Package ‘DramaAnalysis’

September 18, 2020

Type Package
Title Analysis of Dramatic Texts
Version 3.0.2
Date 2020-09-07
Language en-US

Description Analysis of preprocessed dramatic texts, with respect to literary research.
   The package provides functions to analyze and visualize information about characters,
   stage directions, the dramatic structure and the text itself.
   The dramatic texts are expected to be in CSV format, which can be installed from within
   the package, sample texts are provided. The package and the reasoning behind it are described in

URL https://github.com/quadrama/DramaAnalysis

BugReports https://github.com/quadrama/DramaAnalysis/issues
License GPL (>= 3)

LazyData TRUE
Imports stats, utils, reshape2 (>= 1.4.2), readr (>= 1.1.1),
   data.table (>= 1.10.4), httr (>= 1.2.1), git2r (>= 0.24.0),
   xml2 (>= 1.2.0), tokenizers (>= 0.2.1), stringr (>= 1.4.0)

Depends R (>= 3.3.0)

RoxygenNote 7.1.1

Suggests testthat, fmsb, knitr, magrittr, highcharter, rmarkdown (>= 1.8), igraph (>= 1.1.2)

Encoding UTF-8

NeedsCompilation no

Author Nils Reiter [aut, cre] (0000-0003-3193-6170),
   Tim Strohmayer [ctb],
   Janis Pagel [ctb]

Maintainer Nils Reiter <nils.reiter@ims.uni-stuttgart.de>

Repository CRAN

Date/Publication 2020-09-18 15:00:07 UTC
R topics documented:

barplot.QDCharacterStatistics ........................................... 3
base_dictionary ............................................................. 4
characterNames ............................................................. 4
characterStatistics .......................................................... 5
combine ................................................................. 6
configuration ................................................................. 7
correlationAnalysis ........................................................ 8
data ................................................................. 9
dictionaryStatistics .......................................................... 9
dramaNames ................................................................. 11
ensureSuffix ................................................................. 12
filterByDictionary ........................................................... 12
filterCharacters ............................................................... 14
installCollectionData ......................................................... 15
installData ................................................................. 15
isolateCharacterSpeech ....................................................... 16
keyness ................................................................. 17
loadAllInstalledIds ........................................................... 18
loadCharacters ............................................................... 19
loadDrama ................................................................. 19
loadDramaTEI ................................................................. 20
loadFields ................................................................. 20
loadMeta ................................................................. 21
loadSet ................................................................. 22
loadText ................................................................. 23
mapPrefix ................................................................. 23
newCollection ............................................................... 24
numberOfPlays ................................................................. 25
personnelExchange ........................................................... 25
plot.QDHamming ............................................................... 26
plot.QDUtteranceStatistics ................................................... 27
plotSpiderWebs ............................................................... 28
postags ................................................................. 29
presence ................................................................. 29
qd.colors ................................................................. 30
report ................................................................. 30
segment ................................................................. 31
setCollectionDirectory ......................................................... 32
split.QD Drama ................................................................. 32
tfidf ................................................................. 33
utteranceStatistics ........................................................... 34

Index 35
barplot.QDCharacterStatistics

Stacked Bar Plot

Description

This function expects an object of type QDCharacterStatistics and plots the specified column as a stacked bar plot.

Usage

## S3 method for class 'QDCharacterStatistics'
barplot(
  height,  
  col = qd.colors,  
  column = "tokens",  
  order = -1,  
  labels = TRUE,  
  top = 5,  
  ...  
)

Arguments

height   The object of class QDCharacterStatistics that is to be plotted
col      The colors to use
column   Which column of the character statistics should be used?
order    Sort the fields inversely
labels   Whether to add character labels into the plot
top      Limit the labels to the top 5 characters. Otherwise, labels will become unreadable.

...      All remaining options are passed to barplot.default().

Value

See barplot.default().

See Also

barplot.default
**base_dictionary**  
*Base dictionary*

**Description**

A list of word fields, i.e., collections of German lemmas associated with the five concepts Familie (family), Krieg (war), Liebe (love), Ratio (reason) and Religion (religion). The base dictionary is for *demo purposes* only, because it doesn’t contain any umlaut characters.

**Usage**

`base_dictionary`

**Format**

A list with five entries, each of them being a character vector.

---

**characterNames**  
*Format Names*

**Description**

The function `characterNames()` is applicable on all tables with a character table (that are of the class `QDHasCharacter`). It can be used to reformat the character names. The function `FUN` is applied to the character name entries within the `QDDrama` object. The factor levels in the character column of `x` are replaced by the result values of `FUN`.

**Usage**

`characterNames(x, drama, FUN = stringr::str_to_title, sort = 0, ...)`

**Arguments**

- `x`  
The object in which we want to transform names, needs to inherit the type `QDHasCharacter`.

- `drama`  
The QDDrama object with all the information.

- `FUN`  
A function applied to the strings. Defaults to `stringr::str_to_title`, which converts the strings to title case.

- `sort`  
Numeric. If set to a non-zero value, the resulting data.frame will be sorted alphabetically according to the drama and character name. If the value is above 0, the sorting will be ascending, if set to a negative value, the sorting is descending. If sort is set to 0 (the default), the order is unchanged. The ordering can also be specified explicitly, by passing an integer vector with as many elements as `x` has rows.

- `...`  
All other arguments are ignored.
The function returns x, but with modified character names.

See Also

str_to_title

Examples

data(rksp.0)
ustat <- utteranceStatistics(rksp.0)
ustat <- characterNames(ustat, rksp.0)

characterStatistics  Basic Character Statistics

Description

This function extracts character statistics from a drama object.

Usage

characterStatistics(
  drama, 
  normalize = FALSE, 
  segment = c("Drama", "Act", "Scene"),
  filterPunctuation = FALSE
)

Arguments

drama  A QDDrama object
normalize Normalizing the individual columns
segment  "Drama", "Act", or "Scene". Allows calculating statistics on segments of the play
filterPunctuation Whether to exclude all punctuation from token counts

Value

A data frame with the additional classes QDCharacterStatistics and QDHasCharacter. It has following columns and one row for each character: tokens: The number of tokens spoken by that character types : The number of different tokens (= types) spoken by each character utterances: The number of utterances utteranceLengthMean: The mean length of utterances utteranceLengthSd: The standard deviation in utterance length
See Also

characterNames

Examples

```r
data(rksp.0)
stat <- characterStatistics(rksp.0)
d <- combine(rjmw.0, rksp.0)
```

---

**Description**

The function `combine(x, y)` can be used to merge multiple objects of the type QDDrama into one.

**Usage**

`combine(x, y)`

**Arguments**

- `x`: A QDDrama
- `y`: A QDDrama

**Value**

A single QDDrama object that represents both plays.

**Examples**

```r
data(rksp.0)
data(rjmw.0)
d <- combine(rjmw.0, rksp.0)
```
configuration

Character Configuration

Description

The function `configuration(...)` creates a drama configuration matrix as a `QDConfiguration` object, which is also a `data.frame`. The S3 function `as.matrix()` can be used to extract a numeric or logical matrix containing the core.

Usage

```r
configuration(
  d,
  segment = c("Act", "Scene"),
  mode = c("Active", "Passive"),
  onlyPresence = FALSE
)
```

## S3 method for class 'QDConfiguration'

```r
as.matrix(x, ...)
```

Arguments

- `d`: A `QDDrama` object
- `segment`: A character vector, either "Act" or "Scene". Partial matching allowed.
- `mode`: Character vector, should be either "Active" or "Passive". Passive configurations express when characters are mentioned, active ones when they speak themselves. Please note that extracting passive configuration only makes sense if some form of coreference resolution has taken place on the text, either manually or automatic. If not, only very basic references (first person pronouns and proper names) are represented, which usually gives a very wrong impression.
- `onlyPresence`: If TRUE, the function only records whether a character was present. If FALSE (which is the default), the function counts the number of tokens spoken (active) or referenced (passive).
- `x`: An object of class `QDConfiguration`
- `...`: All other arguments are passed to `as.matrix.data.frame`.

Value

Drama configuration matrix as a `QDConfiguration` object (of type `data.frame`).

Active and Passive Configurations

By default, we generate active matrices that are based on the character speech. A character is present in a scene or act, if they make an utterance. Using the argument `mode`, we can also create passive configuration matrices. They look very similar, but are based on who's mentioned in a scene or an act.
correlationAnalysis

See Also

characterNames

Examples

```r
# Active configuration matrix
data(rksp.0)
cfg <- configuration(rksp.0)

# Passive configuration matrix
cfg <- configuration(rksp.0, mode="Passive")
```

---

correlationAnalysis  Correlation analysis

Description

Calculates correlation of a frequency table with an outcome list according to given method. The function currently only works for pairwise correlation, i.e., two categories. Note that the function keyness() is actually better to do the same thing, and this function should not be used anymore in this fashion.

Usage

```
correlationAnalysis(text.ft, categories, method = "spearman", culling = 0, ...)
```

Arguments

text.ft  A matrix, containing words in columns and characters (or plays) in rows. This can be the result of the frequencytable() function.
categories  A factor or numeric vector that represents a list of categories.
method  The correlation method, passed on to cor()
culling  An integer. Words that appear in less items are removed. Defaults to 0 which doesn’t remove anything.
...  Arguments passed to cor()

Value

The function returns a data.frame with three columns: The word, it’s correlation score, and the category it is correlated to. The latter is mainly for an easier use of the results.
Examples

data(rksp.0)
ft <- frequencytable(rksp.0, byCharacter=TRUE)
g <- factor(c("m","m","m","f","m","m","m","f","m","m","f","m","m"))
rksp.0.cor <- correlationAnalysis(ft, g)

# to pre-filter by the total frequency of a word
ft <- frequencytable(rksp.0, byCharacter=TRUE)
ft <- ft[,colSums(ft) > 5]
correlationAnalysis(ft, g)

data

Data sets

Description

rksp.0 represents the data set exported from Lessings Emilia Galotti, rjmw.0 is the one exported from Miss Sara Sampson (also written by Lessing). Please note that in both plays, special characters have been removed for technical reasons. The text is German, but all umlauts have been replaced by another character. This is only a restriction of the pre-packaged files.

Usage

rksp.0

rjmw.0

Format

A list containing data.frames and data.table.
An object of class QDDrama (inherits from list) of length 6.

dictionaryStatistics

Dictionary Use

Description

These methods retrieve count the number of occurrences of the words in the dictionaries, across different speakers and/or segments. The function dictionaryStatistics() calculates statistics for dictionaries with multiple entries, dictionaryStatisticsSingle() only for a single word.

Extract the number part from a QDDictionaryStatistics table as a matrix
dictionaryStatistics

Usage
dictionaryStatistics(
  drama,
  fields = DramaAnalysis::base_dictionary[fieldnames],
  fieldnames = c("Liebe"),
  segment = c("Drama", "Act", "Scene"),
  normalizeByCharacter = FALSE,
  normalizeByField = FALSE,
  byCharacter = TRUE,
  column = "Token.lemma",
  ci = TRUE
)
dictionaryStatisticsSingle(
  drama,
  wordfield = c(),
  segment = c("Drama", "Act", "Scene"),
  normalizeByCharacter = FALSE,
  normalizeByField = FALSE,
  byCharacter = TRUE,
  fieldNormalizer = length(wordfield),
  column = "Token.lemma",
  ci = TRUE,
  colnames = NULL
)

## S3 method for class 'QDDictionaryStatistics'
as.matrix(x, ...)

Arguments
drama A QDDrama object.
fields A list of lists that contains the actual field names. By default, we load the base_dictionary.
fieldnames A list of names for the dictionaries.
segment The segment level that should be used. By default, the entire play will be used. Possible values are "Drama" (default), "Act" or "Scene".
normalizeByCharacter Logical. Whether to normalize by character speech length.
normalizeByField Logical. Whether to normalize by dictionary size. You usually want this.
byCharacter Logical, defaults to TRUE. If false, values will be calculated for the entire segment (play, act, or scene), and not for individual characters.
column The table column we apply the dictionary on. Should be either "Token.surface" or "Token.lemma", the latter is the default.
ci Whether to ignore case. Defaults to TRUE, i.e., case is ignored.
**dramaNames**  

Given a QDDrama object, this function generates a list of nicely formatted names, following the format string.

**Usage**

```r
dramaNames(x, ids = NULL, formatString = "%A: %T (%DM)\n", orderBy = "drama")
```

**Arguments**

- **x**
  - The QDDrama object

- **ids**
  - If specified, should be a character vector of play ids (prefixed with corpus). Then the return value only contains the plays in the vector and in the order specified.

**Examples**

```r
# Check multiple dictionary entries
data(rksp.0)
dstat <- dictionaryStatistics(rksp.0, fieldnames=c("Krieg","Familie"))
# Check a single dictionary entries
data(rksp.0)
fstat <- dictionaryStatisticsSingle(rksp.0, wordfield=c("der"))
mat <- as.matrix(dictionaryStatistics(rksp.0, fieldnames=c("Krieg","Familie")))
```
formatString  A character vector. Contains special symbols that are replaced by meta data entries about the plays. The following symbols can be used: - %T: title of the play - %A: Author name - %P: GND entry of the author (if known) - %DR, - %DM: The minimal date - %L: The language - %I: The id - %C: The corpus prefix

orderBy      The meta data key that the final list will be ordered by

Value

Character vector of formatted drama names

ensureSuffix  Utility functions

Description

ensureSuffix makes certain that a character vector ends in a given suffix

Usage

ensureSuffix(x, suffix)

Arguments

x  The character vector

suffix  The suffix

Value

The input character vector with the desired suffix

filterByDictionary  Word frequencies

Description

The function filterByDictionary() can be used to filter a matrix as produced by frequencytable() by the words in the given dictionary(/-ies).

The function frequencytable() generates a matrix of word frequencies by drama, act or scene and/or by character. The output of this function can be fed to stylo.
Usage

```r
filterByDictionary(
  ft,
  fields = DramaAnalysis::base_dictionary[fieldnames],
  fieldnames = c("Liebe")
)
```

```r
frequencytable(
  drama,
  acceptedPOS = postags$de$words,
  column = "Token.lemma",
  byCharacter = FALSE,
  sep = "|",
  normalize = FALSE,
  sortResult = FALSE,
  segment = c("Drama", "Act", "Scene")
)
```

Arguments

- **ft**: A matrix as produced by `frequencytable()`.
- **fields**: A list of lists that contains the actual field names. By default, we load the base_dictionary (as in `dictionaryStatistics()`).
- **fieldnames**: A list of names for the dictionaries.
- **drama**: A `QDDrama`. May be covering multiple texts.
- **acceptedPOS**: A list of accepted pos tags. Words of all POS tags not in this list are filtered out. Specify NULL or an empty list to include all words.
- **column**: The column name we should use (should be either `Token.surface` or `Token.lemma`).
- **byCharacter**: Logical. Whether the count is by character or by text.
- **sep**: The separation symbol that goes between drama name and character (if applicable). Defaults to the pipe symbol.
- **normalize**: Whether to normalize values or not. If set to TRUE, the values are normalized by row sums.
- **sortResult**: Logical. If true, the columns with the highest sum are ordered left (i.e., frequent words are visible first). If false, the columns are ordered alphabetically by column name.
- **segment**: Character vector. Whether the count is by drama (default), act or scene.

Value

Matrix of word frequencies in the format words X segments

See Also

`stylo`
Examples

data(rksp.0)
ftable <- frequencytable(rksp.0,
byCharacter = TRUE)

filtered <- filterByDictionary(ftable,
    fieldnames=c("Krieg", "Familie"))

data(rksp.0)
st <- frequencytable(rksp.0)

---

filterCharacters  Filter characters

Description

This function can be used to filter characters from all tables that contain a character column (and are of the class QDHasCharacter).

Usage

filterCharacters(
    hasCharacter,
    drama,
    by = c("rank", "tokens", "name"),
    n = ifelse(by == "tokens", 500, ifelse(by == "rank", 10, c())))

Arguments

hasCharacter  The object we want to filter.
drama  The QDDrama object.
by  Character vector. Specifies the filter mechanism.
n  The threshold or a list of character names/ids to keep.

Details

The function supports three filter mechanisms: The filter by rank sorts the characters according to the number of tokens they speak and keeps the top $n$ characters. The filter called tokens keeps all characters that speak $n$ or more tokens. The filter called name keeps the characters that are provided by name as a vector as $n$.

Value

The filtered QDHasCharacter object
installCollectionData

Download and install collection data

Description

Function to download collection data (grouped texts) from github. Overwrites (!) the current collections.

Usage

installCollectionData(
  dataDirectory =getOption("qd.datadir"),
  branchOrCommit = "master",
  repository = "metadata",
  baseUrl = "https://github.com/quadrama/
)

Arguments

dataDirectory    The data directory in which collection and data files are stored
branchOrCommit   The git branch, commit id, or tag that we want to download
repository       The repository
baseUrl          The github user (or group)

Value

NULL

installData      Download preprocessed drama data

Description

This function downloads pre-processed dramatic texts via http and stores them locally in your data directory
 isolateCharacterSpeech

Isolate Character Speech

Description

isolateCharacterSpeech() isolates the speeches of individual characters and optionally saves them in separate text files.

Usage

isolateCharacterSpeech(
  drama,
  segment = c("Drama", "Act", "Scene"),
  minTokenCount = 0,
  countPunctuation = TRUE,
  writeToFiles = TRUE,
  dir = getOption("qd.datadir")
)
keyness

Arguments

- **drama**: A text (or multiple texts, as a QDDrama object)
- **segment**: "Drama", "Act", or "Scene". Determines on what segment-level the speech is isolated.
- **minTokenCount**: The minimal token count for a speech to be considered (default = 0)
- **countPunctuation**: Whether to include punctuation in minTokenCount (default = TRUE)
- **writeToFiles**: Whether to write each isolated speech into a new text file (default = TRUE)
- **dir**: The directory into which the files will be written (default = data directory)

Value

A named list of character vectors, each corresponding to character speeches as defined by segment

Examples

```r
data(rksp.0)
isolateCharacterSpeech(rksp.0, segment="Scene", writeToFiles=FALSE)
```

---

**keyness**  

**Keywords**

**Description**

Given a frequency table (with texts as rows and words as columns), this function calculates log-likelihood and log ratio of one set of rows against the other rows. The return value is a list containing scores for each word. If the method is loglikelihood, the returned scores are unsigned G2 values. To estimate the direction of the keyness, the log ratio is more informative. A nice introduction into log ratio can be found [here](#).

**Usage**

```r
keyness(
  ft,
  categories = c(1, rep(2, nrow(ft) - 1)),
  epsilon = 1e-100,
  siglevel = 0.05,
  method = c(quoteloglikelihoodquot;', "logratio"),
  minimalFrequency = 10
)
```
Arguments

- **ft**: The frequency table
- **categories**: A factor or numeric vector that represents an assignment of categories.
- **epsilon**: null values are replaced by this value, in order to avoid division by zero
- **siglevel**: Return only the keywords above the significance level. Set to 1 to get all words
- **method**: Either "logratio" or "loglikelihood" (default)
- **minimalFrequency**: Words less frequent than this value are not considered at all

Value

A list of keywords, sorted by their log-likelihood or log ratio value, calculated according to [http://ucrel.lancs.ac.uk/llwizard.html](http://ucrel.lancs.ac.uk/llwizard.html).

Examples

```r
data("rksp.0")
ft <- frequencytable(rksp.0, byCharacter = TRUE, normalize = FALSE)
# Calculate log ratio for all words
categories <- factor(c("m", "m", "m", "m", "f", "m", "m", "f", "m", "m", "f", "m", "m"))
keywords <- keyness(ft, method = "logratio",
                    categories = categories,
                    minimalFrequency = 5)
# Remove words that are not significantly different
keywords <- keywords[match(names(keywords), names(keyness(ft, siglevel = 0.01)))]
```

---

**loadAllInstalledIds**  
**Installed texts**

Description

Returns a list of all ids that are installed

Usage

```r
loadAllInstalledIds(
  asDataFrame = FALSE,
  dataDirectory = getOption("qd.datadir")
)
```

Arguments

- **asDataFrame**: Logical value. Controls whether the return value is a list (with colon-joined ids) or a data.frame with two columns (corpus, drama)
- **dataDirectory**: The directory in which precompiled drama data is installed
**loadCharacters**  
*Character data loading*

**Description**
Loads a table of characters and meta data

**Usage**

```r
loadCharacters(
  ids,
  defaultCollection = "tg",
  dataDirectory = getOption("qd.datadir")
)
```

**Arguments**
- `ids` a list or vector of ids
- `defaultCollection` the default collection
- `dataDirectory` the data directory

**Value**
A data.frame extracted from the CSV file about characters

---

**loadDrama**  
*Load drama*

**Description**
This function loads one or more of the installed plays and returns them as a QDDrama object.

**Usage**

```r
loadDrama(ids, defaultCollection = "qd")
```

**Arguments**
- `ids` A vector of ids.
- `defaultCollection` If the ids do not have a collection prefix, the defaultCollection prefix is applied.
Value

The function returns a QDDrama object. This is essentially a list of data.tables, covering the different aspects (utterances, segments, characters, ...). If multiple ids have been supplied as arguments, the tables contain the information of multiple plays.

Examples

```r
# both are equivalent
## Not run:
installData("test")
d <- loadDrama(c("test:rksp.0", "test:rjmw.0"))
d <- loadDrama(c("rksp.0", "rjmw.0"), defaultCollection = "test")

## End(Not run)
```

---

loadDramaTEI  

Load drama

Description

This function parses and loads one or more dramas in raw TEI format.

Usage

`loadDramaTEI(filename)`

Arguments

- `filename` The filename of the drama to load (or a list thereof).

Value

The function returns an object of class QDDrama.

---

loadFields  

Dictionary Handling

Description

`loadFields()` loads dictionaries that are available on the web as plain text files.
Usage

```r
loadFields(
    fieldnames = c("Liebe", "Familie"),
    baseUrl = paste("https://raw.githubusercontent.com/quadrama/metadata/master",
                    ensureSuffix(directory, fileSep), sep = fileSep),
    directory = "fields/",
    fileSuffix = ".txt",
    fileSep = "/"
)
```

Arguments

- **fieldnames**  A list of names for the dictionaries. It is expected that files with that name can be found below the URL.
- **baseUrl** The base path delivering the dictionaries. Should end in a /, field names will be appended and fed into `read.csv()`.
- **directory** The last component of the base url. Useful to retrieve enriched word fields from metadata repo.
- **fileSuffix** The suffix for the dictionary files
- **fileSep** The file separator used to construct the URL Can be overwritten to load local dictionaries.

Value

A named list that holds the loaded dictionaries as character vectors.

File Format

Dictionary files should contain one word per line, with no comments or any other meta information. The entry name for the dictionary is given as the file name. It’s therefore best if it does not contain special characters. The dictionary must be in UTF-8 encoding, and the file needs to end on .txt.

Examples

```r
# retrieves word fields from github
fields <- loadFields(fieldnames=c("Liebe", "Familie", "Krieg"))
```

loadMeta  

Load meta data

Description

helper method to load meta data about dramatic texts (E.g., author, year). Does not load the texts, so it’s much faster.
**loadSet**

**Usage**

`loadMeta(ids)`

**Arguments**

`ids`  
A vector or list of drama ids

**Value**

A data frame

---

**loadSet**  
*Load Collections*

**Description**

Function to load a set from collection files. Can optionally set the set name as a genre in the returned table. `loadSets()` returns a table of all defined collections (and the number of plays in each).

**Usage**

`loadSet(setName, addGenreColumn = FALSE)`  
`loadSets()`

**Arguments**

`setName`  
A character vector. The name of the set(s) to retrieve.

`addGenreColumn`  
Logical. Whether to set the Genre-column in the returned table to the set name. If set to FALSE (default), a vector is returned. In this case, association to collections is not returned. Otherwise, it’s a data.frame.

**Value**

A character vector with play ids that belong to the set.
**loadText**

*Load Text*

### Description

Load Text

### Usage

```r
loadText(
  ids,
  includeTokens = FALSE,
  defaultCollection = "tg",
  unifyCharacterFactors = FALSE,
  variant = "UtterancesWithTokens"
)
```

### Arguments

- **ids**: A vector containing drama ids to be downloaded
- **includeTokens**: This argument has no meaning anymore. Tokens are always included.
- **defaultCollection**: The collection prefix is added if no prefix is found
- **unifyCharacterFactors**: Logical value, defaults to TRUE. Controls whether columns representing characters (i.e., Speaker.* and Mentioned.*) are sharing factor levels
- **variant**: The file variant to load

### Value

A data.frame that is also of class QDHasUtteranceBE.

---

**mapPrefix**

*Replace corpus prefix*

### Description

This function can be used to replace corpus prefixes. If a list of play ids contains textgrid prefixes, for instance, this function can be used to map them onto GerDraCor prefixes. Please note that the function does not check whether the play actually exists in the corpus.

### Usage

```r
mapPrefix(idList, map)
```
Arguments

idList  The list of ids in which we want to replace.
map     A list containing the old prefix as name and the new one as values.

Value

The function returns a list of the same length of the input list, but with replaced play prefixes.

Examples

# returns c("corpus2:play1", "corpus2:play2")
mapPrefix(c("corpus1:play1", "corpus1:play2"), list(corpus1="corpus2"))

newCollection  Create or Extend a Collection

Description

newCollection() can be used to create new collections or add dramas to existing collection files.

Usage

def newCollection(
  drama,
  name = ifelse(inherits(drama, "QDDrama"), paste(unique(drama$meta$drama)),
    paste(drama, collapse = "_")),
  writeToFile = TRUE,
  dir = getOption("qd.collectionDirectory"),
  append = TRUE
)

Arguments

drama  A text (or multiple texts, as data.frame or data.table), or a character vector con-

name  The name of the collection and its filename (default = concatenated drama IDs)

writeToFile = Whether to write the collection to a file (default = TRUE)

dir    The directory into which the collection file will be written (default = collection
directory)

append Whether to extend the collection file if it already exists. If FALSE, the file will

Value

The function returns the ids that belong to the collection as a character vector.
Examples

```
t <- combine(rksp.0, rjmw.0)
newCollection(t, writeToFile=FALSE)
newCollection(c("rksp.0", "rjmw.0"), writeToFile=FALSE) # produces identical file
newCollection(c("a", "b"), name="rksp.0_rjmw.0", writeToFile=FALSE) # adds "a" and "b" to the file
```

---

### numberOfPlays

**Number of plays**

The function `numberOfPlays()` determines how many different plays are contained in a single QDDrama object.

**Usage**

```
numberOfPlays(x)
```

**Arguments**

- `x` The QDDrama object

**Value**

An integer. The number of plays contained in the QDDrama object.

**Examples**

```
# returns 1
numberOfPlays(rksp.0)

# returns 2
numberOfPlays(combine(rksp.0, rjmw.0))
```

---

### personnelExchange

**Measuring Personnel Exchange over Boundaries**

There are multiple ways to quantify the number of characters that are exchanged over a scene or act boundary.

**Usage**

```
hamming(drama, variant = c("Trilcke", "Hamming", "NormalizedHamming"))

scenicDifference(drama, norm = length(unique(drama$text$Speaker.figure_id)))
```
Arguments

- **drama**: The QDDrama Object
- **variant**: For `hamming()`, variants are "Trilcke" (default), "NormalizedHamming", and "Hamming"
- **norm**: For `scenicDifference()`, specifies the normalization constant

Value

A QDHamming object, which is a list of values, one for each scene change. The values indicate the (potentially) normalized number of characters that are exchanged.

Examples

```r
data(rksp.0)
dist_trilcke <- hamming(rksp.0)
dist_hamming <- hamming(rksp.0, variant = "Hamming")
dist_nhamming <- hamming(rksp.0, variant = "NormalizedHamming")
```

plot.QDHamming

Personnel Exchange

Description

Uses the default scatterplot function to plot the personnel exchange in each scene.

Usage

```r
## S3 method for class 'QDHamming'
plot(x, drama = NULL, xlab = "Scene", ylab = "Exchange after Scene", ...)
```

Arguments

- **x**: A numeric vector generated from the function
- **drama**: Optional QDDrama object. If present, act boundaries and correct scene labels are included in the plot.
- **xlab**: A character vector that is used as x axis label. Defaults to "Scene".
- **ylab**: A character vector that is used as y axis label. Defaults to "Exchange".
- **...**: Parameters passed to `plot.default()`.

Value

See `plot.default()`.

See Also

`plot.default`
plot.QDUtteranceStatistics

Examples

```r
data(rksp.0)
h <- hamming(rksp.0)
plot(h, drama=rksp.0)
```

---

plot.QDUtteranceStatistics

*Utterance positions*

---

Description

Uses the function `stripchart` to plot each utterance at their position, in a line representing the character. The dot is marked in the middle of each utterance. Might look weird if very long utterances are present.

Usage

```r
## S3 method for class 'QDUtteranceStatistics'
plot(x, drama = NULL, colors = qd.colors, xlab = "Time", ...)
```

Arguments

- `x` A table generated from the function
- `drama` Optional QDDrama object. If present, segment boundaries are extracted from it and included in the plot.
- `colors` The colors to be used
- `xlab` A character vector that is used as x axis label. Defaults to "Time".
- `...` Parameters passed to `stripchart()`.

Value

See `stripchart()`.

See Also

`stripchart`
plotSpiderWebs  Spider-Webs

Description

Generates spider-web like plot. Spider webs may look cool, but they are terrible to interpret. You should think of using a bar chart to represent the same information. You have been warned.

Usage

```
plotSpiderWebs(
  dstat,
  symbols = c(17, 16, 15, 4, 8),
  cglcol = "black",
  legend = TRUE,
  legend.cex = 0.7,
  legend.pos.x = "bottomright",
  legend.pos.y = NA,
  legend.horizontal = FALSE,
  pcol = qd.colors,
  ...)
```

Arguments

- **dstat**: A data frame containing data, e.g., output from `dictionaryStatistics()`
- **symbols**: Symbols to be used in the plot
- **cglcol**: The color for the spider net
- **legend**: Whether to print a legend
- **legend.cex**: Scaling factor for legend
- **legend.pos.x**: X position of legend
- **legend.pos.y**: Y position of legend
- **legend.horizontal**: Whether to print legend horizontally or vertically
- **pcol**: The line color(s)
- **...**: Miscellaneous arguments to be given for `radarchart()`.

Value

No value is returned.

Note

Radar charts and spider web plots are dangerous, they can easily become misleading. They are in this package for historic reasons, but should not be used anymore.
Examples

data(rksp.0)
fnames <- c("Krieg", "Liebe", "Familie", "Ratio", "Religion")
ds <- dictionaryStatistics(rksp.0, normalizeByField=TRUE,
fieldnames=fnames)
plotSpiderWebs(ds)

postags

Provides lists of groups of pos tags for various word classes.

Description

Provides lists of groups of pos tags for various word classes.

Usage

postags

Format

An object of class list of length 1.

presence

Active and Passive Presence

Description

This function should be called for a single text. It returns a data.frame with one row for each character in the play. The data.frame contains information about the number of scenes in which a character is actively speaking or passively mentions. Please note that the information about passive presence is derived from coreference resolved texts, which is a difficult task and not entirely reliable. The plays included in the package feature manually annotated coreferences (and thus, the presence is calculated on the basis of very well data).

Usage

presence(drama, passiveOnlyWhenNotActive = TRUE)

Arguments

drama A single drama
passiveOnlyWhenNotActive Logical. If true (default), passive presence is only counted if a character is not actively present in the scene.
Value

QDHasCharacter, data.frame. Columns actives, passives and scenes show the absolute number of scenes in which a character is actively/passively present, or the total number of scenes in the play. The column presence is calculated as $\frac{\text{actives} - \text{passives}}{\text{scenes}}$.

Examples

data(rksp.0)
presence(rksp.0)

qd.colors

**QuaDramA colors**

description

color scheme to be used for QuaDramA plots. Taken from http://google.github.io/palette.js/, tol-rainbow, 10 colors

Usage

qd.colors

Format

An object of class character of length 10.

report

**Report**

description

generates a report for a specific dramatic text

Usage

report(
  id = "test:rksp.0",
  of = file.path(getwd(), paste0(unlist(strsplit(id, ":", fixed = TRUE))[2], ".html")),
  type = c("Single", "Compare"),
  ... )
**Arguments**

- **id**: The id of the text or a list of ids
- **of**: The output file
- **type**: The type of the report. "Single" gives a report about a single play, while "Compare" can be used to compare multiple editions of a play. Please note that the "Compare" report is still under development.
- ... Arguments passed through to the rmarkdown document

**Value**

The return value of `render`

---

**segment**

*Segment*

**Description**

This function takes two tables and combines them. The first table is of the class QDHasUtteranceBE and contains text spans that are designated with begin and end character positions. The second table of class QDHasSegments contains information about acts and scenes in the play. This function is used internally in many other functions, but is exported because it might become useful.

**Usage**

```r
segment(hasUtteranceBE, hasSegments)
```

**Arguments**

- **hasUtteranceBE**: Table with utterances
- **hasSegments**: Table with segment info

**Value**

The function returns a `data.table` that has both the play segmentation and the token data in it.

**Examples**

```r
data(rksp.0)
segmentedText <- segment(rksp.0$text, rksp.0$segments)
```
setCollectionDirectory

*This function initializes the paths to data files.*

**Description**

This function initializes the paths to data files.

**Usage**

```r
setCollectionDirectory(
  collectionDirectory = file.path(getOption("qd.datadir"), "collections")
)
```

```r
setDirectories(
  dataDirectory = file.path(path.expand("~"), "QuaDramA", "Data2"),
  collectionDirectory = file.path(dataDirectory, "collections")
)
```

```r
setDataDirectory(
  dataDirectory = file.path(path.expand("~"), "QuaDramA", "Data2")
)
```

**Arguments**

- `collectionDirectory` A path to the directory in which collections are stored. By default, the directory is called "collection" below the data directory.

- `dataDirectory` A path to the directory in which data and metadata are located. "~/QuaDramA/Data2" by default.

**Value**

The `set*Directory()` functions always return `NULL`.

---

**split.QDDrama**

*Split multiple plays*

**Description**

The function `split(x)` expects an object of type `QDDrama` and can be used to split a `QDDrama` object that consists of multiple dramas into a list thereof. It is the counterpart to `combine(x,y)`. 

```r
split.QDDrama(x)
```
Usage

```r
## S3 method for class 'QDDrama'
split(x, ...)
```

Arguments

- `x`: The object of class QDDrama (consisting of multiple dramas). For `split()` it should consist of multiple plays. For `combine()` it can but doesn’t have to.
- `...`: All other arguments are ignored.

Value

Returns a list of individual QDDrama objects, each containing one text.

Examples

```r
data(rksp.0)
data(rjmw.0)
d <- combine(rjmw.0, rksp.0)
dlist <- split(d)
```

---

### `tfidf`

**TF-IDF**

Description

This function calculates a variant of TF-IDF. The input is assumed to contain relative frequencies. IDF is calculated as follows: \( idf_t = \log \frac{N+1}{n_t} \), with \( N \) being the total number of documents (i.e., rows) and \( n_t \) the number of documents containing term \( t \). We add one to the denominator to prevent terms that appear in every document to become 0.

Usage

```r
tfidf(ftable)
```

Arguments

- `ftable`: A matrix, containing "documents" as rows and "terms" as columns. Values are assumed to be normalized by document, i.e., contain relative frequencies.

Value

A matrix containing TF*IDF values instead of relative frequencies.
Examples

```r
data(rksp.0)
ftable <- frequencytable(rksp.0, byCharacter=TRUE, normalize=TRUE)
rksp.0.tfidf <- tfidf(ftable)
mat <- matrix(c(0.10,0.2, 0,
                0, 0.2, 0,
                0.1, 0.2, 0.1,
                0.8, 0.4, 0.9),
               nrow=3,ncol=4)
mat2 <- tfidf(mat)
print(mat2)
```

Utterance Statistics

Description

This method calculates the length of each utterance, organized by character and drama.

Usage

`utteranceStatistics(drama, normalizeByDramaLength = TRUE)`

Arguments

- `drama` The dramatic text(s)
- `normalizeByDramaLength` Logical value. If true, the resulting values will be normalized by the length of the drama.

Value

Returns an object of class QDUtteranceStatistics, which is essentially a data.frame.

See Also

`characterNames`

Examples

```r
data(rksp.0)
ustat <- utteranceStatistics(rksp.0)
boxplot(ustat$utteranceLength ~ ustat$character, 
        col=qd.colors[1:5],
        las=2, frame=FALSE)
```
Index

* datasets
  base_dictionary, 4
  data, 9
  postags, 29
  qd.colors, 30

as.matrix.QDConfiguration
  (configuration), 7
as.matrix.QDDictionaryStatistics
  (dictionaryStatistics), 9

barplot.QDCharacterStatistics, 3
base_dictionary, 4
characterNames, 4, 6, 8, 11, 34
characterStatistics, 5
combine, 6
configuration, 7
cor, 8
correlationAnalysis, 8
data, 9
dictionaryStatistics, 9
dictionaryStatisticsSingle
  (dictionaryStatistics), 9
dramaNames, 11
ensureSuffix, 12
filterByDictionary, 12
filterCharacters, 14
frequencytable, 8
frequencytable (filterByDictionary), 12
hamming (personnelExchange), 25
installCollectionData, 15
installData, 15
isolateCharacterSpeech, 16
keyness, 8, 17
loadAllInstalledIds, 18
loadCharacters, 19
loadDrama, 19
loadDramaTEI, 20
loadFields, 11, 20
loadMeta, 21
loadSet, 22
loadSets (loadSet), 22
loadText, 23
mapPrefix, 23
newCollection, 24
numberOfPlays, 25
personnelExchange, 25
plot.QDHamming, 26
plot.QDUtteranceStatistics, 27
plotSpiderWebs, 28
postags, 29
presence, 29
qd.colors, 30
render, 31
report, 30
rjmw.0 (data), 9
rksp.0 (data), 9
scenicDifference (personnelExchange), 25
segment, 31
setCollectionDirectory, 32
setDataDirectory
  (setCollectionDirectory), 32
setDirectories
  (setCollectionDirectory), 32
split.QDDrama, 32
tfidf, 33
utteranceStatistics, 34