Package 'passport'

November 30, 2017

Type Package

Title Travel Smoothly Between Country Name and Code Formats

Version 0.2.0

Description Smooths the process of working with country names and codes via powerful parsing, standardization, and conversion utilities arranged in a simple, consistent API. Country name formats include multiple sources including the Unicode Common Locale Data Repository (CLDR, <http://cldr.unicode.org/>) common-sense standardized names in hundreds of languages.

Depends R (>= 3.1.0)

Imports stats, utils

Suggests jsonlite, DT, ggplot2, tidyverse, gapminder, testthat, mockr, covr, knitr, rmarkdown

License GPL-3 | file LICENSE

URL https://github.com/alistaire47/passport, https://alistaire47.github.io/passport/

BugReports https://github.com/alistaire47/passport/issues

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1.9000

VignetteBuilder knitr

NeedsCompilation no

Author Edward Visel [aut, cre]

Maintainer Edward Visel <edward.visel@gmail.com>

Repository CRAN

Date/Publication 2017-11-30 13:01:28 UTC

R topics documented:

`as_country_code` *Convert standardized country names to country codes*

Description

`as_country_code` converts a vector of standardized country names or codes to country codes

Usage

```
as_country_code(x, from, to = "iso2c", factor = is.factor(x))
```

Arguments

`x`	A character, factor, or numeric vector of country names or codes
`from`	Format from which to convert. See Details for more options.
`to`	Code format to which to convert. Defaults to `"iso2c"`; see `codes` for more options.
`factor`	If `TRUE`, returns factor instead of character vector.

Details

`as_country_code` takes a character, factor, or numeric vector of country names or codes to translate into the specified code format. The default for `to` is `"iso2c"`, the ISO 3166-1 Alpha-2 character codes, but many alternatives are available.

Several non-unique codes are available as well, including `"continent"`, `"is_independent"`, ISO 4217 currency codes, etc. Backwards conversion will not work for such cases.

See `codes` for all options, or run `DT::datatable(codes)` for a searchable widget.

Value

A vector of country codes. Warns if new `NA` values are added.

See Also

For converting to country names, use `as_country_name()`, which offers control of short and variant forms. For parsing non-standardized country names to codes, use `parse_country()`.

Examples

```
# Codifies standardized names
as_country_code(c("US", "Taiwan", "Myanmar", "Kosovo", "South Korea"), from = "en")

# Translates codes; if passed a factor, returns a releveled one
as_country_code(factor(c("SAH", "PCN", "OMA", "JPN")),
                from = "fifa", to = "iso4217_3c")
```

`as_country_name` *Convert standardized country codes to country names*

Description

`as_country_name` converts a vector of standardized country codes to country names.

Usage

```
as_country_name(x, to = "en", from = "iso2c", short = TRUE,
  variant = FALSE, factor = is.factor(x))
```

Arguments

`x`	A character, factor, or numeric vector of country codes or names
`to`	Language code of country names desired. Defaults to `"en"`; see `codes` for more options.
`from`	Code format from which to convert. Defaults to `"iso2c"`; see `codes` for more options.
`short`	Whether to use short alternative name when available. Can be length 1 or the same length as `x`.
`variant`	Whether to use variant alternative name when available. Can be length 1 or the same length as `x`.
`factor`	If `TRUE`, returns factor instead of character vector. If not supplied, defaults to `is.factor(x)`

Details

`as_country_name` takes a character, factor, or numeric vector of country codes (or names in another standardized format) and converts them to country names in the specified format. If you are trying to standardize an existing set of names, see `parse_country()`.

The default `"en"` is from Unicode Common Locale Data Repository (CLDR), which aspires to use the most customary name e.g. "Switzerland" instead of official ones, which are frequently awkward for common usage, e.g. "Swiss Confederation". CLDR also supplies names in a huge variety of languages, allowing for easy translation. Short and variant alternates are available for some countries; if not, the function will fall back to the standard form. See LICENSE file for terms of use.

Other name sets are available from

- the UN Statistics Division(UNSD), which maintains standardized names in English, Chinese, Russian, French, Spanish, and Arabic, here named as `"en_un"` etc.
- the ISO, `"en_iso"` and `"fr_iso"`, and
- the CIA World Factbook:
  - `"en_cia"`, which include many longer official forms and shorter practical forms,
  - `"en_cia_local"`, which includes transliterations, and
  - `"en_cia_abbreviation"`, which includes commonly-used abbreviations.

See `codes` for all options, or run `DT::datatable(codes)` for a searchable widget.

Value

A character or factor vector of country names. Warns if new `NA` values are added.

See Also

For converting standardized names to codes, use `as_country_code()`. For standardizing names to codes, use `parse_country()`.

Examples

```
# Usable names for tough-to-standardize places
as_country_name(c("US", "TW", "MM", "XK", "KR"))

# If passed a factor, will return a releveled one
as_country_name(factor(c("US", "NF", "CD", "SJ")), short = FALSE, variant = TRUE)

# Speaks a lot of languages, knows a lot of codes
as_country_name(c("SAH", "PCN", "OMA", "JPN"), from = "fifa", to = "cy")    # to Welsh
```

---

codes *Country code and name details and documentation*

Description

A codebook data.frame of codes and details for country code and name conversions available. Contains Internet Engineering Task Force (IETF) language tags (e.g. `"en-nz"` for New Zealand English) for Unicode Common Locale Data Repository (CLDR) names, similar approximations for institutional names (e.g. `"en-iso"`), and short names (e.g. `"iso2c"`) for country codes.

Usage

```
codes
```

Format

A data.frame of 427 rows and 9 variables.

Structure:

- `column`: The column name in the internal `passport:::countries` data.frame. Valid for use in `from` and `to` parameters.
- `code`: `column` with hyphens for underscores, which is a valid IANA language tag for Unicode CLDR country names. Valid for use in `from` and `to` parameters.
- `name`: Full name or code name for non-CLDR options.
- `notes`: Things to note, including deprecations, oddities, etc.
- `language`: Full language name parsed from `code`.
- `region`: Full country or region name parsed from `code`.
- `script`: Full language script name parsed from `code`.
- `variant`: Full variant parsed from `code`. Also used for organization-standardized names.
- `extension`: Further specification of name type.

Details

All functions can accept codes separated with underscores _, hyphens -, or periods ..

Examples

```
# A searchable widget to find a code or name
if (requireNamespace("DT", quietly = TRUE)) {
    DT::datatable(codes)
}
```

`country_format` *Construct formatter function to format country codes as country names*

Description

`country_format` is a constructor function that returns a function to format country codes as country names suitable for passing to ggplot2's scale functions' `label` parameters.

Usage

```
country_format(from = "iso2c", to = "en", short = TRUE, variant = FALSE,
  factor)
```

Arguments

from	Code format from which to convert. Defaults to "iso2c"; see codes for more options.
to	Language code of country names desired. Defaults to "en"; see codes for more options.
short	Whether to use short alternative name when available. Can be length 1 or the same length as x.
variant	Whether to use variant alternative name when available. Can be length 1 or the same length as x.
factor	If TRUE, returns factor instead of character vector. If not supplied, defaults to is.factor(x)

Details

A frequent reason to convert country codes back to country names is to make data visualizations more readable. While both a code and name could be stored in a data frame, the computation and extra storage required can be avoided by transforming codes to names directly within the visualization via a formatter function. as_country_name() could be used without parentheses to format ISO 2-character codes as English names, but format_country allows greater flexibility, returning a formatter function with the specified parameters set.

Value

A function that accepts a vector of country codes and returns them as country names.

See Also

For controlling the order of a discrete scale, pass the results of order_countries() to limits.

Examples

```
if (require(ggplot2, quietly = TRUE)) {
    ggplot(data.frame(country = c("KOR", "MMR", "TWN", "COG"),
                      y = 1:4),
           aes(x = country, y = y)) +
        geom_col() +
        scale_x_discrete(labels = country_format(from = "iso3c"))
}
```

order_countries *Order a vector of countries*

Description

order_countries reorders a vector of countries, returning a result useful for passing to ggplot2's scale functions' limits parameters.

Usage

```
order_countries(x, by, ..., from = "iso2c", short = TRUE, variant = FALSE,
  factor = is.factor(x))
```

Arguments

x	A character, factor, or numeric vector of country codes or names
by	Either a length-one country code from codes or a vector the same length as x by which to order x
...	Parameters passed on to order(), including addition vectors by which to sort, decreasing, and na.last.
from	Code format from which to convert. Defaults to "iso2c"; see codes for more options.
short	Whether to use short alternative name when available. Can be length 1 or the same length as x.
variant	Whether to use variant alternative name when available. Can be length 1 or the same length as x.
factor	If TRUE, returns factor instead of character vector. If not supplied, defaults to is.factor(x)

Details

order_countries orders a vector of countries by

- itself converted to a country code or name if by is a code from codes to which to convert
- a sortable vector if by is a vector of the same length as x
- x itself if neither is supplied.

Value

The original vector of countries, ordered according to the parameters passed. Note that factors are not releveled, but are reordered. To relevel, pass the results to levels<-()

See Also

To change labels of a discrete scale, pass the results of country_format() to the labels parameter.

Examples

```
countries <- c("FR", "CP", "UZ", "BH", "BR")

order_countries(countries)

order_countries(countries, "ja")

order_countries(countries, rnorm(5))

order_countries(countries, grepl("F", countries), 1:5, decreasing = TRUE)

if (require(ggplot2, quietly = TRUE)) {
    df_countries <- data.frame(country = countries,
                               y = exp(1:5))

    ggplot(df_countries, aes(country, y)) +
        geom_col() +
        scale_x_discrete(
            limits = order_countries(df_countries$country,
                                     df_countries$y)[df_countries$y > 5],
            labels = country_format(to = "en-cia-local")
        )
}
```

`parse_country` *Parse country names to standardized form*

Description

`parse_country` parses irregular country names to the ISO 3166-1 Alpha-2 code or other standardized code or name format.

Usage

```
parse_country(x, to = "iso2c", how = c("regex", "google", "dstk"),
  language = c("en", "de"), factor = is.factor(x))
```

Arguments

`x`	A character or factor vector of country names to standardize
`to`	Format to which to convert. Defaults to `"iso2c"`; see `codes` for more options.
`how`	How to parse; defaults to `"regex"`. `"google"`` uses the Google Maps geocoding API;"dstk"“ uses the Data Science Toolkit geocoding API. See "Details" for more information.
`language`	If `how = "regex"`, the language from which to parse country names. Currently accepts `"en"` (default) and `"de"`. Ignored if `how = "google"` or `how = "dstk"`.
`factor`	If `TRUE`, returns factor instead of character vector. If not supplied, defaults to `is.factor(x)`

Details

parse_country tries to parse a character or factor vector of country names to a standardized form: by default, ISO 3166-1 Alpha-2 codes.

When how = "regex" (default), parse_country uses regular expressions to match irregular forms.

If regular expressions are insufficient, how = "google" will use the Google Maps geocoding API instead, which permits a much broader range of input formats and languages. The API allows 2500 calls per day, and should thus be called judiciously. parse_country will make one call per unique input. For more calls, see options that allow passing an API key like ggmap::geocode() with output = "all" or googleway::google_geocode().

If how = "dstk", parse_country will use the Data Science Toolkit geocoding API.

Note that due to their flexibility, the APIs may fail unpredictably, e.g. parse_country("foo", how = "google") returns "CH" and parse_country("foo", how = "dstk") returns "DJ" whereas how = "regex" fails with a graceful NA and warning.

Value

A character vector or factor of ISO 2-character country codes or other specified codes or names. Warns of any parsing failure.

Examples

```
parse_country(c("United States", "USA", "U.S.", "us", "United States of America"))

## Not run:
# Unicode support for parsing accented or non-Latin scripts
parse_country(c("\u65e5\u672c", "Japon", "\u0698\u0627\u067e\u0646"), how = "google")
#> [1] "JP" "JP" "JP" "JP"

# Parse distinct place names via geocoding APIs
parse_country(c("1600 Pennsylvania Ave, DC", "Eiffel Tower"), how = "google")
#> [1] "US" "FR"

## End(Not run)
```

Index