condreg

Compute the condition number with given penalty parameter

Description

Compute the condition number with given penalty parameter

Usage

condreg(data_in, kmax)

Arguments

data_in input data
kmax scalar regularization parameter

Value

list of condition number regularized covariance matrix s and its inverse invS.

Examples

```r
## True covariance matrix
sigma <- diag(5)
sigma[3,2] <- sigma[2,3] <- 0.8

## Generate normal random samples
## Not run:
library(MASS)
X <- mvrnorm(200,rep(0,5),sigma)

## Covariance estimation
crcov <- condreg(X,3)

## Inspect output
str(crcov) # returned object
sigma.hat <- crcov$s # estimate of sigma matrix
omega.hat <- crcov$invS # estimate of inverse of sigma matrix

## End(Not run)
```
Computes multiple solutions

Usage

crbulk(S, k)

Arguments

S  sample covariance matrix
k  vector of regularization parameters

Value

list of orthogonal matrix $Q$, shrunked eigenvalues $L_{\text{bar}}$ (shrinkage depending on penalty parameters) and sample eigenvalues $L$.

datasnp  Standard & Poors index

Return a vector of grid of penalties for cross-validation

Usage

kgrid(gridmax, numpts)

Arguments

gridmax  maximum value in penalty grid
numpts  number of points in penalty grid
Value
vector of penalties between 1 and approximately gridmax with logarithmic spacing

Examples

```r
library(ml_solver)
gmax <- 20  # maximum value for the grid of points
npts <- 10  # number of grid points returned
gridpts <- kgrid(gmax, npts)
```

---

### ml_solver

*Compute shrinkage of eigenvalues for condreg*

**Description**
Compute shrinkage of eigenvalues for condreg

**Usage**

```r
ml_solver(L, k, dir = "forward")
```

**Arguments**

- `L`: vector of eigenvalues
- `k`: vector of penalties
- `dir`: direction of path solver ("forward" or "backward")

**Value**
list of vector of shrinked eigenvalues $L_{\text{bar}}$, optimal $u$ value $u_{\text{opt}}$ and interval indicator $\text{intv}$.

---

### path_backward

*Compute optimal $u$ of Lemma 1 in JRSSB paper using the backward algorithm*

**Description**
Compute optimal $u$ of Lemma 1 in JRSSB paper using the backward algorithm

**Usage**

```r
path_backward(L)
```

**Arguments**

- `L`: vector of eigenvalues
**path_forward**

*Compute optimal u of Lemma 1 in JRSSB paper using the forward algorithm*

**Description**

Compute optimal u of Lemma 1 in JRSSB paper using the forward algorithm

**Usage**

`path_forward(L)`

**Arguments**

- `L`: vector of eigenvalues

**pfweights**

*Compute optimal portfolio weights*

**Description**

Compute optimal portfolio weights

**Usage**

`pfweights(sigma)`

**Arguments**

- `sigma`: covariance matrix

**Value**

new portfolio weights

**R**

*Weekly stock price data*

**Description**

Weekly stock price data
select_condreg

Compute the best condition number regularized based based on cross-validation selected penalty parameter

Description

Compute the best condition number regularized based based on cross-validation selected penalty parameter

Usage

`select_condreg(x, k, ...)`

Arguments

- `x` n-by-p matrix of data
- `k` vector of penalties for cross-validation
- `...` parameters for `select_kmax`

Value

list of condition number regularized covariance matrix S and its inverse invS

Examples

```r
## True covariance matrix
sigma <- diag(5)
sigma[3,2] <- sigma[2,3] <- 0.8

## Generate normal random samples
## Not run:
library(MASS)
X <- mvrnorm(200, rep(0, 5), sigma)

## Covariance estimation
gridpts <- kgrid(50, 100)       ## generate grid of penalties to search over
crcov <- select_condreg(X, gridpts)       ## automatically selects penalty parameter

## Inspect output
str(crcov)            ## returned object
sigma.hat <- crcov$S  ## estimate of sigma matrix
omega.hat <- crcov$invS ## estimate of inverse of sigma matrix

## End(Not run)
```
select_kmax

**Description**

Selection of penalty parameter based on cross-validation

**Usage**

```r
select_kmax(X, k, fold = min(nrow(X), 10))
```

**Arguments**

- `X`: n-by-p data matrix
- `k`: vector of penalties for cross-validation
- `fold`: number of folds for cross-validation

---

transcost

**Description**

Compute transaction cost

**Usage**

```r
transcost(wnew, wold, lastearnings, reltc, wealth)
```

**Arguments**

- `wnew`: new portfolio weights
- `wold`: old portfolio weights
- `lastearnings`: earnings from last period
- `reltc`: relative transaction cost
- `wealth`: current wealth

**Value**

transaction cost of rebalancing portfolio
Index

condreg, 2
crbulk, 3
data.snp, 3
kgrid, 3
ml_solver, 4
path_backward, 4
path_forward, 5
pfweights, 5
R, 5
select_condreg, 6
select_kmax, 7
transcost, 7