Package: glottospace (via r-universe)

October 13, 2024

Type Package

Title Language Mapping and Geospatial Analysis of Linguistic and Cultural Data

Version 0.0.113

Author Sietze Norder, Rui Dong

Maintainer Rui Dong <r.dong@hum.leidenuniv.nl>

Description Streamlined workflows for geolinguistic analysis, including: accessing global linguistic and cultural databases, data import, data entry, data cleaning, data exploration, mapping, visualization and export.

License GPL (>= 3)

URL https://github.com/glottospace/glottospace

BugReports https://github.com/glottospace/glottospace/issues

Depends R (>= 4.1.0)

Imports dplyr, plyr, ggplot2, magrittr, purrr, readxl, rlang, rnaturalearth, sf, tibble, tidyr, units, writexl, rvest, tmap (>= 3.99.9000), TDA, animation, utils

Suggests cluster, grDevices, htmlwidgets, jsonlite, mapedit, plotly, RColorBrewer, s2, scales, testthat (>= 3.0.0), vegan, viridisLite, xml2

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.2.3

Config/testthat/edition 3

Remotes github::r-tmap/tmap

Repository https://glottospace.r-universe.dev

RemoteUrl https://github.com/glottospace/glottospace

RemoteRef HEAD

RemoteSha 40d96c65b957b3262e5d21923fc2795851874db4

Contents

glottobooster *Enhance glottolog data*

Description

This function restructures glottolog data, and optionally adds/removes data. If you want more flexibility in choosing which data to add/remove, you can use glottoboosterflex().

Usage

```
glottobooster(
  glottologdata = NULL,
  space = TRUE,
  addfamname = TRUE,
  addisolates = TRUE,
  L1only = TRUE,
  addfamsize = TRUE,
  addfamsizerank = TRUE,
  rename = TRUE
)
```

Arguments

`glottologdata` data from glottolog, can be downloaded with glottoget("glottolog").

`space` Return spatial object?

`addfamname` Add column with familiy names?

`addisolates` Add column to identify isolates?

`L1only` Keep only L1 languages (remove bookkeeping, unclassifiable, sign languages, etc.).

`addfamsize` Add column with family size?

`addfamsizerank` Add column with family size rank?

`rename` Rename columns "id" to "glottocode" and "iso639p3code" to "isocode"

Details

This function is used to generate 'glottobase' (the reference dataset used throughout the glottospace R package). The default options generate 'glottobase', which can be loaded directly using glottoget("glottobase").

Value

glottologdata object, either a spatial object (class: sf) or a data.frame.

See Also

Other <glottobooster>: `glottoboosterflex()`

Examples

```
glottologdata <- glottoget("glottolog")
glottobase <- glottobooster(glottologdata)
```

glottocheck *Quality check of glottodata or glottosubdata*

Description

This function first checks whether a dataset is glottodata or glottosubdata, and depending on the result calls glottocheck_data or glottocheck_subdata.

Usage

```
glottocheck(glottodata, diagnostic = TRUE, checkmeta = TRUE)
```

Arguments

`glottodata`	User-provided glottodata
`diagnostic`	If TRUE (default) a data viewer will be opened to show the levels of each variable (including NAs), and a data coverage plot will be shown.
`checkmeta`	Should metadata be checked as well?

Details

It subsequently checks whether:

- one column exists with the name "glottocode"
- there are rows without a glottocode (missing IDs)
- there are rows with duplicated glottocodes (duplicate IDs)
- all variables have at least two levels
- all glottocodes are valid

Value

Diagnostic messages highlighting potential issues with glottodata or glottosubdata.

Examples

```
glottodata <- glottoget("demodata")
glottocheck(glottodata, diagnostic = FALSE)
```

`glottoclean` *Clean glottodata/glottosubdata*

Description

This function cleans glottodata/glottosubdata and returns a simplified glottodata/glottosubdata object containing only the cleaned data table and a structure table.

Usage

```
glottoclean(
  glottodata,
  tona = NULL,
  tofalse = NULL,
  totrue = NULL,
  id = NULL,
  glottosample = FALSE,
  one_level_drop = TRUE
)
```

Arguments

`glottodata`	glottodata (either a list or a data.frame)
`tona`	Optional additional values to recode to NA (besides default)
`tofalse`	Optional additional values to recode to FALSE (besides default)
`totrue`	Optional additional values to recode to TRUE (besides default)
`id`	By default, glottoclean looks for a column named 'glottocode', if the id is in a different column, this should be specified.
`glottosample`	Should the sample table be used to subset the data?
`one_level_drop`	A logical value to denote whether or not to drop variables with a single value, the default value is TRUE.

Details

This function has some built in default values that are being recoded: For example, if column type is 'symm' or 'asymm', values such as "No" and 0 are recoded to FALSE Values such as "?" are recoded to NA.

Value

A cleaned-up and simplified version of the original glottodata object

Examples

```
glottodata <- glottoget("demodata", meta = TRUE)
glottodata <- glottoclean(glottodata)

glottosubdata <- glottoget("demosubdata", meta = TRUE)
glottosubdata <- glottoclean(glottosubdata)
```

glottocode_exists *Check whether a set of glottocodes exist in glottolog*

Description

Checks whether a set of glottocodes exist in glottolog (checked at the level of L1 languages)

Usage

```
glottocode_exists(glottocode)
```

Arguments

glottocode	A glottocode or character vector of glottocodes

Value

A logical vector

Examples

```
glottocode_exists(c("yucu1253"))
glottocode_exists(c("yucu1253", "abcd1234"))
```

glottoconvert *Convert a linguistic dataset into glottodata or glottosubdata*

Description

This function is mainly intended for 'messy' datasets that are not in glottodata/glottosubdata structure.

Usage

```
glottoconvert(
  data,
  var = NULL,
  glottocodes = NULL,
  table = NULL,
  glottocolumn = NULL,
  glottosubcolumn = NULL,
  ref = NULL,
  page = NULL,
  remark = NULL,
  contributor = NULL,
  varnamecol = NULL
)
```

Arguments

`data` A dataset that should be converted into glottodata/glottosubdata. This will generally be an excel file loaded with glottoget().

The dataset will be converted into glottodata if:

- all data are stored in a single table, or
- the dataset contains several tables of which one is called 'glottodata', or
- a table argument is provided.

Otherwise, glottospace will attempt to convert the dataset into glottosubdata. This works if:

- table names are glottocodes, and
- an argument is provided to glottocodes, or the dataset contains a sample table from which glottocodes can be obtained.

`var` Character string that distinguishes those columns which contain variable names.

`glottocodes` Optional character vector of glottocodes. If no glottocodes are supplied, glottospace will search for them in the sample table.

`table` In case dataset consists of multiple tables, indicate which table contains the data that should be converted.

`glottocolumn` column name or column id with glottocodes (optional, provide if glottocodes are not stored in a column called 'glottocode')

`glottosubcolumn`

Column name or column id with glottosubcodes (optional, provide if glottosubcodes are not stored in a column called 'glottosubcode')

`ref` Character string that distinguishes those columns which contain references.

`page` Character string that distinguishes those columns which contain page numbers.

`remark` Character string that distinguishes those columns which contain remarks.

`contributor` Character string that distinguishes those columns which contain contributors.

`varnamecol` In case the dataset contains a structure table, but the varnamecol is not called 'varname', its name should be specified.

Value

A glottodata or glottosubdata object (either a list or data.frame)

Examples

```
# Create a messy dataset:
glottodata <- glottoget("demodata")
glottodata <- cbind(glottodata, data.frame("redundant" = c(1:6)))

# In this messy dataset there's no way to determine which columns contain the relevant variables...
# Therefore we manually add a character string to distinguish the relevant columns:
colnames(glottodata)[2:3] <- paste0("var_", colnames(glottodata)[2:3] )

glottoconverted <- glottoconvert(glottodata, var = "var_")
```

glottocreate *Generate empty glottodata or glottosubdata for a set of glottocodes.*

Description

Creates glottodata/glottosubdata and optionally save it as excel file.

Usage

```
glottocreate(
  glottocodes,
  variables,
  meta = TRUE,
  filename = NULL,
  simplify = TRUE,
  groups = NULL,
  n = NULL,
  levels = NULL,
  check = FALSE,
  maintainer = NULL,
  email = NULL,
  citation = NULL,
  url = NULL
)
```

Arguments

`glottocodes` Character vector of glottocodes

`variables` Either a vector with variable names, or a single number indicating the total number of variable columns to be generated

`meta` Should metatables be created?

`filename` Optional name of excel file where to store glottodata `simplify` By default, if a glottodata table is created without metadata, the data will be returned as a data.frame (instead of placing the data inside a list of length 1)

`groups` Character vector of group names (only for glottosubdata)

`n` Optional, number of records to be assigned to each group (only for glottosubdata)

`levels` Optional character vector with levels across all variables

`check` Should glottocodes be checked? Default is FALSE because takes much time to run.

`maintainer` Name of the person/organization maintaining the data (optional, added to readme tab)

`email` Email address of maintainer/contact person (optional, added to readme tab)

`citation` How to cite the data (optional, added to readme tab)

`url` Link to a webpage (optional, added to readme tab).

Details

By default, glottodata will be created. In case a groups argument is provided, glottosubdata will be created.

glottodata has one table for all languages (and a number of metatables if meta = TRUE), with one row per glottocode. glottosubdata has one table for each language (and a number of metatables if meta = TRUE), with one row per glottosubcode.

Run glottoget("demodata") or glottoget("demosubdata") to see examples.

In case you already have your own dataset and want to convert it into glottodata, use: glottoconvert().

Value

A glottodata or glottosubdata object (either with or without metadata). The output can be a list or a data.frame.

Examples

```
# Creates glottodata table without metadata tables
glottocreate(glottocodes = c("yucu1253", "tani1257"),
variables = 3, meta = FALSE)

# Creates glottodata table with metadata tables (stored in a list):
glottocreate(glottocodes = c("yucu1253", "tani1257"), variables = 3)


# Creates glottosubdata table (stored in a list)
glottocreate(glottocodes = c("yucu1253", "tani1257"),
variables = 3, groups = c("a", "b") )

# Create glottodata table and add some information to the readme table:
glottocreate(glottocodes = c("yucu1253", "tani1257"), variables = 3,
maintainer = "Your name", email = "yourname@domain.com")
```

glottocreate_addsample

Add sample table to glottodata or glottosubdata

Description

Add sample table to glottodata or glottosubdata

Usage

```
glottocreate_addsample(glottodata)
```

Arguments

`glottodata`	glottodata or glottosubdata

Value

glottodata/glottosubdata with a sample table

Examples

```
glottodata <- glottoget("demodata")
glottocreate_addsample(glottodata)
```

glottocreate_addstructure

Add structure table to glottodata or glottosubdata

Description

Add structure table to glottodata or glottosubdata

Usage

```
glottocreate_addstructure(glottodata)
```

Arguments

`glottodata`	glottodata or glottosubdata

Value

glottodata/glottosubdata with a structure table

Examples

```
glottodata <- glottoget("demodata")
glottocreate_addstructure(glottodata)
```

`glottodist` *Calculate distances between languages*

Description

Calculate distances between languages

Usage

```
glottodist(glottodata, metric = "gower")
```

Arguments

`glottodata`	glottodata or glottosubdata, either with or without structure table.
`metric`	either "gower" or "anderberg"

Value

object of class `dist`

Details

The function “glottodist” returns a “dist” object with respect to either Gower distance or Anderberg dissimilarity. The Anderberg dissimilarity is defined as follows. Consider a categorical dataset L containing N objects $X_1, \cdots, X_N$ defined over a set of d categorical features where A_k denotes the $k-$th feature. The feature A_k take n_k values in the given dataset which are denoted by $\mathcal{A}_k$. We regard ‘NA‘ as a new value. We also use the following notations:

- $f_k(x)$: The number of times feature A_k takes the value x in the dataset L. If $x \notin \mathcal{A}_k$, $f_k(x) = 0$.
- $\hat{p}_k(x)$: The sample frequency of feature A_k to take the value x in the dataset L. $\hat{p}_k(x) = \frac{f_k(x)}{N}$.

The Anderberg dissimilarity of X and Y is defined in the form of: $d(X_i, X_j) = \frac{D}{D+S}$, where

$$D = \sum_{k \in \{1 \le k \le d : X_k \neq Y_k\}} w_k * \delta_{ij}^{(k)} * \tau_{ij}^{(k)} \left(\frac{1}{2\hat{p}_k(X_k)\hat{p}_k(Y_k)} \right) \frac{2}{n_k(n_k+1)},$$

and

$$S = \sum_{k \in \{1 \le k \le d : X_k = Y_k\}} w_k * \delta_{ij}^{(k)} \left(\frac{1}{\hat{p}_k(X_k)} \right)^2 \frac{2}{n_k(n_k+1)}$$

The numeber w_k gives the weight of the k-th feature, and the numebr $\delta_{ij}^{(k)}$ is equal to either 0 or 1. It is equal to 0 when the type of the k-th feature is asymmetric binary and both values of X_i and X_j

are 0, or when either value of the k-th feature is missing, otherwise, it is equal to 1. When $X_k \neq Y_k$ and the type of A_k is "ordered", $\tau_{ij}^{(k)}$ is equal to the normalized difference of X_k and Y_k, otherwise $\tau_{ij}^{(k)}$ is equal to 1.

References

Andergerg M.R. (1973). Cluster analysis for applications. Academic Press, New York.

Boriah S., Chandola V., Kumar V. (2008). Similarity measures for categorical data: A comparative evaluation. In: Proceedings of the 8th SIAM International Conference on Data Mining, SIAM, p. 243-254.

Examples

```
glottodata <- glottoget("demodata", meta = TRUE)
glottodist <- glottodist(glottodata = glottodata, metric="anderberg")

glottosubdata <- glottoget("demosubdata", meta = TRUE)
glottodist <- glottodist(glottodata = glottosubdata)
```

glottodist_subdata *Calculate construction-based distances between languages*

Description

Calculate construction-based distances between languages

Usage

```
glottodist_subdata(
  glottosubdata,
  metric = NULL,
  index_type = NULL,
  avg_idx = NULL,
  fixed_idx = NULL
)
```

Arguments

`glottosubdata` an glottosubdata object

`metric` either "gower" or "anderberg"

`index_type` either "mci" or "ri" or "fmi"

`avg_idx` the feature indices over which the average of distances is computed, it must be given when index_type is either "ri" or "fmi".

`fixed_idx` the feature indices over which the distance of two constructions is computed, it must be given when index_type is either "ri" or "fmi".

Value

object of class `dist`

Details

The function “glottodist_subdata” returns a “dist” object, the input is a glottosubdata object, it computes the construction-based distance between languages, we refer to the observations of each language as constructions. The distance $d(A_i, B_j)$ between two constructions A_i in a language A and B_j in a language B is determined by the argument “metric”, whose value is either “gower” or “anderberg”. When “index_type” is “mci”, it returns the “matching constructions index”: $MCI(A, B) := \frac{1}{2|A|} \sum_{A_i \in A} \min_{B_j \in B} d(A_i, B_j) + \frac{1}{2|B|} \sum_{B_i \in B} \min_{A_j \in A} d(A_j, B_i)$. When “index_type” is “ri”, it returns the “relative index”: $RI(A, B) = \frac{1}{|M|} \sum_{s \in M} \mathrm{AVG}_{A_i(s)=1 \text{ and } B_j(s)=1} d(A_i^F, B_j^F)$, here M is the indices of a subset of variables given by the argument “avg_idx” and F is the indices of a subset of variables given by the argument “fixed_idx”, the restricted constructions A_i^F and B_j^F are defined as the constructions A_i, B_j restricted to “fixed_idx” F. When “index_type” is “fmi”, it returns the “form-meaning index”: $FMI(A, B) = \frac{1}{|M||F|} \sum_{s \in M, p \in F} \Big(1 - SIM(\{(A_i^M(s) = 1 \text{ and } A_i^F(p) = 1)\}, \{B_j^M(s) = 1 \text{ and } B_j^F(p) = 1\})\Big)$, here $SIM(X, Y) = \min(|X|/|Y|, |Y|/|X|)$, if both X and Y are empty, $SIM(X, Y) = 1$.

Examples

```
glottosubdata_cnstn <- glottoget(glottodata = "demosubdata_cnstn")
glottodist_subdata(glottosubdata = glottosubdata_cnstn, metric = "gower", index_type = "mci")
glottodist_subdata(glottosubdata = glottosubdata_cnstn, metric = "gower", index_type = "ri",
                   avg_idx = 1:4, fixed_idx = 5:7)
glottodist_subdata(glottosubdata = glottosubdata_cnstn, index_type = "fmi",
                   avg_idx = 1:4, fixed_idx = 5:7)
```

`glottofilter` *Filter glottodata by language, glottocode, etc.*

Description

By default, the glottolog data will be used to filter from. But in case the user provides glottodata, this will be used.

Usage

```
glottofilter(
  glottodata = NULL,
  glottocode = NULL,
  location = NULL,
  name = NULL,
  family = NULL,
  family_id = NULL,
  continent = NULL,
  country = NULL,
  sovereignty = NULL,
  macroarea = NULL,
  expression = NULL,
  isocodes = NULL,
  colname = NULL,
  select = NULL,
  drop = NULL
)
```

Arguments

`glottodata`	A glottodata table
`glottocode`	A character vector of glottocodes
`location`	A character vector with a location (either a continent, country, macroarea, or sovereignty)
`name`	A character vector of language names
`family`	A character vector of language families
`family_id`	A character vector of language family IDs
`continent`	A character vector of continents
`country`	A character vector of countries
`sovereignty`	Sovereignty
`macroarea`	Glottolog macroarea
`expression`	A logical expression
`isocodes`	A character vector of iso639p3codes
`colname`	A column name
`select`	Character vector of things to select (only if colname is provided)
`drop`	Character vector of things to drop (only if colname is provided)

Value

A subset of the original glottodata table (data.frame or sf) containing only filtered languages.

See Also

glottofiltermap()

Examples

```
points <- glottofilter(location = "Australia")
points <- glottofilter(glottocode = "wari1268")
points <- glottofilter(family = "Indo-European")
points <- glottofilter(continent = "South America")
points <- glottofilter(family = "Indo-European", continent = "South America")
points <- glottofilter(country = c("Colombia", "Venezuela"))
points <- glottofilter(expression = family %in% c("Arawakan", "Tucanoan"))
points <- glottofilter(expression = family_size > 2)
points <- glottofilter(colname = "family", drop = "Indo-European")
```

glottofiltermap *Filter languages interactively from a map*

Description

Select languages by drawing or clicking on a map. The output should be assigned to a new object. In case you want to select languages based on a (non-spatial) condition, you might want to use glottofilter() instead.

Usage

```
glottofiltermap(glottodata = NULL, mode = NULL, ...)
```

Arguments

glottodata	Spatial glottodata object
mode	You can choose here whether you want to interactively select languages by clicking on them (mode = 'click', default) or by drawing a shape around them (mode = 'draw').
...	Additional arguments to pass to glottofilter

Value

A set of languages selected from the original glottodata object

Examples

```
## Not run:
# Interactive selection by clicking on languages:
selected <- glottofiltermap(continent = "South America")
glottomap(selected)

# Interactive selection by drawing a shape:
selected <- glottofiltermap(continent = "South America", mode = "draw")
glottomap(selected)

## End(Not run)
```

glottoget *Get glottodata from local path or online global databases*

Description

Load locally stored glottodata, download databases from online sources, or load built-in demo data

Usage

```
glottoget(
  glottodata = NULL,
  meta = FALSE,
  download = FALSE,
  dirpath = NULL,
  url = NULL,
  seed = NULL
)
```

Arguments

`glottodata` options are:

- A filepath to locally stored glottodata or glottosubdata with file extension (.xlsx .xls .gpkg .shp). See also: options meta and simplify.
- "glottobase" - Default option, an spatially enhanced version of glottolog. See glottobooster for details. If glottodata = NULL, "glottobase" will be loaded.
- "wals" - This is a spatially enhanced version of WALS.
- "dplace" - This is a spatially enhanced version of D-PLACE.
- "glottolog" - This is a restructured (non-spatial) version of glottolog.
- "glottospace" - A simple dataset with glottocodes and a geometry column. This is a subset of all languages in glottolog with spatial coordinates.
- "grambank" - This is a restructured (non-spatial) version of Grambank.
- "grambankspace" - This is a restructured (spatially enhanced) version of Grambank.
- "phoible_raw" - This is a restructured (non-spatial) raw version of PHOIBLE.
- "phoiblespace_raw" - This is a restructured (spatially enhanced) raw version of PHOIBLE.
- "phoible" - This is a restructured (non-spatial) randomly sampled version of PHOIBLE. When seed is not provided, it will randomly choose a sample for each duplicated glottocode.
- "phoiblespace" - This is a (spatially enhanced) randomly sampled version of PHOIBLE. When seed is not provided, it will randomly choose a sample for each duplicated glottocode.
- "phoible_raw_param_sf" - This returns an sf object of the geographical distribution for all parameter IDs with respect to the raw PHOIBLE.

- "phoible_param_sf" - This returns an sf object of the geographical distribution for all parameter IDs with respect to a sampled version of PHOIBLE. When seed is not provided, it will randomly choose a sample for each duplicated glottocode.
- "demodata" - Built-in artificial glottodata (included for demonstration and testing).
- "demosubdata" - Built-in artificial glottosubdata (included for demonstration and testing)
- "demosubdata_cnstn" - Built-in artificial glottosubdata (included for demonstration and testing)

`meta` In case 'glottodata' is demodata/demosubdata: by default, meta sheets are not loaded. Use meta=TRUE if you want to include them.

`download` By default internally stored versions of global databases are used. Specify download = TRUE in case you want to download the latest version from a remote server.

`dirpath` Optional, if you want to store a global CLDF dataset in a specific directory, or load it from a specific directory.

`url` Zenodo url, something like this: "https://zenodo.org/api/records/3260727"

`seed` the seed number when glottoget phoible dataset, if not provided, the glottoget function will randomly choose one language for each duplicated glottocode.

Value

A glottodata or glottosubdata object (a data.frame or list, depending on which glottodata is requested)

See Also

Other <glottodata>: `glottosave()`

Examples

```
glottoget("glottolog")
```

glottojoin *Join glottodata with other objects, datasets, or databases.*

Description

Join glottodata with other objects, datasets, or databases.

Usage

```
glottojoin(glottodata, with = NULL, id = NULL, na.rm = FALSE, type = "left")
```

Arguments

`glottodata`	glottodata or glottosubdata
`with`	Optional: glottodata (class data.frame), a dist object (class dist), or the name of a glottodatabase ("glottobase" or "glottospace")
`id`	By default, data is joined by a column named "glottocode" or "glottosubcode". In case you want to join using another column, the column name should be specified.
`na.rm`	Only used when joining with a dist object. By default NAs are kept.
`type`	In case two glottodata objects are joined, you can specify the type of join: "left" (default), "right", "full", or "inner"

Value

glottodata or glottosubdata, either with or without metatables. Object is returned as a data.frame or list, depending on the input.

See Also

glottosplit

Examples

```
glottodata <- glottoget("demodata")
glottodata_space <- glottojoin(glottodata, with = "glottospace")
glottodata_base <- glottojoin(glottodata, with = "glottobase")

# Join with a dist object
glottodata <- glottoget("demodata", meta = TRUE)
dist <- glottodist(glottodata)
glottodata_dist <- glottojoin(glottodata, with = dist)

# Join glottosubdata tables:
glottosubdata <- glottocreate(glottocodes = c("yucu1253", "tani1257"),
variables = 3, groups = c("a", "b"), n = 2, meta = FALSE)
glottodatatable <- glottojoin(glottodata = glottosubdata)
```

`glottomap` *Create static and dynamic maps from glottodata, or select languages from a map*

Description

With this function you can easily create static and dynamic maps from glottodata (by setting type to 'static' or 'dynamic'). Alternatively, by specifying type = "filter", you can interactively select languages by drawing a shape around them (mode = "draw"; default) or by clicking on them (mode = "click"). See ?glottofiltermap for more details.

Usage

```
glottomap(
  glottodata = NULL,
  color = NULL,
  label = NULL,
  type = NULL,
  ptsize = NULL,
  alpha = NULL,
  lbsize = NULL,
  palette = NA,
  rivers = FALSE,
  nclass = NULL,
  filename = NULL,
  projection = NULL,
  glotto_title = NULL,
  mode = NULL,
  basemap = "country",
  ...
)
```

Arguments

`glottodata`	Optional, user-provided glottodata. In case no glottodata is provided, you can pass arguments directly to glottofilter.
`color`	glottovar, column name, or column index to be used to color features (optional). See 'Details' below.
`label`	glottovar, column name, or column index to be used to label features (optional). See 'Details' below.
`type`	One of: "static", "dynamic", or "filter". Default is "static".
`ptsize`	Size of points between 0 and 1
`alpha`	Transparency of points between 0 (very transparent) and 1 (not transparent)
`lbsize`	Size of labels between 0 and 1
`palette`	Color palette, see glottocolpal("all") for possible options, and run glottocolpal("turbo") to see what it looks like (replace it with palette name). Alternatively, you could also run tmaptools::palette_explorer(), RColorBrewer::display.brewer.all(), ?viridisLite::viridis, or scales::show_col(viridisLite::viridis(n=20))
`rivers`	Do you want to plot rivers?
`nclass`	Preferred number of classes (default is 5)
`filename`	Optional filename if you want to save resulting map
`projection`	For static maps, you can choose one of the following: 'eqarea' (equal-area Eckert IV, default), 'pacific' (Pacific-centered), or any other Coordinate Reference System, specified using an EPSG code (https://epsg.io/), for example: "ESRI:54009".
`glotto_title`	Optional, the title of legend, the default value is the name of the argument color.

`mode`	In case type = "filter", you can choose here whether you want to interactively select languages by clicking on them (mode = 'click', default) or by drawing a shape around them (mode = 'draw').
`basemap`	The default basemap is "country", which gives the borders of countries. Alternatively, the basemap can be set to be "hydro-basin", this gives global hydro-basins (Level 03).
`...`	Additional parameters to glottofilter

Details

If no glottodata object is provided, then you have the following options for the 'color' and 'label' arguments: ', 'glottocode', 'name', 'macroarea', 'isocode', 'countries', 'family_id', 'classification', 'parent_id', 'family', 'isolate', 'family_size', 'family_size_rank', 'country', 'sovereignty', 'type', 'geounit', 'continent', 'adm0_a3', '

Value

a map created from a glotto(sub)data object and can be saved with glottosave()

Examples

```
## Not run:
glottomap(country = "Netherlands")

glottopoints <- glottofilter(continent = "South America")
glottopols <- glottospace(glottopoints, method = "voronoi")
glottomap(glottodata = glottopols, color = "family_size_rank")
glottomap(glottodata = glottopols, color = "family", palette = "turbo",
type = "dynamic", label = "name")

glottodata <- glottoget()
families <- dplyr::count(glottodata, family, sort = TRUE)

# highlight 10 largest families:
glottodata <- glottospotlight(glottodata = glottodata, spotcol =
"family", spotlight = families$family[1:10], spotcontrast = "family")

# Or, place 10 largest families in background
glottodata <- glottospotlight(glottodata = glottodata, spotcol =
"family", spotlight = families$family[-c(1:10)], spotcontrast = "family")
glottomap(glottodata, color = "legend")

# Interactive selection by clicking on languages:
selected <- glottomap(continent = "South America", type = "filter")
glottomap(selected)

# Interactive selection by drawing a shape:
selected <- glottomap(continent = "South America", type = "filter", mode = "draw")
glottomap(selected)

## End(Not run)
```

glottomap_persist_diagram *Title*

Description

Title

Usage

```
glottomap_persist_diagram(glottodata, maxscale)
```

Arguments

`glottodata`	a glottodata is an object of sf with geometry type as ‘POINT‘
`maxscale`	a numeric number, maximum value of the rips filtration, the default unit is "100km"

Value

a ggplot2 map

Examples

```
glottopoints <- glottofilter(continent = "South America")
awk <- glottopoints[glottopoints$family == "Arawakan", ]
glottomap_persist_diagram(awk, maxscale = 15)
```

glottomap_rips_filt *Title*

Description

Title

Usage

```
glottomap_rips_filt(
  glottodata,
  r = 0,
  maxscale,
  is_animate = FALSE,
  length.out = 20,
  movie.name = "filtration.gif"
)
```

Arguments

`glottodata`	a glottodata is an object of sf with geometry type as ‘POINT‘
`r`	a numerica number, the radius of buffers of all the points in glottodata, the default unit is "100km"
`maxscale`	a numeric number, maximum value of the rips filtration, the default unit is "100km"
`is_animate`	if TRUE, it will generate a GIF file, if FALSE, it will generate a tmap plot, the default value is FALSE
`length.out`	the amount of images to be generated in GIF file when ‘is_animate = TRUE‘, the default value is ‘20‘
`movie.name`	name of the GIF file, the default value is "filtration.gif"

Value

if ‘is_animate = FALSE‘ return a tmap, if ‘is_animate = TRUE‘ return a GIF file

Examples

```
glottopoints <- glottofilter(continent = "South America")
awk <- glottopoints[glottopoints$family == "Arawakan", ]
glottomap_rips_filt(glottodata = awk, r = 6, maxscale = 8)
## Not run:
glottomap_rips_filt(glottodata = awk, r = 6, maxscale = 8, is_animate=TRUE)

## End(Not run)
```

`glottomatch` *glottomatch*

Description

Match a vector of language names to glottocodes and names

Usage

```
glottomatch(namevec, glottodata = NULL, tolerance = NULL)
```

Arguments

`namevec`	Vector of language names
`glottodata`	Optional, where to search for matches. If kept empty, the entire glottolog database will be searched, you could also search within a specific area
`tolerance`	Optional, search tolerance.

Value

a data.frame with exact or closest matches, and their glottocodes.

Examples

```
glottodata <- glottofilter(continent = "South America")
# Finds a single match
glottomatch(name = "yucuni", glottodata = glottodata)
# Finds multiple matches
glottomatch(name = "quechui", glottodata = glottodata)
```

`glottonmds` *Nonmetric Multidimensional Scaling for a glottodist object*

Description

This is a wrapper around the monoMDS function in the vegan package.

Usage

```
glottonmds(glottodist = NULL, k = NULL, na.rm = FALSE, row2id = NULL)
```

Arguments

`glottodist`	A glottodist object
`k`	Number of dimensions. Either 2 or 3 for nmds.
`na.rm`	Whether na's should be removed (default is FALSE)
`row2id`	In case of nmds, specify what each row contains (either 'glottocode' or 'glottosubcode')

Value

a glottonmds object which can be plotted using glottoplot(glottonmds =). See ?monoMDS for more details.

Examples

```
glottodata <- glottoget("demodata", meta = TRUE)
glottodist <- glottodist(glottodata = glottodata)
glottonmds <- glottonmds(glottodist, k = 2, row2id = "glottocode")
glottoplot(glottonmds = glottonmds)
```

`glottoplot` *Visualize glottodata or glottodistances*

Description

This function offers different types of visualizations for linguistic data and linguistic distances.

Usage

```
glottoplot(
  glottodata = NULL,
  glottodist = NULL,
  type = NULL,
  glottonmds = NULL,
  color = NULL,
  ptsize = NULL,
  label = NULL,
  filename = NULL,
  palette = NULL,
  k = NULL,
  na.rm = FALSE,
  row2id = NULL,
  preventoverlap = FALSE,
  alpha = NULL,
  colorvec = NULL,
  expand = NULL,
  lbsize = NULL,
  ptshift = NULL,
  lbshift = NULL
)
```

Arguments

`glottodata` glottodata table

`glottodist` A dist object created with `glottodist`

`type` The type of plot: "heatmap", "nmds", or "missing". Default is heatmap if nothing is provided.

`glottonmds` A glottonmds object created with `glottonmds`

`color` Name of variable to be used to color features (optional). See 'Details' below.

`ptsize` Size of points between 0 and 1 (optional)

`label` Name of variable to be used to label features (optional). See 'Details' below.

`filename` Optional filename if output should be saved.

`palette` Name of color palette, use glottocolpal("all") to see the options

`k` Number of dimensions. Either 2 or 3 for nmds.

`na.rm` Whether na’s should be removed (default is FALSE)

`row2id` In case of nmds, specify what each row contains (either ’glottocode’ or ’glottosubcode’)

`preventoverlap` For nmds with 2 dimensions, should overlap between data points be prevented?

`alpha` For nmds with 2 dimensions: Transparency of points between 0 (very transparent) and 1 (not transparent)

`colorvec` Vector specifying colors for individual values and legend order (non-matching values are omitted), for example: c("Arawakan" = "rosybrown1", "Yucuna" = "red", "Tucanoan" = "lightskyblue1", "Tanimuca-Retuarã" = "blue", "Naduhup" = "gray70", "Kakua-Nukak" = "gray30")

`expand` Optionally expand one or all of the axes. Default is c(0,0,0,0), referring to respectively xmin, xmax, ymin, ymax. If you want to change the maximum of the x-axis, you would do: c(0,1,0,0).

`lbsize` Label size (optional)

`ptshift` (optional) If preventoverlap is TRUE, how much should points be shifted?

`lbshift` (optional) If preventoverlap is TRUE, how much should labels be shifted? See the ’values’ argument in ggplot2::scale_color_manual() for details.

Details

If no glottodata object is provided, then you have the following options for the ’color’ and ’label’ arguments: ’, ’glottocode’, ’name’, ’macroarea’, ’isocode’, ’countries’, ’family_id’, ’classification’, ’parent_id’, ’family’, ’isolate’, ’family_size’, ’family_size_rank’, ’country’, ’sovereignty’, ’type’, ’geounit’, ’continent’, ’adm0_a3’, ’

Value

a visualization of a glotto(sub)data, glottodist or glottonmds object, which can be saved with glottosave()

Examples

```
# Plot glottodist as nmds:
glottodata <- glottoget("demodata", meta = TRUE)
glottodist <- glottodist(glottodata = glottodata)
# glottoplot(glottodist = glottodist, type = "nmds",
#  k = 2, color = "family", label = "name", row2id = "glottocode")

# To create a stress/scree plot, you can run:
# goeveg::dimcheckMDS(matrix = as.matrix(glottodist), k = k)


# Plot missing data:
glottodata <- glottoget("demodata", meta = TRUE)
glottodata <- glottosimplify(glottodata)
glottoplot(glottodata = glottodata, type = "missing")
```

glottorecode_logical *Recode character columns to TRUE/FALSE*

Description

Recode character columns to TRUE/FALSE

Usage

```
glottorecode_logical(glottodata, structure, totrue = NULL, tofalse = NULL)
```

Arguments

glottodata	glottodata list
structure	structure table
totrue	values to recode to TRUE
tofalse	values to recode to FALSE

Examples

```
glottodata <- glottoget("demodata", meta = TRUE)
glottorecode_logical(glottodata, totrue = c("y", "Y", 1), tofalse = c("n", "N", 0),
structure = glottodata[["structure"]])

glottosubdata <- glottoget("demosubdata", meta = TRUE)
glottorecode_logical(glottosubdata, totrue = c("y", "Y", 1), tofalse = c("n", "N", 0),
structure = glottosubdata[["structure"]])
```

glottorecode_missing *Recode missing values to NA*

Description

Recode missing values to NA

Usage

```
glottorecode_missing(glottodata, tona)
```

Arguments

glottodata	glottodata
tona	Optional, additional values to recode to NA

Examples

```
glottodata <- glottoget("demodata", meta = TRUE)
glottorecode_missing(glottodata, tona = "?")

glottosubdata <- glottoget("demosubdata", meta = TRUE)
glottorecode_missing(glottosubdata, tona = "?")
```

glottosave *Save glottodata, maps and plots*

Description

If no filename is provided, the name of the glottodata object will be used.

Usage

```
glottosave(glottodata, filename = NULL)
```

Arguments

`glottodata`	User-provided glottodata
`filename`	Filename either with or without file extension

Details

If no file extension is provided, a sensible default file extension is chosen. Dynamic maps (tmap) are saved in .html format, static maps (tmap) are saved as .png. Spatial data (sf) are saved as geopackage (.GPKG) by default, but .shp is also possible.

Value

No object is returned, it will be save locally at the specified location

See Also

glottoget_glottodata

Other <glottodata>: `glottoget()`

Examples

```
glottodata <- glottoget("demodata", meta = FALSE)
# Saves as .xlsx
glottosave(glottodata, filename = file.path(tempdir(), "glottodata") )

glottospacedata <- glottospace(glottodata)
# Saves as .GPKG
glottosave(glottospacedata, filename = file.path(tempdir(), "glottodata") )
```

```
glottomap <- glottomap(glottodata)
# Saves as .png
glottosave(glottomap, filename = file.path(tempdir(), "glottomap") )

# Saves as .html
glottomap <- glottomap(glottodata, type = "dynamic",
             filename = file.path(tempdir(), "glottomap") )
```

glottosearch *Search within glottodata for languages, glottocodes, etc.*

Description

Search within glottodata for languages, glottocodes, etc.

Usage

```
glottosearch(
  search,
  glottodata = NULL,
  partialmatch = TRUE,
  columns = NULL,
  tolerance = NULL
)
```

Arguments

search	Character string to search for, this can be the name of a language, a family, a glottocode, isocode.
glottodata	Any linguistic or cultural dataset. Default is to search within glottobase.
partialmatch	By default, partial matches will be returned as well. In case you only want exact matches, this argument should be set to FALSE.
columns	By default, the entire dataset is searched, but optionally the search can be limited to specific columns.
tolerance	In case partialmatch is TRUE: what is the maximum difference between search term and match? Default is 0.1

Value

A subset of glottodata that matches search conditions (object returned as a data.frame/tibble)

Examples

```
glottosearch(search = "Yucuni")
glottosearch(search = "Yucuni", columns = "name")
glottosearch(search = "Yucuni", columns = c("name", "family"))
```

glottosimplify *Simplify glottodata structures*

Description

With glottosimplify, the structure of a glottodata object is simplified by removing tables and properties

Usage

```
glottosimplify(
  glottodata,
  droplist = TRUE,
  dropmeta = TRUE,
  dropspatial = TRUE,
  submerge = TRUE,
  dropunits = FALSE
)
```

Arguments

`glottodata`	glottodata or glottosubdata.
`droplist`	By default, if only one sheet is loaded, the data will be returned as a data.frame (instead of placing the data inside a list of length 1)
`dropmeta`	By default all metadata is removed.
`dropspatial`	By default spatial properties are removed.
`submerge`	By default, glottosubdata tables are merged into a single glottodata table.
`dropunits`	By default units are kept.

Value

a simplified version of the original dataset, either a data.frame/tibble or a list (depending on the selected options)

Examples

```
glottodata <- glottoget("demodata", meta = TRUE)
glottosimplify(glottodata)
```

glottospace — *Make glottodata spatial and generate language polygons from points.*

Description

This function takes glottodata (either with or without metadata) and turns it into spatial points or polygons.

Usage

```
glottospace(glottodata, method = NULL, radius = NULL)
```

Arguments

`glottodata`	A glottodata table, or list of a glottodata table and metadata table(s)
`method`	Interpolation method, either "buffer" or "voronoi" (synonymous with "thiessen")
`radius`	In case interpolation method "buffer", the radius in km around the points. If method "thiessen", a buffer will be created into the ocean, particularly relevant for island languages.

Value

A spatial version of glottodata. In case glottodata has metadata, only glottodata will be converted to spatial (but all metadata tables are kept). Object returned as sf object, or a list of which the first element is an sf object, depending on the input.

Examples

```
glottodata <- glottoget("demodata", meta = TRUE)

glottopols <- glottospace(glottodata, method = "voronoi")


glottodata <- glottofilter(country = "Netherlands")
glottopols <- glottospace(glottodata, method = "buffer", radius = 20)
glottomap(glottopols)

glottodata <- glottofilter(continent = "South America")
glottopols <- glottospace(glottodata, method = "thiessen")
glottomap(glottopols)

glottodata <- glottofilter(country = "Philippines")
glottopols <- glottospace(glottodata, radius = 100, method = "thiessen")
glottomap(glottopols)
```

`glottosplitmergemeta` *Split or merge metadata from glottodata (or glottosubdata)*

Description

Usually, you will run this function twice, once to split metadata from glottodata, and a second time to join it again.

Usage

```
glottosplitmergemeta(glottodata, splitted = NULL)
```

Arguments

`glottodata`	glottodata
`splitted`	if provided, the second element of the list will be joined with glottodata

Value

A list of length 2 in case only glottodata is provided, and a merged glottodata object otherwise.

See Also

glottojoin

glottosimplify

Examples

```
glottodata <- glottoget("demodata", meta = TRUE)
splitted <- glottosplitmergemeta(glottodata)
merged <- glottosplitmergemeta(glottodata = glottodata, splitted = splitted)
```

`glottospotlight` *Highlight certain data points in visualizations*

Description

This function creates two separate color scales: one for points to highlight, and a second for the remaining background points. It also creates a legend. This is useful for preparing the data for visualizations such as maps or other plots.

Usage

```
glottospotlight(glottodata, spotcol, spotlight, spotcontrast = NULL)
```

Arguments

`glottodata`	User-provided glottodata
`spotcol`	Name of the column that contains the data to put in the spotlights (as well as remaining background data).
`spotlight`	Selection of data to put in the spotlights.
`spotcontrast`	Optional column to contrast between data points in the spotlight.

Value

A glottodata object with columns added to be used in visualization.

Examples

```
glottodata <- glottofilter(country = c("Netherlands", "Germany", "Belgium") )
glottodata <- glottospotlight(glottodata = glottodata, spotcol = "country",
                              spotlight = "Netherlands")
glottomap(glottodata, color = "legend")
```

`glottostat_dist_permanova` *Permanova across all groups (overall or pairwise)*

Description

This function takes a dist object and performs a Permutational Multivariate Analysis of Variance (PERMANOVA). It can be used to test whether two or more groups are significantly different from each other (by specifying the `comparison` argument with either 'overall' or 'pairwise').

Usage

```
glottostat_dist_permanova(
  glottodist = NULL,
  glottodata = NULL,
  comparison = NULL,
  sample = NULL,
  permutations = NULL,
  by = NULL
)
```

Arguments

`glottodist`	a dist object
`glottodata`	glottodata contains sample
`comparison`	Either "overall" or "pairwise"
`sample`	sample table (optional). By default, searches for sample table in glottodata/glottosubdata.
`permutations`	Number of permutations (default is 999)
`by`	the column name of "sample", over which to compute the permanova.

Details

The argument by is the name of a column in the sample table, which can be either provided by a "sample" sheet in glottodata or given by the argument sample. The default value of by is "group". The function uses by to do the comparisons. The function calls vegan::adonis2(), type ?adonis2 for more details.

Examples

```
glottodata <- glottoget("demodata", meta = TRUE)
glottodist <- glottodist(glottodata, metric = "gower")
glottostat_dist_permanova(glottodist = glottodist, glottodata = glottodata, comparison = "pairwise")
```

---

glottostat_dist_permanova_mci

A temporary version of glottostat_dist_permanova

Description

A temporary version of glottostat_dist_permanova

Usage

```
glottostat_dist_permanova_mci(
  glottodist = NULL,
  glottodata = NULL,
  comparison = NULL,
  sample = NULL,
  permutations = NULL,
  by = NULL
)
```

Arguments

glottodist	a dist object
glottodata	a glottodata
comparison	comparision
sample	sample
permutations	permutations
by	by

glottostat_permanova *Permanova across all groups (overall or pairwise)*

Description

This function takes a glottodata or glottosubdata object and performs a Permutational Multivariate Analysis of Variance (PERMANOVA). It can be used to test whether two or more groups are significantly different from each other (by specifying the ’comparison’ argument with either ’overall’ or ’pairwise’). The function uses the ’group’ column in the sample table to do the comparisons. Before running the analysis, a distance matrix is constructed from the glotto(sub)data object using glottodist(). The function calls vegan::adonis2(), type ?adonis2 for more details.

Usage

```
glottostat_permanova(
  glottodata,
  comparison = NULL,
  sample = NULL,
  permutations = NULL,
  metric = "gower"
)
```

Arguments

`glottodata`	glottodata or glottosubdata
`comparison`	Either "overall" or "pairwise"
`sample`	sample table (optional). By default, searches for sample table in glottodata/glottosubdata.
`permutations`	Number of permutations (default is 999)
`metric`	Either "gower" or "anderberg"

Examples

```
glottodata <- glottoget("demodata", meta = TRUE)
glottostat_permanova(glottodata, comparison = "pairwise")

# Use subgroup (or another column in the structure table) as group
glottodata[["sample"]][,"group"] <- NULL # delete old 'group' column
glottodata[["sample"]][,"group"] <- glottodata[["sample"]][,"subgroup"]
glottostat_permanova(glottodata, comparison = "pairwise")

glottosubdata <- glottoget("demosubdata", meta = TRUE)
glottostat_permanova(glottodata = glottosubdata, comparison = "pairwise")
```

`phoible_param_sf` *Title*

Description

Title

Usage

```
phoible_param_sf(phoible_data)
```

Arguments

`phoible_data`	A non-spatial phoible dataset

Value

an sf object

Examples

```
phoible_sf <- phoible_param_sf(glottospace::phoible_raw)
```

Index