Package 'tidyfinance'

October 26, 2026

Type Package

Title Tidy Finance Helper Functions

Version 1.0.0

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/index.html>.

License MIT + file LICENSE

URL https://r.tidy-finance.org/

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

Depends R (>= 4.1)

Imports cli, dplyr (>= 1.1.4), dbplyr (>= 2.5.0), jsonlite, lifecycle, lubridate (>= 1.9.3), nanoparquet (>= 0.5.1), purrr (>= 1.0.2), rlang (>= 1.1.3), slider (>= 0.3.1), stats, tibble, tidyr (>= 1.3.1), DBI (>= 1.2.2), httr2 (>= 1.1.1), RPostgres (>= 1.4.5), sandwich (>= 3.1-0)

Suggests knitr, readxl, rmarkdown, testthat (>= 3.2.0), withr

Config/testthat/edition 3

Encoding UTF-8

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Christoph Scheuch [aut, cre, cph] (ORCID: <https://orcid.org/0009-0004-0423-6819>), Stefan Voigt [aut, cph] (ORCID: <https://orcid.org/0000-0001-5619-3161>), Patrick Weiss [aut, cph] (ORCID: <https://orcid.org/0000-0002-9282-5872>), Maximilian Mücke [ctb] (ORCID: <https://orcid.org/0009-0000-9432-9795>)

Maintainer Christoph Scheuch <christoph@tidy-intelligence.com>

Repository CRAN

Date/Publication 2026-10-26 15:10:02 UTC

Contents

add_lagged_columns . . . 3
assign_portfolio . . . 5
breakpoint_options . . . 7
compute_breakpoints . . . 8
compute_long_short_returns . . . 10
compute_portfolio_returns . . . 11
compute_rolling_value . . . 15
create_summary_statistics . . . 18
data_options . . . 19
disconnect_connection . . . 21
download_data . . . 22
download_data_constituents . . . 24
download_data_damodaran . . . 25
download_data_factors_ff . . . 26
download_data_factors_q . . . 28
download_data_fred . . . 29
download_data_fred_md . . . 30
download_data_huggingface . . . 32
download_data_jkp . . . 35
download_data_macro_predictors . . . 37
download_data_osap . . . 39
download_data_pastor_stambaugh . . . 40
download_data_pseudo_ccm_links . . . 41
download_data_pseudo_compustat . . . 42
download_data_pseudo_crsp . . . 44
download_data_risk_free . . . 45
download_data_stambaugh_yuan . . . 47
download_data_stock_prices . . . 48
download_data_wrds . . . 49
download_data_wrds_ccm_links . . . 51
download_data_wrds_compustat . . . 52
download_data_wrds_crsp . . . 53
download_data_wrds_fisd . . . 56
download_data_wrds_trace_enhanced . . . 57
download_factor_library_grid . . . 58
download_factor_library_ids . . . 59
estimate_betas . . . 60
estimate_fama_macbeth . . . 62
estimate_model . . . 64
filter_options . . . 66
filter_sorting_data . . . 68

get_available_huggingface_files 70
get_wrds_connection 71
implement_portfolio_sort 73
join_lagged_values 75
list_supported_datasets 77
list_supported_datasets_ff 78
list_supported_datasets_ff_legacy 79
list_supported_datasets_macro_predictors 80
list_supported_datasets_other 80
list_supported_datasets_pseudo 81
list_supported_datasets_q 81
list_supported_datasets_wrds 82
list_supported_indexes 82
list_supported_jkp_factors 83
list_tidy_finance_chapters 84
open_tidy_finance_website 85
portfolio_sort_options 86
set_wrds_credentials 88
trim 89
validate_dates 90
winsorize 90

Index **92**

`add_lagged_columns` *Add Lagged Columns via Join*

Description

Appends lagged versions of specified columns to a data frame using a join-based approach.

Usage

```
add_lagged_columns(
  data,
  cols,
  lag = NULL,
  max_lag = lag,
  by = NULL,
  drop_na = FALSE,
  ff_adjustment = FALSE,
  data_options = NULL
)
```

Arguments

data	A data frame containing the variables to lag.
cols	A character vector specifying the names of the columns to be lagged. Each column produces a new column suffixed with _lag, which must not already exist in data.
lag	An integer or a lubridate::periods() object, e.g., months(1), specifying the minimum lag (inclusive) to apply. Required unless ff_adjustment = TRUE.
max_lag	An integer or a lubridate::periods() object specifying the maximum lag (inclusive) to apply. Defaults to lag (exact lag).
by	An optional character vector specifying grouping columns (e.g., a stock identifier). Lagged values are matched within groups. Defaults to NULL.
drop_na	A logical value. If TRUE, NA values in the source columns are excluded before matching, so the lookup skips over missing observations. Applied independently per column. Defaults to FALSE.
ff_adjustment	A logical value. If TRUE, applies the Fama and French (1993) timing convention for annual accounting data: the last observation per calendar year (within each group defined by by) is used from July of the following year through June of the year after that, whatever the month of the fiscal year-end. lag and max_lag are ignored. Combine with drop_na = TRUE when data is a monthly panel in which the accounting columns are only filled at fiscal year-ends. Defaults to FALSE.
data_options	A list of class tidyfinance_data_options (created via data_options()) specifying column name mappings. The date element is used to specify the date column. Uses data_options() default if NULL: "date" = "date".

Details

When lag == max_lag (the default), an equi-join is used: source dates are shifted forward by lag and matched exactly. When lag < max_lag, an inequality join is used: for each row, the most recent source value within the window [date - max_lag, date - lag] is selected.

The combination of by and date columns must be unique in data. If by is NULL, dates alone must be unique.

For lags expressed in whole months or years (for example months(1) or years(1)), the dates should be the first of the month, as returned by the download_data() functions. On other days of the month, month arithmetic such as date - months(1) can return NA (e.g., 31 January minus one month) or a value from the wrong month; floor the dates with lubridate::floor_date(date, "month") beforehand. This does not apply to ff_adjustment = TRUE, which works on calendar years.

Value

A data frame with the same rows as data and new columns appended, each suffixed with _lag. Unmatched rows receive NA in the lagged columns.

See Also

Other rolling and lagging functions: compute_rolling_value(), join_lagged_values()

Examples

```
set.seed(42)
data <- tibble::tibble(
  permno = rep(1:2, each = 10),
  date = rep(
    seq.Date(as.Date("2023-01-01"), by = "month", length.out = 10),
    2
  ),
  size = runif(20, 100, 200),
  bm = runif(20, 0.5, 1.5)
)

# Exact lag: each row gets the value from exactly 2 months earlier
add_lagged_columns(
  data,
  cols = c("size", "bm"),
  lag = months(2),
  by = "permno"
)

# Window lag: each row gets the most recent value from 2 to 4 months earlier
add_lagged_columns(
  data,
  cols = "size",
  lag = months(2),
  max_lag = months(4),
  by = "permno"
)
```

`assign_portfolio` *Assign Portfolios Based on Sorting Variable*

Description

Assigns data points to portfolios based on a specified sorting variable and the selected function to compute breakpoints. Users can specify a function to compute breakpoints. The function must take `data` and `sorting_variable` as the first two arguments. Additional arguments are passed with a named list `breakpoint_options()`. The function needs to return an ascending vector of breakpoints. By default, breakpoints are computed with `compute_breakpoints()`. The default column names can be modified using `data_options()`.

Usage

```
assign_portfolio(
  data,
  sorting_variable,
  breakpoint_options = NULL,
  breakpoint_function = compute_breakpoints,
```

```
  data_options = 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 based on the breakpoints.
breakpoint_options	An optional named list of arguments passed to breakpoint_function.
breakpoint_function	A function to compute breakpoints. The default is set to compute_breakpoints().
data_options	A list of class tidyfinance_data_options (created via data_options()) specifying column name mappings. Passed through to breakpoint_function. When using the default compute_breakpoints(), the exchange element is used to specify the exchange column, and mktcap_lag is used to specify the market capitalization column. Uses data_options() default if NULL: "exchange" = "exchange" and "mktcap_lag" = "mktcap_lag".

Value

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

See Also

Other portfolio functions: breakpoint_options(), compute_breakpoints(), compute_long_short_returns(), compute_portfolio_returns(), data_options(), filter_options(), filter_sorting_data(), implement_portfolio_sort(), portfolio_sort_options()

Examples

```
set.seed(42)
data <- data.frame(
  id = 1:100,
  exchange = sample(c("NYSE", "NASDAQ"), 100, replace = TRUE),
  market_cap = 1:100
)

assign_portfolio(data, "market_cap", breakpoint_options(n_portfolios = 5))

assign_portfolio(
  data,
  "market_cap",
  breakpoint_options(
    percentiles = c(0.2, 0.4, 0.6, 0.8),
    breakpoints_exchanges = c("NYSE")
  )
)
```

`breakpoint_options` *Create Breakpoint Options for Portfolio Sorting*

Description

Generates a structured list of options for defining breakpoints in portfolio sorting. It includes parameters for the number of portfolios, percentile thresholds, exchange-specific breakpoints, and smooth bunching, along with additional optional parameters.

Usage

```
breakpoint_options(
  n_portfolios = NULL,
  percentiles = NULL,
  breakpoints_exchanges = NULL,
  smooth_bunching = FALSE,
  breakpoints_min_size_threshold = NULL,
  ...
)
```

Arguments

`n_portfolios`	Integer, optional. The number of portfolios to create. Must be a positive integer. If not provided, defaults to `NULL`.
`percentiles`	Numeric vector, optional. A vector of percentile thresholds for defining breakpoints. Each value must be between 0 and 1. If not provided, defaults to `NULL`.
`breakpoints_exchanges`	Character vector, optional. A non-empty vector specifying the exchange from which to compute the breakpoints. If not provided, defaults to `NULL`.
`smooth_bunching`	Logical, optional. Indicates whether smooth bunching should be applied. Defaults to `FALSE`.
`breakpoints_min_size_threshold`	Numeric, optional. When set to a value between 0 and 1, stocks with market capitalization below this quantile are excluded from breakpoint computation. The quantile is computed among `breakpoints_exchanges` stocks if specified, otherwise among all stocks. Requires a market capitalization column in the data (see `data_options()`). Defaults to `NULL` (no size filtering). In dependent bivariate sorts, the quantile for the main sorting variable is computed once per date from the full cross-section, not within each secondary portfolio.
`...`	Additional optional arguments. These will be captured in the resulting structure as a list.

Value

A list of class `"tidyfinance_breakpoint_options"` containing the provided breakpoint options, including any additional arguments passed via `...`.

See Also

Other portfolio functions: assign_portfolio(), compute_breakpoints(), compute_long_short_returns(), compute_portfolio_returns(), data_options(), filter_options(), filter_sorting_data(), implement_portfolio_sort(), portfolio_sort_options()

Examples

```
breakpoint_options(
  n_portfolios = 5,
  percentiles = c(0.2, 0.4, 0.6, 0.8),
  breakpoints_exchanges = "NYSE",
  smooth_bunching = TRUE,
  custom_threshold = 0.5,
  another_option = "example"
)
```

---

compute_breakpoints *Compute Breakpoints Based on Sorting Variable*

Description

Computes breakpoints based on a specified sorting. It can optionally filter the data by exchanges or lagged size quantiles before computing the breakpoints. The function requires either the number of portfolios to be created or specific percentiles for the breakpoints, but not both. The function also optionally handles cases where the sorting variable clusters on the edges, by assigning all extreme values to the edges and attempting to compute equally populated breakpoints with the remaining values.

Usage

```
compute_breakpoints(
  data,
  sorting_variable,
  breakpoint_options,
  data_options = NULL
)
```

Arguments

data	A data frame containing the dataset for breakpoint computation.
sorting_variable	A character string specifying the column name in data to be used for determining breakpoints.
breakpoint_options	A named list of breakpoint_options() for the breakpoints. The arguments include

- `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`.
- `breakpoints_exchanges` An optional character vector specifying exchange names to filter the data before computing breakpoints. Exchanges must be stored in a column given by `data_options` (defaults to `exchange`). If `NULL`, no filtering is applied.
- `smooth_bunching` An optional logical parameter specifying if to attempt smoothing non-extreme portfolios if the sorting variable bunches on the extremes (`TRUE`), or not (`FALSE`, the default). In some cases, smoothing will not result in equal-sized portfolios off the edges due to multiple clusters. If sufficiently large bunching is detected, `percentiles` is ignored and equally-spaced portfolios are returned for these cases with a warning.
- `breakpoints_min_size_threshold` An optional numeric value between 0 and 1 (exclusive). When set, stocks with market capitalization below this quantile are excluded from breakpoint computation. The quantile is computed among `breakpoints_exchanges` stocks if specified, otherwise among all stocks. Requires a market capitalization column in the data (column name determined by `data_options`).

`data_options` A list of class `tidyfinance_data_options` (created via `data_options()`) specifying column name mappings. The `exchange` element is used to specify the exchange column, and `mktcap_lag` is used to specify the market capitalization. Uses `data_options()` default if `NULL`: `"exchange" = "exchange"` and `"mktcap_lag" = "mktcap_lag"`.

Details

Breakpoints are computed from the cross-section passed in `data`; in a sort, pass the cross-section available at formation. When the resulting breakpoints are passed to `assign_portfolio()`, a stock whose sorting value equals an interior breakpoint joins the portfolio *above* that breakpoint (the intervals are left-closed, right-open), and values at or beyond the extreme breakpoints fall in the first and last portfolios.

Value

A numeric vector of breakpoints of the desired length.

Note

This function will stop and throw an error if both `n_portfolios` and `percentiles` are provided or missing simultaneously.

See Also

Other portfolio functions: `assign_portfolio()`, `breakpoint_options()`, `compute_long_short_returns()`, `compute_portfolio_returns()`, `data_options()`, `filter_options()`, `filter_sorting_data()`, `implement_portfolio_sort()`, `portfolio_sort_options()`

Examples

```
set.seed(42)
data <- data.frame(
  id = 1:100,
  exchange = sample(c("NYSE", "NASDAQ"), 100, replace = TRUE),
  market_cap = 1:100
)

compute_breakpoints(data, "market_cap", breakpoint_options(n_portfolios = 5))
compute_breakpoints(
  data,
  "market_cap",
  breakpoint_options(
    percentiles = c(0.2, 0.4, 0.6, 0.8),
    breakpoints_exchanges = c("NYSE")
  )
 )
```

`compute_long_short_returns` *Compute Long-Short Returns*

Description

Calculates long-short returns based on the returns of portfolios. The long-short return is computed as the difference between the returns of the "top" and "bottom" portfolios. The direction of the calculation can be adjusted based on whether the return from the "bottom" portfolio is subtracted from or added to the return from the "top" portfolio.

Usage

```
compute_long_short_returns(
  data,
  direction = "top_minus_bottom",
  data_options = NULL
)
```

Arguments

`data`	A data frame of portfolio returns as returned by `compute_portfolio_returns()`. It must include the portfolio identifier and date columns (named via `data_options`) and at least one of the return columns `ret_excess_vw`, `ret_excess_ew`, and `ret_excess_vw_capped`.
`direction`	A character string specifying the direction of the long-short return calculation. It can be either `"top_minus_bottom"` or `"bottom_minus_top"`. Default is `"top_minus_bottom"`. If set to `"bottom_minus_top"`, the return will be computed as (bottom - top).

data_options A list of class tidyfinance_data_options (created via data_options()) specifying column name mappings. The date element specifies the date column and portfolio the assigned portfolio. Uses data_options() default if NULL: "date" = "date" and "portfolio" = "portfolio".

Value

A data frame with the date column and one column per portfolio return measure present in data (ret_excess_vw, ret_excess_ew, and ret_excess_vw_capped), each holding the long-short return for that measure, arranged by date.

See Also

Other portfolio functions: assign_portfolio(), breakpoint_options(), compute_breakpoints(), compute_portfolio_returns(), data_options(), filter_options(), filter_sorting_data(), implement_portfolio_sort(), portfolio_sort_options()

Examples

```
set.seed(42)
data <- data.frame(
  permno = 1:100,
  date = rep(
    seq.Date(from = as.Date("2020-01-01"), by = "month", length.out = 100),
    each = 10
  ),
  mktcap_lag = runif(100, 100, 1000),
  ret_excess = rnorm(100),
  size = runif(100, 50, 150)
)

portfolio_returns <- compute_portfolio_returns(
  data, "size", "univariate",
  breakpoint_options_main = breakpoint_options(n_portfolios = 5)
)

compute_long_short_returns(portfolio_returns)
```

---

compute_portfolio_returns

Compute Portfolio Returns

Description

Computes individual portfolio returns based on specified sorting variables and sorting methods. The portfolios can be rebalanced every period or on an annual frequency by specifying a rebalancing month, which is only applicable at a monthly return frequency. The function supports univariate and bivariate sorts, with the latter supporting dependent and independent sorting methods.

Usage

```
compute_portfolio_returns(
  data,
  sorting_variables,
  sorting_method,
  rebalancing_month = NULL,
  holding_period = 1L,
  skip = 0L,
  breakpoint_options_main,
  breakpoint_options_secondary = NULL,
  breakpoint_function_main = compute_breakpoints,
  breakpoint_function_secondary = compute_breakpoints,
  min_portfolio_size = 1L,
  cap_weight = 0.8,
  cap_weight_exchanges = "NYSE",
  data_options = NULL,
  quiet = FALSE
)
```

Arguments

`data` A data frame containing the dataset for portfolio assignment and return computation. The panel data must include individual stock identifiers and the time point. It must contain columns for the sorting variables and excess returns. Additionally, lagged market capitalization is required for value-weighted returns (see parameter `data_options`).

`sorting_variables` A character vector specifying the column names in `data` to be used for sorting and determining portfolio assignments. For univariate sorts, provide a single variable. For bivariate sorts, provide two variables, where the first string refers to the main variable and the second string refers to the secondary ("control") variable.

`sorting_method` A string specifying the sorting method to be used. Possible values are:

- `"univariate"`: For a single sorting variable.
- `"bivariate-dependent"`: For two sorting variables, where the main sort depends on the secondary variable.
- `"bivariate-independent"`: For two independent sorting variables.

For bivariate sorts, the portfolio returns are averaged over the controlling sorting variable (i.e., the second sorting variable), and only portfolio returns for the main sorting variable (given as the first element of `sorting_variables`) are returned.

`rebalancing_month` An integer between 1 and 12 specifying the month in which to form portfolios that are held constant for one year. For example, setting it to 7 creates portfolios in July that are held constant until June of the following year. The default `NULL` corresponds to periodic rebalancing.

`holding_period` A positive integer specifying for how many periods portfolios are held. With the default `1L`, portfolios are rebalanced every period. With `holding_period =`

K > 1, portfolios are formed every period and each cohort is held for K periods, so that K overlapping cohorts are held at any time (staggered rebalancing, as in Jegadeesh and Titman, 1993). The return of a portfolio in a period is the equal-weighted average of the returns of the cohorts held in that period; within a cohort, returns are weighted as in the ret_excess_* columns, using the weights of the holding period. A stock stays in its cohort for the whole holding period even if its sorting variables are missing in a later period. Cannot be combined with rebalancing_month.

skip
: A non-negative integer specifying the number of periods between portfolio formation and the first holding period. The default 0L holds a cohort from the period in which it is formed, consistent with sorting variables that are already lagged (i.e., known at the start of the period). Jegadeesh and Titman (1993), e.g., also report strategies that skip one period. Cannot be combined with rebalancing_month.

breakpoint_options_main
: A named list of breakpoint_options() passed to breakpoint_function_main for the main sorting variable.

breakpoint_options_secondary
: An optional named list of breakpoint_options() passed to breakpoint_function_secondary.

breakpoint_function_main
: A function to compute the breakpoints for the main sorting variable. The default is set to compute_breakpoints().

breakpoint_function_secondary
: A function to compute the breakpoints for the secondary sorting variable. The default is set to compute_breakpoints().

min_portfolio_size
: An integer specifying the minimum number of firms required in the reported portfolio cross-section on a given date (default 1L, i.e. at least one observation per reported portfolio). For both univariate and bivariate sorts the threshold is applied to the firm count per (portfolio, date) of the reported cross-section: for univariate sorts that is firms per portfolio-date; for bivariate sorts that is firms per main-portfolio-date summed across the secondary buckets. Cross-sections below the threshold have their returns set to NA. A typical value is 5L (the Fama-French convention). Set to 0L to deactivate the check entirely.

cap_weight
: A numeric value between 0 and 1 specifying the percentile at which market capitalization is capped per date when computing capped value-weighted excess returns (i.e., ret_excess_vw_capped). Defaults to 0.8.

cap_weight_exchanges
: A character vector of exchanges whose stocks determine the cap: market capitalization is capped at the cap_weight percentile of the stocks listed on these exchanges on each date. Defaults to "NYSE", following Jensen, Kelly, and Pedersen (2023), who cap at the 80th percentile of NYSE stocks. NULL uses all stocks. If the exchange column (mapped via data_options()) is missing, or no stock is listed on these exchanges on a date, the percentile of all stocks is used and a message says so.

data_options
: A list of class tidyfinance_data_options (created via data_options()) specifying column name mappings. The id element is used to specify the entity (i.e., firm), date is used to specify the date column, ret_excess is used to specify

the excess return column, and mktcap_lag is used to specify the market capitalization. Uses data_options() default if NULL: "id" = "permno", "date" = "date", "ret_excess" = "ret_excess", and "mktcap_lag" = "mktcap_lag". The exchange element (default "exchange") is used with cap_weight_exchanges.

quiet — A logical value indicating whether to suppress informational messages about missing values in the output panel (default is FALSE).

Details

Holding periods and skips are counted on the grid of dates in data, which must therefore not have gaps (e.g., one row per stock-month in a monthly panel). With staggered rebalancing (holding_period > 1 or skip > 0), the returned panel starts on the first date on which all holding_period cohorts are held.

A portfolio's return on a date is NA when the portfolio holds fewer than min_portfolio_size stocks or when a constituent return is missing, because the equal- and value-weighted averages are taken over all assigned stocks without dropping NAs.

The function checks for consistency in the provided arguments. For univariate sorts, a single sorting variable and a corresponding number of portfolios must be provided. For bivariate sorts, two sorting variables and two corresponding numbers of portfolios (or percentiles) are required. The sorting method determines how portfolios are assigned and how returns are computed. The function handles missing and extreme values appropriately based on the specified sorting method and rebalancing frequency.

Value

A data frame with computed portfolio returns as a complete panel (all portfolio-date combinations), containing the following columns:

- portfolio: The portfolio identifier.
- date: The date of the portfolio return.
- ret_excess_vw: The value-weighted excess return of the portfolio (only computed if data contains a lagged market capitalization column specified by data_options(), which defaults to mktcap_lag). NA if insufficient observations for that portfolio-date.
- ret_excess_ew: The equal-weighted excess return of the portfolio. NA if insufficient observations for that portfolio-date.
- ret_excess_vw_capped: The capped value-weighted excess return of the portfolio (only computed if data contains a lagged market capitalization column specified by data_options(), which defaults to mktcap_lag). Weights are computed using market capitalization capped at the cap_weight percentile per date. NA if insufficient observations for that portfolio-date.

Note

Ensure that data contains all required columns: the specified sorting variables and excess returns (see options and defaults set in data_options). The function will stop and throw an error if any required columns are missing.

References

Jegadeesh, N., & Titman, S. (1993). Returns to buying winners and selling losers: Implications for stock market efficiency. *Journal of Finance*, 48(1), 65-91. doi:10.1111/j.15406261.1993.tb04702.x

Jensen, T. I., Kelly, B., & Pedersen, L. H. (2023). Is there a replication crisis in finance? *Journal of Finance*, 78(5), 2465-2518. doi:10.1111/jofi.13249

See Also

Other portfolio functions: `assign_portfolio()`, `breakpoint_options()`, `compute_breakpoints()`, `compute_long_short_returns()`, `data_options()`, `filter_options()`, `filter_sorting_data()`, `implement_portfolio_sort()`, `portfolio_sort_options()`

Examples

```
set.seed(42)
# Univariate sorting with periodic rebalancing
data <- data.frame(
  permno = 1:500,
  date = rep(
    seq.Date(
      from = as.Date("2020-01-01"),
      by = "month",
      length.out = 100
    ),
    each = 10
  ),
  mktcap_lag = runif(500, 100, 1000),
  ret_excess = rnorm(500),
  size = runif(500, 50, 150)
)

compute_portfolio_returns(
  data, "size", "univariate",
  breakpoint_options_main = breakpoint_options(n_portfolios = 5)
)

# Bivariate dependent sorting with annual rebalancing
compute_portfolio_returns(
  data, c("size", "mktcap_lag"), "bivariate-dependent", 7,
  breakpoint_options_main = breakpoint_options(n_portfolios = 5),
  breakpoint_options_secondary = breakpoint_options(n_portfolios = 3),
)
```

`compute_rolling_value` *Compute a Rolling Value by Period*

Description

Applies an arbitrary summary function over rolling time-period windows. Each window spans `periods` units of `period` (e.g., 12 months). All rows that fall into the same period receive the value of that period's window. Before calling `.f`, rows with any missing values are dropped from the window; if fewer than `min_obs` rows remain, the result is `NA_real_` instead.

Usage

```
compute_rolling_value(
  data,
  .f,
  period = "month",
  periods = 12,
  min_obs = periods,
  data_options = NULL
)
```

Arguments

`data`	A data frame with a date column of class `Date`, named according to `data_options$date` (default `"date"`).
`.f`	A function applied to each window. Receives a data-frame slice (complete cases only) and must return a single scalar value.
`period`	A string specifying the period for rolling windows (e.g., `"month"`, `"quarter"`, `"year"`).
`periods`	Number of periods to include in the rolling window.
`min_obs`	Minimum number of non-missing rows required per window. Defaults to `periods`.
`data_options`	A list of class `tidyfinance_data_options` (created via `data_options()`) specifying column name mappings. The `date` element is used to specify the date column. Uses `data_options()` default if `NULL`: `"date" = "date"`.

Value

A numeric vector aligned with the rows of `data`.

See Also

Other rolling and lagging functions: `add_lagged_columns()`, `join_lagged_values()`

Examples

```
library(dplyr)

# Rolling standard deviation
set.seed(42)
df <- tibble(
  date = seq.Date(
    from = as.Date("2020-01-01"),
```

```
    by = "month",
    length.out = 24
  ),
  value = rnorm(24)
)

df |>
  mutate(
    rolling_sd = compute_rolling_value(
      pick(everything()),
      .f = ~ sd(.x$value, na.rm = TRUE),
      period = "month",
      periods = 4,
      min_obs = 2
    )
  )

# Rolling last residual from a regression
set.seed(42)
df_reg <- tibble(
  date = seq.Date(
    from = as.Date("2020-01-01"),
    by = "month",
    length.out = 60
  ),
  ret_excess = rnorm(60, 0, 0.05),
  mkt_excess = rnorm(60, 0, 0.04),
  smb = rnorm(60, 0, 0.03),
  hml = rnorm(60, 0, 0.03)
)

df_reg |>
  mutate(
    residual = compute_rolling_value(
      pick(everything()),
      .f = \(x) {
        last(lm(ret_excess ~ mkt_excess + smb + hml, data = x)$residuals)
      },
      period = "month",
      periods = 24,
      min_obs = 12
    )
  )

# Rolling cumulative-return-to-SD ratio
set.seed(42)
df_resid <- tibble(
  date = seq.Date(
    from = as.Date("2020-01-01"),
    by = "month",
    length.out = 24
  ),
  int_roll_residual = rnorm(24, 0, 0.02)
```

```
)

df_resid |>
  mutate(
    return_to_sd = compute_rolling_value(
      pick(everything()),
      .f = ~ (prod(1 + .x$int_roll_residual) - 1) / sd(.x$int_roll_residual),
      period = "month",
      periods = 12,
      min_obs = 12
    )
  )
```

`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. It 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 variable (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 each column contains the computed statistics.

See Also

Other utility functions: `get_available_huggingface_files()`, `list_supported_datasets()`, `list_supported_datasets_ff()`, `list_supported_datasets_ff_legacy()`, `list_supported_datasets_macro_pred`
`list_supported_datasets_other()`, `list_supported_datasets_pseudo()`, `list_supported_datasets_wrds()`, `list_supported_indexes()`, `list_supported_jkp_factors()`, `list_tidy_finance_chapters()`, `open_tidy_finance_website()`, `trim()`, `validate_dates()`, `winsorize()`

Examples

```
data <- data.frame(
  ret = c(0.01, -0.02, 0.03, NA, 0.005),
  size = c(100, 200, 150, 300, 250),
  group = c("A", "A", "B", "B", "A")
)

# Basic summary across all observations
create_summary_statistics(data, ret, size)

# Grouped summary
create_summary_statistics(data, ret, size, by = group)

# Detailed quantiles
create_summary_statistics(data, ret, detail = TRUE)
```

`data_options` *Create Data Options*

Description

Creates a list of data options of class `tidyfinance_data_options` used for TidyFinance-related functions. These options map the specific data variables to the TidyFinance naming conventions, allowing functions to flexibly work with different datasets by specifying the relevant column names. Additional options can be passed through `...`.

Usage

```
data_options(
  id = "permno",
  date = "date",
  exchange = "exchange",
  mktcap_lag = "mktcap_lag",
  ret_excess = "ret_excess",
  portfolio = "portfolio",
  siccd = "siccd",
  price = "prc_lag",
  listing_age = "listing_age",
  be = "be",
  earnings = "ib",
  ...
)
```

Arguments

`id`	A character string representing the entity variable (defaults to `"permno"`).
`date`	A character string representing the date variable (defaults to `"date"`).
`exchange`	A character string representing the exchange variable (defaults to `"exchange"`).
`mktcap_lag`	A character string representing the market capitalization lag variable (defaults to `"mktcap_lag"`).
`ret_excess`	A character string representing the excess return variable (defaults to `"ret_excess"`).
`portfolio`	A character string representing the portfolio variable (defaults to `"portfolio"`).
`siccd`	A character string representing the Standard Industrial Classification code variable (defaults to `"siccd"`).
`price`	A character string representing the price variable used by the `min_stock_price` screen of `filter_sorting_data()` (defaults to `"prc_lag"`, the nominal price at the end of the previous month, as returned by `download_data("WRDS", "crsp_monthly")`).
`listing_age`	A character string representing the listing age variable (defaults to `"listing_age"`).
`be`	A character string representing the book equity variable (defaults to `"be"`).
`earnings`	A character string representing the earnings variable (defaults to `"ib"`, the Compustat income before extraordinary items column).
`...`	Additional arguments to be included in the data options list.

Value

A list of class `tidyfinance_data_options` containing the specified data options.

See Also

Other portfolio functions: assign_portfolio(), breakpoint_options(), compute_breakpoints(), compute_long_short_returns(), compute_portfolio_returns(), filter_options(), filter_sorting_data(), implement_portfolio_sort(), portfolio_sort_options()

Examples

```
data_options(
  id = "permno",
  date = "date",
  exchange = "exchange"
)
```

---

disconnect_connection *Disconnect Database Connection*

Description

Safely disconnects an established database connection using the DBI package.

Usage

```
disconnect_connection(con)
```

Arguments

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

Value

TRUE, invisibly. Throws an error if the disconnection fails.

See Also

Other WRDS functions: download_data_wrds(), download_data_wrds_ccm_links(), download_data_wrds_compustat, download_data_wrds_crsp(), download_data_wrds_fisd(), download_data_wrds_trace_enhanced(), get_wrds_connection(), set_wrds_credentials()

Examples

```
## Not run:
con <- get_wrds_connection()
disconnect_connection(con)

## End(Not run)
```

download_data *Download and Process Data Based on Domain and Dataset*

Description

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

Usage

```
download_data(
  domain = NULL,
  dataset = NULL,
  start_date = NULL,
  end_date = NULL,
  type = deprecated(),
  ...
)
```

Arguments

domain	The domain of the dataset to download, using the names returned by list_supported_datasets(): "Fama-French", "Global Q", "Goyal-Welch", "WRDS", "Pseudo Data", "Index Constituents", "FRED", "Stock Prices", "Open Source Asset Pricing", "Global Factor Data", "Pastor-Stambaugh", "Stambaugh-Yuan", "Damodaran", or "Tidy Finance". Use "Pseudo Data" to obtain pseudo data with the same schema as "WRDS" for testing or rendering without a WRDS subscription. The previous machine-readable names (e.g., "famafrench", "wrds", "pseudo", "tidyfinance") are deprecated but still accepted.
dataset	Optional. The specific dataset to download within the domain.
start_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the data. If not provided, the full dataset or a subset is returned, depending on the dataset type.
end_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the data. If not provided, the full dataset or a subset is returned, depending on the dataset type.
type	**[Deprecated]** Use domain and dataset instead.
...	Additional arguments passed to specific download functions depending on the domain. For instance, if domain is "Index Constituents", arguments are passed to download_data_constituents(). If domain is "Global Factor Data", the dataset argument and arguments such as region, factors, classification, frequency, and weighting are passed to download_data_jkp(). If domain is "Tidy Finance" and dataset is "factor_library" or "factor_library_costs", arguments are either filter inputs (e.g., sorting_variable, rebalancing, fill_all)

or an explicit `ids` vector that bypasses the grid filter and downloads the specified portfolios directly. `revision` and `grid_revision` pin the factor library datasets to a release, and `"factor_library_persistence"` takes an optional `sorting_variable`; see `download_data_huggingface()` for details. If `domain` is `"Damodaran"`, arguments such as `region` and `year` are passed to `download_data_damodaran()`, which selects years via `year` instead of `start_date` and `end_date`. If `domain` is `"WRDS"` or `"Pseudo Data"`, `quiet = TRUE` suppresses the messages about the WRDS connection and the download progress, or the pseudo-data notice.

Value

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

See Also

Other download functions: `download_data_constituents()`, `download_data_damodaran()`, `download_data_factors_f`
`download_data_factors_q()`, `download_data_fred()`, `download_data_fred_md()`, `download_data_huggingface()`,
`download_data_jkp()`, `download_data_macro_predictors()`, `download_data_osap()`, `download_data_pastor_stamb`
`download_data_risk_free()`, `download_data_stambaugh_yuan()`, `download_data_stock_prices()`,
`download_factor_library_grid()`, `download_factor_library_ids()`

Examples

```
download_data(
  "Fama-French",
  "Fama/French 5 Factors (2x3) [Daily]",
  "2000-01-01",
  "2020-12-31"
)
download_data("Goyal-Welch", "monthly", "2000-01-01", "2020-12-31")
download_data("Index Constituents", index = "DAX")
download_data("FRED", series = c("GDP", "CPIAUCNS"))
download_data("FRED", "FRED-MD")
download_data("FRED", "FRED-MD", transform = TRUE)
download_data("FRED", "FRED-QD", vintage = "2020-03")
download_data("Stock Prices", symbols = c("AAPL", "MSFT"))
download_data("Damodaran", "wacc", region = "europe", year = 2020:2024)
download_data(
  "Tidy Finance",
  "risk_free",
  "2020-01-01",
  "2020-12-31"
)
download_data(
  "Tidy Finance",
  "high_frequency_sp500",
  "2007-07-26",
  "2007-07-27"
)
download_data(
```

```
  "Tidy Finance",
  "factor_library",
  sorting_variable = "high52",
  rebalancing = "annual"
)
download_data("Tidy Finance", "factor_library", ids = c(1L, 2L, 3L))
download_data(
  "Tidy Finance",
  "factor_library_costs",
  sorting_variable = "bm"
)
download_data("Tidy Finance", "factor_library_grid")
download_data(
  "Tidy Finance",
  "factor_library_persistence",
  sorting_variable = "bm"
)
```

`download_data_constituents` *Download Constituent Data*

Description

Downloads and processes the constituent data for a specified financial index. The data is fetched from a remote CSV file, filtered, and cleaned to provide relevant information about constituents.

Usage

```
download_data_constituents(index = NULL, path = NULL)
```

Arguments

`index`	A character string specifying the name of the financial index for which to download constituent data. The index must be one of the supported indexes listed by `list_supported_indexes()`. Optional when `path` is given.
`path`	Optional. Path to a local iShares or BlackRock holdings CSV, as saved from the fund's web page, to read instead of downloading. Use it as a fallback when the download fails. `index` is optional in this case; when given, entries whose name contains the index name are dropped.

Details

The function retrieves the URL of the CSV file for the specified index from ETF sites, then sends an HTTP GET request to download the CSV file, and processes the CSV file to extract equity constituents.

The approach is inspired by `tidyquant::tq_index()`, which uses a different wrapper around other ETFs.

Value

A tibble with five columns:

symbol The ticker symbol of the equity constituent.

name The name of the equity constituent.

location The location where the company is based.

exchange The exchange where the equity is traded.

currency The currency in which the equity is traded, derived from the exchange.

The tibble is filtered to exclude non-equity entries, blacklisted symbols, empty names, entries containing "CASH", and, when `index` is given, entries containing the index name.

See Also

Other download functions: `download_data()`, `download_data_damodaran()`, `download_data_factors_ff()`, `download_data_factors_q()`, `download_data_fred()`, `download_data_fred_md()`, `download_data_huggingface()`, `download_data_jkp()`, `download_data_macro_predictors()`, `download_data_osap()`, `download_data_pastor_stamb`
`download_data_risk_free()`, `download_data_stambaugh_yuan()`, `download_data_stock_prices()`, `download_factor_library_grid()`, `download_factor_library_ids()`

Examples

```
  download_data_constituents("DAX")

## Not run:
  download_data_constituents(path = "DAXEX_holdings.csv")

## End(Not run)
```

`download_data_damodaran`

Download and Process Damodaran Industry Data

Description

Downloads industry averages from Aswath Damodaran's data library. Currently, the cost of capital (WACC) by industry is supported via `dataset = "wacc"`. The source files are Excel workbooks, so the function requires the **readxl** package.

Usage

```
download_data_damodaran(dataset = "wacc", region = "us", year = NULL)
```

Arguments

dataset	The dataset to download. Currently only "wacc".
region	A character vector of regions: "us" (the default), "europe", "japan", "emerging", "global", "china", "india", or "rest" (Australia, New Zealand, and Canada).
year	Optional. An integer vector of years. If not provided, the most recent data is returned. Years that are not available are skipped with a warning.

Details

Damodaran updates the data each January with the data at the end of the previous year. The year column refers to that previous year, e.g., the file published in January 2026 has year = 2025. Earlier years are read from the archive of the data library. The archive starts in 1998 for the US, in 2013 for Europe, Japan, emerging markets, global, China, and India, and in 2015 for Australia, New Zealand, and Canada. China and India skip some years.

Value

A tibble with one row per region, year, and industry and the columns region, year, industry, number_of_firms, beta, cost_of_equity, equity_to_capital (E/(D+E)), sd_stock (standard deviation of stock returns), cost_of_debt, tax_rate, after_tax_cost_of_debt, debt_to_capital (D/(D+E)), wacc (cost of capital in US dollars), and wacc_local (cost of capital in local currency, NA where the source file does not provide it). Rates are decimals. The rows "Total Market" and, in recent files, "Total Market (without financials)" hold the market-wide averages.

See Also

Other download functions: download_data(), download_data_constituents(), download_data_factors_ff(), download_data_factors_q(), download_data_fred(), download_data_fred_md(), download_data_huggingface(), download_data_jkp(), download_data_macro_predictors(), download_data_osap(), download_data_pastor_stamb download_data_risk_free(), download_data_stambaugh_yuan(), download_data_stock_prices(), download_factor_library_grid(), download_factor_library_ids()

Examples

```
wacc <- download_data_damodaran("wacc")
wacc_panel <- download_data_damodaran(
  "wacc", region = c("us", "europe"), year = 2020:2024
)
```

---

download_data_factors_ff

Download and Process Fama-French Factor Data

Description

Downloads and processes Fama-French factor data based on the specified dataset name and date range. The data is downloaded directly from Kenneth French's data library and processed into a structured format, including date conversion, scaling factor values, and filtering by the specified date range.

Usage

```
download_data_factors_ff(
  dataset = NULL,
  start_date = NULL,
  end_date = NULL,
  type = deprecated()
)
```

Arguments

`dataset`	The name of the Fama-French dataset to download (e.g., "Fama/French 3 Factors").
`start_date`	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the data. If not provided, the full dataset is returned.
`end_date`	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the data. If not provided, the full dataset is returned.
`type`	**[Deprecated]** Use `dataset` instead.

Details

If there are multiple tables in the raw Fama-French data (e.g., value-weighted and equal-weighted returns), then the function only returns the first table because these are the most popular. Download the source ZIP archive directly if you need less commonly used tables.

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. Breakpoints datasets instead return `date`, the firm counts (`count`, or `<=0` and `>0` for ratios that can be non-positive), and the percentile bins `0-5`, `5-10`, ..., `95-100`, unscaled. Annual breakpoints are dated at the end of the year.

References

Fama, E. F., & French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1), 3-56. doi:10.1016/0304405X(93)900235

Fama, E. F., & French, K. R. (2015). A five-factor asset pricing model. *Journal of Financial Economics*, 116(1), 1-22. doi:10.1016/j.jfineco.2014.10.010

Carhart, M. M. (1997). On persistence in mutual fund performance. *Journal of Finance*, 52(1), 57-82. doi:10.1111/j.15406261.1997.tb03808.x

See Also

Other download functions: download_data(), download_data_constituents(), download_data_damodaran(), download_data_factors_q(), download_data_fred(), download_data_fred_md(), download_data_huggingface(), download_data_jkp(), download_data_macro_predictors(), download_data_osap(), download_data_pastor_stamb download_data_risk_free(), download_data_stambaugh_yuan(), download_data_stock_prices(), download_factor_library_grid(), download_factor_library_ids()

Examples

```
download_data_factors_ff(
  "Fama/French 3 Factors", "2000-01-01", "2020-12-31"
)
download_data_factors_ff(
  "10 Industry Portfolios", "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 dataset, date range, and source 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(
  dataset = NULL,
  start_date = NULL,
  end_date = NULL,
  type = deprecated(),
  url = "https://global-q.org/uploads/1/2/2/6/122679606/"
)
```

Arguments

dataset	The name of the dataset to download (e.g., "q5_factors_daily_2023.csv", "q5_factors_monthly_2023.csv"
start_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the data. If not provided, the full dataset is returned.
end_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the data. If not provided, the full dataset is returned.
type	**[Deprecated]** Use dataset instead.
url	The base URL from which to download the dataset files.

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.

References

Hou, K., Xue, C., & Zhang, L. (2015). Digesting anomalies: An investment approach. *Review of Financial Studies*, 28(3), 650-705. doi:10.1093/rfs/hhu068

Hou, K., Mo, H., Xue, C., & Zhang, L. (2019). Which factors? *Review of Finance*, 23(1), 1-35. doi:10.1093/rof/rfy032

See Also

Other download functions: `download_data()`, `download_data_constituents()`, `download_data_damodaran()`, `download_data_factors_ff()`, `download_data_fred()`, `download_data_fred_md()`, `download_data_huggingface()`, `download_data_jkp()`, `download_data_macro_predictors()`, `download_data_osap()`, `download_data_pastor_stamb`, `download_data_risk_free()`, `download_data_stambaugh_yuan()`, `download_data_stock_prices()`, `download_factor_library_grid()`, `download_factor_library_ids()`

Examples

```
download_data_factors_q("q5_factors_daily_2024", "2020-01-01", "2020-12-31")
download_data_factors_q("q5_factors_annual_2024")
```

---

download_data_fred *Download and Process Data from FRED*

Description

Downloads a specified data series from the Federal Reserve Economic Data (FRED) website, processes the data, and returns it as a tibble.

Usage

```
download_data_fred(series, start_date = NULL, end_date = NULL)
```

Arguments

`series`	A character vector specifying the FRED series ID(s) to download.
`start_date`	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the data. If not provided, the full dataset is returned.
`end_date`	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the data. If not provided, the full dataset is returned.

Details

Constructs the URL based on the provided FRED series ID, performs an HTTP GET request to download the data in CSV format, and processes it to a tidy tibble format. The resulting tibble includes the date, the series ID, and the value.

This approach is inspired by `quantmod::getSymbolsFRED()` which uses a different wrapper around the same FRED download data site. If you want to systematically download FRED data via API, please consider using the `fredr` package.

Value

A tibble containing the processed data with three columns:

date The date corresponding to the data point.

series The FRED series ID corresponding to the data.

value The value of the data series at that date.

See Also

Other download functions: `download_data()`, `download_data_constituents()`, `download_data_damodaran()`, `download_data_factors_ff()`, `download_data_factors_q()`, `download_data_fred_md()`, `download_data_huggingfa`
`download_data_jkp()`, `download_data_macro_predictors()`, `download_data_osap()`, `download_data_pastor_stamb`
`download_data_risk_free()`, `download_data_stambaugh_yuan()`, `download_data_stock_prices()`, `download_factor_library_grid()`, `download_factor_library_ids()`

Examples

```
download_data_fred("CPIAUCNS")
download_data_fred(c("GDP", "CPIAUCNS"), "2010-01-01", "2010-12-31")
```

---

download_data_fred_md *Download and Process FRED-MD / FRED-QD (McCracken-Ng) Databases*

Description

Downloads a vintage of the FRED-MD (monthly) or FRED-QD (quarterly) macroeconomic database - a curated, balanced panel of macro series maintained by the Federal Reserve Bank of St. Louis - and returns it as a wide tibble (one column per series), matching the layout of other factor/predictor datasets such as `download_data_factors_ff()`.

Usage

```
download_data_fred_md(
  database = "FRED-MD",
  transform = FALSE,
  vintage = "latest"
)
```

Arguments

database	Which database to download: "FRED-MD" (monthly) or "FRED-QD" (quarterly). The frequency is implied by the database.
transform	Logical. If TRUE, apply each series' McCracken-Ng stationarity transform (tcode 1-7). If FALSE (the default), return raw levels.
vintage	Which release(s) to download: "latest" (the default, the current vintage), a "YYYY-MM" label for one historical release (e.g. "2020-03"; recent ones are hosted individually, older ones are extracted from the St. Louis Fed vintage archives), or "all" for every archived vintage stacked (the full real-time panel).

Details

Each series carries a McCracken-Ng stationarity transform code (tcode, 1-7); transform = TRUE applies it per series (all transforms are causal, i.e. point-in-time safe). FRED-MD/QD publish a new vintage every month, so a specific historical release can be requested via vintage, or the entire real-time panel via vintage = "all", for point-in-time analysis that avoids look-ahead bias from data revisions. Recent vintages are hosted individually; older ones are extracted from the St. Louis Fed historical vintage archive ZIP files.

Value

A tibble [date, <series...>]. For a specific vintage or "all", a vintage column (the "YYYY-MM" release label) is inserted after date.

References

McCracken, M. W., & Ng, S. (2016). FRED-MD: A monthly database for macroeconomic research. *Journal of Business & Economic Statistics*, 34(4), 574-589. doi:10.1080/07350015.2015.1086655

McCracken, M. W., & Ng, S. (2021). FRED-QD: A quarterly database for macroeconomic research. *Federal Reserve Bank of St. Louis Review*, 103(1), 1-44. doi:10.20955/r.103.144

See Also

Other download functions: download_data(), download_data_constituents(), download_data_damodaran(), download_data_factors_ff(), download_data_factors_q(), download_data_fred(), download_data_huggingface(, download_data_jkp(), download_data_macro_predictors(), download_data_osap(), download_data_pastor_stamb, download_data_risk_free(), download_data_stambaugh_yuan(), download_data_stock_prices(), download_factor_library_grid(), download_factor_library_ids()

Examples

```
download_data_fred_md("FRED-MD")
download_data_fred_md("FRED-MD", transform = TRUE)
download_data_fred_md("FRED-MD", vintage = "2020-03")
download_data_fred_md("FRED-MD", vintage = "all")
download_data_fred_md("FRED-QD")
```

download_data_huggingface *Download data from a Hugging Face dataset*

Description

Downloads data from a supported Hugging Face dataset. For "high_frequency_sp500", parquet files are filtered by date range and row-bound. For "factor_library" and "factor_library_costs", portfolio characteristics are selected via filter_factor_library_grid(), the matching returns or trading costs are downloaded, and the result is filtered to start_date/end_date when both are supplied. For "factor_library_grid", the grid itself is returned via download_factor_library_grid(). For "factor_library_persistence", the persistence of the sorting variables is returned.

Usage

```
download_data_huggingface(
  dataset = NULL,
  start_date = NULL,
  end_date = NULL,
  type = deprecated(),
  ...
)
```

Arguments

dataset	Character(1). The dataset to download. Supported values are "high_frequency_sp500", "factor_library", "factor_library_costs", "factor_library_grid", and "factor_library_persistence".
start_date	Date or character. Start date (inclusive) in "YYYY-MM-DD" format. Used for "high_frequency_sp500", "factor_library", "factor_library_costs", and "factor_library_persistence". When omitted for the factor library, the full history is returned; "high_frequency_sp500" falls back to a built-in sample window.
end_date	Date or character. End date (inclusive) in "YYYY-MM-DD" format. See start_date.
type	**[Deprecated]** Use dataset instead.
...	For dataset = "factor_library" and "factor_library_costs": either named arguments used to filter the portfolio grid, or ids = <vector> to bypass the grid filter and download specific portfolios directly. Filter arguments take the form column = value, where value may be a vector to match multiple levels. Optionally pass fill_all = TRUE to leave unspecified columns unrestricted (default: FALSE, i.e. unspecified columns are fixed at the defaults listed below). Passing NULL for any parameter removes that filter entirely, returning all values for that column (e.g., min_size_quantile = NULL includes all size groups). Passing an unrecognised column name raises an error listing the supported names. ids cannot be combined with filter arguments. See the Details section for supported columns and their defaults.

For all four factor library datasets: `revision` and `grid_revision` select the release, see the section Revisions. For `"factor_library_grid"`: `refresh`, see `download_factor_library_grid()`. For `"factor_library_persistence"`: optionally `sorting_variable`, a vector of sorting variables; all are returned without it.

Details

Note on `dataset = "factor_library"` **defaults:** The defaults below reflect one common portfolio construction choice, but may not suit every research question. Always verify that the selected combination matches your intended design.

Supported columns and their defaults for `...`:

- `sorting_variable`: **Required.** The firm characteristic used to sort stocks into portfolios, named like the Open Source Asset Pricing signals (e.g., `"size"` for market equity, `"bm"` for book-to-market). See `download_factor_library_grid()` for all values. No default is applied.
- `min_size_quantile` (defaults to `0.2`): Fraction of the smallest stocks (by market cap) excluded from the portfolio universe. `0.2` drops the bottom 20%.
- `exclude_financials` (defaults to `FALSE`): Whether to drop financial-sector stocks (SIC 6000-6799) from the universe.
- `exclude_utilities` (default: `FALSE`): Whether to drop utility-sector stocks (SIC 4900-4999) from the universe.
- `exclude_negative_earnings` (defaults to `FALSE`): Whether to drop firms with negative earnings before sorting.
- `sorting_variable_lag` (defaults to `"6m"`): Lag applied to the sorting variable before portfolio assignment: `"1m"` (the timing of Open Source Asset Pricing), `"3m"`, `"6m"`, or `"ff"` (Fama-French).
- `rebalancing` (defaults to `"monthly"`): How frequently portfolios are reformed: `"monthly"` or `"annual"`.
- `n_portfolios_main` (defaults to `10`): Number of quantile groups (e.g., `10` for decile portfolios).
- `sorting_method` (defaults to `"univariate"`): Whether portfolios are formed on a single sort (`"univariate"`) or on a double sort with size as the second variable (`"bivariate-dependent"` or `"bivariate-independent"`).
- `n_portfolios_secondary` (defaults to `NULL`): Number of size groups for the secondary sort. Required when `sorting_method` is not `"univariate"`.
- `breakpoints_exchanges` (defaults to: `"NYSE"`): Exchange(s) used to compute breakpoints. `"NYSE"` uses only NYSE-listed stocks to define quantile cutoffs (the conventional Fama-French approach).
- `breakpoints_min_size_threshold` (defaults to `NA`): Minimum size quantile of the stocks that set the main breakpoints (e.g., `0.2`). `NA` means no minimum-size screen is applied.
- `weighting_scheme` (defaults to `"VW"`): Return weighting within portfolios: `"VW"` for value-weighted, `"EW"` for equal-weighted, or `"capped VW"` for value-weighted with capped weights.

Value

A tibble with the downloaded data. For `"high_frequency_sp500"`, contains 5-second aggregated orderbook snapshots filtered to the requested date range. For `"factor_library"`, contains the columns `id`, `date`, and `ret` joined with the full grid metadata for the matched portfolio IDs. For `"factor_library_costs"`, contains the columns `id`, `date`, `turnover`, `cost`, and `microcap_share` joined with the grid metadata in the same way. The costs start in January 1994; `turnover` and `cost` are `0` in months without trades and `microcap_share` is `NA` in months without holdings. For `"factor_library_persistence"`, contains the columns `sorting_variable`, `sorting_variable_lag`, `rebalancing`, `date`, and `persistence`, the Spearman correlation across stocks between the value of the sorting variable under the lag and rebalancing and its up-to-date value.

Revisions

The factor library is published in two Hugging Face datasets. `tidy-finance/factor-library` holds the returns and the trading costs. `tidy-finance/factor-library-grid` holds the grid and the persistence. Each release is a commit of both datasets. `revision` selects the commit of `tidy-finance/factor-library`, and `grid_revision` the commit of `tidy-finance/factor-library-grid`. A revision is a commit hash, a branch, or a tag, and both default to `"main"`, the latest release. The grid and the persistence take only `grid_revision`.

To make a download reproducible, pin both to the commits of one release. The IDs of one release need not match the grid of another.

Download limits

Hugging Face allows anonymous users 3,000 file downloads per five minutes and IP address. The returns and the costs are stored in files of 1,000 portfolio IDs, so all portfolios of one sorting variable take about 27 files of each. The factor library downloads up to 16 files at once and 8 files per second on average, also across calls, which keeps it below the limit. If Hugging Face still refuses a download with HTTP 429, e.g. because of other downloads from the same IP address, the download waits until the limit resets and tries again, twice at most.

See Also

Other download functions: `download_data()`, `download_data_constituents()`, `download_data_damodaran()`, `download_data_factors_ff()`, `download_data_factors_q()`, `download_data_fred()`, `download_data_fred_md()`, `download_data_jkp()`, `download_data_macro_predictors()`, `download_data_osap()`, `download_data_pastor_stamb`
`download_data_risk_free()`, `download_data_stambaugh_yuan()`, `download_data_stock_prices()`, `download_factor_library_grid()`, `download_factor_library_ids()`

Examples

```
## Not run:
  download_data_huggingface(
    "high_frequency_sp500", "2007-07-26", "2007-07-27"
  )
  download_data_huggingface(
    "factor_library",
    sorting_variable = "high52",
```

```
  rebalancing = "annual"
)
download_data_huggingface(
  "factor_library", sorting_variable = "assetgrowth", fill_all = TRUE
)
download_data_huggingface(
  "factor_library",
  sorting_variable = "size",
  start_date = "2000-01-01",
  end_date = "2020-12-31"
)
download_data_huggingface("factor_library", ids = c(1L, 2L, 3L))

# All portfolios of one sorting variable from the release of
# 20 October 2026
download_data_huggingface(
  "factor_library_costs",
  sorting_variable = "bm",
  fill_all = TRUE,
  revision = "e47bd85accd9b9b168ac7f478f12fe89ba5c2430",
  grid_revision = "a5a74cdf1df0c2c3d1ce3d7bf6403a8386fbf99d"
)
download_data_huggingface(
  "factor_library_persistence",
  sorting_variable = "bm",
  grid_revision = "a5a74cdf1df0c2c3d1ce3d7bf6403a8386fbf99d"
)

## End(Not run)
```

`download_data_jkp` *Download and Process Global Factor Data*

Description

Downloads and processes data from Global Factor Data, the public data library accompanying Jensen, Kelly, and Pedersen (2023). The data are stored as zipped CSV files (and a few plain CSV reference files) in a public AWS S3 bucket. For the factor, portfolio, and industry products the function validates the requested selection against the library's live availability manifest, then downloads the matching archive, unzips it, aligns monthly dates to the beginning of the month, and optionally filters by a date range.

Usage

```
download_data_jkp(
  dataset = "factors",
  region = "usa",
  factors = "all_factors",
  classification = "gics",
```

```
  frequency = "monthly",
  weighting = "vw_cap",
  start_date = NULL,
  end_date = NULL
)
```

Arguments

dataset	The Global Factor Data product to download, one of: "factors" (characteristic-managed portfolio returns, the default), "portfolios" (the underlying low/middle/high portfolios that make up each long-short factor), "industry" (industry returns), "nyse_cutoffs" (NYSE size breakpoints), or "return_cutoffs" (return winsorization cutoffs).
region	A character string with the region or country to download, using the codes from the availability manifest (e.g., "usa", "world", "developed", "emerging", or an ISO-3 country code such as "jpn"). Defaults to "usa". Ignored for the reference datasets "nyse_cutoffs" and "return_cutoffs". Call list_supported_jkp_factors() to see the available regions.
factors	A character string selecting the factor content for the "factors" and "portfolios" datasets. For "factors": "mkt" (the market factor), "all_factors" (all 153 factors), "all_themes" (all 13 themes), a single theme (e.g., "value", "momentum"), or a single factor code (e.g., "be_me", "ret_12_1"). For "portfolios": a single factor code. Defaults to "all_factors". Call list_supported_jkp_factors(region, dataset) to see the values available for a region.
classification	The industry classification for the "industry" dataset, either "gics" (default) or "ff49" (Fama-French 49 industries).
frequency	The data frequency, either "monthly" (default) or "daily". The "industry" dataset is only available at monthly frequency. For "return_cutoffs", the frequency selects the monthly or daily cutoff file.
weighting	The portfolio weighting scheme: "vw_cap" (capped value-weighted, the default), "vw" (value-weighted), or "ew" (equal-weighted). Ignored for the reference datasets.
start_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the data. If not provided, the full dataset is returned.
end_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the data. If not provided, the full dataset is returned.

Details

Returns are already expressed as plain numeric (decimal) values in the source data, so no rescaling is applied. The data are licensed under CC BY-NC 4.0 (non-commercial use).

Value

A tibble with the processed data. The date column is aligned to the beginning of the month for monthly data, and all returns are plain numeric (decimal) values. The remaining columns depend on dataset: the "factors" data carry location, name, freq, weighting, direction, n_stocks,

n_stocks_min, and ret; the "portfolios" data add a pf portfolio identifier; the "industry" data carry the classification code alongside ret; and the reference datasets carry breakpoint or cutoff columns.

References

Jensen, T. I., Kelly, B. T., & Pedersen, L. H. (2023). Is there a replication crisis in finance? *Journal of Finance*, 78(5), 2465-2518. doi:10.1111/jofi.13249

See Also

Other download functions: download_data(), download_data_constituents(), download_data_damodaran(), download_data_factors_ff(), download_data_factors_q(), download_data_fred(), download_data_fred_md(), download_data_huggingface(), download_data_macro_predictors(), download_data_osap(), download_data_pastor_stambaugh(), download_data_risk_free(), download_data_stambaugh_yuan(), download_data_stock_prices(), download_factor_library_grid(), download_factor_library_ids()

Examples

```
download_data_jkp(
  region = "usa", factors = "mkt",
  start_date = "2000-01-01", end_date = "2020-12-31"
)
download_data_jkp(
  dataset = "portfolios", region = "usa", factors = "be_me",
  start_date = "2000-01-01", end_date = "2020-12-31"
)
download_data_jkp(
  dataset = "industry", region = "usa", classification = "gics"
)
download_data_jkp(dataset = "nyse_cutoffs")
```

---

download_data_macro_predictors

Download and Process Macro Predictor Data

Description

Downloads and processes macroeconomic predictor data based on the specified dataset (monthly, quarterly, or annual), date range, and source URL. The function downloads the data from 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(
  dataset = NULL,
  start_date = NULL,
  end_date = NULL,
  type = deprecated(),
  sheet_id = "1bM7vCWd3WOt95Sf9qjLPZjoiafgF_8EG"
)
```

Arguments

dataset	The dataset to download ("monthly", "quarterly", "annual").
start_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the data. If not provided, the full dataset is returned.
end_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the data. If not provided, the full dataset is returned.
type	**[Deprecated]** Use `dataset` instead.
sheet_id	The Google Sheets ID from which to download the dataset, with the default "1bM7vCWd3WOt95Sf9qjLPZjoiafgF_8EG".

Value

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

References

Welch, I., & Goyal, A. (2008). A comprehensive look at the empirical performance of equity premium prediction. *Review of Financial Studies*, 21(4), 1455-1508. doi:10.1093/rfs/hhm014

See Also

Other download functions: `download_data()`, `download_data_constituents()`, `download_data_damodaran()`, `download_data_factors_ff()`, `download_data_factors_q()`, `download_data_fred()`, `download_data_fred_md()`, `download_data_huggingface()`, `download_data_jkp()`, `download_data_osap()`, `download_data_pastor_stambaugh()`, `download_data_risk_free()`, `download_data_stambaugh_yuan()`, `download_data_stock_prices()`, `download_factor_library_grid()`, `download_factor_library_ids()`

Examples

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

`download_data_osap` *Download and Process Open Source Asset Pricing Data*

Description

Downloads the data from Open Source Asset Pricing from Google Drive using a specified file ID, processes the data by converting column names to snake_case, aligning the date to the beginning of the month, scaling the percentage long-short returns to numeric values, and optionally filters the data based on a provided date range.

Usage

```
download_data_osap(
  start_date = NULL,
  end_date = NULL,
  sheet_id = "10sOryk_ddjkXagaajTKUk1nwJs2ZLRiI"
)
```

Arguments

`start_date`	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the data. If not provided, the full dataset is returned.
`end_date`	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the data. If not provided, the full dataset is returned. Google Drive limits how often a shared file can be downloaded. When the limit is reached, Google Drive returns an error page instead of the data and the function returns an empty tibble. The limit is usually lifted within 24 hours.
`sheet_id`	A character string with the Google Drive file ID of the wide CSV of monthly long-short returns. The default, `"10sOryk_ddjkXagaajTKUk1nwJs2ZLRiI"`, is the October 2025 release (v2.0.0). Pass the ID of another release to download it instead.

Details

The dataset contains monthly long-short returns of the predictor portfolios. Every column other than `date` is a return expressed in percent, so all of them are divided by 100 to convert them into plain numeric (decimal) returns.

Value

A tibble containing the processed data. The column names are converted to snake_case, the `date` column is aligned to the beginning of the month, all predictor columns (long-short returns in percent) are divided by 100 to obtain plain numeric (decimal) returns, and the data is filtered by the specified date range if `start_date` and `end_date` are provided.

See Also

Other download functions: download_data(), download_data_constituents(), download_data_damodaran(), download_data_factors_ff(), download_data_factors_q(), download_data_fred(), download_data_fred_md(), download_data_huggingface(), download_data_jkp(), download_data_macro_predictors(), download_data_pastor_stambaugh(), download_data_risk_free(), download_data_stambaugh_yuan(), download_data_stock_prices(), download_factor_library_grid(), download_factor_library_ids()

Examples

```
osap <- download_data_osap(
  start_date = "2020-01-01", end_date = "2020-06-30"
)
```

---

download_data_pastor_stambaugh

Download and Process Pastor-Stambaugh Liquidity Factors

Description

Downloads and processes the liquidity factor data of Pastor and Stambaugh (2003) from Stambaugh's data library. The source is a whitespace-delimited text file whose header lines start with a percent sign. The function reads the three liquidity series, aligns the monthly date to the beginning of the month, and optionally filters the data based on a provided date range.

Usage

```
download_data_pastor_stambaugh(
  start_date = NULL,
  end_date = NULL,
  url = paste0("https://faculty.wharton.upenn.edu/wp-content/uploads/2016/11/",
    "liq_data_1962_2025.txt")
)
```

Arguments

start_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the data. If not provided, the full dataset is returned.
end_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the data. If not provided, the full dataset is returned.
url	A character string with the URL of the liquidity data file. Because the file name embeds the last year of data, the default points to the most recent file known at release time; override it when a newer file becomes available.

Details

The series are already expressed as plain numeric (decimal) values in the source data, so no rescaling is applied. The traded liquidity factor is only available from 1968 onward; earlier observations are coded as `-99` in the source file and are returned as `NA`.

Value

A tibble with the columns `date` (aligned to the beginning of the month), `agg_liq` (levels of aggregate liquidity), `innov_liq` (innovations in aggregate liquidity, the non-traded liquidity factor), and `traded_liq` (the traded liquidity factor LIQ_V), filtered by the specified date range if `start_date` and `end_date` are provided.

References

Pastor, L., & Stambaugh, R. F. (2003). Liquidity risk and expected stock returns. *Journal of Political Economy*, 111(3), 642-685. doi:10.1086/374184

See Also

Other download functions: `download_data()`, `download_data_constituents()`, `download_data_damodaran()`, `download_data_factors_ff()`, `download_data_factors_q()`, `download_data_fred()`, `download_data_fred_md()`, `download_data_huggingface()`, `download_data_jkp()`, `download_data_macro_predictors()`, `download_data_osap()`, `download_data_risk_free()`, `download_data_stambaugh_yuan()`, `download_data_stock_pr`
`download_factor_library_grid()`, `download_factor_library_ids()`

Examples

```
pastor_stambaugh <- download_data_pastor_stambaugh(
  start_date = "2020-01-01", end_date = "2020-12-31"
)
```

---

download_data_pseudo_ccm_links

Generate Pseudo CCM Links

Description

Returns a pseudo CRSP-Compustat linking table with the same column layout as `download_data_wrds_ccm_links()`. Every pseudo `permno` is linked to its corresponding `gvkey` for the full sample horizon.

Usage

```
download_data_pseudo_ccm_links(
  n_assets = 1000L,
  seed = 1234L,
  linktype = c("LU", "LC"),
  linkprim = c("P", "C")
)
```

Arguments

n_assets	Integer. Number of pseudo firms in the universe. Defaults to 1000.
seed	Integer. Random seed controlling the pseudo identifier universe; defaults to 1234.
linktype	Accepted for API compatibility with download_data_wrds_ccm_links(); ignored for pseudo data.
linkprim	Accepted for API compatibility with download_data_wrds_ccm_links(); ignored for pseudo data.

Value

A tibble with columns permno, gvkey, linkdt, and linkenddt, one row per pseudo firm.

See Also

Other pseudo functions: download_data_pseudo_compustat(), download_data_pseudo_crsp()

Examples

```
download_data_pseudo_ccm_links(n_assets = 10)
```

---

download_data_pseudo_compustat

Generate Pseudo Compustat Data

Description

Returns pseudo Compustat data with the same column layout as download_data_wrds_compustat(). Useful for testing and for reproducing the workflow of analyses that rely on Compustat without a WRDS subscription. The returned values are simulated and not suitable for inference.

Usage

```
download_data_pseudo_compustat(
  dataset = NULL,
  start_date = NULL,
  end_date = NULL,
  additional_columns = NULL,
  only_usd = FALSE,
  n_assets = 1000L,
  seed = 1234L
)
```

Arguments

dataset	A string specifying the dataset to simulate. Supported: "compustat_annual" and "compustat_quarterly".
start_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the pseudo panel.
end_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the pseudo panel.
additional_columns	Additional Compustat columns to include. Filled with plausible random draws so call sites that pass additional_columns continue to work; the values themselves are not economically meaningful.
only_usd	Accepted for API compatibility with download_data_wrds_compustat(); the pseudo universe is treated as USD-denominated, so this argument has no effect.
n_assets	Integer. Number of pseudo firms in the universe. Defaults to 1000.
seed	Integer. Random seed; defaults to 1234. Identical (seed, n_assets) produces identical output across calls and matches the identifier universe used by download_data_pseudo_crsp() and download_data_pseudo_ccm_links().

Details

Both "compustat_annual" and "compustat_quarterly" are supported.

Value

For "compustat_annual", a tibble with columns gvkey, date, datadate, the financial-statement variables seq, ceq, at, lt, txditc, txdb, itcb, pstkrv, pstkl, pstk, capx, oancf, sale, cogs, xint, xsga, ib, curcd, plus the derived be, op, at_lag, inv, and any requested additional_columns. For "compustat_quarterly", a tibble with columns gvkey, date, datadate, atq, ceqq, and any requested additional_columns.

See Also

Other pseudo functions: download_data_pseudo_ccm_links(), download_data_pseudo_crsp()

Examples

```
download_data_pseudo_compustat(
  "compustat_annual",
  start_date = "2020-01-01",
  end_date = "2024-12-31",
  n_assets = 20
)
download_data_pseudo_compustat(
  "compustat_quarterly",
  start_date = "2020-01-01",
  end_date = "2024-12-31",
  n_assets = 20
)
```

download_data_pseudo_crsp *Generate Pseudo CRSP Data*

Description

Returns pseudo CRSP data with the same column layout as `download_data_wrds_crsp()`. Useful for testing and for reproducing the workflow of analyses that rely on CRSP without a WRDS subscription. The returned values are simulated and not suitable for inference.

Usage

```
download_data_pseudo_crsp(
  dataset = NULL,
  start_date = NULL,
  end_date = NULL,
  version = "v2",
  additional_columns = NULL,
  add_ccm_links = FALSE,
  adjust_volume = FALSE,
  batch_size = 500,
  n_assets = 1000L,
  seed = 1234L
)
```

Arguments

`dataset`	A string specifying the dataset to simulate. Supported: `"crsp_monthly"` and `"crsp_daily"`.
`start_date`	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the pseudo panel.
`end_date`	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the pseudo panel.
`version`	Accepted for API compatibility with `download_data_wrds_crsp()`; the pseudo schema follows the v2 output.
`additional_columns`	Additional CRSP columns to include. Filled with plausible random draws so call sites that pass `additional_columns` continue to work; the values themselves are not economically meaningful.
`add_ccm_links`	A logical indicating whether CRSP-Compustat links should be appended. When `TRUE`, the output gains a gvkey column whose values are derived from the same pseudo identifier universe used by `download_data_pseudo_ccm_links()`.
`adjust_volume`	Accepted for API compatibility with `download_data_wrds_crsp()`; ignored for pseudo data.

batch_size	Accepted for API compatibility with download_data_wrds_crsp(); ignored for pseudo data.
n_assets	Integer. Number of pseudo firms in the universe. Defaults to 1000.
seed	Integer. Random seed; defaults to 1234. Identical (seed, n_assets) produces identical output across calls and matches the identifier universe used by download_data_pseudo_compustat() and download_data_pseudo_ccm_links().

Details

Both "crsp_monthly" and "crsp_daily" are supported. The daily panel uses weekdays (Monday-Friday) only; weekend dates are excluded so the pseudo calendar approximates a trading-day grid.

Value

For "crsp_monthly", a tibble with columns permno, date, calculation_date, ret, shrout, prc, primaryexch, siccd, listing_age, mktcap, mktcap_lag, prc_lag, exchange, industry, and ret_excess. For "crsp_daily", a tibble with columns permno, date, ret, and ret_excess. If add_ccm_links = TRUE, a gvkey column is appended.

See Also

Other pseudo functions: download_data_pseudo_ccm_links(), download_data_pseudo_compustat()

Examples

```
download_data_pseudo_crsp(
  "crsp_monthly",
  start_date = "2020-01-01",
  end_date = "2024-12-31",
  n_assets = 20
)
download_data_pseudo_crsp(
  "crsp_daily",
  start_date = "2020-01-01",
  end_date = "2020-03-31",
  n_assets = 20
)
```

---

download_data_risk_free

Download Risk-Free Rate Data

Description

Downloads pre-processed risk-free rate data from the tidy-finance/risk-free dataset on HuggingFace. The dataset is updated monthly via a scheduled GitHub Actions workflow that splices the 3-Month Treasury Bill Secondary Market Rate (pre-2001) with the 4-Week Treasury Bill Secondary

Market Rate (from 2001 onwards) sourced from FRED. For monthly data, the monthly TB3MS series is spliced with the daily DTB4WK series aggregated to month-end. For daily data, the daily DTB3 series is spliced with the daily DTB4WK series, both at the business-day frequency provided by FRED.

Usage

```
download_data_risk_free(
  start_date = NULL,
  end_date = NULL,
  frequency = "monthly"
)
```

Arguments

`start_date`	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the data. If not provided, the full dataset is returned.
`end_date`	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the data. If not provided, the full dataset is returned.
`frequency`	A character string, either `"monthly"` (default) or `"daily"`, specifying the frequency of the returned data. Daily data starts in 1954-01-04 because of availability of DTB3, while monthly data starts in 1934-01-01.

Details

Both series are quoted as annualised bank discount rates on a 360-day basis. Given an annualised discount rate d and a T-bill with n days to maturity, the holding-period return is `HPR = d * n/360 / (1 - d * n/360)`, which is then converted to the target period length via `(1 + HPR)^(target/source) - 1`.

The series are spliced at 2001-07-01:

- **Pre-2001**: TB3MS (monthly) or DTB3 (daily), 3-month T-bill with n = 90. Monthly conversion uses exponent `1/3`; daily conversion uses exponent `1/63` (approx. trading days per quarter).
- **From 2001**: DTB4WK, 4-week T-bill with n = 28. For monthly data, the last non-NA observation per calendar month is taken and the exponent is `365/(28*12)`. For daily data, observations are used as-is and the exponent is `1/20` (approx. trading days per 4-week period).

Business-day gaps in the daily series (e.g. holidays) are handled by forward-filling the most recent available rate.

Monthly data starts in 1934-01-01 (TB3MS). Daily data starts in 1954-01-04 due to the availability of DTB3.

Value

A tibble with two columns:

date The date of the observation.

risk_free The risk-free rate for the period.

See Also

Other download functions: download_data(), download_data_constituents(), download_data_damodaran(), download_data_factors_ff(), download_data_factors_q(), download_data_fred(), download_data_fred_md(), download_data_huggingface(), download_data_jkp(), download_data_macro_predictors(), download_data_osap(), download_data_pastor_stambaugh(), download_data_stambaugh_yuan(), download_data_stock_prices(), download_factor_library_grid(), download_factor_library_ids()

Examples

```
download_data_risk_free("2020-01-01", "2020-12-31")
download_data_risk_free(
  "2020-01-01", "2020-12-31", frequency = "daily"
)
```

---

download_data_stambaugh_yuan

Download and Process Stambaugh-Yuan Mispricing Factors

Description

Downloads and processes the mispricing factor data of Stambaugh and Yuan (2017) from Stambaugh's data library. The four-factor model (M4) combines the market and size factors with two mispricing factors, `mgmt` (management) and `perf` (performance). The function downloads the requested frequency, aligns the date, renames the columns to the package conventions, and optionally filters the data based on a provided date range.

Usage

```
download_data_stambaugh_yuan(
  dataset = "monthly",
  start_date = NULL,
  end_date = NULL,
  url = "https://faculty.wharton.upenn.edu/wp-content/uploads/2016/11/"
)
```

Arguments

`dataset`	The data frequency to download, either `"monthly"` (the default) or `"daily"`.
`start_date`	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the data. If not provided, the full dataset is returned.
`end_date`	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the data. If not provided, the full dataset is returned.
`url`	The base URL from which to download the dataset files. The file name (`M4.csv` or `M4d.csv`) is appended based on `dataset`.

Details

Returns are already expressed as plain numeric (decimal) values in the source data, so no rescaling is applied. The source files currently end in December 2016; a requested date range that lies entirely outside the available data emits a warning and returns an empty tibble.

Value

A tibble with the columns `date` (aligned to the beginning of the month for monthly data), `mkt_excess` (the market excess return), `smb` (size), `mgmt` (the management mispricing factor), `perf` (the performance mispricing factor), and `risk_free` (the risk-free rate). All returns are plain numeric (decimal) values, filtered by the specified date range if `start_date` and `end_date` are provided.

References

Stambaugh, R. F., & Yuan, Y. (2017). Mispricing factors. *Review of Financial Studies*, 30(4), 1270-1315. doi:10.1093/rfs/hhw107

See Also

Other download functions: `download_data()`, `download_data_constituents()`, `download_data_damodaran()`, `download_data_factors_ff()`, `download_data_factors_q()`, `download_data_fred()`, `download_data_fred_md()`, `download_data_huggingface()`, `download_data_jkp()`, `download_data_macro_predictors()`, `download_data_osap()`, `download_data_pastor_stambaugh()`, `download_data_risk_free()`, `download_data_stock_prices()`, `download_factor_library_grid()`, `download_factor_library_ids()`

Examples

```
download_data_stambaugh_yuan(
  start_date = "2015-01-01", end_date = "2016-12-31"
)
download_data_stambaugh_yuan(
  dataset = "daily", start_date = "2016-01-01", end_date = "2016-12-31"
)
```

---

download_data_stock_prices

Download Stock Data

Description

Downloads historical stock data from Yahoo Finance for given symbols and date range.

Usage

```
download_data_stock_prices(symbols, start_date = NULL, end_date = NULL)
```

Arguments

symbols	A character vector of stock symbols to download data for. At least one symbol must be provided.
start_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the data. If not provided, a one-year subset of the dataset is returned (see validate_dates()).
end_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the data. If not provided, a one-year subset of the dataset is returned (see validate_dates()).

Details

Dates are the trading days in the exchange's local time zone, as reported by Yahoo Finance, and the range is inclusive of both start_date and end_date.

Value

A tibble containing the downloaded stock data with columns: symbol, date, volume, open, low, high, close, and adjusted_close.

See Also

Other download functions: download_data(), download_data_constituents(), download_data_damodaran(), download_data_factors_ff(), download_data_factors_q(), download_data_fred(), download_data_fred_md(), download_data_huggingface(), download_data_jkp(), download_data_macro_predictors(), download_data_osap(), download_data_pastor_stambaugh(), download_data_risk_free(), download_data_stambaugh_yuan(), download_factor_library_grid(), download_factor_library_ids()

Examples

```
download_data_stock_prices(c("AAPL", "MSFT"))
download_data_stock_prices("GOOGL", "2021-01-01", "2022-01-01" )
```

---

download_data_wrds *Download Data from WRDS*

Description

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

Usage

```
download_data_wrds(
  dataset = NULL,
  start_date = NULL,
  end_date = NULL,
  type = deprecated(),
  ...
)
```

Arguments

dataset	A string specifying the dataset to download. Supported values: "crsp_monthly", "crsp_daily" for CRSP data, "compustat_annual", "compustat_quarterly" for Compustat data, "ccm_links" for CCM links data, "fisd" for FISD data, or "trace_enhanced" for TRACE data.
start_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the data. If not provided, a subset of the dataset is returned.
end_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the data. If not provided, a subset of the dataset is returned.
type	**[Deprecated]** Use `dataset` instead.
...	Additional arguments passed to specific download functions depending on the `dataset`. For instance, `quiet = TRUE` suppresses the messages about the WRDS connection and the download progress.

Value

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

See Also

Other WRDS functions: `disconnect_connection()`, `download_data_wrds_ccm_links()`, `download_data_wrds_compus`
`download_data_wrds_crsp()`, `download_data_wrds_fisd()`, `download_data_wrds_trace_enhanced()`,
`get_wrds_connection()`, `set_wrds_credentials()`

Examples

```
## Not run:
  crsp_monthly <- download_data_wrds(
    "crsp_monthly", "2020-01-01", "2020-12-31"
  )
  compustat_annual <- download_data_wrds(
    "compustat_annual", "2020-01-01", "2020-12-31"
  )
  ccm_links <- download_data_wrds("ccm_links")
  fisd <- download_data_wrds("fisd")
  trace_enhanced <- download_data_wrds(
```

```
    "trace_enhanced", cusips = "00101JAH9"
  )

## End(Not run)
```

`download_data_wrds_ccm_links` *Download CCM Links from WRDS*

Description

Downloads data from the WRDS CRSP/Compustat Merged (CCM) links database. It allows users to specify the type of links (`linktype`) and the primacy of the link (`linkprim`).

Usage

```
download_data_wrds_ccm_links(
  linktype = c("LU", "LC"),
  linkprim = c("P", "C"),
  quiet = FALSE
)
```

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.
`quiet`	A logical indicating whether to suppress the messages about the WRDS connection and the download progress. Defaults to `FALSE`.

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.

See Also

Other WRDS functions: `disconnect_connection()`, `download_data_wrds()`, `download_data_wrds_compustat()`, `download_data_wrds_crsp()`, `download_data_wrds_fisd()`, `download_data_wrds_trace_enhanced()`, `get_wrds_connection()`, `set_wrds_credentials()`

Examples

```
## Not run:
  ccm_links <- download_data_wrds_ccm_links(linktype = "LU", linkprim = "P")

## End(Not run)
```

`download_data_wrds_compustat` *Download Data from WRDS Compustat*

Description

Downloads financial data from the WRDS Compustat database for a given dataset, start date, and end date. It filters the data according to industry format, data format, and consolidation level, and returns the most current data for each reporting period. Additionally, the annual data also includes the calculated book equity (be), operating profitability (op), and investment (inv) for each company following Fama & French (1993, 2015), as well as income before extraordinary items (ib).

Usage

```
download_data_wrds_compustat(
  dataset = NULL,
  start_date = NULL,
  end_date = NULL,
  type = deprecated(),
  additional_columns = NULL,
  only_usd = FALSE,
  only_us = deprecated(),
  quiet = FALSE
)
```

Arguments

`dataset`	The dataset to download ("compustat_annual" or "compustat_quarterly").
`start_date`	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the data. If not provided, a subset of the dataset is returned.
`end_date`	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the data. If not provided, a subset of the dataset is returned.
`type`	**[Deprecated]** Use `dataset` instead.
`additional_columns`	Additional columns from the Compustat table as a character vector.
`only_usd`	A logical indicating whether only USD-denominated shares should be returned.
`only_us`	**[Deprecated]** Use `only_usd` instead.
`quiet`	A logical indicating whether to suppress the messages about the WRDS connection and the download progress. Defaults to `FALSE`.

Value

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

References

Fama, E. F., & French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1), 3-56. doi:10.1016/0304405X(93)900235

Fama, E. F., & French, K. R. (2015). A five-factor asset pricing model. *Journal of Financial Economics*, 116(1), 1-22. doi:10.1016/j.jfineco.2014.10.010

See Also

Other WRDS functions: `disconnect_connection()`, `download_data_wrds()`, `download_data_wrds_ccm_links()`, `download_data_wrds_crsp()`, `download_data_wrds_fisd()`, `download_data_wrds_trace_enhanced()`, `get_wrds_connection()`, `set_wrds_credentials()`

Examples

```
## Not run:
download_data_wrds_compustat("compustat_annual", "2020-01-01", "2020-12-31")
download_data_wrds_compustat(
  "compustat_quarterly",
  "2020-01-01",
  "2020-12-31"
)

  # Add additional columns
download_data_wrds_compustat(
  "compustat_annual",
  additional_columns = c("aodo", "aldo")
)

## End(Not run)
```

`download_data_wrds_crsp`

Download Data from WRDS CRSP

Description

Downloads and processes stock return data from the CRSP database for a specified period. Users can choose between monthly and daily datasets. The function also includes delisting returns (see the section on delisting returns) and calculates market capitalization and excess returns over the risk-free rate.

Usage

```
download_data_wrds_crsp(
  dataset = NULL,
  start_date = NULL,
  end_date = NULL,
  type = deprecated(),
```

```
  batch_size = 500,
  version = "v2",
  additional_columns = NULL,
  add_ccm_links = FALSE,
  adjust_volume = FALSE,
  quiet = FALSE
)
```

Arguments

dataset	A string specifying the CRSP dataset to download: "crsp_monthly" or "crsp_daily".
start_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the data. If not provided, a subset of the dataset is returned.
end_date	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the data. If not provided, a subset of the dataset is returned.
type	**[Deprecated]** Use `dataset` instead.
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.
add_ccm_links	A logical indicating whether CRSP-Compustat links should be added automatically using `download_data_wrds_ccm_links()`.
adjust_volume	A logical indicating whether daily CRSP trading volume data should be adjusted according to Gao & Ritter (2010).
quiet	A logical indicating whether to suppress the messages about the WRDS connection and the download progress. Defaults to `FALSE`.

Value

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

Delisting returns

How delisting returns enter the data depends on `version`:

- `"v2"` (CIZ): CRSP already includes the delisting return in the monthly (`mthret`) and daily (`dlyret`) returns of the month and the day in which a stock delists, so `ret` and `ret_excess` include it without further adjustment. The common-stock screens of the security history (`crsp.stksecurityinfohist`) are evaluated on the stock's last trading day for these observations, because the history holds no active record after it. Without this, the month or day that carries the delisting return would be dropped.

- `"v1"` (legacy SIZ), monthly: the delisting return `dlret` from `crsp.msedelist` replaces the return of the delisting month in `ret_adj`, which `ret_excess` is based on. If `dlret` is missing, the return is set to -30% for performance-related delistings (codes 500, 520, 551-574, 580, and 584; Shumway, 1997), kept for code 100, and set to -100% otherwise. `ret` is the unadjusted return.
- `"v1"` (legacy SIZ), daily: the delisting return replaces the return on the delisting date, and later observations are dropped.

References

Shumway, T. (1997). The delisting bias in CRSP data. *Journal of Finance*, 52(1), 327-340. doi:10.1111/j.15406261.1997.tb03818.x

Gao, X., & Ritter, J. R. (2010). The marketing of seasoned equity offerings. *Journal of Financial Economics*, 97(1), 33-52. doi:10.1016/j.jfineco.2010.03.007

See Also

Other WRDS functions: `disconnect_connection()`, `download_data_wrds()`, `download_data_wrds_ccm_links()`, `download_data_wrds_compustat()`, `download_data_wrds_fisd()`, `download_data_wrds_trace_enhanced()`, `get_wrds_connection()`, `set_wrds_credentials()`

Examples

```
## Not run:
crsp_monthly <- download_data_wrds_crsp(
  "crsp_monthly",
  "2020-11-01",
  "2020-12-31"
)
crsp_daily <- download_data_wrds_crsp(
  "crsp_daily",
  "2020-12-01",
  "2020-12-31"
)

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

## End(Not run)
```

`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(additional_columns = NULL, quiet = FALSE)
```

Arguments

`additional_columns`	Additional columns from the FISD table as a character vector.
`quiet`	A logical indicating whether to suppress the messages about the WRDS connection and the download progress. Defaults to `FALSE`.

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 date, offering amount, offering date, dated date, interest frequency, coupon, last interest date, issue ID, issuer ID, and SIC code of the issuer.

See Also

Other WRDS functions: `disconnect_connection()`, `download_data_wrds()`, `download_data_wrds_ccm_links()`, `download_data_wrds_compustat()`, `download_data_wrds_crsp()`, `download_data_wrds_trace_enhanced()`, `get_wrds_connection()`, `set_wrds_credentials()`

Examples

```
## Not run:
fisd <- download_data_wrds_fisd()
fisd_extended <- download_data_wrds_fisd(
  additional_columns = c("asset_backed", "defeased")
)

## End(Not run)
```

`download_data_wrds_trace_enhanced` *Download Enhanced 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_trace_enhanced(
  cusips,
  start_date = NULL,
  end_date = NULL,
  quiet = FALSE
)
```

Arguments

`cusips`	A character vector specifying the 9-digit CUSIPs to download.
`start_date`	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the start date for the data. If not provided, a subset of the dataset is returned.
`end_date`	Optional. A character string or Date object in "YYYY-MM-DD" format specifying the end date for the data. If not provided, a subset of the dataset is returned.
`quiet`	A logical indicating whether to suppress the messages about the WRDS connection and the download progress. Defaults to `FALSE`.

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).

References

Dick-Nielsen, J. (2009). Liquidity biases in TRACE. *Journal of Fixed Income*, 19(2), 43-55. doi:10.3905/jfi.2009.19.2.043

Dick-Nielsen, J. (2014). How to clean enhanced TRACE data. Working Paper. doi:10.2139/ssrn.2337908

See Also

Other WRDS functions: disconnect_connection(), download_data_wrds(), download_data_wrds_ccm_links(), download_data_wrds_compustat(), download_data_wrds_crsp(), download_data_wrds_fisd(), get_wrds_connection(), set_wrds_credentials()

Examples

```
## Not run:
download_data_wrds_trace_enhanced("00101JAH9", "2019-01-01", "2021-12-31")

## End(Not run)
```

---

download_factor_library_grid

Download the Factor Library Grid from Hugging Face

Description

Returns the tidy-finance/factor-library-grid dataset, which describes every portfolio construction available in the factor library (one row per construction, identified by id). Use the returned tibble to discover which (sorting_variable, weighting_scheme, rebalancing, ...) combinations exist before requesting their returns with download_factor_library_ids().

Usage

```
download_factor_library_grid(refresh = FALSE, grid_revision = "main")
```

Arguments

refresh	Logical(1). If TRUE, the grid is downloaded again and replaces the copy cached in the current session. Defaults to FALSE.
grid_revision	Character(1). The revision of tidy-finance/factor-library-grid to read: a commit hash, a branch, or a tag. Defaults to "main", the latest release. Pin it to the release of the returns, see the section Revisions of download_data_huggingface().

Details

Equivalent to calling download_data("Tidy Finance", "factor_library_grid").

The grid is downloaded once per R session and revision and cached in memory, so repeated calls (including those made by download_factor_library_ids() and download_data("Tidy Finance", "factor_library", ...)) reuse it. Set refresh = TRUE to download it again. Each revision is cached separately and takes about 450 MB of memory.

Value

A tibble with one row per portfolio construction in the factor library, including the integer id column used by download_factor_library_ids().

See Also

Other download functions: download_data(), download_data_constituents(), download_data_damodaran(), download_data_factors_ff(), download_data_factors_q(), download_data_fred(), download_data_fred_md(), download_data_huggingface(), download_data_jkp(), download_data_macro_predictors(), download_data_osap(), download_data_pastor_stambaugh(), download_data_risk_free(), download_data_stambaugh_yuan(), download_data_stock_prices(), download_factor_library_ids()

Examples

```
## Not run:
  download_factor_library_grid()
  download_factor_library_grid(
    grid_revision = "a5a74cdf1df0c2c3d1ce3d7bf6403a8386fbf99d"
  )

## End(Not run)
```

---

download_factor_library_ids

Download factor library returns for a vector of portfolio IDs

---

Description

Given a vector of portfolio IDs from the tidy-finance/factor-library-grid Hugging Face dataset, downloads the corresponding return data from the tidy-finance/factor-library dataset on Hugging Face. The returns are stored in files of 1,000 consecutive IDs named after the range they cover (e.g., id_0000001-0001000.parquet), so the function downloads only the files that hold the requested IDs. The grid metadata is joined onto the result.

Usage

```
download_factor_library_ids(ids, revision = "main", grid_revision = "main")
```

Arguments

ids	Integer or numeric vector of portfolio IDs to download. IDs correspond to rows of the tidy-finance/factor-library-grid dataset.
revision	Character(1). The revision of tidy-finance/factor-library to read the returns from: a commit hash, a branch, or a tag. Defaults to "main", the latest release.
grid_revision	Character(1). The revision of tidy-finance/factor-library-grid to read: a commit hash, a branch, or a tag. Defaults to "main", the latest release. Pin it to the release of the returns, see the section Revisions of download_data_huggingface().

Details

Use this function when you already know the portfolio IDs you want (for example, from a previous call to `download_data_huggingface()` with `dataset = "factor_library"`). To resolve IDs from filter criteria (sorting variable, weighting scheme, breakpoints, etc.) and download in a single call, use `download_data_huggingface()` instead.

Raises an error if none of the requested IDs exist in the grid. IDs whose portfolio sort produced no portfolios have no returns; they are absent from the result, with a warning, and the result has no rows if none of the requested IDs has returns. Returns are stored in single precision, and months without a valid long-short return are stored as `0`.

Value

A tibble with the columns `id`, `date`, and `ret` (the monthly long-short excess return) and the grid metadata columns for the requested IDs.

See Also

Other download functions: `download_data()`, `download_data_constituents()`, `download_data_damodaran()`, `download_data_factors_ff()`, `download_data_factors_q()`, `download_data_fred()`, `download_data_fred_md()`, `download_data_huggingface()`, `download_data_jkp()`, `download_data_macro_predictors()`, `download_data_osap()`, `download_data_pastor_stambaugh()`, `download_data_risk_free()`, `download_data_stambaugh_yuan()`, `download_data_stock_prices()`, `download_factor_library_grid()`

Examples

```
## Not run:
  download_factor_library_ids(c(1L, 2L, 3L))
  download_factor_library_ids(
    c(1L, 2L, 3L),
    revision = "e47bd85accd9b9b168ac7f478f12fe89ba5c2430",
    grid_revision = "a5a74cdf1df0c2c3d1ce3d7bf6403a8386fbf99d"
  )

## End(Not run)
```

`estimate_betas` *Estimate Rolling Betas*

Description

Estimates rolling betas for a given linear model using a fast, vectorized approach. Instead of fitting one regression per stock and estimation window, the function collapses the data to additive cumulants (the entries of the moment matrices $X'X$ and $X'y$) per stock and period, aggregates these cumulants over rolling calendar windows in a single pass with `slider::slide_index_sum()`, and recovers the coefficients via the closed-form OLS solution $\hat{\beta} = (X'X)^{-1}X'y$. This produces estimates that are numerically identical to a per-window regression, but fast enough that no per-stock nesting or parallelization is required.

Usage

```
estimate_betas(data, model, lookback, min_obs = NULL, data_options = NULL)
```

Arguments

data	A data frame containing the data with a date identifier (defaults to date), a stock identifier (defaults to permno), and other variables used in the model.
model	A character string describing the model to be estimated (e.g., "ret_excess ~ mkt_excess + hml + smb").
lookback	A Period object specifying the number of months, days, hours, minutes, or seconds to look back when estimating the rolling model.
min_obs	An integer specifying the minimum number of observations required to estimate the model. Defaults to 80% of lookback. Windows with fewer observations are dropped from the output.
data_options	A list of class tidyfinance_data_options (created via data_options()) specifying column name mappings. The id is used to specify the entity (i.e., firm), and the date element is used to specify the date column. Uses data_options() default if NULL: "id" = "permno" and "date" = "date".

Value

A data frame with the estimated betas for each entity and period. It contains the entity and date identifiers, an intercept column (if the model includes one), and one beta_<variable> column per regressor.

See Also

Other estimation functions: estimate_fama_macbeth(), estimate_model()

Examples

```
# Estimate monthly betas using monthly return data
set.seed(1234)
data_monthly <- tibble::tibble(
  date = rep(seq.Date(from = as.Date("2020-01-01"),
                      to = as.Date("2020-12-01"), by = "month"), each = 50),
  permno = rep(1:50, times = 12),
  ret_excess = rnorm(600, 0, 0.1),
  mkt_excess = rnorm(600, 0, 0.1),
  smb = rnorm(600, 0, 0.1),
  hml = rnorm(600, 0, 0.1),
)

estimate_betas(data_monthly,  "ret_excess ~ mkt_excess", months(3))
estimate_betas(
  data_monthly,
  "ret_excess ~ mkt_excess + smb + hml",
  months(6)
)
```

```
data_monthly |>
  dplyr::rename(id = permno) |>
  estimate_betas("ret_excess ~ mkt_excess", months(3),
                 data_options = data_options(id = "id"))

# Estimate monthly betas using daily return data
data_daily <- tibble::tibble(
  date = rep(seq.Date(from = as.Date("2020-01-01"),
                      to = as.Date("2020-12-31"), by = "day"), each = 50),
  permno = rep(1:50, times = 366),
  ret_excess = rnorm(18300, 0, 0.02),
  mkt_excess = rnorm(18300, 0, 0.02),
  smb = rnorm(18300, 0, 0.02),
  hml = rnorm(18300, 0, 0.02),
)

data_daily <- data_daily |>
  dplyr::mutate(date = lubridate::floor_date(date, "month"))

estimate_betas(
  data_daily,
  "ret_excess ~ mkt_excess",
  lubridate::days(90)
)
```

`estimate_fama_macbeth` *Estimate Fama-MacBeth Regressions*

Description

Estimates Fama-MacBeth regressions (Fama and MacBeth, 1973) by first running cross-sectional regressions for each time period and then aggregating the results over time to obtain average risk premia and corresponding t-statistics.

Usage

```
estimate_fama_macbeth(
  data,
  model,
  vcov = "newey-west",
  vcov_options = NULL,
  data_options = NULL,
  detail = FALSE
)
```

Arguments

data	A data frame containing the data for the regression. It must include a column representing the time periods (defaults to date) and the variables specified in the model.
model	A character string describing the model to be estimated in each cross-section (e.g., "ret_excess ~ beta + bm + log_mktcap").
vcov	A character string indicating the type of standard errors to compute. Options are "iid" for independent and identically distributed errors or "newey-west" for Newey-West standard errors. Default is "newey-west".
vcov_options	A list of additional arguments to be passed to the NeweyWest() function when vcov = "newey-west". These can include options such as lag, which specifies the number of lags to use in the Newey-West covariance matrix estimation, and prewhite, which indicates whether to apply a prewhitening transformation. Default is an empty list, which uses the defaults of sandwich::NeweyWest(): prewhitening with a VAR(1) and the automatic bandwidth of Newey and West (1994).
data_options	A list of class tidyfinance_data_options (created via data_options()) specifying column name mappings. The date element is used to specify the date column. Uses data_options() default if NULL: "date" = "date".
detail	A logical value indicating whether to return additional summary statistics. If FALSE (default), the function returns only the coefficient estimates. If TRUE, it returns a list with two elements: coefficients (the usual estimates table) and summary_statistics (a one-row tibble with the average cross-sectional R-squared, the average cross-sectional adjusted R-squared, and the average number of observations per cross-section).

Value

If detail = FALSE (default), a tibble with columns factor, risk_premium, n (number of time periods), standard_error, and t_statistic.

If detail = TRUE, a named list with two elements:

coefficients The same tibble described above.

summary_statistics A one-row tibble with r_squared (mean cross-sectional R-squared), adj_r_squared (mean cross-sectional adjusted R-squared), and n_obs (mean cross-sectional observation count).

References

Fama, E. F., & MacBeth, J. D. (1973). Risk, return, and equilibrium: Empirical tests. *Journal of Political Economy*, 81(3), 607-636. doi:10.1086/260061

Newey, W. K., & West, K. D. (1987). A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica*, 55(3), 703-708. doi:10.2307/1913610

See Also

Other estimation functions: estimate_betas(), estimate_model()

Examples

```
set.seed(1234)

data <- tibble::tibble(
  date = rep(seq.Date(from = as.Date("2020-01-01"),
                      to = as.Date("2020-12-01"), by = "month"), each = 50),
  permno = rep(1:50, times = 12),
  ret_excess = rnorm(600, 0, 0.1),
  beta = rnorm(600, 1, 0.2),
  bm = rnorm(600, 0.5, 0.1),
  log_mktcap = rnorm(600, 10, 1)
)

estimate_fama_macbeth(data, "ret_excess ~ beta + bm + log_mktcap")
estimate_fama_macbeth(
  data,
  "ret_excess ~ beta + bm + log_mktcap",
  vcov = "iid"
)
estimate_fama_macbeth(
  data,
  "ret_excess ~ beta + bm + log_mktcap",
  vcov = "newey-west",
  vcov_options = list(lag = 6, prewhite = FALSE)
)

# Return detailed output including R-squared and observation counts
estimate_fama_macbeth(
  data,
  "ret_excess ~ beta + bm + log_mktcap",
  detail = TRUE
)

# Use different column name for date
data |>
  dplyr::rename(month = date) |>
  estimate_fama_macbeth(
    "ret_excess ~ beta + bm + log_mktcap",
    data_options = data_options(date = "month")
 )
```

`estimate_model` *Estimate a Linear Model*

Description

Estimates 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. Depending on the `output` parameter, it returns the model's coefficients, t-statistics, residuals, or any combination in a named list.

Usage

```
estimate_model(data, model, min_obs = 1, output = "coefficients")
```

Arguments

data	A data frame containing the dependent variable and one or more independent variables.
model	A character that describes the model to be estimated (e.g., "ret_excess ~ mkt_excess + hml + smb").
min_obs	The minimum number of observations required to estimate the model. Defaults to 1.
output	A character vector specifying what to return. Must contain one or more of "coefficients" (default), "residuals", and "tstats". If a single value is provided, the corresponding object is returned directly. If multiple values are provided, a named list is returned.

Value

If output contains a single value: a data frame of coefficients or t-statistics, or a numeric vector of residuals. If output contains multiple values: a named list with the requested elements. Coefficients and t-statistics are returned as data frames with column names corresponding to the model terms. Residuals are returned as a numeric vector of length nrow(data) with NA for rows with missing data or insufficient observations.

See Also

Other estimation functions: estimate_betas(), estimate_fama_macbeth()

Examples

```
set.seed(42)
data <- data.frame(
  ret_excess = rnorm(100),
  mkt_excess = rnorm(100),
  smb = rnorm(100),
  hml = rnorm(100)
)

# Estimate model with a single independent variable
estimate_model(data, "ret_excess ~ mkt_excess")

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

# Estimate model without intercept
estimate_model(data, "ret_excess ~ mkt_excess - 1")

# Calculate residuals
estimate_model(data, "ret_excess ~ mkt_excess + smb + hml",
  output = "residuals"
```

```
)

# Return t-statistics
estimate_model(data, "ret_excess ~ mkt_excess + smb + hml",
  output = "tstats"
)

# Return coefficients, t-statistics, and residuals
estimate_model(data, "ret_excess ~ mkt_excess + smb + hml",
  output = c("coefficients", "tstats", "residuals")
)
```

`filter_options` *Create Filter Options*

Description

Creates a list of filter options of class `tidyfinance_filter_options` used for sample construction in TidyFinance-related functions. These options control which observations are retained before portfolio sorting.

Usage

```
filter_options(
  exclude_financials = FALSE,
  exclude_utilities = FALSE,
  min_stock_price = NULL,
  min_size_quantile = NULL,
  min_listing_age = NULL,
  exclude_negative_book_equity = FALSE,
  exclude_negative_earnings = FALSE,
  timing = "formation",
  ...
)
```

Arguments

`exclude_financials`
: A logical indicating whether to exclude financial firms (SIC codes 6000–6799). Defaults to `FALSE`.

`exclude_utilities`
: A logical indicating whether to exclude utility firms (SIC codes 4900–4999). Defaults to `FALSE`.

`min_stock_price`
: A single positive numeric specifying the minimum stock price required to include an observation. `NULL` (the default) applies no price filter.

`min_size_quantile`
: A single numeric strictly between 0 and 1 specifying the minimum cross-sectional size quantile (based on lagged market cap) required to include an observation. `NULL` (the default) applies no size quantile filter. The cutoff is computed from NYSE stocks only. This requires an `exchange` column in the data (as mapped via `data_options()`); an error is raised if it is missing.

`min_listing_age`
: A single non-negative integer or numeric specifying the minimum number of months a stock must have been listed in CRSP. `NULL` (the default) applies no listing age filter.

`exclude_negative_book_equity`
: A logical indicating whether to exclude observations with non-positive book equity. Defaults to `FALSE`.

`exclude_negative_earnings`
: A logical indicating whether to exclude observations with non-positive earnings. Defaults to `FALSE`.

`timing`
: A character string specifying when the screens apply if portfolios are rebalanced once a year (`rebalancing_month` in `filter_sorting_data()` or `implement_portfolio_sort()`). `"formation"` (the default) evaluates the screens in the formation month only and keeps every stock that passes them for the whole holding period (ex-ante screens). `"every_period"` evaluates them in every month, so a stock that fails a screen in a holding month loses that month's return (ex-post screens). Ex-post screens use information that is not available when the portfolios are formed, so the resulting factor returns can contain look-ahead bias; a warning is issued when they are used. With monthly rebalancing, every month is a formation month and both options are identical.

`...`
: Additional arguments to be included in the filter options list.

Value

A list of class `tidyfinance_filter_options` containing the specified filter options.

See Also

Other portfolio functions: `assign_portfolio()`, `breakpoint_options()`, `compute_breakpoints()`, `compute_long_short_returns()`, `compute_portfolio_returns()`, `data_options()`, `filter_sorting_data()`, `implement_portfolio_sort()`, `portfolio_sort_options()`

Examples

```
filter_options(
  exclude_financials = TRUE,
  exclude_utilities = TRUE,
  min_stock_price = 1,
  min_listing_age = 12
)

# Apply the screens in every month (ex-post), not only at formation
filter_options(min_stock_price = 5, timing = "every_period")
```

`filter_sorting_data` *Filter Sorting Data*

Description

Applies sample construction filters to a data frame before portfolio sorting. Filters are applied in a fixed order: financials exclusion, utilities exclusion, minimum stock price, minimum size quantile, minimum listing age, positive book equity, and positive earnings. An informational message is emitted for each filter that actually removes at least one observation.

Usage

```
filter_sorting_data(
  data,
  filter_options = NULL,
  data_options = NULL,
  rebalancing_month = NULL,
  quiet = FALSE
)
```

Arguments

`data`	A data frame containing the stock-level panel data to be filtered.
`filter_options`	A list of class `tidyfinance_filter_options` created by `filter_options()`. If `NULL` (the default), the defaults from `filter_options()` are used (i.e., no filters are applied). The arguments accepted by `filter_options()` include • `exclude_financials` A logical indicating whether to exclude financial firms (SIC codes 6000–6799). Defaults to `FALSE`. • `exclude_utilities` A logical indicating whether to exclude utility firms (SIC codes 4900–4999). Defaults to `FALSE`. • `min_stock_price` A single positive numeric specifying the minimum stock price required to include an observation. `NULL` (the default) applies no price filter. • `min_size_quantile` A single numeric strictly between 0 and 1 specifying the minimum cross-sectional size quantile (based on lagged market cap) required to include an observation. `NULL` (the default) applies no size quantile filter. The cutoff is computed from NYSE stocks only; the `exchange` column (mapped via `data_options()`) must be present in the data or an error is raised. • `min_listing_age` A single non-negative integer or numeric specifying the minimum number of months a stock must have been listed in CRSP. `NULL` (the default) applies no listing age filter. • `exclude_negative_book_equity` A logical indicating whether to exclude observations with non-positive book equity. Defaults to `FALSE`. • `exclude_negative_earnings` A logical indicating whether to exclude observations with non-positive earnings. Defaults to `FALSE`.

- `timing` Either `"formation"` (the default) to apply the screens only in the formation month when portfolios are rebalanced once a year, or `"every_period"` to apply them in every month, which can introduce look-ahead bias.

`data_options` A list of class `tidyfinance_data_options` (created via `data_options()`) specifying column name mappings. The `siccd` element is used to specify the SIC code column, `price` is used to specify the price column of the `min_stock_price` screen, `mktcap_lag` is used to specify the market capitalization column, `date` is used to specify the date column, `listing_age` is used to specify the listing age column, `be` is used to specify the book equity column, and `earnings` is used to specify the earnings column. Uses `data_options()` default if `NULL`: `"siccd"` = `"siccd"`, `"price"` = `"prc_lag"`, `"exchange"` = `"exchange"` `"mktcap_lag"` = `"mktcap_lag"`, `"date"` = `"date"`, `"listing_age"` = `"listing_age"`, `"be"` = `"be"`, and `"earnings"` = `"ib"`. With `rebalancing_month`, the `id` element (default `"permno"`) is used to keep a stock across its holding period.

`rebalancing_month` An optional integer between 1 and 12 specifying the month in which portfolios are formed when they are rebalanced once a year (see `compute_portfolio_returns()`). `NULL` (the default) means monthly rebalancing. Together with the `timing` element of `filter_options`, it determines when the screens apply; see Details.

`quiet` A logical indicating whether informational messages should be suppressed. Defaults to `FALSE`.

Details

Without `rebalancing_month`, every month is a formation month and each row is screened on its own values. The screens are then free of look-ahead bias only if the screened columns are known at the start of the month: the default price column `prc_lag` and `mktcap_lag` are lagged by one month, and accounting variables such as book equity and earnings should be lagged as well (e.g., with `join_lagged_values()`).

With `rebalancing_month`, portfolios are formed once a year and held for twelve months. By default (`filter_options(timing = "formation")`), each screen is evaluated in the formation month only, and a stock that passes all screens keeps all its rows until the next formation month, even if it fails a screen in a holding month. Rows of stocks that fail a screen at formation, and rows before the first formation month, are removed. With `timing = "every_period"`, the screens are evaluated in every month (ex-post) and a warning is issued, because dropping holding-month returns based on information from those months can bias factor returns.

The threshold and sign screens (`min_stock_price`, `min_size_quantile`, `min_listing_age`, `exclude_negative_book_eq` `exclude_negative_earnings`) drop an observation when the screened value is missing; the industry exclusions (`exclude_financials`, `exclude_utilities`) keep an observation whose SIC code is missing.

Value

The filtered data frame, preserving the class and structure of the input.

See Also

Other portfolio functions: assign_portfolio(), breakpoint_options(), compute_breakpoints(), compute_long_short_returns(), compute_portfolio_returns(), data_options(), filter_options(), implement_portfolio_sort(), portfolio_sort_options()

Examples

```
data <- data.frame(
  permno = 1:5,
  date = as.Date("2020-01-01"),
  siccd = c(6100, 2000, 4950, 3000, 6500),
  prc_lag = c(5, 0.5, 15, 20, 10)
)

data |>
  filter_sorting_data(
    filter_options = filter_options(
      exclude_financials = TRUE,
      min_stock_price = 1
    )
  )

# Annual rebalancing in July: the price screen is evaluated in July only,
# so stock 1 is kept in August although its price falls below 1
data_annual <- data.frame(
  permno = rep(1:2, each = 2),
  date = as.Date(rep(c("2020-07-01", "2020-08-01"), 2)),
  prc_lag = c(2, 0.5, 0.5, 2)
)

data_annual |>
  filter_sorting_data(
    filter_options = filter_options(min_stock_price = 1),
    rebalancing_month = 7
  )
```

---

get_available_huggingface_files

List Parquet Files in a Hugging Face Dataset

Description

Query the Hugging Face Datasets API and return a tibble of files with a `.parquet` suffix. The function follows pagination links returned in the response `Link` header and returns path, size, and a resolved URL.

Usage

```
get_available_huggingface_files(organization, dataset)
```

Arguments

organization	Character(1). Hugging Face organization or user name.
dataset	Character(1). Dataset name under the organization.

Details

Uses httr2 to perform HTTP requests. Requires internet access and the dataset to be publicly accessible or accessible with appropriate authentication.

Value

A tibble with columns: path (character), size (numeric), and url (character).

See Also

Other utility functions: create_summary_statistics(), list_supported_datasets(), list_supported_datasets_ff(
list_supported_datasets_ff_legacy(), list_supported_datasets_macro_predictors(), list_supported_datas
list_supported_datasets_pseudo(), list_supported_datasets_wrds(), list_supported_indexes(),
list_supported_jkp_factors(), list_tidy_finance_chapters(), open_tidy_finance_website(),
trim(), validate_dates(), winsorize()

Examples

```
## Not run:
get_available_huggingface_files("voigtstefan", "sp500")

## End(Not run)
```

---

get_wrds_connection *Establish a Connection to the WRDS Database*

Description

Establishes a connection to the Wharton Research Data Services (WRDS) database using the RPostgres package. It requires that the RPostgres package be installed and that valid WRDS credentials be set as environment variables.

Usage

```
get_wrds_connection(quiet = FALSE)
```

Arguments

quiet	A logical indicating whether to suppress the messages about the WRDS connection and the download progress. Defaults to FALSE.

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.

WRDS account setup

A WRDS username and password alone are not enough to connect. Before the first connection:

1. Get a WRDS account through an institution that subscribes to the data you need (e.g., CRSP and Compustat).
2. Enroll in two-factor authentication (Duo) for your WRDS account. WRDS requires it for all connections, including connections from R.
3. Log in to the WRDS website (https://wrds-www.wharton.upenn.edu) once and accept the terms of use, if WRDS asks you to.
4. Store your username and password with set_wrds_credentials(), or set the WRDS_USER and WRDS_PASSWORD environment variables, and restart R.
5. When you connect, approve the Duo push notification on your registered device. The connection waits until you approve it.

If the connection fails, the error message lists these steps. Unless quiet = TRUE, the function says when it starts connecting and when the connection succeeds, so you know when to approve the Duo request and when the download starts.

See Also

RPostgres::Postgres(), DBI::dbDisconnect() for more information on managing database connections.

Other WRDS functions: disconnect_connection(), download_data_wrds(), download_data_wrds_ccm_links(), download_data_wrds_compustat(), download_data_wrds_crsp(), download_data_wrds_fisd(), download_data_wrds_trace_enhanced(), set_wrds_credentials()

Examples

```
## Not run:
  # 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)

## End(Not run)
```

implement_portfolio_sort *Implement Portfolio Sort*

Description

A convenience wrapper that combines sample construction filtering and portfolio return computation into a single call. Equivalent to calling `filter_sorting_data()` followed by `compute_portfolio_returns()`.

Usage

```
implement_portfolio_sort(
  data,
  sorting_variables,
  sorting_method,
  rebalancing_month = NULL,
  holding_period = 1L,
  skip = 0L,
  portfolio_sort_options,
  breakpoint_function_main = compute_breakpoints,
  breakpoint_function_secondary = compute_breakpoints,
  min_portfolio_size = 1L,
  cap_weight = 0.8,
  cap_weight_exchanges = "NYSE",
  data_options = NULL,
  quiet = FALSE
)
```

Arguments

`data`	A data frame containing the stock-level panel data.
`sorting_variables`	A character vector of one or two column names to sort portfolios on.
`sorting_method`	A string specifying the sorting method: `"univariate"`, `"bivariate-dependent"`, or `"bivariate-independent"`.
`rebalancing_month`	An optional integer specifying the month in which portfolios are rebalanced annually. `NULL` (the default) means monthly rebalancing. Also determines the formation month in which the screens of `filter_options` are applied.
`holding_period, skip`	Staggered rebalancing: portfolios are formed every period and each cohort is held for `holding_period` periods, starting `skip` periods after formation. See `compute_portfolio_returns()`. With the default `filter_options(timing = "formation")`, the screens only decide which stocks enter the cohort formed in a period; a stock that fails a screen later stays in its earlier cohorts until their holding period ends.

portfolio_sort_options
: A list of class tidyfinance_portfolio_sort_options created by portfolio_sort_options(), bundling filter and breakpoint specifications. The arguments accepted by portfolio_sort_options() include

  - filter_options A list of class tidyfinance_filter_options created by filter_options(), or NULL (the default, which applies no filters). Options include exclude_financials, exclude_utilities, min_stock_price, min_size_quantile, min_listing_age, exclude_negative_book_equity, exclude_negative_earnings, and timing. With rebalancing_month, the screens apply in the formation month only by default; see filter_sorting_data().
  - breakpoint_options_main A list of class tidyfinance_breakpoint_options created by breakpoint_options(), specifying breakpoints for the primary sorting variable. Options include n_portfolios, percentiles, breakpoints_exchanges, smooth_bunching, and breakpoints_min_size_threshold.
  - breakpoint_options_secondary A list of class tidyfinance_breakpoint_options created by breakpoint_options(), specifying breakpoints for the secondary sorting variable, or NULL (the default) for univariate sorts. Options are the same as for breakpoint_options_main.

breakpoint_function_main
: The function used to compute breakpoints for the main sorting variable. Defaults to compute_breakpoints().

breakpoint_function_secondary
: The function used to compute breakpoints for the secondary sorting variable. Defaults to compute_breakpoints().

min_portfolio_size
: An integer specifying the minimum number of firms in the reported portfolio cross-section on a given date. Defaults to 1L (at least one observation per reported portfolio). For univariate sorts that is firms per portfolio-date; for bivariate sorts that is firms per main-portfolio-date summed across the secondary buckets. Cross-sections below the threshold have their returns set to NA. Set to 0L to deactivate the check. See compute_portfolio_returns().

cap_weight
: A numeric between 0 and 1 specifying the quantile at which portfolio weights are capped for the capped value-weighted return. Defaults to 0.8.

cap_weight_exchanges
: A character vector of exchanges whose stocks determine the cap, or NULL for all stocks. Defaults to "NYSE". See compute_portfolio_returns().

data_options
: A list of class tidyfinance_data_options (created via data_options()) specifying column name mappings. All elements are forwarded to filter_sorting_data() and compute_portfolio_returns(). Uses data_options() default if NULL: "id" = "permno", "date" = "date", "exchange" = "exchange", "mktcap_lag" = "mktcap_lag", "ret_excess" = "ret_excess", "siccd" = "siccd", "price" = "prc_lag", "listing_age" = "listing_age", "be" = "be", and "earnings" = "ib".

quiet
: A logical indicating whether informational messages should be suppressed. Defaults to FALSE.

Value

A data frame of portfolio returns as returned by compute_portfolio_returns().

See Also

Other portfolio functions: assign_portfolio(), breakpoint_options(), compute_breakpoints(), compute_long_short_returns(), compute_portfolio_returns(), data_options(), filter_options(), filter_sorting_data(), portfolio_sort_options()

Examples

```
set.seed(123)
data <- data.frame(
  permno = 1:500,
  date = rep(
    seq.Date(
      from = as.Date("2020-01-01"),
      by = "month",
      length.out = 100
    ),
    each = 10
  ),
  mktcap_lag = runif(500, 100, 1000),
  ret_excess = rnorm(500),
  prc_lag = runif(500, 0.5, 50),
  size = runif(500, 50, 150)
)
implement_portfolio_sort(
  data = data,
  sorting_variables = "size",
  sorting_method = "univariate",
  portfolio_sort_options = portfolio_sort_options(
    filter_options = filter_options(
      min_stock_price = 1
    ),
    breakpoint_options_main = breakpoint_options(n_portfolios = 5)
  )
)
```

---

join_lagged_values *Join Lagged Variable Values over a Date Range*

Description

Joins lagged values of selected variables from one dataset (new_data) into another (original_data), based on date ranges defined by min_lag and max_lag. Unlike add_lagged_columns(), this function supports joining across data frames with different date grids (e.g., monthly source data into quarterly target data).

Usage

```
join_lagged_values(
  original_data,
  new_data,
  id_keys,
  min_lag = NULL,
  max_lag = NULL,
  ff_adjustment = FALSE,
  data_options = NULL
)
```

Arguments

original_data	A data frame containing the target panel data.
new_data	A data frame containing the source variables to lag and merge. All columns besides id_keys and the date column will be lagged and joined.
id_keys	A character vector specifying the identifier column(s).
min_lag	A lubridate::Period specifying the lower lag bound (inclusive). Required unless ff_adjustment = TRUE.
max_lag	A lubridate::Period specifying the upper lag bound (inclusive). Required unless ff_adjustment = TRUE.
ff_adjustment	Logical; if TRUE, applies the Fama and French (1993) timing convention: the last observation per identifier and calendar year (the fiscal year-end) is used from July of the following year through June of the year after that, whatever the month of the fiscal year-end. A value for a fiscal year ending in December 2020 or in June 2020 is thus used from July 2021 to June 2022. min_lag and max_lag are ignored. Defaults to FALSE.
data_options	A list of class tidyfinance_data_options (created via data_options()) specifying column name mappings. The date element is used to identify the date column. Uses data_options() default if NULL: "date" = "date".

Details

For min_lag/max_lag expressed in whole months or years, the dates in both data frames should be the first of the month, as returned by the download_data() functions. On other days of the month the monthly offset can miss the intended match (for example, a source dated 30 April never matches a target dated 31 May); floor the dates with lubridate::floor_date(date, "month") beforehand. This does not apply to ff_adjustment = TRUE, which works on calendar years.

Value

A data frame with all columns from original_data plus the lagged columns from new_data (keeping their original names).

See Also

Other rolling and lagging functions: add_lagged_columns(), compute_rolling_value()

Examples

```
set.seed(42)
library(dplyr)
library(lubridate)

df1 <- tibble(
  id = rep(1:2, each = 6),
  date = rep(seq(as.Date("2020-01-01"), by = "month", length.out = 6), 2)
)

df2 <- df1 |>
  mutate(x = rnorm(n()))

join_lagged_values(
  original_data = df1,
  new_data = df2,
  id_keys = "id",
  min_lag = months(1),
  max_lag = months(3)
)

# Fama-French timing: annual values are used from July to June
annual <- tibble(
  id = 1:2,
  date = as.Date(c("2019-12-01", "2019-06-01")),
  x = c(10, 20)
)
monthly <- tibble(
  id = rep(1:2, each = 3),
  date = rep(as.Date(c("2020-06-01", "2020-07-01", "2021-06-01")), 2)
)
join_lagged_values(monthly, annual, id_keys = "id", ff_adjustment = TRUE)
```

---

list_supported_datasets

List All Supported Datasets

Description

Aggregates and returns a comprehensive tibble of all supported datasets 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).

`list_supported_types()` is a deprecated alias for `list_supported_datasets()`.

Usage

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

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

Arguments

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

Value

A tibble aggregating all supported datasets 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").

See Also

Other utility functions: create_summary_statistics(), get_available_huggingface_files(), list_supported_datasets_ff(), list_supported_datasets_ff_legacy(), list_supported_datasets_macro_pred
list_supported_datasets_other(), list_supported_datasets_pseudo(), list_supported_datasets_wrds(), list_supported_indexes(), list_supported_jkp_factors(), list_tidy_finance_chapters(), open_tidy_finance_website(), trim(), validate_dates(), winsorize()

Examples

```
# List all supported datasets as a data frame
list_supported_datasets()

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

# List supported datasets as a vector
list_supported_datasets(as_vector = TRUE)
```

---

list_supported_datasets_ff

List Supported Fama-French Datasets

Description

Returns a tibble with the supported Fama-French datasets, 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_datasets_ff()
```

Value

A tibble with columns: `type` (the type of dataset), `dataset_name` (a descriptive name of the dataset), `file_url` (the path of the source ZIP relative to Kenneth French's data library), and `domain` (the domain to which the dataset belongs, always "Fama-French").

See Also

Other utility functions: `create_summary_statistics()`, `get_available_huggingface_files()`, `list_supported_datasets()`, `list_supported_datasets_ff_legacy()`, `list_supported_datasets_macro_predicto`
`list_supported_datasets_other()`, `list_supported_datasets_pseudo()`, `list_supported_datasets_wrds()`, `list_supported_indexes()`, `list_supported_jkp_factors()`, `list_tidy_finance_chapters()`, `open_tidy_finance_website()`, `trim()`, `validate_dates()`, `winsorize()`

---

list_supported_datasets_ff_legacy

List Supported Legacy Fama-French Datasets

Description

Returns a tibble with the legacy names of initially supported Fama-French datasets, 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". Not included in the exported `list_supported_datasets()` function.

Usage

```
list_supported_datasets_ff_legacy()
```

Value

A tibble with columns: `type` (the type of dataset), `dataset_name` (a descriptive name of the dataset), `file_url` (the path of the source ZIP relative to Kenneth French's data library), and `domain` (the domain to which the dataset belongs, always "Fama-French").

See Also

Other utility functions: `create_summary_statistics()`, `get_available_huggingface_files()`, `list_supported_datasets()`, `list_supported_datasets_ff()`, `list_supported_datasets_macro_predictors()`, `list_supported_datasets_other()`, `list_supported_datasets_pseudo()`, `list_supported_datasets_wrds()`, `list_supported_indexes()`, `list_supported_jkp_factors()`, `list_tidy_finance_chapters()`, `open_tidy_finance_website()`, `trim()`, `validate_dates()`, `winsorize()`

list_supported_datasets_macro_predictors
List Supported Macro Predictor Datasets

Description

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

Usage

```
list_supported_datasets_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").

See Also

Other utility functions: `create_summary_statistics()`, `get_available_huggingface_files()`, `list_supported_datasets()`, `list_supported_datasets_ff()`, `list_supported_datasets_ff_legacy()`, `list_supported_datasets_other()`, `list_supported_datasets_pseudo()`, `list_supported_datasets_wrds()`, `list_supported_indexes()`, `list_supported_jkp_factors()`, `list_tidy_finance_chapters()`, `open_tidy_finance_website()`, `trim()`, `validate_dates()`, `winsorize()`

list_supported_datasets_other
List Supported Other Datasets

Description

Returns a tibble listing the supported other datasets and their corresponding dataset names.

Usage

```
list_supported_datasets_other()
```

Value

A tibble with columns: `type` (the type of dataset), `dataset_name` (the name of the data source), and `domain` (the domain to which the dataset belongs).

See Also

Other utility functions: create_summary_statistics(), get_available_huggingface_files(), list_supported_datasets(), list_supported_datasets_ff(), list_supported_datasets_ff_legacy(), list_supported_datasets_macro_predictors(), list_supported_datasets_pseudo(), list_supported_datasets_
list_supported_indexes(), list_supported_jkp_factors(), list_tidy_finance_chapters(), open_tidy_finance_website(), trim(), validate_dates(), winsorize()

`list_supported_datasets_pseudo`

List Supported Pseudo WRDS Datasets

Description

Returns a tibble with the supported pseudo WRDS datasets generated by `download_data(domain = "Pseudo Data", ...)`. Annotated with the domain "Pseudo Data".

Usage

```
list_supported_datasets_pseudo()
```

Value

A tibble with columns: `type`, `dataset_name`, and `domain`.

See Also

Other utility functions: create_summary_statistics(), get_available_huggingface_files(), list_supported_datasets(), list_supported_datasets_ff(), list_supported_datasets_ff_legacy(), list_supported_datasets_macro_predictors(), list_supported_datasets_other(), list_supported_datasets_
list_supported_indexes(), list_supported_jkp_factors(), list_tidy_finance_chapters(), open_tidy_finance_website(), trim(), validate_dates(), winsorize()

`list_supported_datasets_q`

List Supported Global Q Datasets

Description

Returns a tibble with the supported Global Q datasets, 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 2023. Additionally, it annotates each dataset with the domain "Global Q".

Usage

```
list_supported_datasets_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_datasets_wrds`

List Supported WRDS Datasets

Description

Returns a tibble with the supported datasets provided via WRDS. Additionally, it annotates each dataset with the domain "WRDS".

Usage

```
list_supported_datasets_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").

See Also

Other utility functions: `create_summary_statistics()`, `get_available_huggingface_files()`, `list_supported_datasets()`, `list_supported_datasets_ff()`, `list_supported_datasets_ff_legacy()`, `list_supported_datasets_macro_predictors()`, `list_supported_datasets_other()`, `list_supported_datasets_`
`list_supported_indexes()`, `list_supported_jkp_factors()`, `list_tidy_finance_chapters()`, `open_tidy_finance_website()`, `trim()`, `validate_dates()`, `winsorize()`

`list_supported_indexes`

List Supported Indexes

Description

Returns a tibble containing information about supported financial indexes. Each index is associated with a URL that points to a CSV file containing the holdings of the index. Additionally, each index has a corresponding `skip` value, which indicates the number of preamble lines in the CSV file. It is kept for compatibility; `download_data_constituents()` detects the header row automatically.

Usage

```
list_supported_indexes()
```

Value

A tibble with three columns:

index The name of the financial index (e.g., "DAX", "S&P 500").

url The URL to the CSV file containing the holdings data for the index.

skip The number of preamble lines in the CSV file. Kept for compatibility; the header row is detected automatically.

See Also

Other utility functions: `create_summary_statistics()`, `get_available_huggingface_files()`, `list_supported_datasets()`, `list_supported_datasets_ff()`, `list_supported_datasets_ff_legacy()`, `list_supported_datasets_macro_predictors()`, `list_supported_datasets_other()`, `list_supported_datasets_`
`list_supported_datasets_wrds()`, `list_supported_jkp_factors()`, `list_tidy_finance_chapters()`, `open_tidy_finance_website()`, `trim()`, `validate_dates()`, `winsorize()`

Examples

```
supported_indexes <- list_supported_indexes()
print(supported_indexes)
```

---

list_supported_jkp_factors

List Supported Global Factor Data Regions and Factors

Description

Queries the live availability manifest of Global Factor Data and returns the regions and selectors that can be passed to `download_data_jkp()`.

Usage

```
list_supported_jkp_factors(region = NULL, dataset = "factors")
```

Arguments

`region`	Optional. A region or country code. If provided, the function returns the selectors (factor codes, or industry classifications when `dataset = "industry"`) available for that region. If `NULL` (the default), it returns the available region codes.
`dataset`	The Global Factor Data product to query, one of `"factors"` (default), `"portfolios"`, or `"industry"`.

Value

A tibble. When `region` is `NULL`, it has a single `region` column listing the available region codes. When `region` is provided, it has a `region` column and a `factor` column listing the selectors (factor codes, or industry classifications when `dataset = "industry"`) available for that region.

See Also

Other utility functions: `create_summary_statistics()`, `get_available_huggingface_files()`, `list_supported_datasets()`, `list_supported_datasets_ff()`, `list_supported_datasets_ff_legacy()`, `list_supported_datasets_macro_predictors()`, `list_supported_datasets_other()`, `list_supported_datasets_p`
`list_supported_datasets_wrds()`, `list_supported_indexes()`, `list_tidy_finance_chapters()`, `open_tidy_finance_website()`, `trim()`, `validate_dates()`, `winsorize()`

Examples

```
list_supported_jkp_factors()
list_supported_jkp_factors("usa")
list_supported_jkp_factors("usa", dataset = "portfolios")
```

---

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. It 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, in the order of the website. These names correspond to the chapter pages at `https://www.tidy-finance.org/chapt`

See Also

Other utility functions: `create_summary_statistics()`, `get_available_huggingface_files()`, `list_supported_datasets()`, `list_supported_datasets_ff()`, `list_supported_datasets_ff_legacy()`, `list_supported_datasets_macro_predictors()`, `list_supported_datasets_other()`, `list_supported_datasets_p`
`list_supported_datasets_wrds()`, `list_supported_indexes()`, `list_supported_jkp_factors()`, `open_tidy_finance_website()`, `trim()`, `validate_dates()`, `winsorize()`

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 chapter overview of Tidy Finance. 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.

See Also

Other utility functions: `create_summary_statistics()`, `get_available_huggingface_files()`, `list_supported_datasets()`, `list_supported_datasets_ff()`, `list_supported_datasets_ff_legacy()`, `list_supported_datasets_macro_predictors()`, `list_supported_datasets_other()`, `list_supported_datasets_`
`list_supported_datasets_wrds()`, `list_supported_indexes()`, `list_supported_jkp_factors()`, `list_tidy_finance_chapters()`, `trim()`, `validate_dates()`, `winsorize()`

Examples

```
## Not run:
open_tidy_finance_website()
open_tidy_finance_website("beta-estimation")

## End(Not run)
```

portfolio_sort_options *Create Portfolio Sort Options*

Description

Creates a list of options of class `tidyfinance_portfolio_sort_options` that bundles sample construction filters and breakpoint specifications for use with `implement_portfolio_sort()`.

Usage

```
portfolio_sort_options(
  filter_options = NULL,
  breakpoint_options_main,
  breakpoint_options_secondary = NULL,
  ...
)
```

Arguments

`filter_options` A list of class `tidyfinance_filter_options` created by `filter_options()`, or `NULL` (the default, which applies no filters). The arguments accepted by `filter_options()` include

- `exclude_financials` A logical indicating whether to exclude financial firms (SIC codes 6000–6799). Defaults to `FALSE`.
- `exclude_utilities` A logical indicating whether to exclude utility firms (SIC codes 4900–4999). Defaults to `FALSE`.
- `min_stock_price` A single positive numeric specifying the minimum stock price required to include an observation. `NULL` (the default) applies no price filter.
- `min_size_quantile` A single numeric strictly between 0 and 1 specifying the minimum cross-sectional size quantile (based on lagged market cap) required to include an observation. `NULL` (the default) applies no size quantile filter.
- `min_listing_age` A single non-negative integer or numeric specifying the minimum number of months a stock must have been listed in CRSP. `NULL` (the default) applies no listing age filter.
- `exclude_negative_book_equity` A logical indicating whether to exclude observations with non-positive book equity. Defaults to `FALSE`.
- `exclude_negative_earnings` A logical indicating whether to exclude observations with non-positive earnings. Defaults to `FALSE`.
- `timing` Either `"formation"` (the default) to apply the screens only in the formation month when portfolios are rebalanced once a year, or `"every_period"` to apply them in every month, which can introduce look-ahead bias.

breakpoint_options_main
: A list of class tidyfinance_breakpoint_options created by breakpoint_options(), specifying breakpoints for the primary sorting variable. The arguments accepted by breakpoint_options() include

  - 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.
  - breakpoints_exchanges An optional character vector specifying exchange names to filter the data before computing breakpoints. Exchanges must be stored in a column given by data_options (defaults to exchange). If NULL, no filtering is applied.
  - smooth_bunching An optional logical parameter specifying if to attempt smoothing non-extreme portfolios if the sorting variable bunches on the extremes (TRUE), or not (FALSE, the default).
  - breakpoints_min_size_threshold An optional numeric value between 0 and 1 (exclusive). When set, stocks with market capitalization below this quantile are excluded from breakpoint computation.

breakpoint_options_secondary
: A list of class tidyfinance_breakpoint_options created by breakpoint_options(), specifying breakpoints for the secondary sorting variable, or NULL (the default) for univariate sorts. The arguments accepted by breakpoint_options() are the same as for breakpoint_options_main.

...
: Additional arguments to be included in the options list.

Value

A list of class tidyfinance_portfolio_sort_options containing the specified options.

See Also

Other portfolio functions: assign_portfolio(), breakpoint_options(), compute_breakpoints(), compute_long_short_returns(), compute_portfolio_returns(), data_options(), filter_options(), filter_sorting_data(), implement_portfolio_sort()

Examples

```
portfolio_sort_options(
  filter_options = filter_options(exclude_financials = TRUE),
  breakpoint_options_main = breakpoint_options(n_portfolios = 10)
)
```

`set_wrds_credentials` *Set WRDS Credentials*

Description

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()
```

Details

Storing credentials is not enough to connect to WRDS: you also need to enroll in two-factor authentication (Duo) and accept the WRDS terms of use. See the "WRDS account setup" section of `get_wrds_connection()`.

Value

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

See Also

Other WRDS functions: `disconnect_connection()`, `download_data_wrds()`, `download_data_wrds_ccm_links()`, `download_data_wrds_compustat()`, `download_data_wrds_crsp()`, `download_data_wrds_fisd()`, `download_data_wrds_trace_enhanced()`, `get_wrds_connection()`

Examples

```
## Not run:
set_wrds_credentials()

## End(Not run)
```

`trim` *Trim a Numeric Vector*

Description

Sets the values in a numeric vector that are beyond the specified quantiles to `NA`, based on the `cut` parameter. Values below the `cut` quantile and above the `1 - cut` quantile become `NA`; the length of `x` is unchanged, which is the useful behaviour inside `dplyr::mutate()`.

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 in [0, 0.5].

Value

A numeric vector the same length as `x`, with the values outside the `cut` and `1 - cut` quantiles set to `NA`.

See Also

Other utility functions: `create_summary_statistics()`, `get_available_huggingface_files()`, `list_supported_datasets()`, `list_supported_datasets_ff()`, `list_supported_datasets_ff_legacy()`, `list_supported_datasets_macro_predictors()`, `list_supported_datasets_other()`, `list_supported_datasets_`
`list_supported_datasets_wrds()`, `list_supported_indexes()`, `list_supported_jkp_factors()`, `list_tidy_finance_chapters()`, `open_tidy_finance_website()`, `validate_dates()`, `winsorize()`

Examples

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

validate_dates *Validate and Coerce Date Range Arguments*

Description

Checks that `start_date` and `end_date` are a valid pair, coerces them to `Date`, and handles the case where both are `NULL`. When both are `NULL` and `use_default_range = TRUE`, a default range spanning from two years ago to one year ago is applied and reported to the user.

Usage

```
validate_dates(start_date, end_date, use_default_range = FALSE)
```

Arguments

`start_date`	A scalar coercible to `Date` via `as.Date()`, or `NULL`.
`end_date`	A scalar coercible to `Date` via `as.Date()`, or `NULL`.
`use_default_range`	A logical scalar. If `TRUE` and both date arguments are `NULL`, a default range spanning from two years ago to one year ago is used instead of returning `NULL`. Defaults to `FALSE`.

Value

A named list with elements `start_date` and `end_date`, both of class `Date` (or `NULL` when no dates are provided and `use_default_range = FALSE`).

See Also

Other utility functions: `create_summary_statistics()`, `get_available_huggingface_files()`, `list_supported_datasets()`, `list_supported_datasets_ff()`, `list_supported_datasets_ff_legacy()`, `list_supported_datasets_macro_predictors()`, `list_supported_datasets_other()`, `list_supported_datasets_p`
`list_supported_datasets_wrds()`, `list_supported_indexes()`, `list_supported_jkp_factors()`, `list_tidy_finance_chapters()`, `open_tidy_finance_website()`, `trim()`, `winsorize()`

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 in [0, 0.5].

Value

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

See Also

Other utility functions: create_summary_statistics(), get_available_huggingface_files(), list_supported_datasets(), list_supported_datasets_ff(), list_supported_datasets_ff_legacy(), list_supported_datasets_macro_predictors(), list_supported_datasets_other(), list_supported_datasets_p list_supported_datasets_wrds(), list_supported_indexes(), list_supported_jkp_factors(), list_tidy_finance_chapters(), open_tidy_finance_website(), trim(), validate_dates()

Examples

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

Index

∗ **WRDS functions**
disconnect_connection, 21
download_data_wrds, 49
download_data_wrds_ccm_links, 51
download_data_wrds_compustat, 52
download_data_wrds_crsp, 53
download_data_wrds_fisd, 56
download_data_wrds_trace_enhanced, 57
get_wrds_connection, 71
set_wrds_credentials, 88
∗ **download functions**
download_data, 22
download_data_constituents, 24
download_data_damodaran, 25
download_data_factors_ff, 26
download_data_factors_q, 28
download_data_fred, 29
download_data_fred_md, 30
download_data_huggingface, 32
download_data_jkp, 35
download_data_macro_predictors, 37
download_data_osap, 39
download_data_pastor_stambaugh, 40
download_data_risk_free, 45
download_data_stambaugh_yuan, 47
download_data_stock_prices, 48
download_factor_library_grid, 58
download_factor_library_ids, 59
∗ **estimation functions**
estimate_betas, 60
estimate_fama_macbeth, 62
estimate_model, 64
∗ **portfolio functions**
assign_portfolio, 5
breakpoint_options, 7
compute_breakpoints, 8
compute_long_short_returns, 10
compute_portfolio_returns, 11
data_options, 19
filter_options, 66
filter_sorting_data, 68
implement_portfolio_sort, 73
portfolio_sort_options, 86
∗ **pseudo functions**
download_data_pseudo_ccm_links, 41
download_data_pseudo_compustat, 42
download_data_pseudo_crsp, 44
∗ **rolling and lagging functions**
add_lagged_columns, 3
compute_rolling_value, 15
join_lagged_values, 75
∗ **utility functions**
create_summary_statistics, 18
get_available_huggingface_files, 70
list_supported_datasets, 77
list_supported_datasets_ff, 78
list_supported_datasets_ff_legacy, 79
list_supported_datasets_macro_predictors, 80
list_supported_datasets_other, 80
list_supported_datasets_pseudo, 81
list_supported_datasets_wrds, 82
list_supported_indexes, 82
list_supported_jkp_factors, 83
list_tidy_finance_chapters, 84
open_tidy_finance_website, 85
trim, 89
validate_dates, 90
winsorize, 90

add_lagged_columns, 3
add_lagged_columns(), *16*, *75, 76*
assign_portfolio, 5
assign_portfolio(), *8, 9*, *11*, *15*, *21*, *67*, *70*, *75*, *87*

breakpoint_options, 7
breakpoint_options(), *5, 6*, *8, 9*, *11*, *13*, *15*, *21*, *67*, *70*, *74, 75*, *87*

compute_breakpoints, 8
compute_breakpoints(), *5, 6*, *8*, *11*, *13*, *15*, *21*, *67*, *70*, *74, 75*, *87*
compute_long_short_returns, 10
compute_long_short_returns(), *6*, *8, 9*, *15*, *21*, *67*, *70*, *75*, *87*
compute_portfolio_returns, 11
compute_portfolio_returns(), *6*, *8–11*, *21*, *67*, *69, 70*, *73–75*, *87*
compute_rolling_value, 15
compute_rolling_value(), *4*, *76*
create_summary_statistics, 18
create_summary_statistics(), *71*, *78–85*, *89–91*

data_options, 19
data_options(), *4–9*, *11*, *13–16*, *61*, *63*, *67–70*, *74–76*, *87*
DBI::dbDisconnect(), *72*
disconnect_connection, 21
disconnect_connection(), *50, 51*, *53*, *55, 56*, *58*, *72*, *88*
download_data, 22
download_data(), *25, 26*, *28–31*, *34*, *37, 38*, *40, 41*, *47–49*, *59, 60*
download_data_constituents, 24
download_data_constituents(), *23*, *26*, *28–31*, *34*, *37, 38*, *40, 41*, *47–49*, *59, 60*, *82*
download_data_damodaran, 25
download_data_damodaran(), *23*, *25*, *28–31*, *34*, *37, 38*, *40, 41*, *47–49*, *59, 60*
download_data_factors_ff, 26
download_data_factors_ff(), *23*, *25, 26*, *29–31*, *34*, *37, 38*, *40, 41*, *47–49*, *59, 60*
download_data_factors_q, 28
download_data_factors_q(), *23*, *25, 26*, *28*, *30, 31*, *34*, *37, 38*, *40, 41*, *47–49*, *59, 60*
download_data_fred, 29
download_data_fred(), *23*, *25, 26*, *28, 29*, *31*, *34*, *37, 38*, *40, 41*, *47–49*, *59, 60*
download_data_fred_md, 30
download_data_fred_md(), *23*, *25, 26*, *28–30*, *34*, *37, 38*, *40, 41*, *47–49*, *59, 60*
download_data_huggingface, 32
download_data_huggingface(), *23*, *25, 26*, *28–31*, *37, 38*, *40, 41*, *47–49*, *58–60*
download_data_jkp, 35
download_data_jkp(), *23*, *25, 26*, *28–31*, *34*, *38*, *40, 41*, *47–49*, *59, 60*, *83*
download_data_macro_predictors, 37
download_data_macro_predictors(), *23*, *25, 26*, *28–31*, *34*, *37*, *40, 41*, *47–49*, *59, 60*
download_data_osap, 39
download_data_osap(), *23*, *25, 26*, *28–31*, *34*, *37, 38*, *41*, *47–49*, *59, 60*
download_data_pastor_stambaugh, 40
download_data_pastor_stambaugh(), *23*, *25, 26*, *28–31*, *34*, *37, 38*, *40*, *47–49*, *59, 60*
download_data_pseudo_ccm_links, 41
download_data_pseudo_ccm_links(), *43–45*
download_data_pseudo_compustat, 42
download_data_pseudo_compustat(), *42*, *45*
download_data_pseudo_crsp, 44
download_data_pseudo_crsp(), *42, 43*
download_data_risk_free, 45
download_data_risk_free(), *23*, *25, 26*, *28–31*, *34*, *37, 38*, *40, 41*, *48, 49*, *59, 60*
download_data_stambaugh_yuan, 47
download_data_stambaugh_yuan(), *23*, *25, 26*, *28–31*, *34*, *37, 38*, *40, 41*, *47*, *49*, *59, 60*
download_data_stock_prices, 48
download_data_stock_prices(), *23*, *25, 26*, *28–31*, *34*, *37, 38*, *40, 41*, *47, 48*, *59, 60*
download_data_wrds, 49
download_data_wrds(), *21*, *51*, *53*, *55, 56*, *58*, *72*, *88*
download_data_wrds_ccm_links, 51
download_data_wrds_ccm_links(), *21*, *41, 42*, *50*, *53–56*, *58*, *72*, *88*
download_data_wrds_compustat, 52
download_data_wrds_compustat(), *21*, *42*,

43, *50, 51*, *55, 56*, *58*, *72*, *88*
download_data_wrds_crsp, 53
download_data_wrds_crsp(), *21*, *44, 45*, *50, 51*, *53*, *56*, *58*, *72*, *88*
download_data_wrds_fisd, 56
download_data_wrds_fisd(), *21*, *50, 51*, *53*, *55*, *58*, *72*, *88*
download_data_wrds_trace_enhanced, 57
download_data_wrds_trace_enhanced(), *21*, *50, 51*, *53*, *55, 56*, *72*, *88*
download_factor_library_grid, 58
download_factor_library_grid(), *23*, *25, 26*, *28–34*, *37, 38*, *40, 41*, *47–49*, *60*
download_factor_library_ids, 59
download_factor_library_ids(), *23*, *25, 26*, *28–31*, *34*, *37, 38*, *40, 41*, *47–49*, *58, 59*

estimate_betas, 60
estimate_betas(), *63*, *65*
estimate_fama_macbeth, 62
estimate_fama_macbeth(), *61*, *65*
estimate_model, 64
estimate_model(), *61*, *63*

filter_options, 66
filter_options(), *6*, *8, 9*, *11*, *15*, *21*, *68*, *70*, *74, 75*, *86, 87*
filter_sorting_data, 68
filter_sorting_data(), *6*, *8, 9*, *11*, *15*, *20, 21*, *67*, *73–75*, *87*

get_available_huggingface_files, 70
get_available_huggingface_files(), *19*, *78–85*, *89–91*
get_wrds_connection, 71
get_wrds_connection(), *21*, *50, 51*, *53*, *55, 56*, *58*, *88*

implement_portfolio_sort, 73
implement_portfolio_sort(), *6*, *8, 9*, *11*, *15*, *21*, *67*, *70*, *86, 87*

join_lagged_values, 75
join_lagged_values(), *4*, *16*, *69*

list_supported_datasets, 77
list_supported_datasets(), *19*, *22*, *71*, *79–85*, *89–91*
list_supported_datasets_ff, 78
list_supported_datasets_ff(), *19*, *71*, *78–85*, *89–91*
list_supported_datasets_ff_legacy, 79
list_supported_datasets_ff_legacy(), *19*, *71*, *78–85*, *89–91*
list_supported_datasets_macro_predictors, 80
list_supported_datasets_macro_predictors(), *19*, *71*, *78, 79*, *81–85*, *89–91*
list_supported_datasets_other, 80
list_supported_datasets_other(), *19*, *71*, *78–85*, *89–91*
list_supported_datasets_pseudo, 81
list_supported_datasets_pseudo(), *19*, *71*, *78–85*, *89–91*
list_supported_datasets_q, 81
list_supported_datasets_wrds, 82
list_supported_datasets_wrds(), *19*, *71*, *78–81*, *83–85*, *89–91*
list_supported_indexes, 82
list_supported_indexes(), *19*, *24*, *71*, *78–82*, *84, 85*, *89–91*
list_supported_jkp_factors, 83
list_supported_jkp_factors(), *19*, *71*, *78–85*, *89–91*
list_supported_types *(*list_supported_datasets*)*, 77
list_tidy_finance_chapters, 84
list_tidy_finance_chapters(), *19*, *71*, *78–85*, *89–91*

open_tidy_finance_website, 85
open_tidy_finance_website(), *19*, *71*, *78–84*, *89–91*

portfolio_sort_options, 86
portfolio_sort_options(), *6*, *8, 9*, *11*, *15*, *21*, *67*, *70*, *74, 75*

RPostgres::Postgres(), *72*

set_wrds_credentials, 88
set_wrds_credentials(), *21*, *50, 51*, *53*, *55, 56*, *58*, *72*
slider::slide_index_sum(), *60*

trim, 89
trim(), *19*, *71*, *78–85*, *90, 91*

validate_dates, 90

validate_dates(), *19*, *49*, *71*, *78–85*, *89*, *91*

winsorize, 90
winsorize(), *19*, *71*, *78–85*, *89, 90*