Package 'tidyfinance'

July 3, 2024

Type Package

Title Tidy Finance Helper Functions

Version 0.2.1

Description Helper functions for empirical research in financial economics, addressing a variety of topics covered in Scheuch, Voigt, and Weiss (2023) <doi:10.1201/b23237>. The package is designed to provide shortcuts for issues extensively discussed in the book, facilitating easier application of its concepts. For more information and resources related to the book, visit <https://www.tidy-finance.org/r/index.html>.

License MIT + file LICENSE

URL https://www.tidy-finance.org/r/, https://github.com/tidy-finance/r-tidyfinance

BugReports https://github.com/tidy-finance/r-tidyfinance/issues

Depends R (>= 4.1)

Imports dplyr (>= 1.1.4), rlang (>= 1.1.3), tidyr (>= 1.3.1), purrr (>= 1.0.2), lubridate (>= 1.9.3), tibble (>= 3.2.1), lifecycle

Suggests frenchdata (>= 0.2.0), readxl (>= 1.4.0), RPostgres (>= 1.4.5), dbplyr (>= 2.5.0), DBI (>= 1.2.2), testthat (>= 3.2.0), knitr, rmarkdown

Encoding UTF-8

Config/testthat/edition 3

RoxygenNote 7.3.1

VignetteBuilder knitr

NeedsCompilation no

Author Christoph Scheuch [aut, cre] (<https://orcid.org/0009-0004-0423-6819>), Stefan Voigt [aut] (<https://orcid.org/0000-0001-5619-3161>), Patrick Weiss [aut] (<https://orcid.org/0000-0002-9282-5872>)

Maintainer Christoph Scheuch <christoph.scheuch@gmail.com>

Repository CRAN

Date/Publication 2024-07-03 09:50:02 UTC

Contents

`assign_portfolio` *Assign Portfolios Based on Sorting Variable*

Description

[Experimental]

This function assigns data points to portfolios based on a specified sorting variable. It can optionally filter the data by exchanges before assignment. The function requires either the number of portfolios to be created or specific percentiles for the breakpoints, but not both.

Usage

```
assign_portfolio(
  data,
  sorting_variable,
  n_portfolios = NULL,
  percentiles = NULL,
  exchanges = NULL
)
```

Arguments

`data`	A data frame containing the dataset for portfolio assignment.
`sorting_variable`	A string specifying the column name in `data` to be used for sorting and determining portfolio assignments.
`n_portfolios`	An optional integer specifying the number of equally sized portfolios to create. This parameter is mutually exclusive with `percentiles`.
`percentiles`	An optional numeric vector specifying the percentiles for determining the breakpoints of the portfolios. This parameter is mutually exclusive with `n_portfolios`.
`exchanges`	An optional character vector specifying exchange names to filter the data before computing breakpoints and assigning portfolios. Exchanges must be stored in a column named `exchange` in `data`. If `NULL`, no filtering is applied.

Value

A vector of portfolio assignments for each row in the input `data`.

Note

This function will stop and throw an error if both `n_portfolios` and `percentiles` are provided or if neither is provided. Ensure that you only use one of these parameters for specifying portfolio breakpoints.

Examples

```
data <- data.frame(
  id = 1:100,
  exchange = sample(c("NYSE", "NASDAQ"), 100, replace = TRUE),
  market_cap = runif(100, 1e6, 1e9)
)
assign_portfolio(data, "market_cap", n_portfolios = 5)
assign_portfolio(data, "market_cap", percentiles = c(0.2, 0.4, 0.6, 0.8), exchanges = c("NYSE"))
```

check_if_package_installed *Check If a Package is Installed*

Description

Checks if a specified package is installed and available for use. If the package is not installed, the function stops execution and prompts the user to install the package.

Usage

```
check_if_package_installed(package, type)
```

Arguments

package	The name of the package to check.
type	The type for which the package needs to be available.

Value

Invisible TRUE if the package is installed; otherwise, it stops execution with an error message advising the installation of the package. Since the function is designed to stop if the package is not found, it does not explicitly return a value upon successful completion.

check_supported_type *Check if a Dataset Type is Supported*

Description

This function checks if a given dataset type is supported by verifying against a list of all supported dataset types from different domains. If the specified type is not supported, it stops execution and returns an error message listing all supported types.

Usage

```
check_supported_type(type)
```

Arguments

type	The dataset type to check for support.

Value

Does not return a value; instead, it either passes silently if the type is supported or stops execution with an error message if the type is unsupported.

`create_summary_statistics` *Create Summary Statistics for Specified Variables*

Description

Computes a set of summary statistics for numeric and integer variables in a data frame. This function allows users to select specific variables for summarization and can calculate statistics for the whole dataset or within groups specified by the by argument. Additional detail levels for quantiles can be included.

Usage

```
create_summary_statistics(
  data,
  ...,
  by = NULL,
  detail = FALSE,
  drop_na = FALSE
)
```

Arguments

`data`	A data frame containing the variables to be summarized.
`...`	Comma-separated list of unquoted variable names in the data frame to summarize. These variables must be either numeric, integer, or logical.
`by`	An optional unquoted variable name to group the data before summarizing. If NULL (the default), summary statistics are computed across all observations.
`detail`	A logical flag indicating whether to compute detailed summary statistics including additional quantiles. Defaults to FALSE, which computes basic statistics (n, mean, sd, min, median, max). When TRUE, additional quantiles (1%, 5%, 10%, 25%, 75%, 90%, 95%, 99%) are computed.
`drop_na`	A logical flag indicating whether to drop missing values for each variabl (default is FALSE).

Details

The function first checks that all specified variables are of type numeric, integer, or logical. If any variables do not meet this criterion, the function stops and returns an error message indicating the non-conforming variables.

The basic set of summary statistics includes the count of non-NA values (n), mean, standard deviation (sd), minimum (min), median (q50), and maximum (max). If `detail` is TRUE, the function also computes the 1st, 5th, 10th, 25th, 75th, 90th, 95th, and 99th percentiles.

Summary statistics are computed for each variable specified in `...`. If a `by` variable is provided, statistics are computed within each level of the `by` variable.

Value

A tibble with summary statistics for each selected variable. If by is specified, the output includes the grouping variable as well. Each row represents a variable (and a group if by is used), and columns include the computed statistics.

`create_wrds_dummy_database`

Create WRDS Dummy Database

Description

Downloads the WRDS dummy database from the respective Tidy Finance GitHub repository and saves it to the specified path. If the file already exists, the user is prompted before it is replaced.

Usage

```
create_wrds_dummy_database(path)
```

Arguments

`path`	The file path where the SQLite database should be saved. If not provided, the default path is "data/tidy_finance_r.sqlite".

Value

Invisible NULL. Side effect: downloads a file to the specified path.

Examples

```
path <- paste0(tempdir(), "/tidy_finance_r.sqlite")
create_wrds_dummy_database(path)
```

`disconnection_connection`

Disconnect Database Connection

Description

This function safely disconnects an established database connection using the DBI package.

Usage

```
disconnection_connection(con)
```

Arguments

con	A database connection object created by DBI::dbConnect or any similar function that establishes a connection to a database.

Value

A logical value: TRUE if disconnection was successful, FALSE otherwise.

---

download_data *Download and Process Data Based on Type*

Description

Downloads and processes data based on the specified type (e.g., Fama-French factors, Global Q factors, or macro predictors), and date range. This function checks if the specified type is supported and then delegates to the appropriate function for downloading and processing the data.

Usage

```
download_data(type, start_date, end_date)
```

Arguments

type	The type of dataset to download, indicating either factor data or macroeconomic predictors.
start_date	The start date for filtering the data, in "YYYY-MM-DD" format.
end_date	The end date for filtering the data, in "YYYY-MM-DD" format.

Value

A tibble with processed data, including dates and the relevant financial metrics, filtered by the specified date range.

Examples

```
download_data("factors_ff3_monthly", "2000-01-01", "2020-12-31")
download_data("macro_predictors_monthly", "2000-01-01", "2020-12-31")
```

download_data_factors *Download and Process Factor Data*

Description

Downloads and processes factor data based on the specified type (Fama-French or Global Q), and date range. This function delegates to specific functions based on the type of factors requested: Fama-French or Global Q. It checks if the specified type is supported before proceeding with the download and processing.

Usage

```
download_data_factors(type, start_date, end_date)
```

Arguments

type	The type of dataset to download, indicating the factor model and frequency.
start_date	The start date for filtering the data, in "YYYY-MM-DD" format.
end_date	The end date for filtering the data, in "YYYY-MM-DD" format.

Value

A tibble with processed factor data, including dates, risk-free rates, market excess returns, and other factors, filtered by the specified date range.

Examples

```
download_data_factors("factors_ff3_monthly", "2000-01-01", "2020-12-31")
download_data_factors("factors_q5_daily", "2020-01-01", "2020-12-31")
```

download_data_factors_ff *Download and Process Fama-French Factor Data*

Description

Downloads and processes Fama-French factor data based on the specified type (e.g., "factors_ff3_monthly"), and date range. The function first checks if the specified type is supported and requires the 'frenchdata' package to download the data. It processes the raw data into a structured format, including date conversion, scaling factor values, and filtering by the specified date range.

Usage

```
download_data_factors_ff(type, start_date, end_date)
```

Arguments

type	The type of dataset to download, corresponding to the specific Fama-French model and frequency.
start_date	The start date for filtering the data, in "YYYY-MM-DD" format.
end_date	The end date for filtering the data, in "YYYY-MM-DD" format.

Value

A tibble with processed factor data, including the date, risk-free rate, market excess return, and other factors, filtered by the specified date range.

Examples

```
download_data_factors_ff("factors_ff3_monthly", "2000-01-01", "2020-12-31")
download_data_factors_ff("factors_ff_industry_10_monthly", "2000-01-01", "2020-12-31")
```

---

download_data_factors_q

Download and Process Global Q Factor Data

Description

Downloads and processes Global Q factor data based on the specified type (daily, monthly, etc.), date range, and source URL. The function first checks if the specified type is supported, identifies the dataset name from the supported types, then downloads and processes the data from the provided URL. The processing includes date conversion, renaming variables to a standardized format, scaling factor values, and filtering by the specified date range.

Usage

```
download_data_factors_q(
  type,
  start_date,
  end_date,
  url = "http://global-q.org/uploads/1/2/2/6/122679606/"
)
```

Arguments

type	The type of dataset to download (e.g., "factors_q5_daily", "factors_q5_monthly").
start_date	The start date for filtering the data, in "YYYY-MM-DD" format.
end_date	The end date for filtering the data, in "YYYY-MM-DD" format.
url	The base URL from which to download the dataset files, with a specific path for Global Q datasets.

Value

A tibble with processed factor data, including the date, risk-free rate, market excess return, and other factors, filtered by the specified date range.

Examples

```
download_data_factors_q("factors_q5_daily", "2020-01-01", "2020-12-31")
```

`download_data_macro_predictors`

Download and Process Macro Predictor Data

Description

Downloads and processes macroeconomic predictor data based on the specified type (monthly, quarterly, or annual), date range, and source URL. The function first checks if the specified type is supported, then downloads the data from the provided URL (defaulting to a Google Sheets export link). It processes the raw data into a structured format, calculating additional financial metrics and filtering by the specified date range.

Usage

```
download_data_macro_predictors(
  type,
  start_date,
  end_date,
 url = "https://docs.google.com/spreadsheets/d/1g4LOaRj4TvwJr9RIaA_nwrXXWTOy46bP"
)
```

Arguments

`type`	The type of dataset to download ("macro_predictors_monthly", "macro_predictors_quarterly", "macro_predictors_annual").
`start_date`	The start date for filtering the data, in "YYYY-MM-DD" format.
`end_date`	The end date for filtering the data, in "YYYY-MM-DD" format.
`url`	The URL from which to download the dataset, with a default Google Sheets export link.

Value

A tibble with processed data, filtered by the specified date range and including financial metrics.

Examples

```
download_data_macro_predictors("macro_predictors_monthly", "2000-01-01", "2020-12-31")
```

`download_data_wrds` *Download Data from WRDS*

Description

This function acts as a wrapper to download data from various WRDS datasets including CRSP, Compustat, and CCM links based on the specified type. It is designed to handle different data types by redirecting to the appropriate specific data download function.

Usage

```
download_data_wrds(type, start_date, end_date)
```

Arguments

`type`	A string specifying the type of data to download. It should match one of the predefined patterns to indicate the dataset: "wrds_crsp" for CRSP data, "wrds_compustat" for Compustat data, or "wrds_ccm_links" for CCM links data.
`start_date`	A date in 'YYYY-MM-DD' format indicating the start of the period for which data is requested.
`end_date`	A date in 'YYYY-MM-DD' format indicating the end of the period for which data is requested.

Value

A data frame containing the requested data, with the structure and contents depending on the specified `type`.

Examples

```
 crsp_monthly <- download_data_wrds("wrds_crsp_monthly", "2020-01-01", "2020-12-31")
compustat_annual <- download_data_wrds("wrds_compustat_annual", "2020-01-01", "2020-12-31")
 ccm_links <- download_data_wrds("wrds_ccm_links", "2020-01-01", "2020-12-31")
```

`download_data_wrds_ccm_links` *Download CCM Links from WRDS*

Description

This function downloads data from the WRDS CRSP/Compustat Merged (CCM) links database. It allows users to specify the type of links (`linktype`), the primacy of the link (`linkprim`), and whether to use flagged links (`usedflag`).

Usage

```
download_data_wrds_ccm_links(
  linktype = c("LU", "LC"),
  linkprim = c("P", "C"),
  usedflag = 1
)
```

Arguments

`linktype`	A character vector indicating the type of link to download. The default is `c("LU", "LC")`, where "LU" stands for "Link Up" and "LC" for "Link CRSP".
`linkprim`	A character vector indicating the primacy of the link. Default is `c("P", "C")`, where "P" indicates primary and "C" indicates conditional links.
`usedflag`	An integer indicating whether to use flagged links. The default is `1`, indicating that only flagged links should be used.

Value

A data frame with the columns `permno`, `gvkey`, `linkdt`, and `linkenddt`, where `linkenddt` is the end date of the link, and missing end dates are replaced with today's date.

Examples

```
ccm_links <- download_data_wrds_ccm_links(linktype = "LU", linkprim = "P", usedflag = 1)
```

download_data_wrds_clean_trace

Download Cleaned TRACE Data from WRDS

Description

Establishes a connection to the WRDS database to download the specified CUSIPs trade messages from the Trade Reporting and Compliance Engine (TRACE). The trade data is cleaned as suggested by Dick-Nielsen (2009, 2014).

Usage

```
download_data_wrds_clean_trace(cusips, start_date, end_date)
```

Arguments

cusips	A character vector specifying the 9-digit CUSIPs to download.
start_date	The start date for filtering the data, in "YYYY-MM-DD" format.
end_date	The end date for filtering the data, in "YYYY-MM-DD" format.

Value

A data frame containing the cleaned trade messages from TRACE for the selected CUSIPs over the time window specified. Output variables include identifying information (i.e., CUSIP, trade date/time) and trade-specific information (i.e., price/yield, volume, counterparty, and reporting side).

Examples

```
one_bond <- download_data_wrds_clean_trace("00101JAH9", "2019-01-01", "2021-12-31")
```

download_data_wrds_compustat

Download Data from WRDS Compustat

Description

This function downloads financial data from the WRDS Compustat database for a given type of financial data, start date, and end date. It filters the data according to industry format, data format, and consolidation level, and calculates book equity (be), operating profitability (op), and investment (inv) for each company.

Usage

```
download_data_wrds_compustat(
  type,
  start_date,
  end_date,
  additional_columns = NULL
)
```

Arguments

type	The type of financial data to download.
start_date	The start date for the data retrieval in "YYYY-MM-DD" format.
end_date	The end date for the data retrieval in "YYYY-MM-DD" format.
additional_columns	Additional columns from the Compustat table as a character vector.

Value

A data frame with financial data for the specified period, including variables for book equity (be), operating profitability (op), investment (inv), and others.

Examples

```
compustat <- download_data_wrds_compustat("wrds_compustat_annual", "2020-01-01", "2020-12-31")

# Add additional columns
download_data_wrds_compustat("wrds_compustat_annual", "2020-01-01", "2020-12-31",
                             additional_columns = c("aodo", "aldo"))
```

---

download_data_wrds_crsp

Download Data from WRDS CRSP

Description

This function downloads and processes stock return data from the CRSP database for a specified period. Users can choose between monthly and daily data types. The function also adjusts returns for delisting and calculates market capitalization and excess returns over the risk-free rate.

Usage

```
download_data_wrds_crsp(
  type,
  start_date,
  end_date,
  batch_size = 500,
  version = "v2",
  additional_columns = NULL
)
```

Arguments

type	A string specifying the type of CRSP data to download: "crsp_monthly" or "crsp_daily".
start_date	The start date for the data retrieval in "YYYY-MM-DD" format.
end_date	The end date for the data retrieval in "YYYY-MM-DD" format.
batch_size	An optional integer specifying the batch size for processing daily data, with a default of 500.
version	An optional character specifying which CRSP version to use. "v2" (the default) uses the updated second version of CRSP, and "v1" downloads the legacy version of CRSP.
additional_columns	Additional columns from the CRSP monthly or daily data as a character vector.

Value

A data frame containing CRSP stock returns, adjusted for delistings, along with calculated market capitalization and excess returns over the risk-free rate. The structure of the returned data frame depends on the selected data type.

Examples

```
crsp_monthly <- download_data_wrds_crsp("wrds_crsp_monthly", "2020-11-01", "2020-12-31")
crsp_daily <- download_data_wrds_crsp("wrds_crsp_daily", "2020-12-01", "2020-12-31")

# Add additional columns
download_data_wrds_crsp("wrds_crsp_monthly", "2020-11-01", "2020-12-31",
                        additional_columns = c("mthvol", "mthvolflg"))
```

download_data_wrds_fisd

Download Filtered FISD Data from WRDS

Description

Establishes a connection to the WRDS database to download a filtered subset of the FISD (Fixed Income Securities Database). The function filters the `fisd_mergedissue` and `fisd_mergedissuer` tables based on several criteria related to the securities, such as security level, bond type, coupon type, and others, focusing on specific attributes that denote the nature of the securities. It finally returns a data frame with selected fields from the `fisd_mergedissue` table after joining it with issuer information from the `fisd_mergedissuer` table for issuers domiciled in the USA.

Usage

```
download_data_wrds_fisd()
```

Value

A data frame containing a subset of FISD data with fields related to the bond's characteristics and issuer information. This includes complete CUSIP, maturity, offering amount, offering date, dated date, interest frequency, coupon, last interest date, issue ID, issuer ID, SIC code of the issuer.

Examples

```
fisd <- download_data_wrds_fisd()
```

estimate_model

Estimate Model Coefficients

Description

[Experimental]

This function estimates the coefficients of a linear model specified by one or more independent variables. It checks for the presence of the specified independent variables in the dataset and whether the dataset has a sufficient number of observations. It returns the model's coefficients as either a numeric value (for a single independent variable) or a data frame (for multiple independent variables).

Usage

```
estimate_model(data, model, min_obs = 1)
```

Arguments

data	A data frame containing the dependent variable and one or more independent variables.
model	A character that describes the model to estimate (e.g. "ret_excess ~ mkt_excess + hmb + sml").
min_obs	The minimum number of observations required to estimate the model. Defaults to 1.

Value

If a single independent variable is specified, a numeric value representing the coefficient of that variable. If multiple independent variables are specified, a data frame with a row for each coefficient and column names corresponding to the independent variables.

See Also

lm for details on the underlying linear model fitting used.

Examples

```
data <- data.frame(
  ret_excess = rnorm(100),
  mkt_excess = rnorm(100),
  smb = rnorm(100),
  hml = rnorm(100)
)
# Estimate model with a single independent variable
single_var_model <- estimate_model(data, "ret_excess ~ mkt_excess")

# Estimate model with multiple independent variables
multi_var_model <- estimate_model(data, "ret_excess ~ mkt_excess + smb + hml")
```

---

get_wrds_connection *Establish a Connection to the WRDS Database*

Description

This function establishes a connection to the Wharton Research Data Services (WRDS) database using the RPostgres package. It requires that the RPostgres package is installed and that valid WRDS credentials are set as environment variables.

Usage

```
get_wrds_connection()
```

Details

The function checks if the RPostgres package is installed before attempting to establish a connection. It uses the host, dbname, port, and sslmode as fixed parameters for the connection. Users must set their WRDS username and password as environment variables WRDS_USER and WRDS_PASSWORD, respectively, before using this function.

Value

An object of class DBIConnection representing the connection to the WRDS database. This object can be used with other DBI-compliant functions to interact with the database.

See Also

Postgres, dbDisconnect for more information on managing database connections.

Examples

```
# Before using this function, set your WRDS credentials:
# Sys.setenv(WRDS_USER = "your_username", WRDS_PASSWORD = "your_password")

con <- get_wrds_connection()
# Use `con` with DBI-compliant functions to interact with the WRDS database
# Remember to disconnect after use:
# disconnect_connection(con)
```

---

list_supported_types *List All Supported Dataset Types*

Description

This function aggregates and returns a comprehensive tibble of all supported dataset types from different domains. It includes various datasets across different frequencies (daily, weekly, monthly, quarterly, annual) and models (e.g., q5 factors, Fama-French 3 and 5 factors, macro predictors).

Usage

```
list_supported_types(domain = NULL, as_vector = FALSE)
```

Arguments

domain	A character vector to filter for domain specific types (e.g. c("WRDS", "Fama-French"))
as_vector	Logical indicating whether types should be returned as a character vector instead of data frame.

Value

A tibble aggregating all supported dataset types with columns: `type` (the type of dataset), `dataset_name` (a descriptive name or file name of the dataset), and `domain` (the domain to which the dataset belongs, e.g., "Global Q", "Fama-French", "Goyal-Welch").

Examples

```
# List all supported types as a data frame
list_supported_types()

# Filter by domain
list_supported_types(domain = "WRDS")

# List supported types as a vector
list_supported_types(as_vector = TRUE)
```

---

list_supported_types_ff

List Supported Fama-French Dataset Types

Description

This function returns a tibble with the supported Fama-French dataset types, including their names and frequencies (daily, weekly, monthly). Each dataset type is associated with a specific Fama-French model (e.g., 3 factors, 5 factors). Additionally, it annotates each dataset with the domain "Fama-French".

Usage

```
list_supported_types_ff()
```

Value

A tibble with columns: `type` (the type of dataset), `dataset_name` (a descriptive name of the dataset), and `domain` (the domain to which the dataset belongs, always "Fama-French").

list_supported_types_macro_predictors

List Supported Macro Predictor Dataset Types

Description

This function returns a tibble with the supported macro predictor dataset types provided by Goyal-Welch, including their frequencies (monthly, quarterly, annual). All dataset types reference the same source file "PredictorData2022.xlsx" for the year 2022. Additionally, it annotates each dataset with the domain "Goyal-Welch".

Usage

```
list_supported_types_macro_predictors()
```

Value

A tibble with columns: `type` (the type of dataset), `dataset_name` (the file name of the dataset, which is the same for all types), and `domain` (the domain to which the dataset belongs, always "Goyal-Welch").

list_supported_types_q

List Supported Global Q Dataset Types

Description

This function returns a tibble with the supported Global Q dataset types, including their names and frequencies (daily, weekly, weekly week-to-week, monthly, quarterly, annual). Each dataset type is associated with the Global Q model, specifically the q5 factors model for the year 2022. Additionally, it annotates each dataset with the domain "Global Q".

Usage

```
list_supported_types_q()
```

Value

A tibble with columns: `type` (the type of dataset), `dataset_name` (the file name of the dataset), and `domain` (the domain to which the dataset belongs, always "Global Q").

list_supported_types_wrds
List Supported WRDS Dataset Types

Description

This function returns a tibble with the supported dataset types provided via WRDS. Additionally, it annotates each dataset with the domain "WRDS".

Usage

```
list_supported_types_wrds()
```

Value

A tibble with columns: `type` (the type of dataset), `dataset_name` (the file name of the dataset), and `domain` (the domain to which the dataset belongs, always "WRDS").

list_tidy_finance_chapters
List Chapters of Tidy Finance

Description

Returns a character vector containing the names of the chapters available in the Tidy Finance resource. This function provides a quick reference to the various topics covered.

Usage

```
list_tidy_finance_chapters()
```

Value

A character vector where each element is the name of a chapter available in the Tidy Finance resource. These names correspond to specific chapters in Tidy Finance with R.

Examples

```
list_tidy_finance_chapters()
```

open_tidy_finance_website *Open Tidy Finance Website or Specific Chapter in Browser*

Description

Opens the main Tidy Finance website or a specific chapter within the site in the user's default web browser. If a chapter is specified, the function constructs the URL to access the chapter directly.

Usage

```
open_tidy_finance_website(chapter = NULL)
```

Arguments

chapter	An optional character string specifying the chapter to open. If NULL (the default), the function opens the main page of Tidy Finance with R. If a chapter name is provided (e.g., "beta-estimation"), the function opens the corresponding chapter's page (e.g., "beta-estimation.html"). If the chapter name does not exist, then the function opens the main page.

Value

Invisible NULL. The function is called for its side effect of opening a web page.

Examples

```
open_tidy_finance_website()
open_tidy_finance_website("beta-estimation")
```

set_wrds_credentials *Set WRDS Credentials*

Description

This function prompts the user to input their WRDS (Wharton Research Data Services) username and password, and stores these credentials in a .Renviron file. The user can choose to store the .Renviron file in either the project directory or the home directory. If the .Renviron file already contains WRDS credentials, the user will be asked if they want to overwrite the existing credentials. Additionally, the user has the option to add the .Renviron file to the .gitignore file to prevent it from being tracked by version control.

Usage

```
set_wrds_credentials()
```

Value

Invisibly returns TRUE. Displays messages to the user based on their input and actions taken.

Examples

```
## Not run:
set_wrds_credentials()

## End(Not run)
```

`trim` *Trim a Numeric Vector*

Description

Removes the values in a numeric vector that are beyond the specified quantiles, effectively trimming the distribution based on the `cut` parameter. This process reduces the length of the vector, excluding extreme values from both tails of the distribution.

Usage

```
trim(x, cut)
```

Arguments

`x`	A numeric vector to be trimmed.
`cut`	The proportion of data to be trimmed from both ends of the distribution. For example, a `cut` of 0.05 will remove the lowest and highest 5% of the data. Must be between [0, 0.5].

Value

A numeric vector with the extreme values removed.

Examples

```
set.seed(123)
data <- rnorm(100)
trimmed_data <- trim(x = data, cut = 0.05)
```

`winsorize` *Winsorize a Numeric Vector*

Description

Replaces the values in a numeric vector that are beyond the specified quantiles with the boundary values of those quantiles. This is done for both tails of the distribution based on the `cut` parameter.

Usage

```
winsorize(x, cut)
```

Arguments

`x`	A numeric vector to be winsorized.
`cut`	The proportion of data to be winsorized from both ends of the distribution. For example, a `cut` of 0.05 will winsorize the lowest and highest 5% of the data. Must be inside [0, 0.5].

Value

A numeric vector with the extreme values replaced by the corresponding quantile values.

Examples

```
set.seed(123)
data <- rnorm(100)
winsorized_data <- winsorize(data, 0.05)
```

Index