Package: rolloptim (via r-universe)

July 11, 2026

Type Package

Title Rolling Optimizations

Version 1.0.0

Description Analytical computation of rolling optimization for time-series data. The 'rolloptim' package solves constrained quadratic and linear programs in closed form by applying Lagrangian multipliers and the Karush-Kuhn-Tucker conditions (Kuhn and Tucker, 1951, <doi:10.1525/9780520411586-036>) to perform mean-variance portfolio optimization (Markowitz, 1952, <doi:10.1111/j.1540-6261.1952.tb01525.x>) over rolling windows. For each window, the analytical solution computes the optimal weights that minimize variance, maximize expected return, minimize residual sum of squares, or maximize quadratic utility, subject to a total-weight equality constraint and box bounds on each weight. Use cases include mean-variance portfolio optimization, expected-return maximization, and constrained regression. The package supports rolling optimizations with constraints via the total, lower, and upper arguments. The implementation accepts rolling moments computed via the 'roll' package and uses 'RcppArmadillo' for linear algebra, with parallelism across windows provided by 'RcppParallel'.

License GPL (>= 2)

URL https://github.com/jasonjfoster/rolloptim

BugReports https://github.com/jasonjfoster/rolloptim/issues

Depends R (>= 3.5.0)

Imports Rcpp, RcppParallel

Suggests covr, CVXR, ROI, ROI.plugin.glpk, ROI.plugin.qpoases, ROI.plugin.quadprog, roll (>= 1.1.7), testthat, zoo

LinkingTo Rcpp, RcppArmadillo, RcppParallel

Config/roxygen2/old_usage TRUE

Config/roxygen2/version 8.0.0

Encoding UTF-8

SystemRequirements GNU make

NeedsCompilation yes

Author Jason Foster [aut, cre]

Maintainer Jason Foster <jason.j.foster@gmail.com>

Config/pak/sysreqs make

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

Date/Publication 2026-07-11 08:30:08 UTC

RemoteUrl https://github.com/cran/rolloptim

RemoteRef HEAD

RemoteSha 70c2a43936a62200b4af7e442d35d847d1b1a81e

Contents

`rolloptim-package` *Rolling Optimizations*

Description

Analytical computation of rolling optimization for time-series data. The 'rolloptim' package solves constrained quadratic and linear programs in closed form by applying Lagrangian multipliers and the Karush-Kuhn-Tucker conditions (Kuhn and Tucker, 1951, <doi:10.1525/9780520411586-036>) to perform mean-variance portfolio optimization (Markowitz, 1952, <doi:10.1111/j.1540-6261.1952.tb01525.x>) over rolling windows. For each window, the analytical solution computes the optimal weights that minimize variance, maximize expected return, minimize residual sum of squares, or maximize quadratic utility, subject to a total-weight equality constraint and box bounds on each weight. Use cases include mean-variance portfolio optimization, expected-return maximization, and constrained regression. The package supports rolling optimizations with constraints via the total, lower, and upper arguments. The implementation accepts rolling moments computed via the 'roll' package and uses 'RcppArmadillo' for linear algebra, with parallelism across windows provided by 'RcppParallel'.

Details

`rolloptim` is a package that provides analytical computation of rolling optimization for time-series data.

Author(s)

Jason Foster [aut, cre]

References

Kuhn, H.W. and Tucker, A.W. (1951). "Nonlinear Programming." In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, edited by J. Neyman, 481-492. University of California Press. doi:10.1525/9780520411586036

Markowitz, H.M. (1952). "Portfolio Selection." *The Journal of Finance*, 7(1), 77-91. doi:10.1111/j.15406261.1952.tb01525.x

---

`roll_max_mean` *Rolling Optimizations to Maximize Mean*

Description

A function for computing rolling optimizations to maximize mean.

Usage

```
roll_max_mean(mu, sigma = NULL, target = NULL, total = 1, lower = 0,
  upper = 1)
```

Arguments

`mu`	matrix. Rows are means and columns are variables.
`sigma`	cube. Slices are covariance matrices.
`target`	vector. Rows are target variances.
`total`	numeric. Sum of the weights.
`lower`	numeric. Lower bound of the weights.
`upper`	numeric. Upper bound of the weights.

Value

An object of the same class and dimension as `mu` with the rolling optimizations to maximize mean.

Examples

```
if (requireNamespace("roll", quietly = TRUE)) {

n_vars <- 3
n_obs <- 15
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)

mu <- roll::roll_mean(x, 5)
```

```
# rolling optimizations to maximize mean
roll_max_mean(mu)

}
```

`roll_max_utility` *Rolling Optimizations to Maximize Utility*

Description

A function for computing rolling optimizations to maximize utility.

Usage

```
roll_max_utility(mu, sigma, lambda = 1, total = 1, lower = 0,
  upper = 1)
```

Arguments

`mu`	matrix. Rows are means and columns are variables.
`sigma`	cube. Slices are covariance matrices.
`lambda`	numeric. Risk aversion parameter.
`total`	numeric. Sum of the weights.
`lower`	numeric. Lower bound of the weights.
`upper`	numeric. Upper bound of the weights.

Value

An object of the same class and dimension as `mu` with the rolling optimizations to maximize utility.

Examples

```
if (requireNamespace("roll", quietly = TRUE)) {

n_vars <- 3
n_obs <- 15
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)

mu <- roll::roll_mean(x, 5)
sigma <- roll::roll_cov(x, width = 5)

# rolling optimizations to maximize utility
roll_max_utility(mu, sigma, lambda = 1)

}
```

roll_min_rss *Rolling Optimizations to Minimize Residual Sum of Squares*

Description

A function for computing rolling optimizations to minimize residual sum of squares.

Usage

```
roll_min_rss(xx, xy, total = 1, lower = 0, upper = 1)
```

Arguments

xx	cube. Slices are crossproducts of x and x.
xy	cube. Slices are crossproducts of x and y.
total	numeric. Sum of the weights.
lower	numeric. Lower bound of the weights.
upper	numeric. Upper bound of the weights.

Value

An object of the same class and dimension as x with the rolling optimizations to minimize residual sum of squares.

Examples

```
if (requireNamespace("roll", quietly = TRUE)) {

n_vars <- 3
n_obs <- 15
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)
y <- rnorm(n_obs)

xx <- roll::roll_crossprod(x, x, 5)
xy <- roll::roll_crossprod(x, y, 5)

# rolling optimizations to minimize residual sum of squares
roll_min_rss(xx, xy)

}
```

roll_min_var *Rolling Optimizations to Minimize Variance*

Description

A function for computing rolling optimizations to minimize variance.

Usage

```
roll_min_var(sigma, mu = NULL, target = NULL, total = 1, lower = 0,
  upper = 1)
```

Arguments

sigma	cube. Slices are covariance matrices.
mu	matrix. Rows are means and columns are variables.
target	vector. Rows are target means.
total	numeric. Sum of the weights.
lower	numeric. Lower bound of the weights.
upper	numeric. Upper bound of the weights.

Value

An object of the same class and dimension as mu with the rolling optimizations to minimize variance.

Examples

```
if (requireNamespace("roll", quietly = TRUE)) {

n_vars <- 3
n_obs <- 15
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)

sigma <- roll::roll_cov(x, width = 5)

# rolling optimizations to minimize variance
roll_min_var(sigma)

}
```

Index