Package: REN (via r-universe)

December 10, 2024

Type Package

Title Regularization Ensemble for Robust Portfolio Optimization

Version 0.1.0

Maintainer Bonsoo Koo <bonsoo.koo@monash.edu>

Description Portfolio optimization is achieved through a combination of regularization techniques and ensemble methods that are designed to generate stable out-of-sample return predictions, particularly in the presence of strong correlations among assets. The package includes functions for data preparation, parallel processing, and portfolio analysis using methods such as Mean-Variance, James-Stein, LASSO, Ridge Regression, and Equal Weighting. It also provides visualization tools and performance metrics, such as the Sharpe ratio, volatility, and maximum drawdown, to assess the results.

License AGPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports lubridate, glmnet, quadprog, doParallel, Matrix, tictoc, corpcor, ggplot2, reshape2, foreach, stats, parallel

Suggests knitr, rmarkdown, KernSmooth, cluster, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 2.10)

LazyData true

NeedsCompilation no

Author Hardik Dixit [aut], Shijia Wang [aut], Bonsoo Koo [aut, cre], Cash Looi [aut], Hong Wang [aut]

Repository CRAN

Date/Publication 2024-10-10 16:30:05 UTC

Config/pak/sysreqs libicu-dev

Contents

buh.clust . . . 2
FF25 . . . 3
insert.at . . . 4
perform_analysis . . . 4
po.avg . . . 6
po.bhu . . . 6
po.cols . . . 7
po.covShrink . . . 7
po.grossExp . . . 8
po.JM . . . 8
po.SW . . . 9
po.SW.lasso . . . 9
po.TZT . . . 10
prepare_data . . . 10
ren . . . 11
setup_parallel . . . 12

Index **13**

`buh.clust` *Perform Hierarchical Clustering on Asset Correlations*

Description

This function performs hierarchical clustering on asset correlations and returns the clustered groups.

Usage

```
buh.clust(x)
```

Arguments

`x`	A numeric matrix of asset returns.

Value

A list of asset clusters.

`FF25` *FF25 Dataset*

Description

A dataset containing financial data for portfolio analysis with 25 portfolios formed on size and book-to-market ratios.

Usage

```
FF25
```

Format

A data frame with 25,670 rows and 27 variables:

X Date in the format YYYYMMDD (int)

X.1 Another column (check what this is; update description as needed)

SMALL.LoBM Portfolio returns for small firms with low book-to-market ratios (numeric)

ME1.BM2 Portfolio returns for medium-sized firms with book-to-market group 2 (numeric)

ME1.BM3 Portfolio returns for medium-sized firms with book-to-market group 3 (numeric)

ME1.BM4 Portfolio returns for medium-sized firms with book-to-market group 4 (numeric)

SMALL.HiBM Portfolio returns for small firms with high book-to-market ratios (numeric)

ME2.BM1 Portfolio returns for medium-sized firms with book-to-market group 1 (numeric)

ME2.BM2 Portfolio returns for medium-sized firms with book-to-market group 2 (numeric)

ME2.BM3 Portfolio returns for medium-sized firms with book-to-market group 3 (numeric)

ME2.BM4 Portfolio returns for medium-sized firms with book-to-market group 4 (numeric)

ME2.BM5 Portfolio returns for medium-sized firms with book-to-market group 5 (numeric)

ME3.BM1 Portfolio returns for large firms with book-to-market group 1 (numeric)

ME3.BM2 Portfolio returns for large firms with book-to-market group 2 (numeric)

ME3.BM3 Portfolio returns for large firms with book-to-market group 3 (numeric)

ME3.BM4 Portfolio returns for large firms with book-to-market group 4 (numeric)

ME3.BM5 Portfolio returns for large firms with book-to-market group 5 (numeric)

ME4.BM1 Portfolio returns for firms in the fourth size and first book-to-market group (numeric)

ME4.BM2 Portfolio returns for firms in the fourth size and second book-to-market group (numeric)

ME4.BM3 Portfolio returns for firms in the fourth size and third book-to-market group (numeric)

ME4.BM4 Portfolio returns for firms in the fourth size and fourth book-to-market group (numeric)

ME4.BM5 Portfolio returns for firms in the fourth size and fifth book-to-market group (numeric)

BIG.LoBM Portfolio returns for large firms with low book-to-market ratios (numeric)

ME5.BM2 Portfolio returns for large firms with book-to-market group 2 (numeric)

ME5.BM3 Portfolio returns for large firms with book-to-market group 3 (numeric)

ME5.BM4 Portfolio returns for large firms with book-to-market group 4 (numeric)

BIG.HiBM Portfolio returns for large firms with high book-to-market ratios (numeric)

Source

<Provide any relevant source here, e.g., URL or description of data origin>

Examples

```
data(FF25)
head(FF25)
```

`insert.at` *Insert Values at Specified Positions in a Vector*

Description

This function inserts specified values at given positions in a vector.

Usage

```
insert.at(a, pos, ...)
```

Arguments

`a`	A vector.
`pos`	A numeric vector specifying the positions to insert the values.
`...`	Values to be inserted at the specified positions.

Value

A vector with the inserted values.

`perform_analysis` *Perform Portfolio Analysis*

Description

This function performs portfolio analysis using various methods such as Mean-Variance (MV), James-Stein (JM), LASSO, Ridge Regression, Equal Weighting (EW), among others. It calculates weights, turnover, returns, Sharpe ratios, volatility, and maximum drawdown for each method.

Usage

```
perform_analysis(x, mon, count, Date, num_cores = 7)
```

Arguments

x	A numeric matrix where each column represents asset returns and rows represent time periods.
mon	A numeric vector representing the number of months since the start date for each time period.
count	A numeric vector indicating the number of entries per month.
Date	A vector of Date objects representing the dates of the time periods.
num_cores	The number of cores to use for parallel processing. Default is 7.

Details

The function iterates through different time periods and calculates portfolio weights, turnover, and returns for multiple methods including Mean-Variance (MV), James-Stein (JM), and various regularization techniques. It also computes performance metrics like the Sharpe ratio, volatility, maximum drawdown, and cumulative turnover for each method. Visualization of the cumulative returns and turnover is generated using ggplot2.

Value

A list containing the following components:

cumulative_return_plot A ggplot object representing the cumulative returns for each method.

cumulative_turnover_plot A ggplot object representing the cumulative turnover for each method.

turnover_mean A numeric vector of the mean turnover for each method.

sharpe_ratio A numeric vector of the Sharpe ratio for each method.

volatility A numeric vector of the annualized volatility for each method.

max_drawdown A numeric vector of the maximum drawdown for each method.

vw_to_mean The mean turnover for the volume-weighted (VW) portfolio.

vw_sharpe_ratio The Sharpe ratio for the VW portfolio.

vw_volatility The annualized volatility for the VW portfolio.

vw_max_drawdown The maximum drawdown for the VW portfolio.

Examples

```
# Create a larger example dataset that aligns with the function's expectations
set.seed(123)
x <- matrix(runif(700), ncol = 10)  # 10 columns (assets), 70 rows (observations)
mon <- rep(1:10, each = 7)         # Example month identifiers, 7 observations per month
count <- rep(7, 10)                 # Example count per month (7 entries per month)
Date <- as.Date('2020-01-01') + 0:69 # Example date sequence (70 days)

# Run the analysis with 2 cores
result <- perform_analysis(x, mon, count, Date, num_cores = 2)

# Display results
print(result$cumulative_return_plot)
print(result$cumulative_turnover_plot)
```

po.avg *Perform LASSO or Ridge Regression for Portfolio Optimization*

Description

This function performs LASSO, Ridge, or Elastic Net regression for portfolio optimization.

Usage

```
po.avg(y0, x0, method = "LASSO")
```

Arguments

y0	A numeric vector of response values.
x0	A numeric matrix of predictors.
method	The regularization method: "LASSO", "RIDGE", or "EN" (Elastic Net).

Value

A numeric vector of optimized portfolio weights.

po.bhu *Perform Portfolio Optimization Using Clusters and LASSO*

Description

This function performs portfolio optimization using clustering and LASSO regularization.

Usage

```
po.bhu(y0, x0, group, rep)
```

Arguments

y0	A numeric vector of response values.
x0	A numeric matrix of predictors.
group	A list of asset clusters.
rep	The number of repetitions for optimization.

Value

A numeric vector of optimized portfolio weights.

po.cols *Perform Simple Linear Model for Portfolio Optimization*

Description

This function uses simple linear regression to perform portfolio optimization.

Usage

```
po.cols(y0, x0)
```

Arguments

y0	A numeric vector of response values.
x0	A numeric matrix of predictors.

Value

A numeric vector of optimized portfolio weights.

po.covShrink *Perform Shrinkage-Based Portfolio Optimization*

Description

This function uses covariance shrinkage techniques for portfolio optimization.

Usage

```
po.covShrink(y0, x0)
```

Arguments

y0	A numeric vector of response values.
x0	A numeric matrix of predictors.

Value

A numeric vector of optimized portfolio weights.

po.grossExp *Perform Gross Exposure Portfolio Optimization*

Description

This function performs gross exposure portfolio optimization using LASSO.

Usage

```
po.grossExp(y0, x0, method = "NOSHORT")
```

Arguments

y0	A numeric vector of response values.
x0	A numeric matrix of predictors.
method	The regularization method: "NOSHORT" or "EQUAL".

Value

A numeric vector of optimized portfolio weights.

po.JM *Perform James-Stein Portfolio Optimization*

Description

This function uses James-Stein estimation for portfolio optimization.

Usage

```
po.JM(x0)
```

Arguments

x0	A numeric matrix of asset returns.

Value

A numeric vector of optimized portfolio weights.

po.SW — *Perform Stochastic Weight Portfolio Optimization*

Description

This function performs stochastic weight portfolio optimization.

Usage

```
po.SW(x0, b, sample)
```

Arguments

x0	A numeric matrix of asset returns.
b	Number of assets to select in each sample.
sample	Number of random samples to generate.

Value

A numeric vector of optimized portfolio weights.

po.SW.lasso — *Perform LASSO Regularization with Stochastic Weight Portfolio Optimization*

Description

This function performs portfolio optimization using LASSO regularization and stochastic weight selection.

Usage

```
po.SW.lasso(y0, x0, b, sample)
```

Arguments

y0	A numeric vector of response values.
x0	A numeric matrix of predictors.
b	Number of assets to select in each sample.
sample	Number of random samples to generate.

Value

A numeric vector of optimized portfolio weights.

po.TZT *Perform Portfolio Optimization Using TZT Method*

Description

This function performs portfolio optimization using the TZT method.

Usage

```
po.TZT(x0, gamma)
```

Arguments

x0	A numeric matrix of asset returns.
gamma	A numeric parameter for the TZT method.

Value

A numeric vector of optimized portfolio weights.

prepare_data *Prepare Data for Portfolio Analysis*

Description

This function prepares the input data by filtering based on a specified date range, removing the date column, and handling missing values. It also generates time-related columns and returns the processed data.

Usage

```
prepare_data(
  dat,
  date_column_index = 1,
  start_date = "19990101",
  end_date = "20231231"
)
```

Arguments

dat	A data frame or matrix where the first column is the date and the remaining columns are the data.
date_column_index	The index of the date column in the input data. Default is 1.
start_date	A character string specifying the start date for filtering the data in 'YYYYMMDD' format. Default is '19990101'.
end_date	A character string specifying the end date for filtering the data in 'YYYYMMDD' format. Default is '20231231'.

Value

A list containing the following components:

x A matrix of the filtered data with missing values handled.

mon A vector of integers representing the number of months from the first date in the data.

count A vector of the number of entries per month.

Date A vector of Date objects representing the filtered dates.

Examples

```
data <- data.frame(Date = c("19990101", "19990115", "19990201", "19990301", "19990315", "19990401"),
Var1 = c(1, 2, -99.99, 4, 5, -99.99),
Var2 = c(3, -99.99, 6, 7, 8, 9),
Var3 = c(10, 11, 12, 13, -99.99, 15))
result <- prepare_data(data, date_column_index = 1, start_date = '19990101', end_date = '19990430')
print(result)
```

---

ren *Main Function for Portfolio Analysis*

Description

This function integrates data preparation, parallel setup, and portfolio analysis. It takes raw data as input, prepares it using ‘prepare_data‘, sets up parallel processing using ‘setup_parallel‘, and performs the analysis using ‘perform_analysis‘.

Usage

```
ren(
  dat,
  date_column_index = 1,
  start_date = "19990101",
  end_date = "20231231",
  num_cores = 2
)
```

Arguments

dat	A data frame or matrix where the first column is the date and the remaining columns are the data.
date_column_index	The index of the date column in the input data. Default is 1.
start_date	A character string specifying the start date for filtering the data in 'YYYYMMDD' format. Default is '19990101'.
end_date	A character string specifying the end date for filtering the data in 'YYYYMMDD' format. Default is '20231231'.
num_cores	The number of cores to use for parallel processing. Default is 2.

Value

The results from 'perform_analysis', including plots and performance metrics.

Examples

```
## Not run:
# load the sample data
dat(FF25)
# Run the main function
result <- ren(FF25)

# Display results
print(result$cumulative_return_plot)
print(result$cumulative_turnover_plot)

## End(Not run)
```

`setup_parallel` *Setup Parallel Processing for Portfolio Analysis*

Description

This function sets up parallel processing by loading necessary libraries, allowing the user to specify the number of cores to use, and creating a parallel backend for faster computation.

Usage

```
setup_parallel(num_cores = 7)
```

Arguments

`num_cores`	The default number of cores to use for parallel processing. Default is 7.

Details

This function allows the user to specify the number of cores for parallel processing either through the argument `num_cores` or via interactive user input. The function also loads a set of libraries required for portfolio analysis.

Value

A parallel cluster object that can be used with functions that support parallel computation.

Examples

```
# Set up parallel processing with a specified number of cores
cl <- setup_parallel(num_cores = 2)  # Use 2 cores for the example
print(cl)  # Print the cluster information
parallel::stopCluster(cl)  # Stop the cluster after use to clean up
```

Index

∗ **datasets**
FF25, 3

buh.clust, 2

FF25, 3

insert.at, 4

perform_analysis, 4
po.avg, 6
po.bhu, 6
po.cols, 7
po.covShrink, 7
po.grossExp, 8
po.JM, 8
po.SW, 9
po.SW.lasso, 9
po.TZT, 10
prepare_data, 10

ren, 11

setup_parallel, 12