Package 'qtkit'

December 8, 2024

Title Quantitative Text Kit

Version 1.1.0

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, gutenbergr, kableExtra, knitr, Matrix, openai, rlang, xml2

Suggests httptest, rmarkdown, testthat (>= 3.0.0), webshot2, fs, tibble, glue, readr

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 2024-12-08 00:10:02 UTC

Contents

`add_pkg_to_bib` *Add Package Citations to BibTeX File*

Description

Adds citation information for R packages to a BibTeX file. Uses the `knitr::write_bib` function to generate and append package citations in BibTeX format.

Usage

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

Arguments

`pkg_name`	Character string. The name of the R package to add to the BibTeX file.
`bib_file`	Character string. The path and name of the BibTeX file to write to. Default is "packages.bib".

Details

The function will create the BibTeX file if it doesn't exist, or append to it if it does. It includes citations for both the specified package and all currently loaded packages.

Value

Invisible NULL. The function is called for its side effect of writing to the BibTeX file.

Examples

```
# Create a temporary BibTeX file
my_bib_file <- tempfile(fileext = ".bib")

# Add citations for dplyr package
add_pkg_to_bib("dplyr", my_bib_file)

# View the contents of the BibTeX 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_normalized_entropy` *Calculate Normalized Entropy for Categorical Variables*

Description

Computes the normalized entropy (uncertainty measure) for categorical variables, providing a standardized measure of dispersion or randomness in the data.

Usage

```
calc_normalized_entropy(x)
```

Arguments

`x`	A character vector or factor containing categorical data.

Details

The function:

- Handles both character vectors and factors as input
- Treats NA values as a separate category
- Normalizes entropy to range (0,1) where:
  - 0 indicates complete certainty (one category dominates)
  - 1 indicates maximum uncertainty (equal distribution)

The calculation process:

1. Computes category proportions
2. Calculates raw entropy using Shannon's formula
3. Normalizes by dividing by maximum possible entropy

Value

A numeric value between 0 and 1 representing the normalized entropy:

- Values closer to 0 indicate less diversity/uncertainty
- Values closer to 1 indicate more diversity/uncertainty

Examples

```
# Calculate entropy for a simple categorical vector
x <- c("A", "B", "B", "C", "C", "C", "D", "D", "D", "D")
calc_normalized_entropy(x)

# Handle missing values
y <- c("A", "B", NA, "C", "C", NA, "D", "D")
calc_normalized_entropy(y)

# Works with factors too
z <- factor(c("Low", "Med", "Med", "High", "High", "High"))
calc_normalized_entropy(z)
```

---

calc_type_metrics *Calculate Frequency and Dispersion Metrics for Text Types*

Description

Calculates various frequency and dispersion metrics for types (terms/tokens) in tokenized text data. Provides a comprehensive analysis of how types are distributed across documents in a corpus.

Usage

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

Arguments

`data`	Data frame. Contains the tokenized text data with document IDs and types/terms.
`type`	Symbol. Column in `data` containing the types to analyze (e.g., terms, lemmas).
`document`	Symbol. Column in `data` containing the document identifiers.
`frequency`	Character vector. Frequency metrics to calculate: - NULL (default): Returns only type counts - 'all': All available metrics - 'rf': Relative frequency - 'orf': Observed relative frequency (per 100)
`dispersion`	Character vector. Dispersion metrics to calculate: - NULL (default): Returns only type counts - 'all': All available metrics - 'df': Document frequency - 'idf': Inverse document frequency - 'dp': Gries' deviation of proportions

Details

The function creates a term-document matrix internally and calculates the requested metrics. Frequency metrics show how often types occur, while dispersion metrics show how evenly they are distributed across documents.

The 'dp' metric (Gries' Deviation of Proportions) ranges from 0 (perfectly even distribution) to 1 (completely clumped distribution).

Value

Data frame containing requested metrics:

- type: Unique types from input data
- n: Raw frequency count
- rf: Relative frequency (if requested)
- orf: Observed relative frequency per 100 (if requested)
- df: Document frequency (if requested)
- idf: Inverse document frequency (if requested)
- dp: Deviation of proportions (if requested)

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")
df <- readRDS(data_path)
calc_type_metrics(
  data = df,
  type = letter,
  document = doc_id,
  frequency = c("rf", "orf"),
  dispersion = "dp"
)
```

---

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 Documentation*

Description

Creates a standardized data origin documentation file in CSV format, containing essential metadata about a dataset's source, format, and usage rights.

Usage

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

Arguments

file_path	Character string. Path where the CSV file should be saved.
return	Logical. If TRUE, returns the data frame in addition to saving. Default is FALSE.
force	Logical. If TRUE, overwrites existing file at path. Default is FALSE.

Details

Generates a template with the following metadata fields:

- Resource name
- Data source (URL/DOI)
- Sampling frame (language, modality, genre)
- Collection dates
- Data format
- Schema description
- License information
- Attribution requirements

Value

If return=TRUE, returns a data frame containing the data origin template. Otherwise returns invisible(NULL).

Examples

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

---

curate_enntt_data *Curate ENNTT Data*

Description

This function processes and curates ENNTT (European Parliament) data from a specified directory. It handles both .dat files (containing XML metadata) and .tok files '(containing text content).

Usage

```
curate_enntt_data(dir_path)
```

Arguments

dir_path	A string. The path to the directory containing the ENNTT data files. Must be an existing directory.

Details

The function expects a directory containing paired .dat and .tok files with matching names, as found in the raw ENNTT data https://github.com/senisioi/enntt-release. The .dat files should contain XML-formatted metadata with attributes:

- session_id: Unique identifier for the parliamentary session
- mepid: Member of European Parliament ID
- state: Country or state representation
- seq_speaker_id: Sequential ID within the session

The .tok files should contain the corresponding text content, one entry per line.

Value

A tibble containing the curated ENNTT data with columns:

- session_id: Parliamentary session identifier
- speaker_id: Speaker's MEP ID
- state: Representative's state/country
- session_seq: Sequential position in session
- text: Speech content
- type: Corpus type identifier

Examples

```
# Example using simulated data bundled with the package
example_data <- system.file("extdata", "simul_enntt", package = "qtkit")
curated_data <- curate_enntt_data(example_data)

str(curated_data)
```

`curate_swda_data` *Curate SWDA data*

Description

Process and curate Switchboard Dialog Act (SWDA) data by reading all .utt files from a specified directory and converting them into a structured format.

Usage

```
curate_swda_data(dir_path)
```

Arguments

dir_path	Character string. Path to the directory containing .utt files. Must be an existing directory.

Details

The function expects a directory containing .utt files or subdirectories with .utt files, as found in the raw SWDA data (Linguistic Data Consortium. LDC97S62: Switchboard Dialog Act Corpus.)

Value

A data frame containing the curated SWDA data with columns:

- doc_id: Document identifier
- damsl_tag: Dialog act annotation
- speaker_id: Unique speaker identifier
- speaker: Speaker designation (A or B)
- turn_num: Turn number in conversation
- utterance_num: Utterance number
- utterance_text: Actual spoken text

Examples

```
# Example using simulated data bundled with the package
example_data <- system.file("extdata", "simul_swda", package = "qtkit")
swda_data <- curate_swda_data(example_data)

str(swda_data)
```

---

find_outliers *Detect Statistical Outliers Using IQR Method*

Description

Identifies statistical outliers in a numeric variable using the Interquartile Range (IQR) method. Provides detailed diagnostics about the outlier detection process.

Usage

```
find_outliers(data, variable_name, verbose = TRUE)
```

Arguments

data	Data frame containing the variable to analyze.
variable_name	Unquoted name of the numeric variable to check for outliers.
verbose	Logical. If TRUE, prints diagnostic information about quartiles, fences, and number of outliers found. Default is TRUE.

Details

The function uses the standard IQR method for outlier detection:

- Calculates Q1 (25th percentile) and Q3 (75th percentile)
- Computes IQR = Q3 - Q1
- Defines outliers as values outside (Q1 - 1.5*IQR, Q3 + 1.5*IQR)

Value

If outliers are found:

- Data frame containing rows with outlier values
- Prints diagnostic information about quartiles and fences

If no outliers:

- Returns NULL
- Prints confirmation message

Diagnostic Output

- Variable name
- Q1 and Q3 values
- IQR value
- Upper and lower fence values
- Number of outliers found

Examples

```
data(mtcars)
find_outliers(mtcars, mpg)
find_outliers(mtcars, wt, verbose = FALSE)
```

---

get_archive_data *Download and Extract Archive Files*

Description

Downloads compressed archive files from a URL and extracts their contents to a specified directory. Supports multiple archive formats and handles permission confirmation.

Usage

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

Arguments

`url`	Character string. Full URL to the compressed archive file.
`target_dir`	Character string. Directory where the archive contents should be extracted.
`force`	Logical. If TRUE, overwrites existing data in target directory. Default is FALSE.
`confirmed`	Logical. If TRUE, skips permission confirmation prompt. Useful for reproducible workflows. Default is FALSE.

Details

Supported archive formats:

- ZIP (.zip)
- Gzip (.gz)
- Tar (.tar)
- Compressed tar (.tgz)

The function includes safety features:

- Permission confirmation for data usage
- Directory existence checks
- Archive format validation
- Automatic file cleanup

Value

Invisible NULL. Called for side effects:

- Downloads archive file
- Creates target directory if needed
- Extracts archive contents
- Cleans up temporary files

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 https://www.loc.gov/catdir/cpso/lcco/ 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 *Save ggplot Objects to Files*

Description

A wrapper around ggsave that facilitates saving ggplot objects within knitr documents. Automatically handles file naming and directory creation, with support for multiple output formats.

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.

Details

This function extends ggplot2::ggsave by:

- Using knitr code block labels for automatic file naming
- Creating target directories if they don't exist
- Supporting multiple output formats (PDF, PNG, JPEG, TIFF, SVG)
- Applying custom themes to plots before saving

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

A wrapper around kableExtra::save_kable that facilitates saving kable objects within knitr documents. Automatically handles file naming, directory creation, and supports multiple output formats with Bootstrap theming options.

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.

Details

The function extends save_kable functionality by:

- Using knitr code block labels for automatic file naming
- Creating target directories if they don't exist
- Supporting multiple output formats (PDF, HTML, LaTeX, PNG, JPEG)
- Applying Bootstrap themes for HTML output
- Preserving table styling and formatting

For HTML output, the function supports all Bootstrap themes available in kableExtra. The default theme is "bootstrap".

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

A wrapper around dput that facilitates saving R objects within knitr documents. Automatically handles file naming and directory creation, with support for preserving object structure and attributes.

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.

Details

This function extends dput functionality by:

- Using knitr code block labels for automatic file naming
- Creating target directories if they don't exist
- Preserving complex object structures and attributes
- Supporting all R object types (vectors, lists, data frames, etc.)

Objects saved with this function can be read back using the standard dget function.

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