Package: AssetAllocation (via r-universe)

October 25, 2024

Type Package

Title Backtesting Simple Asset Allocation Strategies

Version 1.1.1

Author Alexandre Rubesam

Maintainer Alexandre Rubesam <alexandre.rubesam@gmail.com>

Description Easy and quick testing of customizable asset allocation strategies. Users can rely on their own data, or have the package automatically download data from Yahoo Finance (<https://finance.yahoo.com/>). Several pre-loaded portfolios with data are available, including some which are discussed in Faber (2015, ISBN:9780988679924).

License GPL (>= 3)

Encoding UTF-8

LazyData true

Depends R (>= 2.10)

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

RoxygenNote 7.1.2

Imports PerformanceAnalytics, quantmod, RiskPortfolios, xts, zoo, NMOF, riskParityPortfolio, curl

Config/testthat/edition 3

URL https://github.com/rubetron/AssetAllocation

BugReports https://github.com/rubetron/AssetAllocation/issues

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

RemoteUrl https://github.com/rubetron/assetallocation

RemoteRef HEAD

RemoteSha 19832294551056e22464f8925667ad469cbc39a1

Contents

`asset_allocations` *Pre-loaded Static and Tactical Asset Allocations*

Description

Basic static and tactical asset allocation strategies that work with the pre-loaded data in the object `ETFs`. Each element is itself a list with the following fields: name, tickers, default_weights, rebalance_frequency, portfolio_rule_fn.

The static allocations included are:

- United States 60/40 portfolio
- Golden Butterfly portfolio
- Rob Arnott Portfolio
- Global Asset Allocation
- Permanent Portfolio
- Desert Portfolio
- Larry Portfolio
- Big Rocks Portfolio
- Sandwich Portfolio
- Balanced Tax Aware Portfolio
- Balanced Portfolio
- Income with Growth Portfolio
- Income with Growth Tax Aware Portfolio

- Conservative Income
- Conservative Income Tax Aware
- All Weather Portfolio

The tactical asset allocations included are:

- Ivy Portfolio
- Robust Asset Allocation
- Dual Momentum
- Adaptive Asset Allocation
- The Trend is Your Friend (original)
- The Trend is Your Friend (real risk parity)
- JPMorgan Efficiente 5

Usage

```
data("asset_allocations")
```

Format

Object of class "List" with two fields containing static and tactical asset allocations, respectively. Each asset allocation is represented by a list with the following fields: ..$ name : chr ..$ tickers : chr ..$ default_weights : num ..$ rebalance_frequency: chr (default is "month") ..$ portfolio_rule_fn : chr (default is "identity")

Examples

```
data(asset_allocations)
# basic static allocation is the U.S. 60/40 portfolio:
us_60_40 <- asset_allocations$static$us_60_40

# basic tactical allocation is the Ivy portfolio:
ivy <- asset_allocations$tactical$ivy
```

---

backtest_allocation *Backtesting of asset allocation strategies*

Description

backtest_allocation computes a backtest of a given portfolio allocation rule.

Usage

```
backtest_allocation(strat, P, R, risk_free = 0, start_date = NULL)
```

Arguments

strat	A list representing an asset allocation strategy.
P	An xts object with daily prices of the tickers in strat.
R	An xts object with daily returns of the tickers in strat.
risk_free	Either an xts object with daily returns of the risk-free asset, or a scalar numeric with the annual risk-free rate in decimals.
start_date	Optional starting date

Details

The function first determines the rebalancing dates based on strat$rebalance_frequency. Then, it cycles through intermediate dates and calculates daily returns based on the allocation. If the optional parameter start_date is provided, the backtest will start on that date. Otherwise, it will start from the date from which data on all assets becomes available.

Value

An object of class "List" with the following elements:

strat	The strat provided to the function
returns	An xts object with the daily returns of the strategy
table_performance	A table with performance metrics
rebalance_dates	Vector of rebalancing dates
rebalance_weights	Vector of rebalancing dates

Examples

```
# Example 1: backtesting one of the asset allocations in the package
us_60_40 <- asset_allocations$static$us_60_40
bt_us_60_40 <- backtest_allocation(us_60_40,
                                   ETFs$Prices,
                                   ETFs$Returns,
                                   ETFs$risk_free)

# show table with performance metrics
bt_us_60_40$table_performance
# Example 2: creating and backtesting an asset allocation from scratch

# create a strategy that invests equally in momentum (MTUM), value (VLUE),
# low volatility (USMV) and quality (QUAL) ETFs.

factor_strat  <- list(name = "EW Factors",
                      tickers = c("MTUM", "VLUE", "USMV", "QUAL"),
                      default_weights = c(0.25, 0.25, 0.25, 0.25),
                      rebalance_frequency = "month",
```

```
                    portfolio_rule_fn = "constant_weights")

# get data for tickers using getSymbols
factor_ETFs <- get_data_from_tickers(factor_strat$tickers,
                                     starting_date = "2020-01-01")
# backtest the strategy
bt_factor_strat <- backtest_allocation(factor_strat,
                                       factor_ETFs$P,
                                       factor_ETFs$R)
# show table with performance metrics
bt_factor_strat$table_performance
```

`constant_weights` *Returns constant weights for static asset allocations*

Description

`constant_weights` applies the identity function to the default weights in a strategy.

Usage

```
constant_weights(strat, reb_date = NULL, P, R, risk_free)
```

Arguments

`strat`	A list representing an asset allocation strategy.
`reb_date`	A date on which the allocation rule is applied.
`P`	An xts object with daily prices of the tickers in strat.
`R`	An xts object with daily returns of the tickers in strat.
`risk_free`	Either an xts object with daily returns of the risk-free asset, or a scalar numeric with the annual risk-free rate in decimals.

Value

A numeric vector of weights after applying the rule.

Examples

```
us_60_40  <- asset_allocations$static$us_60_40
reb_date <- as.Date("2022-03-31")
constant_weights(us_60_40,
                 reb_date,
                 ETFs$Prices[, us_60_40$tickers],
                 ETFs$Returns[, us_60_40$tickers],
                 ETFs$risk_free)
```

daily_account_calc *Calculation of account value for backtesting asset allocation strategies*

Description

daily_account_calc is a helper function used by backtest_allocation to calculate theoretical the theoretical account value given an initial allocation to assets. It is not intended to be called directly by the user.

Usage

```
daily_account_calc(w, R)
```

Arguments

w	A vector of weights
R	An xts object with daily returns of the tickers in strat.

Details

The function simulates the value of a theoretical account from the initial weights and the daily returns of a set of assets.

Value

A numeric vector with the daily value of the account.

ETFs *Daily prices and total returns for 24 ETFs.*

Description

Data set containing daily prices and total returns for 37 exchange-traded funds (ETFs) as well as daily returns for U.S. Treasury bills (risk-free asset).

Usage

```
data(ETFs)
```

Format

An object of class "list"

Prices xts object with daily prices

Returns xts object with daily total returns

Description data.frame with information about the ETFs

risk_free xts object with daily returns of U.S. Treasury bills

Examples

```
data(ETFs)
head(ETFs$Prices)
ETFs$Description
```

`get_data_from_tickers` *Downloads prices in xts format from a list of tickers from Yahoo Finance (<https://finance.yahoo.com/>).*

Description

`get_data_from_tickers` retrieves adjusted closing prices from Yahoo Finance for a set of tickers and returns the prices and returns.

Usage

```
get_data_from_tickers(tickers, starting_date = "2007-01-01")
```

Arguments

`tickers`	A vector containing a tickers.
`starting_date`	A date on which the allocation rule is applied.

Details

The function retrieves data from Yahoo Finance (<https://finance.yahoo.com/>) using the getSymbols function from the `quantmod` package. It calculates returns from adjusted prices. The ticker names must correspond to those found in Yahoo Finance.

Value

An object of class `"List"` containing two objects of class `"xts"` with respectively the prices and returns of the assets, with column names corresponding to the tickers.

Examples

```
## download data for the following exchange-traded-funds: MTUM, VLUE, USMV, and QUAL.
factor_ETFs <- get_data_from_tickers(c("MTUM", "VLUE", "USMV", "QUAL"),
                                     starting_date = "2020-01-01")
```

get_rebalance_dates *Portfolio rebalancing dates*

Description

get_rebalance_dates determines rebalancing dates based on rebalancing frequency chosen by the user. This is a helper function used by backtest_allocation and is not intended to be called directly by the user.

Usage

```
get_rebalance_dates(dates, reb_freq, k = 1)
```

Arguments

dates	A vector of dates
reb_freq	Character with rebalancing frequency. Options are "days", "weeks", "months", "quarters", and "years"
k	An integer with number of periods to skip.

Value

A vector of dates.

min_variance *Returns minimum variance portfolio weights on a given date*

Description

min_variance determines asset allocations that minimize the variance of aportfolio.

Usage

```
min_variance(strat, reb_date, P, R, risk_free = NULL)
```

Arguments

strat	A list representing an asset allocation strategy.
reb_date	A date on which the allocation rule is applied.
P	An xts object with daily prices of the tickers in strat.
R	An xts object with daily returns of the tickers in strat.
risk_free	Either an xts object with daily returns of the risk-free asset, or a scalar numeric with the annual risk-free rate in decimals.

Details

The function calculates the covariance matrix of returns using the last two years (or minimum of one year) of daily returns. It relies on the `minvar` function from the `NMOF` package.

Value

A numeric vector of weights after applying the rule.

Examples

```
ivy  <- asset_allocations$tactical$ivy
reb_date <- as.Date("2022-03-31")
risk_parity(ivy, reb_date, ETFs$Prices[, ivy$tickers], ETFs$Returns[, ivy$tickers])
```

`risk_parity` *Returns risk parity weights on a given date*

Description

`risk_parity` determines asset allocations using a risk parity rule. It obtains the weights such that all assets provide the same risk contribution to the risk of the portfolio.

Usage

```
risk_parity(strat, reb_date, P, R, risk_free = NULL)
```

Arguments

`strat`	A list representing an asset allocation strategy.
`reb_date`	A date on which the allocation rule is applied.
`P`	An xts object with daily prices of the tickers in strat.
`R`	An xts object with daily returns of the tickers in strat.
`risk_free`	Either an xts object with daily returns of the risk-free asset, or a scalar numeric with the annual risk-free rate in decimals.

Details

The function calculates the covariance matrix of returns using the last two years (or minimum of one year) of daily returns.

Value

A numeric vector of weights after applying the rule.

Examples

```
ivy  <- asset_allocations$tactical$ivy
reb_date <- as.Date("2022-03-31")
risk_parity(ivy, reb_date, ETFs$Prices[, ivy$tickers], ETFs$Returns[, ivy$tickers])
```

`tactical_AAA` *Returns allocations for the Adaptive Asset Allocation strategy on a given date*

Description

`tactical_AAA` determines asset allocations according to the Adaptive Asset Allocation approach described in Butler, Philbrick, Gordillo, and Varadi (2012) <https://dx.doi.org/10.2139/ssrn.2328254>.

Usage

```
tactical_AAA(strat, reb_date, P, R, risk_free)
```

Arguments

`strat`	A list representing an asset allocation strategy. For this particular strategy, `strat$asset_class` must contain a character vector containing the corresponding asset classes.
`reb_date`	A date on which the allocation rule is applied.
`P`	An xts object with daily prices of the tickers in strat.
`R`	An xts object with daily returns of the tickers in strat.
`risk_free`	Either an xts object with daily returns of the risk-free asset, or a scalar numeric with the annual risk-free rate in decimals.

Details

The Adaptive Asset Allocation strategy sorts a specific list of assets based on 6-month momentum, selects the top 5 assets, and then calculates weights that yield the minimum portfolio variance. The parameters controlling the number of months for the momentum calculation (`n_months_mom`, default = 6), number of months of daily data used to estimate the covariance matrix (`n_months_mom`, default value = 1), and the number of assets to select using the momentum rule (`n_assets`, default = 5) can be changed by adding them to a list called `param` in the `strat` object. This allows the user to apply the simple principle of the strategy (momentum and minimum variance) to any set of assets.

Value

A numeric vector of weights after applying the rule.

`tactical_DualMomentum` *Returns allocations for the dual momentum strategy on a given date*

Description

`tactical_DualMomentum` determines asset allocations for a strategy according to the dual momentum approach described in Antonacci (2016) <https://dx.doi.org/10.2139/ssrn.2042750>.

Usage

```
tactical_DualMomentum(strat, reb_date, P, R, risk_free)
```

Arguments

`strat`	A list representing an asset allocation strategy. For this particular strategy, `strat$asset_class` must contain a character vector containing the corresponding asset classes.
`reb_date`	A date on which the allocation rule is applied.
`P`	An xts object with daily prices of the tickers in strat.
`R`	An xts object with daily returns of the tickers in strat.
`risk_free`	Either an xts object with daily returns of the risk-free asset, or a scalar numeric with the annual risk-free rate in decimals.

Details

Dual momentum sorts assets within each asset class described in `strat` on a relative basis (i.e. which asset outperforms others within the same asset class) over the last 12 months, as well as whether an asset has positive excess return over the last 12 months. Dual momentum invests in the top performing asset within the asset class, as long as it also has positive excess return over the risk-free rate. Otherwise, the allocation is shifted to the risk-free asset. Any amounts not allocated to risky assets are allocated to the risk-free asset as implemented in the `backtest_allocation` function.

Value

A numeric vector of weights after applying the rule.

Examples

```
dual_mom  <- asset_allocations$tactical$dual_mom
reb_date <- as.Date("2022-03-31")
tactical_DualMomentum(dual_mom,
             reb_date,
             ETFs$Prices[, dual_mom$tickers],
             ETFs$Returns[, dual_mom$tickers],
             ETFs$risk_free)
```

`tactical_ivy` *Returns allocations for the Ivy Portfolio on a given date*

Description

`tactical_ivy` determines asset allocations for a strategy according to the Ivy Portfolio rule of Faber (2013, ISBN:978-1118008850).

Usage

```
tactical_ivy(strat, reb_date, P, R, risk_free = NULL)
```

Arguments

`strat`	A list representing an asset allocation strategy.
`reb_date`	A date on which the allocation rule is applied.
`P`	An xts object with daily prices of the tickers in strat.
`R`	An xts object with daily returns of the tickers in strat.
`risk_free`	Either an xts object with daily returns of the risk-free

Details

The function compares prices at the end of a month to their moving averages. If the price of an asset is below its moving average, the corresponding allocation in `strat$default_weights` is set to zero.

Value

A numeric vector of weights after applying the rule.

Examples

```
ivy  <- asset_allocations$tactical$ivy
reb_date <- as.Date("2022-03-31")
tactical_ivy(ivy, reb_date, ETFs$Prices[, ivy$tickers], ETFs$Returns[, ivy$tickers])
```

`tactical_JPM5` *Calculates asset allocations for the JPMorgan ETF Efficiente® 5 portfolio.*

Description

`tactical_JPM5` determines asset allocations using a replication of the JPMorgan ETF Efficiente® 5 index methodology described in publicly available documentation (<https://sp.jpmorgan.com/spweb/content/307403.pdf>).

Usage

```
tactical_JPM5(strat, reb_date, P, R, risk_free = NULL)
```

Arguments

`strat`	A list representing an asset allocation strategy.
`reb_date`	A date on which the allocation rule is applied.
`P`	An xts object with daily prices of the tickers in strat.
`R`	An xts object with daily returns of the tickers in strat.
`risk_free`	Either an xts object with daily returns of the risk-free asset, or a scalar numeric with the annual risk-free rate in decimals.

Details

The strategy uses a window of six months of daily data to compute inputs to perform a constrained mean-variance optimization. It relies on the `mvFrontier` function from the `NMOF` package.

Value

A numeric vector of weights after applying the rule.

Examples

```
JPM_Eff5  <- asset_allocations$tactical$JPM_Eff5
reb_date <- as.Date("2022-03-31")
tactical_JPM5(JPM_Eff5, reb_date, ETFs$Prices[, JPM_Eff5$tickers], ETFs$Returns[, JPM_Eff5$tickers])
```

`tactical_RAA` *Returns allocations for the Robust Asset Allocation on a given date*

Description

`tactical_RAA` determines asset allocations for a strategy according to the Robust Asset Allocation (RAA) approach of Gray and Vogel (2015, ISBN:978-1119071501).

Usage

```
tactical_RAA(strat, reb_date, P, R, risk_free)
```

Arguments

`strat`	A list representing an asset allocation strategy.
`reb_date`	A date on which the allocation rule is applied.
`P`	An xts object with daily prices of the tickers in strat.
`R`	An xts object with daily returns of the tickers in strat.
`risk_free`	Either an xts object with daily returns of the risk-free asset, or a scalar numeric with the annual risk-free rate in decimals.

Details

RAA uses two trend-following rules. The first one is based on comparing the current price of assets with their 12-month moving average. The second one compares returns with the returns of the risk-free asset. The allocation rule keeps either 100 if both rules provide a positive signal, only one rule provided a positive signal, or both rules provide a negative signal, respectively. Any amounts not allocated to risky assets are allocated to the risk-free asset as implemented in the `backtest_allocation` function.

Value

A numeric vector of weights after applying the rule.

Examples

```
raa  <- asset_allocations$tactical$raa
reb_date <- as.Date("2022-03-31")
tactical_RAA(raa,
             reb_date,
             ETFs$Prices[, raa$tickers],
             ETFs$Returns[, raa$tickers],
             ETFs$risk_free)
```

`tactical_TrendFriend` *Returns allocations for the Ivy Portfolio on a given date*

Description

`tactical_TrendFriend` determines asset allocations for a strategy according to the strategy in Clare et al (2016, <https://doi.org/10.1016/j.jbef.2016.01.002)>.

Usage

```
tactical_TrendFriend(strat, reb_date, P, R, risk_free = NULL)
```

Arguments

`strat`	A list representing an asset allocation strategy.
`reb_date`	A date on which the allocation rule is applied.
`P`	An xts object with daily prices of the tickers in strat.
`R`	An xts object with daily returns of the tickers in strat.
`risk_free`	Either an xts object with daily returns of the risk-free

Details

The allocation strategy proposed in the paper is based on using a a time series momentum rule to select assets from a universe, and an allocation rule which gives weights proportional to the inverse volatility of the assets. The time-series (trend) momentum rule is based on whether the price of the asset on the rebalancing date is above its 10-month moving average. If not, the corresponding allocation in `strat$default_weights` is set to zero (and is therefore allocated to the risk-free asset).

Value

A numeric vector of weights after applying the rule.

Examples

```
trend_friend  <- asset_allocations$tactical$trend_friend
reb_date <- as.Date("2022-03-31")
tactical_TrendFriend(trend_friend,
                     reb_date,
                     ETFs$Prices[, trend_friend$tickers],
                     ETFs$Returns[, trend_friend$tickers]
                    )
```

`tactical_TrendFriend_RP` *Returns allocations for the Ivy Portfolio on a given date*

Description

`tactical_TrendFriend_RP` determines asset allocations for a strategy according to a modified version of the the strategy in Clare et al (2016, <https://doi.org/10.1016/j.jbef.2016.01.002)>. The modified version uses full risk parity instead of the inverse-volatility rule in the paper.

Usage

```
tactical_TrendFriend_RP(strat, reb_date, P, R, risk_free = NULL)
```

Arguments

`strat`	A list representing an asset allocation strategy.
`reb_date`	A date on which the allocation rule is applied.
`P`	An xts object with daily prices of the tickers in strat.
`R`	An xts object with daily returns of the tickers in strat.
`risk_free`	Either an xts object with daily returns of the risk-free

Details

The allocation strategy proposed in the paper is based on using a a time series momentum rule to select assets from a universe, and an allocation rule which gives weights proportional to the inverse volatility of the assets. The time-series (trend) momentum rule is based on whether the price of the asset on the rebalancing date is above its 10-month moving average. If not, the corresponding allocation in `strat$default_weights` is set to zero (and is therefore allocated to the risk-free asset).

Value

A numeric vector of weights after applying the rule.

Examples

```
trend_friend  <- asset_allocations$tactical$trend_friend
reb_date <- as.Date("2022-03-31")
tactical_TrendFriend_RP(trend_friend,
                        reb_date,
                        ETFs$Prices[, trend_friend$tickers],
                        ETFs$Returns[, trend_friend$tickers]
                        )
```

Index