Package ‘lingtypology’

February 23, 2017

Type Package

Title Linguistic Typology and Mapping

Version 1.0.2

Depends R (>= 2.10)

Imports leaflet,
stats,
stringdist,
magrittr,
grDevices,
rowr

Author George Moroz

Maintainer George Moroz <agricolamz@gmail.com>

Description Provides R with the Glottolog database <http://glottolog.org> and some more abilities for purposes of linguistic cartography. The Glottolog database contains the catalogue of languages of the world. This package helps researchers to make a linguistic maps, using philosophy of the Cross-Linguistic Linked Data project <http://clld.org/>, which allows for while at the same time facilitating uniform access to the data across publications. A tutorial for this package is available on GitHub pages <https://agricolamz.github.io/lingtypology/> and package vignette.

License GPL (>= 2)

URL https://CRAN.R-project.org/package=lingtypology, https://github.com/agricolamz/lingtypology/

BugReports https://github.com/agricolamz/lingtypology/issues

LazyData TRUE

RoxygenNote 5.0.1

Suggests knitr,
rmarkdown,
testthat,
covr

VignetteBuilder knitr

R topics documented:

`aff.lang` *Get affiliation by languoid*

Description

Takes any vector of languoids and return affiliation.

Usage

```
aff.lang(x, glottolog.source = "modified")
```

Arguments

`x`	A character vector of the languoids (can be written in lower case)
`glottolog.source`	A character vector that define which glottolog database is used: 'original' or 'modified' (by default)

Author(s)

George Moroz <agricolamz@gmail.com>

See Also

`area.lang`, `country.lang`, `iso.lang`, `lat.lang`, `long.lang`

Examples

```
aff.lang('Korean')
aff.lang(c('Korean', 'Polish'))
```

`area.lang` *Get macro area by languoid*

Description

Takes any vector of languoids and return macro area.

Usage

```
area.lang(x, glottolog.source = "modified")
```

Arguments

`x`	character vector of the languoids (can be written in lower case)
`glottolog.source`	A character vector that define which glottolog database is used: 'original' or 'modified' (by default)

Author(s)

George Moroz <agricolamz@gmail.com>

See Also

`aff.lang`, `country.lang`, `iso.lang`, `lat.lang`, `long.lang`

Examples

```
area.lang('Adyghe')
area.lang(c('Adyghe', 'Aduge'))
```

`circassian` *Circassian villages in Russia*

Description

A dataset containes the list of the Circassian villages in Russia with genealogical affiliation, coordinates and district names. Most data collected during the fieldworks (2011–2016).

Usage

```
circassian
```

Format

A data frame with 157 rows and 6 variables:

longitude longitude

latitude latitude

village name of the village

district names of the subjects of the Russian Federation: kbr — Kabardino–Balkar Republic, kch — Karachay–Cherkess Republic, kk — Krasnodar Krai, ra — Republic of Adygea, stv — Stavropol Krai

languoid names of the Circassian dialects

language according standard Circassian devision there are Adyghe and Kabardian languages

`countries` *Catalogue of countries names.*

Description

Catalogue of countries names.

Usage

```
countries
```

Format

A data frame with 86 rows and 3 variables:

common common name

official official name

abbreviation abreviated name

`country.lang` *Get country by languoid*

Description

Takes any vector of languoids and return affiliation.

Usage

```
country.lang(x, intersection = FALSE, glottolog.source = "modified")
```

Arguments

`x`	character vector of the languoids (can be written in lower case)
`intersection`	logical. If TRUE, function reterns vector of countries, where all languoids from x argument are spoken.
`glottolog.source`	A character vector that define which glottolog database is used: 'original' or 'modified' (by default)

Author(s)

George Moroz <agricolamz@gmail.com>

See Also

`aff.lang`, `area.lang`, `iso.lang`, `lat.lang`, `long.lang`

Examples

```
country.lang('Udi')
country.lang(c('Udi', 'Laz'))
country.lang(c('Udi', 'Laz'), intersection = TRUE)
```

`glottolog.modified` *Catalogue of languages of the world*

Description

A dataset containes the modified catalogue of languages of the world involving genealogical affiliation, macro-area, country, iso code, and coordinates.

Usage

```
glottolog.modified
```

Format

A data frame with 8304 rows and 7 variables:

iso code based on ISO 639–3 `http://www-01.sil.org/iso639-3/`

languoid name of the languoid

affiliation genealogical affiliation

macro_area have six values Africa, Australia, Eurasia, North America, Papunesia, South America

country list of countries, where the language is spoken

latitude latitude

longitude longitude

Details

Glottolog 2.7. Hammarstrom, Harald & Forkel, Robert & Haspelmath, Martin & Bank, Sebastian. 2016. Max Planck Institute for the Science of Human History. Accessed on 2016-06-15.

Source

`http://glottolog.org/`

`glottolog.original` *Catalogue of languages of the world*

Description

A dataset containes the original catalogue of languages of the world involving genealogical affiliation, macro-area, country, iso code, and coordinates.

Usage

```
glottolog.original
```

Format

A data frame with 8285 rows and 7 variables:

iso code based on ISO 639–3 `http://www-01.sil.org/iso639-3/`

languoid name of the languoid

affiliation genealogical affiliation

macro_area have six values Africa, Australia, Eurasia, North America, Papunesia, South America

country list of countries, where the language is spoken

latitude latitude

longitude longitude

Details

Glottolog 2.7. Hammarstrom, Harald & Forkel, Robert & Haspelmath, Martin & Bank, Sebastian. 2016. Max Planck Institute for the Science of Human History. Accessed on 2016-06-15.

Source

http://glottolog.org/

`is.glottolog` *Are these langoids in glottolog?*

Description

Takes any vector of linguoids or ISO codes and return a logical vector.

Usage

```
is.glottolog(x, response = FALSE, glottolog.source = "modified")
```

Arguments

`x`	A character vector of linguoids (can be written in lower case)or ISO codes
`response`	logical. If TRUE, when languoid is absent, return warnings with a possible candidates.
`glottolog.source`	A character vector that define which glottolog database is used: ’original’ or ’modified’ (by default)

Author(s)

George Moroz <agricolamz@gmail.com>

Examples

```
is.glottolog(c('Adyghe', 'Russsian'))

# Add warning message with sugestions
is.glottolog(c('Adyge', 'Russian'), response = TRUE)
# > FALSE TRUE
# Warning message:
# In is.glottolog(c('Adyge', 'Russian'), response = TRUE) :
# Languoid Adyge is absent in our database. Did you mean Aduge, Adyghe?
```

iso.lang *Get ISO 639–3 code by languoid*

Description

Takes any vector of languoids and return ISO code.

Usage

```
iso.lang(x, glottolog.source = "modified")
```

Arguments

`x`	A character vector of the languoids (can be written in lower case)
`glottolog.source`	A character vector that define which glottolog database is used: 'original' or 'modified' (by default)

Author(s)

George Moroz <agricolamz@gmail.com>

See Also

`aff.lang`, `area.lang`, `country.lang`, `lat.lang`, `long.lang`

Examples

```
iso.lang('Adyghe')
iso.lang(c('Adyghe', 'Udi'))
```

lang.aff *Get languoids by affiliation*

Description

Takes any vector of affiliations and return languoids.

Usage

```
lang.aff(x, list = FALSE, glottolog.source = "modified")
```

Arguments

`x`	A character vector of the affiliations (can be written in lower case)
`list`	logical. If TRUE, returns a list of languoids, if FALSE return a named vector.
`glottolog.source`	A character vector that define which glottolog database is used: 'original' or 'modified' (by default)

Author(s)

George Moroz <agricolamz@gmail.com>

See Also

`lang.country`, `lang.iso`

Examples

```
lang.aff('Balto-Slavic')
lang.aff(c('East Slavic', 'West Slavic'))
lang.aff(c('East Slavic', 'West Slavic'), list = TRUE)
```

`lang.country` *Get languoids by country*

Description

Takes any vector of countries and return languoids.

Usage

```
lang.country(x, list = FALSE, glottolog.source = "modified")
```

Arguments

`x`	character vector of the countries (can be written in lower case)
`list`	logical. If TRUE, returns a list of languoids, if FALSE return a vector.
`glottolog.source`	A character vector that define which glottolog database is used: 'original' or 'modified' (by default)

Author(s)

George Moroz <agricolamz@gmail.com>

See Also

`lang.aff`, `lang.iso`

Examples

```
lang.country('North Korea')
lang.country(c('North Korea', 'Luxembourg'))
lang.country(c('North Korea', 'Luxembourg'), list = TRUE)
## What languoids are both in North Korea and in South Korea?
lang.country('Korea')
```

`lang.iso` *Get languoid by ISO 639–3 code*

Description

Takes any vector of ISO codes and return languoids.

Usage

```
lang.iso(x, glottolog.source = "modified")
```

Arguments

`x`	A character vector of the ISO codes.
`glottolog.source`	A character vector that define which glottolog database is used: 'original' or 'modified' (by default)

Author(s)

George Moroz <agricolamz@gmail.com>

See Also

`lang.aff`, `lang.country`

Examples

```
lang.iso('ady')
lang.iso(c('ady', 'rus'))
```

`lat.lang` *Get latitude by languoid*

Description

Takes any vector of languoids and return latitude.

Usage

```
lat.lang(x, glottolog.source = "modified")
```

Arguments

`x`	A character vector of the languoids (can be written in lower case)
`glottolog.source`	A character vector that define which glottolog database is used: 'original' or 'modified' (by default)

Author(s)

George Moroz <agricolamz@gmail.com>

See Also

`aff.lang`, `area.lang`, `country.lang`, `iso.lang`, `long.lang`

Examples

```
lat.lang('Adyghe')
long.lang('Adyghe')
lat.lang(c('Adyghe', 'Russian'))
long.lang(c('Adyghe', 'Russian'))
```

`long.lang` *Get longitude by languoid*

Description

Takes any vector of languoids and return longitude.

Usage

```
long.lang(x, glottolog.source = "modified")
```

Arguments

`x`	A character vector of the languoids (can be written in lower case)
`glottolog.source`	A character vector that define which glottolog database is used: ’original’ or ’modified’ (by default)

Author(s)

George Moroz <agricolamz@gmail.com>

See Also

`aff.lang`, `area.lang`, `country.lang`, `iso.lang`, `lat.lang`

Examples

```
lat.lang('Adyghe')
long.lang('Adyghe')
lat.lang(c('Adyghe', 'Russian'))
long.lang(c('Adyghe', 'Russian'))
```

`makelink` *Make a link for a languoid*

Description

Takes any vector of linguoids and return links to glottolog pages.

Usage

```
makelink(x, popup = "", glottolog.source = "modified")
```

Arguments

`x`	A character vector of linguoids (can be written in lower case)
`popup`	character vector of strings that will appear in pop-up window of the function map.feature
`glottolog.source`	A character vector that define which glottolog database is used: ’original’ or ’modified’ (by default)

Author(s)

George Moroz <agricolamz@gmail.com>

map.feature *Create a map*

Description

Map a set of linguoids and color them by feature or two sets of features.

Usage

```
map.feature(languages, features = "none", popup = "", label = "",
  label.hide = FALSE, label.fsize = 14, label.position = "right",
  stroke.features = NULL, latitude = NULL, longitude = NULL,
  color = NULL, stroke.color = NULL, image.url = NULL,
  image.width = 100, image.height = 100, image.X.shift = 0,
  image.Y.shift = 0, title = NULL, stroke.title = NULL, control = FALSE,
  legend = TRUE, legend.opacity = 1, legend.position = "topright",
  stroke.legend = TRUE, stroke.legend.opacity = 1,
  stroke.legend.position = "bottomleft", radius = 5, stroke.radius = 9.5,
  opacity = 1, stroke.opacity = 1, scale.bar = TRUE,
  scale.bar.position = "bottomleft", minimap = FALSE,
  minimap.position = "bottomright", minimap.width = 150,
  minimap.height = 150, tile = "OpenStreetMap.Mapnik", tile.name = NULL,
  glottolog.source = "modified")
```

Arguments

`languages`	character vector of linguoids (can be written in lower case)
`features`	character vector of features
`popup`	character vector of strings that will appear in pop-up window
`label`	character vector of strings that will appear near points
`label.hide`	logical. If FALSE, labels are displayed allways. If TRUE, labels are displayed on mouse over. By default is TRUE.
`label.fsize`	numeric value of the label font size. By default is 14.
`label.position`	the position of labels: "left", "right", "top", "bottom"
`stroke.features`	additional independent stroke features
`latitude`	numeric vector of latitudes
`longitude`	numeric vector of longitudes
`color`	vector of colors
`stroke.color`	vector of stroke colors
`image.url`	character vector of URLs with an images
`image.width`	numeric vector of image widths
`image.height`	numeric vector of image heights

`image.X.shift`	numeric vector of image's X axis shift relative to the latitude-longitude point
`image.Y.shift`	numeric vector of image's Y axis shift relative to the latitude-longitude point
`title`	title of a legend
`stroke.title`	title of a stroke-feature legend
`control`	logical. If TRUE, function show layer control buttons. By default is TRUE.
`legend`	logical. If TRUE, function show legend. By default is FALSE.
`legend.opacity`	a numeric vector of legend opacity.
`legend.position`	the position of the legend: "topright", "bottomright", "bottomleft","topleft"
`stroke.legend`	logical. If TRUE, function show stroke.legend. By default is FALSE.
`stroke.legend.opacity`	a numeric vector of stroke.legend opacity.
`stroke.legend.position`	the position of the stroke.legend: "topright", "bottomright", "bottomleft","topleft"
`radius`	a numeric vector of radii for the circles.
`stroke.radius`	a numeric vector of stroke radii for the circles.
`opacity`	a numeric vector of marker opacity.
`stroke.opacity`	a numeric vector of stroke opacity.
`scale.bar`	logical. If TRUE, function show scale-bar. By default is TRUE.
`scale.bar.position`	the position of the scale-bar: "topright", "bottomright", "bottomleft","topleft"
`minimap`	= FALSE,
`minimap.position`	the position of the minimap: "topright", "bottomright", "bottomleft","topleft"
`minimap.width`	The width of the minimap in pixels.
`minimap.height`	The height of the minimap in pixels.
`tile`	a character verctor with a map tiles, popularized by Google Maps. See here for the complete set.
`tile.name`	a character verctor with a user's map tiles' names
`glottolog.source`	A character vector that define which glottolog database is used: "original" or "modified" (by default)

Author(s)

George Moroz <agricolamz@gmail.com>

Examples

```
map.feature(c("Adyghe", "Russian"))

## All Sign languages
map.feature(lang.aff("Sign"))

## Map all Slavic languages
map.feature(lang.aff(c("Slavic")))

## Add control buttons
map.feature(c("Adyghe", "Russian"), control = TRUE)

## Color linguoids by feature
df <- data.frame(lang = c("Adyghe", "Kabardian", "Polish", "Russian", "Bulgarian"),
feature = c("polysynthetic", "polysynthetic", "fusion", "fusion", "fusion"))
map.feature(df$lang, df$feature)
## ... or add a control buttons for features
map.feature(df$lang, df$feature, control = TRUE)

## Adding pop-up
df <- data.frame(lang = c("Adyghe", "Kabardian", "Polish", "Russian", "Bulgarian"),
feature = c("polysynthetic", "polysynthetic", "fusion", "fusion", "fusion"),
popup = c("Circassian", "Circassian", "Slavic", "Slavic", "Slavic"))
map.feature(df$lang, df$feature, df$popup)

## Adding labels
df <- data.frame(lang = c("Adyghe", "Kabardian", "Polish", "Russian", "Bulgarian"),
feature = c("polysynthetic", "polysynthetic", "fusion", "fusion", "fusion"),
popup = c("Circassian", "Circassian", "Slavic", "Slavic", "Slavic"))
map.feature(df$lang, df$feature, label = df$lang)

## Adding title
df <- data.frame(lang = c("Adyghe", "Kabardian", "Polish", "Russian", "Bulgarian"),
feature = c("polysynthetic", "polysynthetic", "fusion", "fusion", "fusion"),
popup = c("Circassian", "Circassian", "Slavic", "Slavic", "Slavic"))
map.feature(df$lang, df$feature, df$popup, title = "type of a language")

## Add your own coordinates
map.feature("Adyghe", latitude = 43, longitude = 57)

## Change map tile
map.feature("Adyghe", tile = "Thunderforest.OpenCycleMap")
map.feature("Adyghe", tile = c("OpenStreetMap.BlackAndWhite", "Thunderforest.OpenCycleMap"))
map.feature("Adyghe", tile = "Thunderforest.OpenCycleMap", tile.name = "colored")

## Add you own colors
df <- data.frame(lang = c("Adyghe", "Kabardian", "Polish", "Russian", "Bulgarian"),
feature = c("polysynthetic", "polysynthetic", "fusion", "fusion", "fusion"),
popup = c("Circassian", "Circassian", "Slavic", "Slavic", "Slavic"))
map.feature(df$lang, df$feature, df$popup, color = c("green", "navy"))

## Map two sets of features
```

```
df <- data.frame(lang = c("Adyghe", "Kabardian", "Polish", "Russian", "Bulgarian"),
feature = c("polysynthetic", "polysynthetic", "fusion", "fusion", "fusion"),
popup = c("Circassian", "Circassian", "Slavic", "Slavic", "Slavic"))
map.feature(df$lang, df$feature, df$popup,
stroke.features = df$popup)

## Add a pictures to plot
df <- data.frame(lang = c("Russian", "Russian"),
lat  = c(55.75, 59.95),
long = c(37.616667, 30.3),
urls = c("https://goo.gl/5OUv1E", "https://goo.gl/UWmvDw"))
map.feature(languages = df$lang,
latitude = df$lat,
longitude = df$long,
image.url = df$urls)

## Add a minimap to plot
map.feature(c("Adyghe", "Russian"), minimap = TRUE)

## Remove scale bar
map.feature(c("Adyghe", "Russian"), scale.bar = FALSE)
```

Index