

Package: uncorbets (via r-universe)

August 21, 2024

Type Package

Title Uncorrelated Bets via Minimum Torsion Algorithm

Version 0.1.2.9000

Description Implements Minimum Torsion for portfolio diversification as described in Meucci, Attilio (2013) [<doi:10.2139/ssrn.2276632>](https://doi.org/10.2139/ssrn.2276632).

License MIT + file LICENSE

URL <https://github.com/Reckziegel/uncorbets>,
<https://reckziegel.github.io/uncorbets/>

BugReports <https://github.com/Reckziegel/uncorbets/issues>

Imports assertthat (>= 0.2.1), NlcOptim (>= 0.6), stats

Suggests spelling, covr, testthat (>= 3.0.10)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.1

Language en-US

Depends R (>= 2.10)

Repository <https://reckziegel.r-universe.dev>

RemoteUrl <https://github.com/reckziegel/uncorbets>

RemoteRef HEAD

RemoteSha e1282710096d56045e631858ba6bf2ed4d24a110

Contents

effective_bets	2
max_effective_bets	3
torsion	4

Index	5
--------------	----------

effective_bets	<i>Effective Number of Bets</i>
----------------	---------------------------------

Description

Computes the diversification probability distribution and the effective number of bets of an allocation.

Usage

```
effective_bets(b, sigma, t)
```

Arguments

b	A vector of exposures (allocations).
sigma	A $n \times n$ covariance matrix.
t	A $n \times n$ torsion matrix.

Value

A list of length 2 with:

- p: the diversification probability distribution;
- enb: the effective number of bets.

Examples

```
# extract the invariants from the data
set.seed(123)
log_ret <- matrix(rnorm(400), ncol = 4) / 10

# compute the covariance matrix
sigma <- stats::cov(log_ret)

# torsion
torsion_cov <- torsion(sigma = sigma, model = 'minimum-torsion', method = 'exact')

# 1/N reference
b <- rep(1 / ncol(log_ret), ncol(log_ret))

# ENB
effective_bets(b = b, sigma = sigma, t = torsion_cov)
```

max_effective_bets *Risk-Diversification powered by the Minimum Torsion Algorithm*

Description

Finds the allocation that maximizes the [effective_bets](#).

Usage

```
max_effective_bets(x0, sigma, t, tol = 1e-20, maxeval = 5000L, maxiter = 5000L)
```

Arguments

x0	A numeric vector for the search starting point. Usually the "one over n" allocation.
sigma	A n x n covariance matrix.
t	A n x n torsion matrix.
tol	An interger with the convergence tolerance.
maxeval	An integer with the maximum number of evaluations of the objective function.
maxiter	An integer with the maximum number of iterations.

Value

A list with the following components:

- `weights`: the optimal allocation policy
- `enb`: the optimal effective number of bets
- `counts`: the number of iterations of the objective and the gradient
- `lambda_lb`: the lower bound Lagrange multipliers
- `lambda_ub`: the upper bound Lagrange multipliers
- `lambda_eq`: the equality Lagrange multipliers
- `gradient`: the gradient of the objective function at the optimum
- `hessian`: hessian of the objective function at the optimum

See Also

[solnl](#)

Examples

```
# extract the invariants from the data
set.seed(123)
log_ret <- matrix(stats::rnorm(400), ncol = 4) / 10

# compute the covariance matrix
sigma <- stats::cov(log_ret)

# torsion
torsion_cov <- torsion(sigma = sigma, model = 'minimum-torsion', method = 'exact')

# 1/N reference
b <- rep(1 / ncol(log_ret), ncol(log_ret))

max_effective_bets(x0 = b, sigma = sigma, t = torsion_cov)
```

torsion	<i>Computes the Minimum Torsion Matrix</i>
---------	--

Description

Computes the Principal Components Torsion and the Minimum Torsion for diversification analysis.

Usage

```
torsion(sigma, model = "minimum-torsion", method = "exact", max_niter = 10000L)
```

Arguments

sigma	A n x n covariance matrix.
model	One of: "pca" or "minimum-torsion".
method	One of: "approximate" or "exact". Only used when model = "minimum-torsion".
max_niter	An integer with the maximum number of iterations.

Value

A n x n torsion matrix.

Examples

```
# extract the invariants from the data
set.seed(123)
log_ret <- matrix(rnorm(400), ncol = 4) / 10

# calculate the covariance matrix
sigma <- stats::cov(log_ret)

# torsion
torsion(sigma = sigma, model = 'minimum-torsion', method = 'exact')
```

Index

effective_bets, [2](#), [3](#)

max_effective_bets, [3](#)

solnl, [3](#)

torsion, [4](#)