Package 'ritwals'

June 17, 2020

Title R Interface to WALS

Version 0.0.0.9100

Author Henri Kauhanen <henri@henr.in>

Maintainer Henri Kauhanen <henri@henr.in>

Description Provides an intuitive R interface to the World Atlas of Language Structures (WALS). The current version makes available the 2014 version of WALS Online (the most recent version as of this writing).

Depends R (>= 2.12)

License GPL-3 | file LICENSE

Disclaimer Portions of this package constitute a derivative work of the World Atlas of Language Structures (Dryer & Haspelmath 2013).

Changes Semantically, the WALS has not been changed upon incorporation into this software package. What has been done is to collate all the various data files WALS provides (on languages, on features, on geographical information...) into a single data frame that supplies the entire atlas in one place, and enhance this with functions that assist in accessing the atlas data.

Reference Dryer, Matthew S. & Haspelmath, Martin (eds.) 2013. The World Atlas of Language Structures Online. Leipzig: Max Planck Institute for Evolutionary Anthropology. http://wals.info.

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

URL https://github.com/hkauhanen/ritwals

R topics documented:

contingency_table 2
country_codes 3
feature_metadata 3
filter_WALS 4
get_feature 5
great_circle_distance 6
intersect_features 6

language_metadata 7
nearest_geo_neighbours 8
nearest_geo_neighbours_discoidal 9
sample_random 10
sample_WALS_100 11
sample_WALS_200 11
WALS 12
WALS_features 13
WALS_languages 13

Index **15**

`contingency_table` *Contingency Tables*

Description

Produce a contingency table from the values of two or more WALS features.

Usage

```
contingency_table(ids, names = TRUE, absolute = TRUE, data = WALS)
```

Arguments

`ids`	Vector of feature IDs
`names`	Boolean; should the output use prose value descriptions? If `FALSE`, value IDs are used instead.
`absolute`	Boolean; should the output be in the form of absolute frequencies? If `FALSE`, relative frequencies are outputted.
`data`	Optionally a data frame, e.g. a subset of `WALS`

Details

This function calls `intersect_features` internally to establish the database intersection based on which a contingency table may be created.

Value

An *n*-by-*n* contingency table, where *n* is the length of `ids`

Examples

```
# 2x2 contingency table (absolute frequencies) of the first two features:
tab <- contingency_table(c("1A", "2A"))

# Run a chi-square test on the above:
chisq.test(tab)

# 3x3 contingency table of the first three features, showing relative
# frequencies and value IDs instead of value descriptions:
contingency_table(c("1A", "2A", "3A"), names=FALSE, absolute=FALSE)
```

---

country_codes *Country Codes*

Description

Return the country codes associated with a language

Usage

```
country_codes(id)
```

Arguments

id	Language's WALS ID, or a vector of IDs

Value

A list of character vectors

Examples

```
country_codes("fre")
unlist(country_codes("fre"))
country_codes(c("fre", "ger"))
```

---

feature_metadata *WALS Feature Metadata*

Description

Returns information about a WALS feature. Use this to find the possible values the feature can take.

Usage

```
feature_metadata(id)
```

Arguments

id	Feature's WALS ID, or a vector of IDs

Value

A dataframe with the following columns:

`feature_ID` Feature's WALS ID

`feature` Feature's name

`value_ID` Value ID

`value` Prose description of value

Examples

```
feature_metadata("13A")
feature_metadata("13A")$value_ID
feature_metadata(c("13A", "9A", "83A"))
```

---

filter_WALS *Filter WALS Data*

Description

Filter the WALS dataset down to a subset by a logical condition.

Usage

```
filter_WALS(expr, data = WALS)
```

Arguments

expr	Logical expression, must evaluate to a Boolean
data	Optionally, a data frame (e.g. any subset of WALS)

Details

The expr argument should refer to columns in the WALS data frame.

Value

A portion of the WALS data frame

See Also

See WALS for a description of the columns of the returned data frame.

Examples

```
# Get the subset of WALS pertaining to Afro-Asiatic languages:
filter_WALS(family=="Afro-Asiatic")

# Get the Semitic languages on the African continent:
filter_WALS(genus=="Semitic" & macroarea=="Africa")

# Get the set of Celtic and Romance languages:
filter_WALS(genus=="Celtic" | genus=="Romance")

# The same thing:
filter_WALS(genus %in% c("Celtic", "Romance"))

# Get languages whose latitudes fall between 10 and 20 degrees:
filter_WALS(latitude >= 10 & latitude <= 20)

# Get languages whose WALS language ID begins with the letters 'la'
filter_WALS(substr(language_ID, 1, 2) == "la")
```

`get_feature` *Get Feature*

Description

Return the representation of a feature in WALS.

Usage

```
get_feature(id, data = WALS)
```

Arguments

`id`	Feature's WALS ID
`data`	Optionally, a data frame (such as a subset of `WALS`)

Details

Returns the relevant portion of the `WALS` data frame.

Value

A data frame

great_circle_distance *Great-Circle Distance*

Description

Calculates the great-circle distance between two languages as determined by their WALS latitude-longitude coordinates.

Usage

```
great_circle_distance(id1, id2)
```

Arguments

id1	WALS ID of first language
id2	WALS ID of second language

Details

WALS defines the geographical representation of a language as a point in latitude-longitude space. Calculation of great-circle distance is by the haversine formula assuming a perfectly spherical Earth of radius 6,371 km.

Examples

```
a <- great_circle_distance("fre", "ger")
b <- great_circle_distance("ger", "rus")
c <- great_circle_distance("fre", "rus")
if (a + b >= c) {
  print("The triangle inequality works!")
}
```

intersect_features *Intersect Features*

Description

Return a subset of WALS consisting of only those languages for which data is attested in each of a given set of features; in effect, produce a set-theoretic intersection of the features.

Usage

```
intersect_features(ids, data = WALS)
```

Arguments

ids	Vector of IDs of features to intersect
data	Optionally, a data frame (such as a subset of WALS)

Value

A portion of the WALS data frame.

Examples

```
# Intersection of features 23A and 32A:
intersect_features(c("23A", "32A"))

# Intersection of features 1A, 9A and 13A:
intersect_features(c("1A", "9A", "13A"))

# Intersection of features 1A ... 20A:
df <- intersect_features(paste0(1:20, "A"))

# Find out which languages belong to that intersection:
unique(df$language_ID)

# And get their metadata:
language_metadata(unique(df$language_ID))

# Prove there is no language in WALS for which all features are attested:
df <- intersect_features(unique(WALS$feature_ID))
nrow(df) # == 0
```

---

language_metadata *WALS Language Metadata*

Description

Returns information about a WALS language.

Usage

```
language_metadata(id)
```

Arguments

id	Language's WALS ID, or a vector of IDs

Value

A dataframe with the following columns:

`language_ID` Language's WALS ID

`iso_code` Language's ISO code

`glottocode` Language's Glottocode

`language` Language's name

`latitude` Language's latitude

`longitude` Language's longitude

`genus` Language's genus

`family` Language's family

`macroarea` Language's macroarea

`countrycodes` Country codes associated with the language

`in_WALS_100` Boolean flag: `TRUE` if language is included in the WALS 100 sample

`in_WALS_200` Boolean flag: `TRUE` if language is included in the WALS 200 sample

Examples

```
language_metadata("fin")
language_metadata("fin")$family
language_metadata(c("fin", "swe"))
```

---

nearest_geo_neighbours

Nearest Geographical Neighbours

Description

Return the *k* nearest geographical neighbours of a language.

Usage

```
nearest_geo_neighbours(id, k, data = WALS)
```

Arguments

`id`	Language's WALS ID
`k`	How many neighbours to return
`data`	Dataset to consider; by default the entire `WALS`

Details

Computed using great_circle_distance. By default (data = WALS), all languages in the WALS atlas will be considered in the computation of the nearest neighbours. By supplying a non-default data argument, it is possible to override this behaviour, e.g. to supply only a subset of the languages.

If the language does not have as many as *k* neighbours (e.g. because of a strict filtering by data), the neighbours that *are* found are returned with a warning.

Examples

```
# 10 nearest neighbours of Egyptian Arabic:
nearest_geo_neighbours("aeg", k=10)

# 10 nearest neighbours of Egyptian Arabic among those languages for
# which feature 13A is attested:
nearest_geo_neighbours("aeg", k=10, data=filter_WALS(feature_ID=="13A"))

# 10 nearest neighbours of Egyptian Arabic among those languages for
# which feature 13A is attested, among the Afro-Asiatic family:
nearest_geo_neighbours("aeg", k=10,
  data=filter_WALS(feature_ID=="13A" & family=="Afro-Asiatic"))
```

---

nearest_geo_neighbours_discoidal

Nearest Geographical Neighbours (Discoidal)

Description

Return the neighbours of a language residing within a disc of a specified radius from that language.

Usage

```
nearest_geo_neighbours_discoidal(id, radius, data = WALS)
```

Arguments

id	Language's WALS ID
radius	Radius of disc in km
data	Dataset to consider; by default the entire WALS

Details

Computed using great_circle_distance. By default (data = WALS), all languages in the WALS atlas will be considered in the computation of the nearest neighbours. By supplying a non-default data argument, it is possible to override this behaviour, e.g. to supply only a subset of the languages.

Examples

```
# Neighbours of Finnish within 1000 km:
nearest_geo_neighbours_discoidal("fin", radius=1000)

# Neighbours of Finnish within 1000 km, within the Uralic family:
nearest_geo_neighbours_discoidal("fin", radius=1000,
  data=filter_WALS(family=="Uralic"))
```

`sample_random` *Random Samples*

Description

Take a random sample of *N* languages from WALS.

Usage

```
sample_random(N, data = WALS)
```

Arguments

`N`	Sample size
`data`	Optionally, a data frame (e.g. the output of `filter_WALS`)

Details

Languages in `data` are sampled uniformly at random without replacement, and their representation in `WALS` is returned.

Value

A subset of the `WALS` data frame.

Examples

```
# Just a random random sample of 100 languages:
random_WALS <- sample_random(100)

# Random sample of 50 tonal languages (feature 13A has value 2 or 3):
tonal_lges <- filter_WALS(feature_ID == "13A" & value_ID %in% 2:3)
random_tonals <- sample_random(50, data=tonal_lges)

# Random sample of 20 languages from the WALS 100-language sample:
sample_random(20, sample_WALS_100())
```

sample_WALS_100 *WALS 100-Language Sample*

Description

Return the WALS 100-language sample.

Usage

```
sample_WALS_100()
```

Details

Returns the portion of WALS where in_WALS_100 is TRUE.

Value

A subset of the WALS data frame.

sample_WALS_200 *WALS 200-Language Sample*

Description

Return the WALS 200-language sample.

Usage

```
sample_WALS_200()
```

Details

Returns the portion of WALS where in_WALS_200 is TRUE. Note that the WALS 200-language sample contains *202* languages.

Value

A subset of the WALS data frame.

WALS *World Atlas of Language Structures*

Description

The full WALS dataset.

Format

A data frame of 76465 rows (each row representing one language-feature combination) and 20 variables:

`combined_ID` Identifier combining `feature_ID` and `language_ID`; identifies a unique row in the data frame

`feature_ID` Feature ID

`feature` Feature name

`value_ID` Value ID

`value` Value name

`code_ID` Identifier combining `feature_ID` and `value_ID`

`language_ID` Language ID

`iso_code` Language's ISO code

`glottocode` Language's Glottolog code

`language` Language name

`family` Language's family

`genus` Language's genus

`macroarea` Language's macroarea

`countrycodes` Country codes associated with language

`latitude` Language's latitude in degrees

`longitude` Language's longitude in degrees

`in_WALS_100` Boolean: `TRUE` if language belongs to the WALS 100-language sample

`in_WALS_200` Boolean: `TRUE` if language belongs to the WALS 200-language sample

`source` Information source for this language-feature combination

`contributors` WALS contributors for this language-feature combination

Source

http://wals.info

WALS_features *WALS Feature Metadata*

Description

Information about WALS features and their possible values.

Format

A data frame with the following variables:

`feature_ID` Feature's WALS ID

`feature` Feature's name

`value_ID` Value ID

`value` Value's name

`code_ID` Identifier combining `feature_ID` and `value_ID`

Source

http://wals.info

WALS_languages *WALS Language Metadata*

Description

Information about the languages in WALS.

Format

A data frame of 2679 rows (one row per language) with the following 12 variables:

`language_ID` Language's WALS ID (or 'WALS code')

`iso_code` Language's ISO code

`glottocode` Language's Glottolog code

`language` Language's name

`family` Language's family

`genus` Language's genus

`macroarea` Language's macroarea

`countrycodes` Country codes associated with language

`latitude` Language's latitude in degrees

`longitude` Language's longitude in degrees

`in_WALS_100` Boolean: `TRUE` if language belongs to the WALS 100-language sample

`in_WALS_200` Boolean: `TRUE` if language belongs to the WALS 200-language sample

Source

http://wals.info

Examples

```
# The number of WALS genera:
length(unique(WALS_languages$genus))

# Frequency table of languages per family:
table(WALS_languages$family)
```

Index

contingency_table, 2
country_codes, 3

feature_metadata, 3
filter_WALS, 4, *10*

get_feature, 5
great_circle_distance, 6, *9*

intersect_features, *2*, 6

language_metadata, 7

nearest_geo_neighbours, 8
nearest_geo_neighbours_discoidal, 9

sample_random, 10
sample_WALS_100, 11
sample_WALS_200, 11

WALS, *2*, *4*, *5*, *7–11*, 12
WALS_features, 13
WALS_languages, 13