(c("a","b","c"),10, replace=TRUE),
c=c(rep("a",5),rep("b",5))),
cols=c("a","c"))
```

list_countries *Get a list of country names*

Description

This function returns a vector of country names in different nomenclatures.

Usage

```
list_countries(nomenclature = "name_en")
```

Arguments

nomenclature	String indicating the nomenclature from which the list of countries should be taken. Not all countries are present in all nomenclatures, for example Taiwan is not recognised by the UN, so it will not be returned with "WTO_en". The function accepts any of the nomenclatures supported country_name. For a list of accepted values, refer to this page. The default is name_en, which is the English list of names in the ISO standard 3166.

Value

A vector of country names in the desired nomenclature.

See Also

random_countries, country_name

Examples

```
list_countries("ISO3")
list_countries("UN_en")
list_countries()
```

list_fields *List of accepted fields for the function country_info*

Description

This function queries REST Countries API and returns a list of all possible fields that can be used in the function country_info. **NOTE:** Internet access is needed to download information from the API.

Usage

```
list_fields()
```

Value

A vector of accepted fields for the function `country_info()`

See Also

country_info

Examples

```
# Run example only if a connection to the API is available
if (check_countries_api(warnings = FALSE)){

list_fields()

}
```

`match_table` *Create a conversion table for country names*

Description

This function returns a conversion table for country names to the desired naming conventions and languages. The use of fuzzy matching allows more flexibility in recognising and identifying country names.

Usage

```
match_table(
  x,
  to = c("simple", "ISO3"),
  fuzzy_match = TRUE,
  verbose = FALSE,
  matching_info = FALSE,
  simplify = TRUE,
  na_fill = FALSE,
  poor_matches = TRUE,
  custom_table = NULL
)
```

Arguments

`x`	A vector of country names
`to`	A vector containing one or more desired naming conventions to which `x` should be converted to (e.g. `"ISO3"`, `"name_en"`, `"UN_fr"`, ...). For a list of all possible values click here or refer to the vignette on country names `vignette("dealing_with_names")`. Default is `c("simple", "ISO3")`.

fuzzy_match	Logical value indicating whether fuzzy matching of country names should be allowed (TRUE), or only exact matches are allowed (FALSE). Default is TRUE. Switching to FALSE will result in much faster execution.
verbose	Logical value indicating whether the function should print to the console a report on the matching process. Default is FALSE.
matching_info	Logical value. If set to true the output match table will include additional information on the matching of x's entries. Default is FALSE.
simplify	Logical value. If set to TRUE the function will return the match table as a data.frame object. If set to FALSE, the function will return a list object containing the match table and additional details on the country matching process. Default is TRUE.
na_fill	Logical value. If set to TRUE, any NA in the output names will be filled with the original country name supplied in x. The default is FALSE (no filling). In general, NAs are produced if: 1) the country is not present in the nomenclature requested in to (e.g. country_name("Abkhazia", to = "ISO3")), 2) the input country name is NA, 3) No exact match is found and the user sets the option fuzzy_match = FALSE, 4) When the fuzzy match algorithm does not find a good match and the user sets the option poor_match = FALSE. The na_fill argument gives the option to replace the resulting NA with the original value in x.
poor_matches	Logical value. If set to TRUE (the default option), the function will always return the closest matching country name, even if the matching is poor. If set to FALSE, the function will return NA in case of poor matching.
custom_table	Custom conversion table to be used. This needs to be a data.frame object. Default is NULL.

Value

Returns a conversion table for countries names to the desired naming conventions. If simplify=FALSE it returns a list object.

See Also

country_name, is_country

Examples

```
match_table(x=c("UK","Estados Unidos","Zaire","C#te d^ivoire"), to= c("UN_en","ISO3"))
```

Mode *Statistical mode of a vector*

Description

This function returns the mode of vectors. That is to say, for any given vector of values, it returns the value that appears most frequently. The function works with strings, numerical and mixed inputs. NA values are treated as distinct values.

Usage

```
Mode(x, na.rm = FALSE, first_only = FALSE)
```

Arguments

x	A vector
na.rm	Logical value indicating whether NA values should be omitted. Default is FALSE.
first_only	Logical value indicating whether only the first mode should be returned if x has multiple modes (i.e. there are multiple values with the highest number of observations). Default is FALSE.

Value

Returns the mode of the vector x

Examples

```
countries::Mode(c("a","a",2,3))
countries::Mode(c(1,1,2,3,NA,2))
countries::Mode(c(NA,NA,NA,1,1,2))
```

---

palettes_countries *Discrete colour palettes*

Description

This function provides access to the discrete colours palettes used in this packages' 11 themes.

Usage

```
palettes_countries(n, theme = 1, reverse = FALSE)
```

Arguments

n	Number of desired colours
theme	A numeric value or name identifying the theme's colours. Can be a number between 1 and 11, or one of the theme's names: c("Default", "Greyscale", "Candy", "RedBlue", "Dark", "Reds", "Blues", "Greens", "Viridis", "Cividis", "Distinct", "Distinct2", "Paired").
reverse	Logical value indicating whether to reverse the order of the palette's colours Default is FALSE.

Value

Returns n colours from the requested theme

See Also

quick_map

Examples

```
palettes_countries(5, theme = 1)
```

`quick_map` *Easily visualise country data with a map*

Description

`quick_map()` allows to plot country chloropleth maps with one line of code. The only inputs required are a `data.frame` object and the name of the column to plot. The function uses `country_name()`'s capabilities to automatically match country names to one of the territories in the ISO standard 3166-1. This allows fuzzy matching of country names in multiple languages and nomenclatures. For some map examples, see this article.

Usage

```
quick_map(
  data,
  plot_col,
  theme = 1,
  zoom = "Default",
  verbose = FALSE,
  save_to = NULL,
  width_plot = 30,
  name_legend = NULL,
  reverse_palette = FALSE,
  col_breaks = NULL,
  col_border = "black",
  col_na = "grey97",
  width_border = 0.1
)
```

Arguments

`data`	Table (data.frame) containing the data to plot. Each row in the table should correspond to a country. One of the columns should contain country names.
`plot_col`	Name of the column to plot.
`theme`	A numeric value or name identifying one of the predefined visual themes for the map. Can be a number between 1 and 11, or one of the predefined theme's names: `c("Default", "Greyscale", "Candy", "RedBlue", "Dark", "Reds", "Blues", "Greens", "Viridis", "Cividis", "Distinct", "Distinct2", "Paired")`. If `0` or `"NoTheme"` is passed, no theme will be applied (default 'ggplot2''s settings are used).

zoom	This argument defines the zoom applied to the map. It can be either a string identifying one of the predefined zoom boxes ("Default", "World", "Africa", "Asia", "Europe", "SEAsia", "NAmerica", "CAmerica", "SAmerica", "Oceania"). Alternatively, the user may provide a numeric vector of length 4 describing the min/max longitude and latitude (e.g. c(-80, -35, -55, 10) defines a zoom on South America).
verbose	Logical value indicating whether to print messages to the console. Default is FALSE.
save_to	Path to the file where the plot is to be saved. This need to be in an existing directory. The default is NULL, which does not save the plot.
width_plot	Width (in cm) when plot is saved to a file. The ratio between height and width is fixed. This argument is only relevant if save_to is different from NULL. Default is 30. For custom saving options the function ggsave() can be used.
name_legend	String giving the name to be used for the plotted variable in the legend of the map. If nothing is provided, the default is to use the name in plot_col.
reverse_palette	Logical value indicating whether to reverse the order of the colours in the palette. Default is FALSE.
col_breaks	Only relevant for numeric data. This argument allows the user to provide manual breaks for the colour scale. Needs to be a numeric vector (c(0, 100, 500, 1000)). Default is NULL, which will result in breaks being automatically selected by the function. Note that data with 6 or less unique values will be treated as factor by the function.
col_border	Colour of border line separating countries and landmasses. Default is "black".
col_na	Colour for countries with missing data (NAs). Default is "grey97".
width_border	Numeric value giving the width of the border lines between countries. Default is '0.1'.

Details

Good to know

quick_map() only allows plotting of territories in the ISO standard 3166-1. It does not support plotting of other regions. The output of the function is a ggplot object. This means means that users can then customise the look of the output by applying any of ggplot's methods.

Disclaimer

Territories' borders and shapes are intended for illustrative purpose. They might be outdated and do not imply the expression of any opinion on the part of the package developers.

Value

ggplot object

Examples

```
# creating some sample data to plot
example_data <- data.frame(country = random_countries(100), population = runif(100))

# make a map
quick_map(example_data, "population")

# The function provides several predefined themes
quick_map(example_data, "population", theme = 3)
quick_map(example_data, "population", theme = "Reds")

# provide breaks for the colour scale
quick_map(example_data, "population", col_breaks = c(0, 1e5, 1e6, 1e7, 1e8, 1e9))
```

---

random_countries *Output random country names*

Description

This function returns the mode of vectors. That is to say, for any given vector of values, it returns the value that appears most frequently. The function works with strings, numerical and mixed inputs. NA values are treated as distinct values.

Usage

```
random_countries(n, replace = FALSE, nomenclature = "name_en", seed = NULL)
```

Arguments

n	Number of desired (pseudo)random country names.
replace	Logical value indicating whether sampling should be with replacement.
nomenclature	Nomenclature from which the list of countries should be taken. Not all countries are present in all nomenclature, for example Taiwan is not recognised by the UN, so it will not be returned with "WTO_en". The function accept any of the nomenclatures of country_name. For a list of accepted values, refer to this page. The default is name_en, which is the English list of names in the ISO standard 3166.
seed	Single numerical value to be used as seed.

Value

A vector of n (pseudo)random country names.

See Also

list_countries, country_name

Examples

```
random_countries(10)
random_countries(n = 500, replace = TRUE)
random_countries(n = 5, nomenclature = "ISO3", seed = 5)
```

`which_min` *Return location of minimum, maximum and mode values' index*

Description

These function return the position (index) of all the minimum, maximum, and mode values of the vector `x`. `which_min()` and `which_max()` only support numeric and logical vectors. These functions are identical to `which.min()` and `which.max()`, except that ALL minima/maxima are returned instead of only the first one.

Usage

```
which_min(x, first_only = FALSE)

which_max(x, first_only = FALSE)

which_mode(x, first_only = FALSE)
```

Arguments

`x`	A numeric or vector
`first_only`	Logical value indicating whether only the first value should be returned (i.e. if `TRUE` the function behaves like `which.min()` and `which.max()`). Default is FALSE.

Value

Returns the position of the minimum, maximum and mode values of a vector `x`

See Also

Mode, which.min, which.max

Examples

```
which_mode(c("a","a",2,3))
which_min(c(1,1,2,3,NA,2))
which_max(c(NA,NA,NA,1,1,2))
```

`world` *World map data*

Description

A table containing points to draw a world map. The data comes from the package maps ("world") An additional column is added with ISO 3-digit country codes.

Usage

```
world
```

Format

A data frame with six columns providing information to plot world maps.

long Longitude

lat Latitude

group Numeric value used to identify polygons

order Order in which lines should be traced

region Name of the polygon's geographic region

ISO3 3-digits ISO country code for the region

Index