Package 'scicomptools'

October 31, 2024

Type Package

Title Tools Developed by the NCEAS Scientific Computing Support Team

Version 1.1.0

Date 2024-10-30

Description Set of tools to import, summarize, wrangle, and visualize data. These functions were originally written based on the needs of the various synthesis working groups that were supported by the National Center for Ecological Analysis and Synthesis (NCEAS). These tools are meant to be useful inside and outside of the context for which they were designed.

License BSD_3_clause + file LICENSE

Encoding UTF-8

Language en-US

URL https://github.com/NCEAS/scicomptools, https://nceas.github.io/scicomptools/

BugReports https://github.com/NCEAS/scicomptools/issues

RoxygenNote 7.3.2

VignetteBuilder knitr

Imports chromote, data.tree, dplyr, gitcreds, googledrive, ggplot2, ggwordcloud, magrittr, methods, purrr, readxl, stringr, SemNetCleaner, tibble, tidyr, tidytext, tidyxl

Suggests ecodist, knitr, lme4, lmerTest, nlme, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Angel Chen [aut, cre] (angelchen7.github.io), Nicholas J Lyon [aut] (njlyon0.github.io), Gabriel Antunes Daldegan [aut] (<https://orcid.org/0000-0001-5345-4880>), Julien Brun [aut] (<https://orcid.org/0000-0002-7751-6238>),

Gabe De La Rosa [ctb] (www.gabrieldelarosa.com/),
Kara Koenig [aut] (<https://orcid.org/0000-0002-6371-7821>),
Kendall Miller [aut],
Timothy D Nguyen [aut] (www.linkedin.com/in/timothy-d-nguyen),
National Science Foundation [fnd] (NSF 1929393, 09/01/2019 - 08/31/2024),
University of California, Santa Barbara [cph]

Maintainer Angel Chen <angel777chen@gmail.com>

Repository CRAN

Date/Publication 2024-10-31 22:40:02 UTC

Contents

`drive_toc` *Identify all Folders within Specified Google Drive Folder*

Description

Identifies all sub-folders within a user-supplied Drive folder (typically the top-level URL). Also allows for exclusion of folders by name; useful if a "Backups" or "Archive" folder is complex and a table of contents is unwanted for that folder(s).

Usage

```
drive_toc(url = NULL, ignore_names = NULL, quiet = FALSE)
```

Arguments

`url`	(drive_id) Google Drive folder link modified by ‘googledrive::as_id‘ to be a true "Drive ID" (e.g., ‘url = as_id("url text")‘)
`ignore_names`	(character) Vector of name(s) of folder(s) to be excluded from list of folders

quiet (logical) Whether to message which folder it is currently listing (defaults to 'FALSE'). Complex folder structures will take time to fully process but the informative per-folder message provides solace that this function has not stopped working

Value

(node / R6) Special object class used by the 'data.tree' package

Examples

```
## Not run:
# Supply a single Google Drive folder link to identify all its sub-folders
drive_toc(url = googledrive::as_id("https://drive.google.com/drive/u/0/folders/your-folder"))

## End(Not run)
```

---

issue_extract *Export GitHub issues as PDF Files*

Description

Exports specified GitHub issues as PDF files when given the URL of a GitHub repository and a numeric vector of GitHub issue numbers. This function will export the first 10 issues as a default.

Usage

```
issue_extract(
  repo_url = NULL,
  issue_nums = 1:10,
  export_folder = NULL,
  cookies = NULL,
  quiet = FALSE
)
```

Arguments

repo_url (character) URL of the GitHub repository as a character string.

issue_nums (numeric) Numeric vector of the issue numbers to be exported. Default is issue #1 through #10.

export_folder (character) Name of the folder that will be created to contain the output PDF files. Default is "exported_issues".

cookies (character) Optional file path to the cookies to load into the Chrome session. This is only required when accessing GitHub repositories that require a login. See this link for more details: https://github.com/rstudio/chromote/blob/main/README.md#websites-that-require-authentication.

quiet (logical) Whether to silence informative messages while issues are being exported. Default is FALSE.

Value

No return value, called for side effects

Examples

```
## Not run:
# Export GitHub issue #7000 and #7080 through #7089 for the public `dplyr` repository
issue_extract(repo_url = "https://github.com/tidyverse/dplyr",
              issue_nums = c(7000, 7080:7089),
              export_folder = "dplyr_issues")

## End(Not run)
```

---

molec_wt *Identify Molecular Weight for a Given Element*

Description

Identifies molecular weight for the specified element based on the element's name, its symbol, or its atomic number. Returns only the molecular weight as a numeric value.

Usage

```
molec_wt(element = NULL)
```

Arguments

element	(character/numeric) element name, symbol, or atomic number for which to retrieve molecular weight

Value

(numeric) molecular weight value for the relevant element

Examples

```
# Identify molecular weight for carbon by name
molec_wt(element = "Carbon")

# Identify molecular weight for hydrogen by atomic number
molec_wt(element = 1)
```

read_xl_format *Read Formatting of All Sheets in an Excel Workbook*

Description

Retrieves all sheets of a Microsoft Excel workbook and identifies the formatting of each value (including column headers and blank cells).

Usage

```
read_xl_format(file_name = NULL)
```

Arguments

file_name	(character) Name of (and path to) the Excel workbook

Value

(data frame) One row per cell in the dataframe with a column for each type of relevant formatting and its 'address' within the original Excel workbook

Examples

```
# Identify the formatting of every cell in all sheets of an Excel file
read_xl_format(file_name = system.file("extdata", "excel_book.xlsx", package = "scicomptools"))
```

read_xl_sheets *Read All Sheets from an Excel Workbook*

Description

Retrieves all of the sheets in a given Microsoft Excel workbook and stores them as elements in a list. Note that the guts of this function were created by the developers of 'readxl::read_excel()' and we merely created a wrapper function to invoke their work more easily.

Usage

```
read_xl_sheets(file_name = NULL)
```

Arguments

file_name	(character) Name of (and path to) the Excel workbook

Value

(list) One tibble per sheet in the Excel workbook stored as separate elements in a list

Examples

```
# Read in each sheet as an element in a list
read_xl_sheets(file_name = system.file("extdata", "excel_book.xlsx", package = "scicomptools"))
```

`stat_extract` *Extract Summary Statistics from Model Fit Object*

Description

Accepts model fit object and extracts core statistical information. This includes P value, test statistic, degrees of freedom, etc. Currently accepts the following model types: ‘stats::t.test‘, ‘stats::lm‘, ‘stats_nls‘, ‘nlme::lme‘, ‘lmerTest::lmer‘, ‘ecodist::MRM‘, or ‘RRPP::trajectory.analysis‘

Usage

```
stat_extract(mod_fit = NULL, traj_angle = "deg")
```

Arguments

`mod_fit`	(lme, trajectory.analysis) Model fit object of supported class (see function description text)
`traj_angle`	(character) Either "deg" or "rad" for whether trajectory analysis angle information should be extracted in degrees or radians. Only required if model is trajectory analysis

Value

(data.frame) Dataframe of core summary statistics for the given model

Examples

```
# Create some example data
x <- c(3.5, 2.1, 7.5, 5.6, 3.3, 6.0, 5.6)
y <- c(2.3, 4.7, 7.8, 9.1, 4.5, 3.6, 5.1)

# Fit a linear model
mod <- lm(y ~ x)

# Extract the relevant information
stat_extract(mod_fit = mod)
```

token_check *Check Token Status*

Description

To make some direct-from-API workflows functional (e.g., Qualtrics surveys, etc.). It is necessary to quickly test whether a given R session "knows" the API token. This function returns an error if the specified token type isn't found and prints a message if one is found

Usage

```
token_check(api = "qualtrics", secret = TRUE)
```

Arguments

api	(character) API the token is for (currently only supports "qualtrics" and "github")
secret	(logical) Whether to include the token character string in the success message. FALSE prints the token, TRUE keeps it secret but returns a success message

Value

No return value, called for side effects

Examples

```
## Not run:
# Check whether a GitHub token is attached or not
token_check(api = "github", secret = TRUE)

## End(Not run)
## Not run:
# Check whether a Qualtrics token is attached or not
token_check(api = "qualtrics", secret = TRUE)

## End(Not run)
```

wd_loc *Define Local or Remote Working Directories*

Description

While working on the same script both in a remote server and locally on your home computer, defining file paths can be unwieldy and may even require duplicate scripts–one for each location–that require maintenance in parallel. This function allows you to define whether you are working locally or not and specify the path to use in either case.

Usage

```
wd_loc(local = TRUE, local_path = getwd(), remote_path = NULL)
```

Arguments

local	(logical) Whether you are working locally or on a remote server
local_path	(character) File path to use if ‘local‘ is ‘TRUE‘ (defaults to ‘getwd()‘)
remote_path	(character) File path to use if ‘local‘ is ‘FALSE‘

Value

(character) Either the entry of ‘local_path‘ or ‘remote_path‘ depending on whether ‘local‘ is set as true or false

Examples

```
# Set two working directory paths to toggle between

# If you are working in your local computer, set `local` to "TRUE"
wd_loc(local = TRUE,
       local_path = file.path("local path"),
       remote_path = file.path("path on server"))

# If you are working in a remote server, set `local` to "FALSE"
wd_loc(local = FALSE,
       local_path = file.path("local path"),
       remote_path = file.path("path on server"))
```

---

word_cloud_plot *Text Mine a Given Column and Create a Word Cloud*

Description

Mines a user-defined column of text and creates a word cloud from the identified words and bigrams.

Usage

```
word_cloud_plot(
  data = NULL,
  text_column = NULL,
  word_count = 50,
  known_bigrams = c("working group")
)
```

Arguments

data	dataframe containing at least one column
text_column	character, name of column in dataframe given to ‘data‘ that contains the text to be mined
word_count	numeric, number of words to be returned (counts from most to least frequent)
known_bigrams	character vector, all bigrams (two-word phrases) to be mined before mining for single words

Value

dataframe of one column (named ’word’) that can be used for word cloud creation. One row per bigram supplied in ‘known_bigrams‘ or single word (not including "stop words")

---

word_cloud_prep *Perform Text Mining of a Given Column*

Description

Mines a user-defined column to create a dataframe that is ready for creating a word cloud. It also identifies any user-defined "bigrams" (i.e., two-word phrases) supplied as a vector.

Usage

```
word_cloud_prep(
  data = NULL,
  text_column = NULL,
  word_count = 50,
  known_bigrams = c("working group")
)
```

Arguments

data	(dataframe) Data object containing at least one column
text_column	(character) Name of column in dataframe given to ‘data‘ that contains the text to be mined
word_count	(numeric) Number of words to be returned (counts from most to least frequent)
known_bigrams	(character) Vector of all bigrams (two-word phrases) to be mined before mining for single words

Value

dataframe of one column (named ’word’) that can be used for word cloud creation. One row per bigram supplied in ‘known_bigrams‘ or single word (not including "stop words")

Examples

```
# Create a dataframe containing some example text
text <- data.frame(article_num = 1:6,
                   article_title = c("Why pigeons are the best birds",
                                     "10 ways to show your pet budgie love",
                                     "Should you feed ducks at the park?",
                                     "Locations and tips for birdwatching",
                                     "How to tell which pet bird is right for you",
                                     "Do birds make good pets?"))

# Prepare the dataframe for word cloud plotting
word_cloud_prep(data = text, text_column = "article_title")
```

Index