Package: fundsr (via r-universe)

May 27, 2026

Type Package

Title Rolling Differences (CAGR and Log), Survival and Other Financial Planning Plots

Version 0.5.0

Description A small tidyverse-based framework for importing and plotting UCITS ETF and index data, ETF liquidity measures, as well as survival curves and other plots related to financial planning.

License MIT + file LICENSE

Imports dplyr, ggplot2, glue, lifecycle, lubridate, magrittr, purrr, readr, readxl, rlang, scales, stats, stringr, tibble, tidyr, xml2

Encoding UTF-8

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Suggests curl, DiagrammeR, kableExtra, knitr, pkgdown, rmarkdown, svglite, testthat (>= 3.0.0), withr

VignetteBuilder knitr

URL https://stantraykov.github.io/fundsr/, https://github.com/StanTraykov/fundsr

BugReports https://github.com/StanTraykov/fundsr/issues

Config/testthat/edition 3

Config/pak/sysreqs libicu-dev libxml2-dev libx11-dev

Repository https://stantraykov.r-universe.dev

Date/Publication 2026-03-28 05:51:23 UTC

RemoteUrl https://github.com/StanTraykov/fundsr

RemoteRef v0.5.0

RemoteSha c13c9e33067ccc0bd9ae6d47b8538d3f6952661f

Contents

`add_data_loader` *Register a data loader*

Description

Appends `fun` to the internal data-loader registry (`session$state$data_loaders`). Registered functions are intended to be run sequentially in registration order.

Usage

```
add_data_loader(fun, session = NULL)
```

Arguments

fun	A function to register. Must take no arguments.
session	Optional fundsr_session object. Defaults to the package default session when NULL.

Details

If a loader with the same function body is already registered, fun is not added again.

Value

Invisibly returns the updated session$state$data_loaders list.

See Also

Other fund/index workflow functions: adjust_for_split(), build_all_series(), clear_data_loaders(), clear_storage(), get_storage(), import_fund(), join_env(), run_data_loaders(), store_timeseries()

Examples

```
add_data_loader(function() NULL)
```

---

add_fund_index_map *Add to fund-index map*

Description

Merges fund-index pairs into the session fund-index map (session$state$fund_index_map). Existing entries with the same names are replaced.

Usage

```
add_fund_index_map(fund_index_map, session = NULL)
```

Arguments

fund_index_map	Named character vector of fund-index pairs to merge into session$state$fund_index_map. Names are fund identifiers; values are index identifiers.
session	Optional fundsr_session object. Defaults to the package default session when NULL.

Value

Invisibly returns NULL. Called for side effects.

See Also

Other fund-index map functions: `clear_fund_index_map()`, `get_fund_index_map()`

`add_fund_urls` *Add entries to the fund download list*

Description

Adds one or more named download specifications to the `fundsr.fund_urls` option. Existing entries are preserved; entries in `fund_urls` replace any existing entries with the same name.

Usage

```
add_fund_urls(fund_urls)
```

Arguments

`fund_urls`	A named character vector mapping download identifiers to URLs.

Details

Names are converted to uppercase before storing.

Value

Invisibly returns a named list (as returned by `fundsr_options()`) containing the previous value of `fundsr.fund_urls`.

See Also

`fundsr_options()` to set `fundsr.fund_urls` and other fundsr options in one call. `download_fund_data()` to download files from the added URLs.

Other download functions: `download_fund_data()`

adjust_for_split *Adjust a time series for a stock split*

Description

Adjusts values in a numeric column for observations strictly before a given split date by dividing them by the supplied split ratio.

Usage

```
adjust_for_split(data, split_date, split_ratio, value_col, date_col = "date")
```

Arguments

data	A data frame.
split_date	A single date coercible via lubridate::as_date().
split_ratio	A positive numeric scalar giving the split ratio.
value_col	String. Name of the numeric column to adjust.
date_col	String. Name of the date column in data.

Details

The function parses split_date and data[[date_col]] with lubridate::as_date(). Rows with missing dates are left unchanged. Rows with unparseable non-missing dates trigger an error.

For rows where the parsed date is strictly earlier than split_date, the values in value_col are divided by split_ratio.

Value

A data frame with the same columns as data, where value_col has been adjusted for rows with dates strictly before split_date.

See Also

Other fund/index workflow functions: add_data_loader(), build_all_series(), clear_data_loaders(), clear_storage(), get_storage(), import_fund(), join_env(), run_data_loaders(), store_timeseries()

Examples

```
df <- data.frame(
  date = c("2024-01-01", "2024-01-02", "2024-01-03"),
  price = c(300, 330, 120)
)

adjust_for_split(
  data = df,
  split_date = "2024-01-03",
```

```
  split_ratio = 3,
  value_col = "price"
)
```

`build_all_series` *Run all registered data loaders and join all loaded series into a big tibble.*

Description

Runs `run_data_loaders()`, joins the resulting environment into a single data frame via `join_env()`, and sorts the result by by.

Usage

```
build_all_series(reload = FALSE, by = "date", session = NULL, ...)
```

Arguments

`reload`	Logical; if `TRUE`, forces a full reload by temporarily setting `options(fundsr.reload = TRUE)`.
`by`	Character vector of column names to join by and sort by.
`session`	Optional `fundsr_session` object. Defaults to the package default session when `NULL`.
`...`	Additional arguments forwarded to `join_env()` (e.g. `late`, `join_precedence`, etc.).

Details

This function is a convenience wrapper for the most common workflow.

Value

A tibble containing all joined series, sorted by by.

See Also

Other fund/index workflow functions: `add_data_loader()`, `adjust_for_split()`, `clear_data_loaders()`, `clear_storage()`, `get_storage()`, `import_fund()`, `join_env()`, `run_data_loaders()`, `store_timeseries()`

Examples

```
## Not run:

s1 <- build_all_series()
download_fund_data(redownload = TRUE)
s2 <- build_all_series(by = "date", late = "ftaw", join_precedence = c(".y", ".x")) %>%
  filter(date >= as_date("2013-01-01"))

## End(Not run)
```

`chance_alive` *Compute conditional survival (chance alive) by age*

Description

Computes the conditional probability of being alive at each age `x` >= `age0`, given survival to `age0`, from an HMD-style period life table. For each year, the returned series is: `chance_alive(x | age0) = lx(x) / lx(age0)`.

Usage

```
chance_alive(lt, pop_name, age0)
```

Arguments

`lt`	A life table tibble as returned by `read_life_table()`, containing at least `PopName`, `Year`, `Age`, and `lx`.
`pop_name`	Population code (HMD `PopName`) to filter within `lt` (e.g. `"BGR"`, `"USA"`, `"DEUTNP"`).
`age0`	Baseline age (integer). Returned ages start at `age0`.

Value

A tibble with columns `Year`, `Age`, and `chance_alive`, sorted by `Year` then `Age`.

See Also

Other survival curve functions: `chance_alive_es_aasmr()`, `plot_chance_alive()`, `plot_chance_alive_es_aasmr()`, `read_es_aasmr()`, `read_life_table()`

Examples

```
## Not run:
lt_m <- read_life_table(file.path("data", "life"), sex = "m", look_back = 10)
ca <- chance_alive(lt_m, pop_name = "BGR", age0 = 27)
ca

## End(Not run)
```

chance_alive_es_aasmr *Compute cohort-style survival from Eurostat EUROPOP2023 mortality assumptions*

Description

Computes conditional survival (chance alive) for a person aged `age0` in `start_year` using Eurostat EUROPOP2023 age-specific mortality rate assumptions (dataset `proj_23naasmr`). The computation follows a cohort path (diagonal): for age `age0 + k` it uses the mortality rate for year `start_year + k`.

Usage

```
chance_alive_es_aasmr(es, geo, sex, age0, start_year = NULL)
```

Arguments

`es`	A tibble as returned by `read_es_aasmr()`.
`geo`	Eurostat geo code (e.g. `"BG"`, `"NL"`).
`sex`	Sex code: `"m"` or `"f"` (case-insensitive).
`age0`	Baseline age (integer).
`start_year`	Starting calendar year (integer). If `NULL`, uses the earliest year available for the selected `geo/sex`.

Details

The result includes two projection variants: baseline (`BSL`) and lower mortality (`LMRT`).

Value

A tibble with columns `geo`, `sex`, `projection`, `Year`, `Age`, `mx`, `qx`, `chance_alive`.

See Also

Other survival curve functions: `chance_alive()`, `plot_chance_alive()`, `plot_chance_alive_es_aasmr()`, `read_es_aasmr()`, `read_life_table()`

Examples

```
## Not run:
es <- read_es_aasmr(file.path("data", "life"))
ca <- chance_alive_es_aasmr(es, geo = "NL", sex = "m", age0 = 42, start_year = 2022)
p <- plot_chance_alive_es_aasmr(ca, sex = "m", population = "NL")
p

## End(Not run)
```

clear_data_loaders *Clear registered data loaders*

Description

Clears the internal data-loader registry (`session$state$data_loaders`), removing all previously registered data loader functions.

Usage

```
clear_data_loaders(session = NULL)
```

Arguments

`session`	Optional `fundsr_session` object. Defaults to the package default session when `NULL`.

Value

Invisibly returns `NULL`. Called for side effects.

See Also

Other fund/index workflow functions: `add_data_loader()`, `adjust_for_split()`, `build_all_series()`, `clear_storage()`, `get_storage()`, `import_fund()`, `join_env()`, `run_data_loaders()`, `store_timeseries()`

Examples

```
clear_data_loaders()
```

clear_fund_index_map *Clear fund-index map*

Description

Clears the fund-index map stored in `session$state$fund_index_map`.

Usage

```
clear_fund_index_map(session = NULL)
```

Arguments

`session`	Optional `fundsr_session` object. Defaults to the package default session when `NULL`.

Value

Invisibly returns NULL. Called for side effects.

See Also

Other fund-index map functions: add_fund_index_map(), get_fund_index_map()

---

clear_inkscape_queue *Clear queued Inkscape exports*

Description

Clears the internal Inkscape export queue (session$state$inkscape_queue), removing all queued export commands.

Usage

```
clear_inkscape_queue(session = NULL)
```

Arguments

session	Optional fundsr_session object. Defaults to the package default session when NULL.

Value

Invisibly returns NULL. Called for side effects.

See Also

Other plot export utilities: export_pngs(), run_plots(), save_plot()

Examples

```
clear_inkscape_queue()
```

clear_storage *Clear storage*

Description

Removes all objects from the package's storage environment (`session$storage`). Optionally also clears the fund-index map (`session$state$fund_index_map`).

Usage

```
clear_storage(clear_map = FALSE, session = NULL)
```

Arguments

`clear_map`	Logical scalar; if `TRUE`, also clears `session$state$fund_index_map`.
`session`	Optional `fundsr_session` object. Defaults to the package default session when `NULL`.

Value

Invisibly returns `NULL`. Called for side effects.

See Also

Other fund/index workflow functions: `add_data_loader()`, `adjust_for_split()`, `build_all_series()`, `clear_data_loaders()`, `get_storage()`, `import_fund()`, `join_env()`, `run_data_loaders()`, `store_timeseries()`

Examples

```
clear_storage()
clear_storage(clear_map = TRUE)
```

---

download_fund_data *Download fund data according to the configured download list*

Description

Retrieves all Excel files with fund identifiers and URLs listed in the `fundsr.fund_urls` option and saves them into the directory specified by the `fundsr.data_dir` option.

Usage

```
download_fund_data(redownload = FALSE)
```

Arguments

redownload	Logical; if TRUE, existing files are overwritten. If FALSE, only missing files are downloaded.

Value

Invisibly returns NULL. Files are written as a side effect.

See Also

add_fund_urls() to add/update entries in fundsr.fund_urls.

Other download functions: add_fund_urls()

---

export_pngs *Export queued SVGs to PNG via Inkscape*

Description

Runs all pending Inkscape export actions stored in the internal queue, invoking Inkscape to produce PNG files from previously saved SVGs.

Usage

```
export_pngs(background = "white", session = NULL)
```

Arguments

background	Background color for PNG export, passed to Inkscape as an export-background:{...} action. If NULL, no background action is added. Defaults to "white".
session	Optional fundsr_session object. Defaults to the package default session when NULL.

Details

This function reads queued Inkscape actions from session$state$inkscape_queue, optionally prepends an export-background:{background} action, and executes Inkscape using base::system2().

The Inkscape executable is searched for in the following order:

1. user-supplied config (fundsr.inkscape option or INKSCAPE environment variable)
2. an inkscape executable available on the system PATH
3. common installation locations (Windows, macOS, Linux, etc.)

On success, the internal queue is cleared. On failure, the queue is left intact and a message is printed.

Value

The exit status returned by Inkscape (0 indicates success). Invisibly returns `NULL` if the queue is empty or if Inkscape cannot be located.

See Also

`clear_inkscape_queue()`, `save_plot()`

Other plot export utilities: `clear_inkscape_queue()`, `run_plots()`, `save_plot()`

`fundsr_default_session` *Get the default fundsr session*

Description

Returns the package-global default `fundsr_session` object.

Usage

```
fundsr_default_session()
```

Value

An object of class `"fundsr_session"`.

See Also

Other config functions: `fundsr_options()`, `fundsr_session()`, `reset_state()`

`fundsr_example_data` *Get the path to an example file shipped with the package.*

Description

Get the path to an example file shipped with the package.

Usage

```
fundsr_example_data(file = ".")
```

Arguments

`file`	The name of the example file.

Examples

```
fundsr_example_data("FNDA.xlsx")
fundsr_example_data()
```

`fundsr_options` *Set fundsr package options*

Description

Convenience wrapper around `options()` for setting common `fundsr.*` options.

Usage

```
fundsr_options(
  data_dir = NULL,
  out_dir = NULL,
  px_width = NULL,
  internal_png = NULL,
  export_svg = NULL,
  ticker_map = NULL,
  inkscape = NULL,
  reload = NULL,
  fund_urls = NULL,
  verbosity = NULL
)
```

Arguments

`data_dir`	Directory containing fund data files (sets `fundsr.data_dir`). Note: fundsr may try to write to this directory (see `download_fund_data()`).
`out_dir`	Output directory for plots/exports (sets `fundsr.out_dir`).
`px_width`	Default PNG export width in pixels (sets `fundsr.px_width`).
`internal_png`	Logical; whether to save an internal PNG immediately when exporting plots (sets `fundsr.internal_png`).
`export_svg`	Logical; whether to save SVGs and queue Inkscape exports (sets `fundsr.export_svg`).
`ticker_map`	Named character vector mapping "primary" tickers (series table column names) to translated tickers (sets `fundsr.ticker_map`).
`inkscape`	Inkscape executable (path or command name) used by export helpers (sets `fundsr.inkscape`).
`reload`	Logical; default value for forcing re-import of cached objects (sets `fundsr.reload`).
`fund_urls`	Named character vector or named list of URLs for fund data downloads (sets `fundsr.fund_urls`).
`verbosity`	Integer verbosity level (sets `fundsr.verbosity`). Use 0 to silence informational messages, 2 to emit more detailed progress/diagnostic messages, and 4 to force all message types even when they are disabled via function arguments (e.g. `messages` for `roll_diffs()`).

Details

All arguments default to `NULL`. If an argument is left as `NULL`, the corresponding `fundsr.*` option is left unchanged.

Value

Invisibly returns a named list of the previous values of the options that were changed (as returned by `options()`).

See Also

`add_fund_urls()` to add/update entries in `fundsr.fund_urls`.

Other config functions: `fundsr_default_session()`, `fundsr_session()`, `reset_state()`

Examples

```
fundsr_options(verbosity = 4)
fundsr_options(
    data_dir = file.path("data", "funds"),
    out_dir = "output",
    px_width = 1300,
    ticker_map = c(
        ticker1 = "ticker2",
        ticker3 = "ticker4"
    )
)
```

`fundsr_session` *Create a fundsr session*

Description

Constructs a `fundsr_session` object, which encapsulates a mutable state environment and a storage environment.

Usage

```
fundsr_session(
  state = new.env(parent = emptyenv()),
  storage = new.env(parent = emptyenv())
)
```

Arguments

`state`	Environment for mutable fundsr state (fund-index map, loader registry, export queues, etc.).
`storage`	Environment for cached series storage.

Value

An object of class `"fundsr_session"`.

See Also

Other config functions: fundsr_default_session(), fundsr_options(), reset_state()

`get_fund_index_map` *Get the internal fund index map*

Description

Returns the package's fund index lookup table stored in `session$state$fund_index_map`.

Usage

```
get_fund_index_map(session = NULL)
```

Arguments

`session`	Optional `fundsr_session` object. Defaults to the package default session when `NULL`.

Value

A named character vector representing the internal fund index mapping.

See Also

Other fund-index map functions: add_fund_index_map(), clear_fund_index_map()

`get_storage` *Get the internal fund storage*

Description

Returns the fundsr's fund storage environment (`session$storage`).

Usage

```
get_storage(session = NULL)
```

Arguments

`session`	Optional `fundsr_session` object. Defaults to the package default session when `NULL`.

Value

The storage environment.

See Also

Other fund/index workflow functions: add_data_loader(), adjust_for_split(), build_all_series(), clear_data_loaders(), clear_storage(), import_fund(), join_env(), run_data_loaders(), store_timeseries()

`import_fund` *Import a fund's NAV data and optionally register its benchmark mapping*

Description

Imports a fund's NAV time series from an Excel file and stores it in the storage environment via `store_timeseries()`. Optionally, a benchmark column can also be imported, and a fund/index mapping is recorded in `session$state$fund_index_map`.

Usage

```
import_fund(
  ticker,
  file = NULL,
  sheet = 1,
  date_col = "^Date",
  nav_col = "^NAV",
  benchmark = NULL,
  benchmark_col = NULL,
  retrieve_benchmark = FALSE,
  date_order = "dmy",
  var_name = NULL,
  data_sheet = deprecated(),
  ...
)
```

Arguments

`ticker`	Fund ticker symbol. Used (in lower case) as the storage key and (in upper case) to derive the default filename.
`file`	Optional filename. If `NULL` (the default), it is inferred from `ticker` as described above.
`sheet`	Sheet index or name containing the NAV data.
`date_col`	Regular expression identifying the date column.
`nav_col`	Regular expression identifying the fund's NAV column.
`benchmark`	Optional benchmark key that this fund should be associated with in the fund/index map. When `retrieve_benchmark = TRUE`, the same value is also used as the name under which the benchmark series is imported.

benchmark_col
: Regular expression identifying the benchmark column in the Excel sheet. Only used when retrieve_benchmark = TRUE.

retrieve_benchmark
: Logical; if TRUE, both benchmark and benchmark_col must be supplied and the benchmark column is imported alongside the fund.

date_order
: Date parsing order passed to the importer.

var_name
: Specify a custom variable name for the storage environment.

data_sheet
: Deprecated; use sheet.

...
: Arguments passed on to store_timeseries

  overwrite Logical scalar. If TRUE, recompute and replace any existing cached value, regardless of fundsr.reload.

  postprocess Function applied to the computed value before caching. Only used when the value is (re)computed (i.e. not applied when a cached value is reused). Defaults to base::identity().

  session Optional fundsr_session object. Defaults to the package default session when NULL.

Details

If file is NULL, the function searches fundsr.data_dir for exactly one of paste0(toupper(ticker), ".xlsx") or paste0(toupper(ticker), ".xls").

The function builds a column-translation mapping from the fund NAV column and, if requested, a benchmark column. It then calls read_timeseries_excel() to read the Excel file and store_timeseries() to cache the imported object under var_name, if supplied, otherwise tolower(ticker). When benchmark is provided, a corresponding entry is added to session$state$fund_index_map to link the fund to its benchmark key.

Value

Invisibly returns NULL. The imported data are stored in session$storage under tolower(ticker). A fund/index mapping is recorded in session$state$fund_index_map when benchmark is supplied.

See Also

store_timeseries(), read_timeseries_excel()

Other fund/index workflow functions: add_data_loader(), adjust_for_split(), build_all_series(), clear_data_loaders(), clear_storage(), get_storage(), join_env(), run_data_loaders(), store_timeseries()

Examples

```
fundsr_options(data_dir = fundsr_example_data(), verbosity = 2)

import_fund("FNDA",
          "FNDA.xlsx",
          benchmark = "IDX1",
```

```
            sheet = "historical",
            date_col = "^As Of",
            nav_col = "^NAV")

import_fund("FNDB",
            benchmark = "IDX1",
            date_col = "^date",
            nav_col = "^net asset val",
            date_order = "mdy")
```

`join_env` *Join all data frames in an environment with optional late joins*

Description

Performs a `dplyr::full_join()` across all objects in `env` (in alphabetical order), excluding any listed in `late`. Late objects are then joined sequentially (via `dplyr::left_join()` by default) in the order given. Full-join clashes use suffixes `c(".x", ".y")`; late joins use `c(".early", ".late")`.

Usage

```
join_env(
  env,
  by = "date",
  late = NULL,
  join_precedence = NULL,
  coalesce_suffixed = deprecated(),
  late_join = dplyr::left_join
)
```

Arguments

`env`	Environment containing *only* data frames (incl. tibbles) to join.
`by`	Character vector of join keys (passed to `dplyr::full_join()`).
`late`	Character vector of object names in `env` that should be *excluded* from the initial full join and instead left-joined afterwards. Defaults to `NULL`.
`join_precedence`	Optional character vector of length 2 giving join suffixes to coalesce (for example, `c(".early", ".late")` or `c(".late", ".early")`). When non-`NULL`, any pairs of columns whose names end in these suffixes (and share the same base name) are replaced by a single unsuffixed column containing the coalesced values, preferring the left suffix when both values are available. If `NULL` (the default), no automatic coalescing is performed.
`coalesce_suffixed`	Deprecated; use `join_precedence` (same meaning).
`late_join`	Function to use for joining late objects, e.g. `dplyr::left_join` (the default). Must accept dplyr-style `suffix` and `by` arguments.

Details

Optionally, column pairs with specified suffixes can be coalesced into unsuffixed base columns via `join_precedence`.

Value

A tibble: the full join of all non-late objects, followed by sequential left-joins (or other joins specified by `late_join`) of the late objects. If `join_precedence` is supplied, suffixed join columns are coalesced into unsuffixed base columns as described above.

See Also

Other fund/index workflow functions: `add_data_loader()`, `adjust_for_split()`, `build_all_series()`, `clear_data_loaders()`, `clear_storage()`, `get_storage()`, `import_fund()`, `run_data_loaders()`, `store_timeseries()`

Examples

```
e <- new.env()
e$members <- dplyr::band_members
e$instruments <- dplyr::band_instruments
e$other_instr <- dplyr::band_instruments |>
    dplyr::mutate(plays = dplyr::case_match(name,
                                            "John" ~ "banjo",
                                            "Paul" ~ "mellotron",
                                            "Keith" ~ "harpsichord")) |>
    dplyr::add_row(name = "Mick", plays = "harmonica") |>
    dplyr::add_row(name = "Stu", plays = "piano")

full <- join_env(e, by = "name")
late <- join_env(e, by = "name", late = "other_instr")
late_coalesced <- join_env(e,
                           by = "name",
                           late = "other_instr",
                           join_precedence = c(".early", ".late"))
print(list(full = full, late = late, late_coalesced = late_coalesced))
```

---

load_fund *Deprecated alias for* `import_fund()`.

Description

`load_fund()` has been renamed to `import_fund()`.

Usage

```
load_fund(
  ticker,
  file = NULL,
  sheet = 1,
  date_col = "^Date",
  nav_col = "^NAV",
  benchmark = NULL,
  benchmark_col = NULL,
  retrieve_benchmark = FALSE,
  date_order = "dmy",
  var_name = NULL,
  data_sheet = lifecycle::deprecated(),
  ...
)
```

Arguments

ticker	Fund ticker symbol. Used (in lower case) as the storage key and (in upper case) to derive the default filename.
file	Optional filename. If NULL (the default), it is inferred from ticker as described above.
sheet	Sheet index or name containing the NAV data.
date_col	Regular expression identifying the date column.
nav_col	Regular expression identifying the fund's NAV column.
benchmark	Optional benchmark key that this fund should be associated with in the fund/index map. When retrieve_benchmark = TRUE, the same value is also used as the name under which the benchmark series is imported.
benchmark_col	Regular expression identifying the benchmark column in the Excel sheet. Only used when retrieve_benchmark = TRUE.
retrieve_benchmark	Logical; if TRUE, both benchmark and benchmark_col must be supplied and the benchmark column is imported alongside the fund.
date_order	Date parsing order passed to the importer.
var_name	Specify a custom variable name for the storage environment.
data_sheet	Deprecated; use sheet.
...	Arguments passed on to import_fund

Value

Invisibly returns NULL. The imported data are stored in session$storage under tolower(ticker). A fund/index mapping is recorded in session$state$fund_index_map when benchmark is supplied.

`plot_chance_alive` *Plot chance alive by age*

Description

Plots conditional survival curves produced by `chance_alive()`. The most recent year is highlighted in black; earlier years are shown with a teal-to-orange-to-light gradient. Horizontal reference lines mark 10% and 5% survival levels.

Usage

```
plot_chance_alive(ca, sex = c("m", "f"), population)
```

Arguments

`ca`	A tibble as returned by `chance_alive()`, with columns `Year`, `Age`, and `chance_alive`.
`sex`	Sex code: `"m"` (male) or `"f"` (female). Used for labeling.
`population`	Population label/code to display in the subtitle (e.g. `"BGR"`, `"USA"`, `"DEUTNP"`).

Value

A ggplot object.

See Also

Other survival curve functions: `chance_alive()`, `chance_alive_es_aasmr()`, `plot_chance_alive_es_aasmr()`, `read_es_aasmr()`, `read_life_table()`

Examples

```
## Not run:
lt_m <- read_life_table(file.path("data", "life"), sex = "m", look_back = 10)
ca <- chance_alive(lt_m, pop_name = "BGR", age0 = 27)
p <- plot_chance_alive(ca, sex = "m", population = "BGR")
p

## End(Not run)
```

plot_chance_alive_es_aasmr *Plot cohort-style survival from Eurostat EUROPOP2023 assumptions*

Description

Plots the conditional survival curves returned by `chance_alive_es_aasmr()`. The baseline projection (`BSL`) is shown in black and the lower-mortality variant (`LMRT`) is shown in dark cyan.

Usage

```
plot_chance_alive_es_aasmr(ca, sex, population)
```

Arguments

`ca`	A tibble as returned by `chance_alive_es_aasmr()`, with columns `projection`, `Age`, and `chance_alive` (and typically `geo`, `sex`, `Year`).
`sex`	Sex code: `"m"` or `"f"` (case-insensitive). Used for labeling.
`population`	Population label/code to display in the subtitle (e.g. `"BG"`).

Value

A ggplot object.

See Also

Other survival curve functions: `chance_alive()`, `chance_alive_es_aasmr()`, `plot_chance_alive()`, `read_es_aasmr()`, `read_life_table()`

Examples

```
## Not run:
es_aasmr <- read_es_aasmr(file.path("data", "life"))
ca <- chance_alive_es_aasmr(es_aasmr, geo = "BG", sex = "m", age0 = 42, start_year = 2022)
p <- plot_chance_alive_es_aasmr(ca, sex = "m", population = "BG")
p

## End(Not run)
```

`plot_roll_diffs` *Plot rolling return differences against benchmark*

Description

Plots rolling annualized tracking differences (CAGR-style or log-return) for selected funds against their benchmark series. The plot uses quantile-based y-limits and formats the y-axis in basis points.

Usage

```
plot_roll_diffs(
  data,
  n_days,
  funds,
  use_log = FALSE,
  gg_params = NULL,
  title_add = NULL,
  date_brk = NULL,
  qprob = c(0.005, 0.995),
  bmark_type = c("net", "gross")
)
```

Arguments

`data`	Input data frame containing a `date` column and one rolling-difference column per fund (specified by `funds`).
`n_days`	Window length (in days) used to compute rolling differences (used for labelling).
`funds`	Character vector of fund column names to include.
`use_log`	Logical; if `TRUE`, labels the plot as log-return differences. If `FALSE`, labels the plot as CAGR differences.
`gg_params`	Optional ggplot components to add to the plot.
`title_add`	Optional title suffix. Can be a single string or a named character vector specifying the title in multiple languages (e.g. `c(en=..., bg=...)`).
`date_brk`	Optional date-break specification for the x-axis (e.g. `"3 months"`). If `NULL`, it is chosen automatically based on the time span and available data.
`qprob`	Two-element numeric vector giving lower and upper quantiles used to set the baseline y-axis limits. Defaults to `c(0.005, 0.995)`.
`bmark_type`	Benchmark type used in the title: `"net"` or `"gross"`.

Details

The function reshapes `data` to long format and produces a scatter plot coloured by fund. The y-axis limits are primarily determined from quantiles of the rolling differences (as specified by `qprob`), always including 0.

To avoid clipping recent extremes, the y-limits are expanded (if needed) to also include the full range observed in the most recent 30 days of data, even when those values fall outside the qprob quantiles.

The x-axis breaks are chosen as follows when date_brk is NULL: for spans of up to 3 years, breaks default to "3 months". For longer spans, breaks are anchored to calendar months (semiannual or quarterly depending on span) and are included only when the data range extends beyond the midpoint to the neighboring break.

Value

A ggplot object.

See Also

Other rolling difference functions: roll_diffs()

---

read_es_aasmr *Read Eurostat EUROPOP2023 mortality-assumption table (proj_23naasmr)*

Description

Reads the Eurostat TSV export for EUROPOP2023 age-specific mortality rate assumptions (dataset proj_23naasmr), typically downloaded as estat_proj_23naasmr.tsv.gz. Returns a tidy long table with metadata columns plus numeric Age, Year, and mx.

Usage

```
read_es_aasmr(directory)
```

Arguments

directory	Directory containing estat_proj_23naasmr.tsv.gz.

Details

Eurostat value flags (e.g. provisional/estimated markers) are tolerated: the numeric part is parsed into mx, while missing values encoded as : are returned as NA.

The Eurostat age dimension uses codes like Y_LT1 (age < 1), Y1 (age 1), and Y_GE85 (age 85+). This function maps Y_LT1 to Age = 0 and parses Y{n} and Y_GE{n} to integer ages.

Value

A tibble with columns: freq, projection, sex, unit, geo, age, Age, Year, mx.

See Also

Other survival curve functions: chance_alive(), chance_alive_es_aasmr(), plot_chance_alive(), plot_chance_alive_es_aasmr(), read_life_table()

Examples

```
## Not run:
es_aasmr <- read_es_aasmr(file.path("data", "life"))
es_aasmr %>% dplyr::count(geo, sex, projection, sort = TRUE)

## End(Not run)
```

---

read_life_table *Read HMD period life tables (1x1) from disk*

Description

Reads a Human Mortality Database (HMD) period life table file (1x1, by single year of age) for the selected sex and returns only the last look_back years based on the latest year present in the file. The open-ended age group (e.g. "110+") is parsed as its numeric lower bound (e.g. 110).

Usage

```
read_life_table(directory, sex = c("f", "m"), look_back = 20)
```

Arguments

directory	Directory containing the HMD life table files ("mltper_1x1.txt" / "fltper_1x1.txt" or gzipped variants "mltper_1x1.txt.gz" / "fltper_1x1.txt.gz").
sex	Sex code: "m" (male) or "f" (female).
look_back	Number of most recent years to keep (inclusive of the latest available year). Must be >= 1.

Value

A tibble with columns: PopName, Year, Age, mx, qx, ax, lx, dx, Lx, Tx, ex. Age is returned as integer.

See Also

Other survival curve functions: chance_alive(), chance_alive_es_aasmr(), plot_chance_alive(), plot_chance_alive_es_aasmr(), read_es_aasmr()

Examples

```
## Not run:
lt_m <- read_life_table(file.path("data", "life"), sex = "m", look_back = 20)
lt_m %>% dplyr::distinct(PopName) %>% dplyr::arrange(PopName)

## End(Not run)
```

`read_timeseries` *Read a time series file (CSV/TSV) with a date + one or more value columns*

Description

Loads a delimited file from the directory specified by `fundsr.data_dir`, parses the date column into a proper `Date`, and coerces all other columns to numeric.

Usage

```
read_timeseries(
  file,
  date_col = "date",
  time_unit = c("ms", "s", "us", "ns"),
  orders = NULL,
  force_text_date = FALSE,
  line_filter = NULL,
  ext_override = NULL
)
```

Arguments

`file`	Filename to read (relative to `fundsr.data_dir` option).
`date_col`	Name of the date column in the file.
`time_unit`	Character scalar giving the unit of a *numeric* date column (Unix epoch). One of `"ms"` (default), `"s"`, `"us"`, `"ns"`.
`orders`	Character vector of lubridate parsing orders for a *text* date column (passed to `lubridate::parse_date_time()`). If `NULL`, a default set of common dmy-order formats is used.
`force_text_date`	Logical scalar. If `TRUE`, the date column is always parsed as text using `orders` (no Unix-epoch numeric interpretation is attempted).
`line_filter`	Optional regular expression used to pre-filter the raw file by lines before parsing. If supplied, the file is read with `readr::read_lines()` and only lines matching `line_filter` are kept. The regex must match both data lines and the header line, e.g. `"^[0-9]|^Date,"`
`ext_override`	File format to assume instead of the one implied by the filename extension: either `tsv`, `csv`.

Details

The reader is chosen by file extension: `.csv` uses `readr::read_csv()` and `.tsv`/`.tab`/`.txt` uses `readr::read_tsv()`. Gzipped variants such as `.csv.gz` and `.tsv.gz` are also supported. This can be overriden by `ext_override`.

The function assumes a date column exists (default: `date`). By default, if the date column looks numeric (i.e., coercion to numeric yields at least one non-`NA`), it is interpreted as a Unix timestamp (scaled by `time_unit`). Otherwise it is parsed as text using `orders`. If `force_text_date = TRUE`, it is always parsed as text using `orders`.

All non-date columns are coerced with `as.numeric()` (non-parsable values become `NA`).

Value

A tibble with parsed date column and numeric value columns.

See Also

Other fund/index file readers: `read_timeseries_excel()`

---

read_timeseries_excel *Read a time series from an Excel workbook*

Description

Reads an Excel sheet, detects the header row by searching for a date header, parses the date column, selects/renames value columns by regex, and optionally coerces value columns to numeric.

Usage

```
read_timeseries_excel(
  file,
  sheet,
  date_col,
  col_trans,
  date_order = "dmy",
  force_numeric = TRUE,
  comma_rep = "."
)
```

Arguments

`file`	Path to the Excel workbook, relative to `fundsr.data_dir`.
`sheet`	Sheet identifier to read from (sheet name or 1-based index).
`date_col`	String used to detect the header row and identify the date column (matched via regex against cell contents for header-row detection, and against column names after headers are assigned).

col_trans	Named character vector (or list) mapping output column names to regex patterns used to select columns from the sheet. Names are returned column names; values are patterns matched against header names.
date_order	Character scalar indicating day/month/year order used to generate candidate date formats for parsing text dates (passed to make_date_fmts()). Default is "dmy".
force_numeric	Logical. If TRUE (default), always replace matched value columns with their numeric coercions (non-parsable values become NA). If FALSE, only replace when coercion succeeds for all non-NA values.
comma_rep	Character scalar used when converting character numerics: commas are replaced by this string before conversion. Default "." (treat comma as decimal separator).

Details

The sheet is read using read_excel_or_xml() (tries readxl first, then an XML fallback). Completely empty columns are dropped. The first row containing date_col (any cell match) is treated as the header row; data starts below it.

Date parsing:

- If the detected date column is numeric (or looks numeric), it is interpreted as an Excel serial date with origin "1899-12-30".
- Otherwise the date strings are cleaned (truncated to 24 chars, "Sept" → "Sep", trailing " 12:00:00 AM" removed) and parsed with as.Date() using formats from make_date_fmts(date_order). After parsing, the function drops all rows from the first unparseable date onward (i.e., it truncates at the first NA date), then filters remaining NA dates.

Column selection/renaming: col_trans maps desired output names to regex patterns matched against the detected header names. If a pattern matches multiple columns, they are kept and suffixed (name, name2, name3, ...).

Numeric coercion: For non-date columns, character values have "$" / "USD " stripped, commas replaced by comma_rep, then are converted with as.numeric(). If force_numeric = TRUE, the converted numeric column is kept even if some values fail to parse; otherwise the column is only replaced when all non-NA values parse successfully.

Value

A tibble with a date column (class Date) and the selected value columns (possibly numeric), with names determined by col_trans.

See Also

read_timeseries() for CSV/TSV time series import.

Other fund/index file readers: read_timeseries()

Examples

```
## Not run:
x <- read_timeseries_excel(
  file = "example.xlsx",
  sheet = 1,
  date_col = "^Date$",
  col_trans = c(nav = "NAV", tr = "TR"),
  date_order = "dmy"
)

## End(Not run)
```

`reset_state` *Clear fundsr session state*

Description

Convenience helper that clears mutable internal fundsr state for a session: storage, fund-index map, import-function registry, Inkscape export queue, and the XLM bookkeeping vector.

Usage

```
reset_state(session = NULL)
```

Arguments

`session`	Optional `fundsr_session` object. Defaults to the package default session when `NULL`.

Value

Invisibly returns `NULL`. Called for side effects.

See Also

Other config functions: `fundsr_default_session()`, `fundsr_options()`, `fundsr_session()`

Examples

```
reset_state()
```

`roll_diffs` *Compute rolling annualized tracking difference statistics*

Description

For each fund–index pair in `fund_index_map`, computes rolling, annualized tracking differences over a backward-looking window of `n_days` calendar days. Both log-return and CAGR forms are returned.

Usage

```
roll_diffs(
  df,
  n_days,
  fund_index_map,
  date_col = "date",
  index_level = c("net", "gross"),
  annual_days = 365,
  messages = c("roll", "skip"),
  gross_suffix = "-GR"
)
```

Arguments

`df`	Data frame containing a date column, fund columns, and benchmark/index columns referenced in `fund_index_map`.
`n_days`	Rolling lookback window in calendar days.
`fund_index_map`	Named character vector mapping fund column names to their corresponding benchmark/index base column names.
`date_col`	Name of the date column in `df`. Must be of class `Date` and sorted (strictly increasing).
`index_level`	Which index level to use, one of `"net"` or `"gross"`. If `"gross"`, `gross_suffix` is appended to the mapped index base name before lookup in `df`.
`annual_days`	Number of days used for annualization.
`messages`	Character vector controlling emitted messages. Any of `"roll"` (per-pair progress) and `"skip"` (skip reasons). Use `messages = "roll"` to show only progress, `messages = "skip"` to show only skip reasons, or `messages = character()` or `NULL` to silence all messages.
`gross_suffix`	Suffix appended to the mapped index base name when `index_level = "gross"`.

Details

For each date t, a target anchor threshold $t - n\_days$ is formed. The anchor date t_0 is chosen as the **last available observation at or before** $t - n\_days$ among rows where **both** fund and index

values are present. Let $\Delta = t - t_0$ in calendar days (Δ can be greater than `n_days` when data are missing around the threshold).

The annualized tracking differences are:

- Log-return difference:

$$\left[\ln\left(\frac{f_t}{f_{t_0}}\right) - \ln\left(\frac{i_t}{i_{t_0}}\right)\right] \times \frac{annual\_days}{\Delta}$$

- CAGR difference:

$$\left(\frac{f_t}{f_{t_0}}\right)^{annual\_days/\Delta} - \left(\frac{i_t}{i_{t_0}}\right)^{annual\_days/\Delta}$$

Values are `NA` when an anchor cannot be found, current-date inputs are missing, or inputs are invalid for the chosen formula (e.g. any non-positive level for log returns, or non-finite / non-positive ratios for CAGR).

Funds are skipped (optionally with a message) when the fund column is missing, the mapped index column is missing (after applying `index_level` / `gross_suffix`), or when `fund == index` (self-tracking).

Emitted messages will be visible at verbosity level >= 1 (option `fundsr.verbosity`). Verbosity level >= 4 forces both message types regardless of the `messages` argument.

Value

A named list with two data frames, `cagr` and `log`. Each data frame contains `date_col` followed by one column per fund (named as in `fund_index_map`), holding the rolling annualized tracking differences.

See Also

Other rolling difference functions: `plot_roll_diffs()`

`run_data_loaders` *Run registered data loaders*

Description

Runs the data loader registry (`session$state$data_loaders`) to populate (or refresh) the package's storage environment (`session$storage`).

Usage

```
run_data_loaders(reload = FALSE, session = NULL)
```

Arguments

`reload`	Logical scalar. If `TRUE`, forces a full reload by setting `options(fundsr.reload = TRUE)` for the duration of this call.
`session`	Optional `fundsr_session` object. Defaults to the package default session when `NULL`.

Details

The function temporarily sets the `fundsr.reload` option so that data loaders can decide whether to recompute cached objects.

The previous value of option "fundsr.reload" is restored on exit, even if a data loader errors.

Data loaders are taken from `session$state$data_loaders` and are called sequentially in registration order. Each registered function must take no arguments.

Value

Invisibly returns `session$storage` after running the data loaders.

See Also

Other fund/index workflow functions: `add_data_loader()`, `adjust_for_split()`, `build_all_series()`, `clear_data_loaders()`, `clear_storage()`, `get_storage()`, `import_fund()`, `join_env()`, `store_timeseries()`

---

run_plots *Generate and export rolling-differences from a specification*

Description

Iterates over plot specifications and produces rolling-difference plots for both CAGR and log-return variants. Each plot is saved via `save_plot()`, and optional extra plots based on `extra_data` may also be generated. All generated plot objects are stored in an environment and returned.

Usage

```
run_plots(
  roll_diffs,
  n_days,
  plot_spec,
  extra_data = NULL,
  add_gg_params = ggplot2::geom_blank(),
  bmark_type = c("net", "gross"),
  suffix = "",
  extra_prefix = "extra_",
  extra_fun = NULL,
  session = NULL,
  ...
)
```

Arguments

`roll_diffs`	A list of length 2 containing data frames, named `cagr` and `log` (CAGR differences and log-return differences).
`n_days`	Rolling-window length in days (passed to `plot_roll_diffs()` for labelling).

plot_spec	A data frame or a list of data frames describing plot parameters. Expected columns include: plot_id, title, data_filter, gg_params, width, height, funds.
extra_data	Optional data frame used to produce extra plots. Defaults to NULL.
add_gg_params	Optional ggplot component (or list of components) appended to each generated plot in addition to the per-plot gg_params defined in plot_spec. Defaults to ggplot2::geom_blank().
bmark_type	Benchmark type used in plot titles: "net" or "gross".
suffix	Character string appended to each plot_id when constructing filenames and names in the returned environment. Defaults to "".
extra_prefix	Character string prepended to each plot_id when constructing filenames and names in the returned environment for extra plots. Defaults to "extra_".
extra_fun	Function to call for extra plots.
session	Optional fundsr_session object. Defaults to the package default session when NULL.
...	Additional arguments passed to plot_roll_diffs().

Details

For each row in plot_spec, the function constructs both a CAGR-based and a log-return-based variant using plot_roll_diffs() and writes the resulting plots via save_plot(), using a filename suffix (_L) to distinguish the log-return variant. The optional suffix is appended to plot_id before filenames (and environment keys) are formed. Additional arguments in ... are forwarded to plot_roll_diffs().

If plot_spec is provided as a list of data frames, the function binds them into a single specification. The title column may be provided as a list column (e.g. to keep a multilingual named vector as a single per-row value).

If extra_data is supplied, an extra plot is generated once per unique set of tickers using extra_fun. Fund tickers are translated using mappings from the fundsr.ticker_map option (tickers not present in the map are used as-is). The first plot specification encountered for a given ticker set determines the base filename {extra_prefix}_<plot_id{suffix}> used for saving and storing the resulting extra plot.

Value

An environment containing ggplot objects. Objects are stored under names corresponding to the base filenames used in save_plot():

- plot_id{suffix} for the CAGR variant,
- plot_id{suffix}_L for the log-return variant,
- and (when generated) {extra_prefix}_plot_id{suffix} for extra plots.

See Also

Other plot export utilities: clear_inkscape_queue(), export_pngs(), save_plot()

Examples

```
## Not run:
plots <- run_plots(roll_diffs, n_days, plot_spec, extra_data = extra_data)
plots[["funds"]]
plots[["funds_L"]]
plots[["extra_funds"]]

## End(Not run)
```

`save_plot` *Save a plot as SVG and/or PNG and queue for Inkscape conversion*

Description

Saves `plot` as an SVG file when `save_svg = TRUE`. When an SVG is saved, an Inkscape export action is also queued so PNG generation can be performed later in batch via `export_pngs()`. Optionally, when `save_png = TRUE`, the function also saves a PNG immediately via `ggplot2::ggsave()` (independently of queueing).

Usage

```
save_plot(
  file,
  plot,
  px_width = fundsr_get_option("px_width", 1300),
  height = 12,
  width = 12,
  units = "in",
  out_dir = fundsr_get_option("out_dir"),
  save_png = fundsr_get_option("internal_png", FALSE),
  save_svg = fundsr_get_option("export_svg", TRUE),
  background = "white",
  session = NULL
)
```

Arguments

`file`	Base filename (without extension) used for output files.
`plot`	A plot object (typically a ggplot) to be saved.
`px_width`	Target width in pixels for PNG output. Used as the queued Inkscape `export-width`, and (if `save_png = TRUE`) used to compute the DPI for immediate PNG saving.
`height`	Height of the saved plot in `units`.
`width`	Width of the saved plot in `units`.
`units`	Units for `width`/`height` (e.g. `"in"`). For immediate PNG saving, only `"in"`, `"cm"`, and `"mm"` are supported (to compute DPI from `px_width`).

out_dir	Output directory where files are written.
save_png	Logical scalar; if TRUE, also saves a PNG immediately.
save_svg	Logical scalar; if TRUE, saves the SVG and queues an Inkscape export action.
background	Background color used for immediate PNG saving via ggplot2::ggsave() (bg).
session	Optional fundsr_session object. Defaults to the package default session when NULL.

Details

If save_svg = TRUE, the SVG is written as "{file}.svg". An Inkscape action string is then stored in session$state$inkscape_queue[file] so the SVG can later be exported to "{file}.png" at px_width pixels wide when export_pngs() is run.

Queueing is refused if either output path contains a semicolon (;), since Inkscape actions are separated by semicolons.

If save_png = TRUE, a PNG is also written immediately as "{file}.png". The PNG uses the same width, height, and units as the SVG, and sets dpi = px_width / width_in so that the pixel width is approximately px_width while keeping comparable physical-size typography across outputs. The PNG background is set via background.

If both save_svg and save_png are FALSE, the function issues a warning and returns without writing files or queueing exports.

Value

Invisibly returns NULL. Called for side effects.

See Also

Other plot export utilities: clear_inkscape_queue(), export_pngs(), run_plots()

---

store_timeseries *Store a cached object in the package storage environment*

Description

Evaluate an expression and cache its result in the package storage environment (session$storage) under a given name. The expression is only re-evaluated when the cached value is missing, when overwrite = TRUE, or when the global option fundsr.reload is TRUE. Optionally merges additional fund/index mappings into session$state$fund_index_map.

Usage

```
store_timeseries(
  var_name,
  expr,
  fund_index_map = NULL,
  overwrite = FALSE,
  postprocess = identity,
  session = NULL
)
```

Arguments

var_name	Character scalar. Name of the variable to store in session$storage.
expr	An expression. Evaluated in the caller's environment when (re)computing the cached value.
fund_index_map	Optional named vector of fund/index pairs to merge into session$state$fund_index_map. Names are used as keys, values should be indices.
overwrite	Logical scalar. If TRUE, recompute and replace any existing cached value, regardless of fundsr.reload.
postprocess	Function applied to the computed value before caching. Only used when the value is (re)computed (i.e. not applied when a cached value is reused). Defaults to base::identity().
session	Optional fundsr_session object. Defaults to the package default session when NULL.

Details

expr is evaluated in the environment where store_timeseries() is called (i.e. the caller's environment), then assigned into session$storage under var_name.

Caching behavior is controlled by:

- overwrite = TRUE (always recompute),
- options(fundsr.reload = TRUE) (force recomputation globally), or
- absence of var_name in session$storage (compute once).

If fund_index_map is supplied, it is merged into session$state$fund_index_map via name-based assignment: existing entries with the same names are replaced.

Value

Invisibly returns NULL (called for its side effects).

See Also

Other fund/index workflow functions: add_data_loader(), adjust_for_split(), build_all_series(), clear_data_loaders(), clear_storage(), get_storage(), import_fund(), join_env(), run_data_loaders()

Index