Package 'qtkit'

August 17, 2024

Title Quantitative Text Kit

Version 1.0.0

Description Support package for the textbook ``An Introduction to Quantitative Text Analysis for Linguists: Reproducible Research Using R" (Francom, 2024) <doi:10.4324/9781003393764>. Includes functions to acquire, clean, and analyze text data as well as functions to document and share the results of text analysis. The package is designed to be used in conjunction with the book, but can also be used as a standalone package for text analysis.

License GPL (>= 3)

URL https://cran.r-project.org/package=qtkit

BugReports https://github.com/qtalr/qtkit/issues

SystemRequirements Chromium-based browser (e.g., Chrome, Chromium, or Brave)

Depends R (>= 4.1)

Imports chromote, dplyr, ggplot2, glue, gutenbergr, kableExtra, knitr, Matrix, openai, purrr, readr, rlang, stringr, tibble, tidytext

Suggests httptest, rmarkdown, testthat (>= 3.0.0), webshot2

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.3.1

VignetteBuilder knitr

Author Jerid Francom [aut, cre, cph] (<https://orcid.org/0000-0001-5972-6330>)

Maintainer Jerid Francom <francojc@wfu.edu>

NeedsCompilation no

Repository CRAN

Date/Publication 2024-08-17 17:10:02 UTC

Contents

`add_pkg_to_bib` *Add package to BibTeX file*

Description

This function adds a package to a BibTeX file. It uses the `knitr::write_bib` function to write the package name to the file.

Usage

```
add_pkg_to_bib(pkg_name, bib_file = "packages.bib")
```

Arguments

`pkg_name`	The name of the package to add to the BibTeX file.
`bib_file`	The name of the BibTeX file to write to.

Examples

```
my_bib_file <- tempfile(fileext = ".bib")
add_pkg_to_bib("dplyr", my_bib_file)
readLines(my_bib_file) |> cat(sep = "\n")
```

`calc_assoc_metrics` *Calculate Association Metrics for Bigrams*

Description

This function calculates various association metrics (PMI, Dice's Coefficient, G-score) for bigrams in a given corpus.

Usage

```
calc_assoc_metrics(
  data,
  doc_index,
  token_index,
  type,
  association = "all",
  verbose = FALSE
)
```

Arguments

`data`	A data frame containing the corpus.
`doc_index`	Column in 'data' which represents the document index.
`token_index`	Column in 'data' which represents the token index.
`type`	Column in 'data' which represents the tokens or terms.
`association`	A character vector specifying which metrics to calculate. Can be any combination of 'pmi', 'dice_coeff', 'g_score', or 'all'. Default is 'all'.
`verbose`	A logical value indicating whether to keep the intermediate probability columns. Default is FALSE.

Value

A data frame with one row per bigram and columns for each calculated metric.

Examples

```
data_path <- system.file("extdata", "bigrams_data.rds", package = "qtkit")
data <- readRDS(data_path)

calc_assoc_metrics(data, doc_index, token_index, type)
```

`calc_type_metrics` *Calculate Type Metrics for Text Data*

Description

This function calculates type metrics for tokenized text data.

Usage

```
calc_type_metrics(data, type, document, frequency = NULL, dispersion = NULL)
```

Arguments

`data`	A data frame containing the tokenized text data
`type`	The variable in `data` that contains the type (e.g., term, lemma) to analyze.
`document`	The variable in `data` that contains the document IDs.
`frequency`	A character vector indicating which frequency metrics to use. If NULL (default), only the `type` and `n` are returned. Other options: 'all', 'rf' calculates relative frequency, 'orf' calculates observed relative frequency. Can specify multiple options: c("rf", "orf").
`dispersion`	A character vector indicating which dispersion metrics to use. If NULL (default), only the `type` and `n` are returned. Other options: 'all', 'df' calculates Document Frequency. 'idf' calculates Inverse Document Frequency. 'dp' calculates Gries' Deviation of Proportions. Can specify multiple options: c("df", "idf").

Value

A data frame with columns:

- `type`: The unique types from the input data.
- `n`: The frequency of each type across all document. Optionally (based on the `frequency` and `dispersion` arguments):
- `rf`: The relative frequency of each type across all document.
- `orf`: The observed relative frequency (per 100) of each type across all document.
- `df`: The document frequency of each type.
- `idf`: The inverse document frequency of each type.
- `dp`: Gries' Deviation of Proportions of each type.

References

Gries, Stefan Th. (2023). Statistical Methods in Corpus Linguistics. In Readings in Corpus Linguistics: A Teaching and Research Guide for Scholars in Nigeria and Beyond, pp. 78-114.

Examples

```
data_path <- system.file("extdata", "types_data.rds", package = "qtkit")
data <- readRDS(data_path)
calc_type_metrics(
  data = data,
  type = type,
  document = document,
  frequency = c("rf", "orf"),
  dispersion = c("df", "idf")
)
```

`create_data_dictionary`

Create Data Dictionary

Description

This function takes a data frame and creates a data dictionary. The data dictionary includes the variable name, a human-readable name, the variable type, and a description. If a model is specified, the function uses OpenAI's API to generate the information based on the characteristics of the data frame.

Usage

```
create_data_dictionary(
  data,
  file_path,
  model = NULL,
  sample_n = 5,
  grouping = NULL,
  force = FALSE
)
```

Arguments

`data`	A data frame to create a data dictionary for.
`file_path`	The file path to save the data dictionary to.
`model`	The ID of the OpenAI chat completion models to use for generating descriptions (see `openai::list_models()`). If NULL (default), a scaffolding for the data dictionary is created.
`sample_n`	The number of rows to sample from the data frame to use as input for the model. Default NULL.
`grouping`	A character vector of column names to group by when sampling rows from the data frame for the model. Default NULL.
`force`	If TRUE, overwrite the file at `file_path` if it already exists. Default FALSE.

Value

A data frame containing the variable name, human-readable name, variable type, and description for each variable in the input data frame.

---

create_data_origin *Create data origin file*

Description

Data frame with attributes about the data origin, written to a CSV file and optionally returned.

Usage

```
create_data_origin(file_path, return = FALSE, force = FALSE)
```

Arguments

file_path	File path where the data origin file should be saved.
return	Logical value indicating whether the data origin should be returned.
force	Logical value indicating whether to overwrite the file if it already exists.

Value

A data frame containing the data origin information.

Examples

```
tmp_file <- tempfile(fileext = ".csv")
create_data_origin(tmp_file)
read.csv(tmp_file)
```

---

find_outliers *Identify Outliers in a Numeric Variable*

Description

This function identifies outliers in a numeric variable of a data.frame using the interquartile range (IQR) method.

Usage

```
find_outliers(data, variable_name)
```

Arguments

data	A data.frame object.
variable_name	A symbol representing a numeric variable in data.

Value

A data.frame containing the outliers in variable_name. If no outliers are found, the function returns NULL. The function also prints diagnostic information about the variable and the number of outliers found.

Examples

```
data(mtcars)
find_outliers(mtcars, mpg)
find_outliers(mtcars, wt)
```

---

get_archive_data *Download an archive file and unarchive its contents*

Description

Possible file types include .zip, .gz, .tar, and .tgz

Usage

```
get_archive_data(url, target_dir, force = FALSE, confirmed = FALSE)
```

Arguments

url	A character vector representing the full url to the compressed file
target_dir	The directory where the archive file should be downloaded
force	An optional argument which forcefully overwrites existing data
confirmed	If TRUE, the user has confirmed that they have permission to use the data. If FALSE, the function will prompt the user to confirm permission. Setting this to TRUE is useful for reproducible workflows.

Value

NULL, the archive file is unarchived in the target directory

Examples

```
## Not run:
data_dir <- file.path(tempdir(), "data")
url <-
 "https://raw.githubusercontent.com/qtalr/qtkit/main/inst/extdata/test_data.zip"
get_archive_data(
 url = url,
 target_dir = data_dir,
 confirmed = TRUE)

## End(Not run)
```

---

get_gutenberg_data *Get Works from Project Gutenberg*

Description

Retrieves works from Project Gutenberg based on specified criteria and saves the data to a CSV file. This function is a wrapper for the gutenbergr package.

Usage

```
get_gutenberg_data(
  target_dir,
  lcc_subject,
  birth_year = NULL,
  death_year = NULL,
  n_works = 100,
  force = FALSE,
  confirmed = FALSE
)
```

Arguments

target_dir	The directory where the CSV file will be saved.
lcc_subject	A character vector specifying the Library of Congress Classification (LCC) subjects to filter the works.
birth_year	An optional integer specifying the minimum birth year of authors to include.
death_year	An optional integer specifying the maximum death year of authors to include.
n_works	An integer specifying the number of works to retrieve. Default is 100.
force	A logical value indicating whether to overwrite existing data if it already exists.
confirmed	If TRUE, the user has confirmed that they have permission to use the data. If FALSE, the function will prompt the user to confirm permission. Setting this to TRUE is useful for reproducible workflows.

Details

This function retrieves Gutenberg works based on the specified LCC subjects and optional author birth and death years. It checks if the data already exists in the target directory and provides an option to overwrite it. The function also creates the target directory if it doesn't exist. If the number of works is greater than 1000 and the 'confirmed' parameter is not set to TRUE, it prompts the user for confirmation. The retrieved works are filtered based on public domain rights in the USA and availability of text. The resulting works are downloaded and saved as a CSV file in the target directory.

For more information on Library of Congress Classification (LCC) subjects, refer to the Library of Congress Classification Guide.

Value

A message indicating whether the data was acquired or already existed on disk, writes the data files to disk in the specified target directory.

Examples

```
## Not run:
data_dir <- file.path(tempdir(), "data")

get_gutenberg_data(
 target_dir = data_dir,
 lcc_subject = "JC"
 n_works = 5,
 confirmed = TRUE)

## End(Not run)
```

---

write_gg *Write a ggplot object to a file*

Description

This function is a wrapper around `ggsave` from the `ggplot2` package that allows you to write a ggplot object as part of a knitr document as an output for later use. It is designed to be used in a code block. The file name, if not specified, will be the label of the code block.

Usage

```
write_gg(
  gg_obj = NULL,
  file = NULL,
  target_dir = NULL,
  device = "pdf",
  theme = NULL,
  ...
)
```

Arguments

gg_obj	The ggplot to be written. If not specified, the last ggplot created will be written.
file	The name of the file to be written. If not specified, the label of the code block will be used.
target_dir	The directory where the file will be written. If not specified, the current working directory will be used.
device	The device to be used for saving the ggplot. Options include "pdf" (default), "png", "jpeg", "tiff", and "svg".
theme	The ggplot2 theme to be applied to the ggplot. Default is the theme specified in the ggplot2 options.
...	Additional arguments to be passed to the ggsave function from the ggplot2 package.

Value

The path of the written file.

Examples

```
## Not run:
library(ggplot2)

plot_dir <- file.path(tempdir(), "plot")

# Write a ggplot object as a PDF file
p <- ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point()

write_gg(
 gg_obj = p,
 file = "plot_file",
 target_dir = plot_dir,
 device = "pdf")

unlink(plot_dir)

## End(Not run)
```

---

write_kbl *Write a kable object to a file*

Description

This function is a wrapper around save_kable from the kableExtra package that allows you to write a kable object as part of a knitr document as an output for later use. It is designed to be used in a code block. The file name, if not specified, will be the label of the code block.

Usage

```
write_kbl(
  kbl_obj,
  file = NULL,
  target_dir = NULL,
  device = "pdf",
  bs_theme = "bootstrap",
  ...
)
```

Arguments

kbl_obj	The knitr_kable object to be written.
file	The name of the file to be written. If not specified, the name will be based on the current knitr code block label.
target_dir	The directory where the file will be written. If not specified, the current working directory will be used.
device	The device to be used for saving the file. Options include "pdf" (default), "html", "latex", "png", and "jpeg". Note that a Chromium-based browser (e.g., Google Chrome, Chromium, Microsoft Edge or Brave) is required on your system for all options except "latex'. If a suitable browser is not available, the function will stop and return an error message.
bs_theme	The Bootstrap theme to be applied to the kable object (only applicable for HTML output). Default is "bootstrap".
...	Additional arguments to be passed to the save_kable function from the kableExtra package.

Value

The path of the written file.

Examples

```
## Not run:
library(knitr)

table_dir <- file.path(tempdir(), "table")

mtcars_kbl <- kable(
 x = mtcars[1:5, ],
 format = "html")

# Write a kable object as a PDF file
write_kbl(
 kbl_obj = mtcars_kbl,
 file = "kable_pdf",
 target_dir = table_dir,
 device = "pdf")
```

```
# Write a kable as an HTML file with a custom Bootstrap theme
write_kbl(
 kbl_obj = mtcars_kbl,
 file = "kable_html",
 target_dir = table_dir,
 device = "html",
 bs_theme = "flatly")

unlink(table_dir)

## End(Not run)
```

`write_obj` *Write an R object as a file*

Description

This function is a wrapper around `dput` that allows you to write an R object as part of a knitr document as an output for later use. It is designed to be used in a code block. The file name, if not specified, will be the label of the code block. Use the standard `dget` function to read the file back into an R session.

Usage

```
write_obj(obj, file = NULL, target_dir = NULL, ...)
```

Arguments

`obj`	The R object to be written.
`file`	The name of the file to be written. If not specified, the label of the code block will be used.
`target_dir`	The directory where the file will be written. If not specified, the current working directory will be used.
`...`	Additional arguments to be passed to `dput`.

Value

The path of the written file.

Examples

```
## Not run:
obj_dir <- file.path(tempdir(), "obj")

# Write a data frame as a file
write_obj(
 obj = mtcars,
```

```
 file = "mtcars_data",
 target_dir = obj_dir)

# Read the file back into an R session
my_mtcars <- dget(file.path(obj_dir, "mtcars_data"))

unlink(obj_dir)

## End(Not run)
```

Index