Package 'cvCovEst'

July 25, 2021

Title Cross-Validated Covariance Matrix Estimation

Version 1.0.0

Description An efficient cross-validated approach for covariance matrix estimation, particularly useful in high-dimensional settings. This method relies upon the theory of loss-based estimator selection to identify the optimal estimator of the covariance matrix from among a prespecified set of candidates.

Depends R (>= 4.0.0)

Imports matrixStats, Matrix, stats, methods, origami, coop, Rdpack, rlang, dplyr, stringr, purrr, tibble, assertthat, RSpectra, ggplot2, ggpubr, RColorBrewer

Suggests future, future.apply, MASS, testthat, knitr, rmarkdown, covr, spelling

License MIT + file LICENSE

URL https://github.com/PhilBoileau/cvCovEst

BugReports https://github.com/PhilBoileau/cvCovEst/issues

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.1.1

RdMacros Rdpack

Language en-US

NeedsCompilation no

Author Philippe Boileau [aut, cre, cph] (<https://orcid.org/0000-0002-4850-2507>), Nima Hejazi [aut] (<https://orcid.org/0000-0002-7127-2789>), Brian Collica [aut] (<https://orcid.org/0000-0003-1127-2557>), Jamarcus Liu [ctb], Mark van der Laan [ctb, ths] (<https://orcid.org/0000-0003-1432-5511>), Sandrine Dudoit [ctb, ths] (<https://orcid.org/0000-0002-6069-8629>)

Maintainer Philippe Boileau <philippe_boileau@berkeley.edu>

Repository CRAN

Date/Publication 2021-07-25 13:30:06 UTC

R topics documented:

`adaptiveLassoEst` *Adaptive LASSO Estimator*

Description

`adaptiveLassoEst()` applied the adaptive LASSO to the entries of the sample covariance matrix. The thresholding function is inspired by the penalized regression introduced by Zou (2006). The thresholding function assigns a weight to each entry of the sample covariance matrix based on its initial value. This weight then determines the relative size of the penalty resulting in larger values being penalized less and reducing bias (Rothman et al. 2009).

Usage

```
adaptiveLassoEst(dat, lambda, n)
```

Arguments

`dat`	A numeric `data.frame`, `matrix`, or similar object.
`lambda`	A non-negative `numeric` defining the amount of thresholding applied to each element of `dat`'s sample covariance matrix.
`n`	A non-negative `numeric` defining the exponent of the adaptive weight applied to each element of `dat`'s sample covariance matrix.

Value

A matrix corresponding to the estimate of the covariance matrix.

References

Rothman AJ, Levina E, Zhu J (2009). "Generalized Thresholding of Large Covariance Matrices." *Journal of the American Statistical Association*, **104**(485), 177–186. doi: 10.1198/jasa.2009.0101, https://doi.org/10.1198/jasa.2009.0101.

Zou H (2006). "The Adaptive Lasso and Its Oracle Properties." *Journal of the American Statistical Association*, **101**(476), 1418–1429. doi: 10.1198/016214506000000735, https://doi.org/10.1198/016214506000000735.

Examples

```
adaptiveLassoEst(dat = mtcars, lambda = 0.9, n = 0.9)
```

---

bandingEst *Banding Estimator*

Description

bandingEst() estimates the covariance matrix of data with ordered variables by forcing off-diagonal entries to be zero for indices that are far removed from one another. The i, j entry of the estimated covariance matrix will be zero if the absolute value of i - j is greater than some non-negative constant k. This estimator was proposed by Bickel and Levina (2008).

Usage

```
bandingEst(dat, k)
```

Arguments

dat	A numeric data.frame, matrix, or similar object.
k	A non-negative, numeric integer.

Value

A matrix corresponding to the estimate of the covariance matrix.

References

Bickel PJ, Levina E (2008). "Regularized estimation of large covariance matrices." *Annals of Statistics*, **36**(1), 199–227. doi: 10.1214/009053607000000758, https://doi.org/10.1214/009053607000000758.

Examples

```
bandingEst(dat = mtcars, k = 2L)
```

cvCovEst *Cross-Validated Covariance Matrix Estimator Selector*

Description

cvCovEst() identifies the optimal covariance matrix estimator from among a set of candidate estimators.

Usage

```
cvCovEst(
  dat,
  estimators = c(linearShrinkEst, thresholdingEst, sampleCovEst),
  estimator_params = list(linearShrinkEst = list(alpha = 0), thresholdingEst =
    list(gamma = 0)),
  cv_loss = cvMatrixFrobeniusLoss,
  cv_scheme = "v_fold",
  mc_split = 0.5,
  v_folds = 10L,
  center = TRUE,
  scale = FALSE,
  parallel = FALSE
)
```

Arguments

dat	A numeric data.frame, matrix, or similar object.
estimators	A list of estimator functions to be considered in the cross-validated estimator selection procedure.
estimator_params	A named list of arguments corresponding to the hyperparameters of covariance matrix estimators in estimators. The name of each list element should match the name of an estimator passed to estimators. Each element of the estimator_params is itself a named list, with the names corresponding to a given estimator's hyperparameter(s). The hyperparameter(s) may be in the form of a single numeric or a numeric vector. If no hyperparameter is needed for a given estimator, then the estimator need not be listed.
cv_loss	A function indicating the loss function to be used. This defaults to the Frobenius loss, cvMatrixFrobeniusLoss(). An observation-based version, cvFrobeniusLoss(), is also made available. Additionally, the cvScaledMatrixFrobeniusLoss() is included for situations in which dat's variables are of different scales.
cv_scheme	A character indicating the cross-validation scheme to be employed. There are two options: (1) V-fold cross-validation, via "v_folds"; and (2) Monte Carlo cross-validation, via "mc". Defaults to Monte Carlo cross-validation.
mc_split	A numeric between 0 and 1 indicating the proportion of observations to be included in the validation set of each Monte Carlo cross-validation fold.

v_folds	An integer larger than or equal to 1 indicating the number of folds to use for cross-validation. The default is 10, regardless of the choice of cross-validation scheme.
center	A logical indicating whether to center the columns of dat to have mean zero.
scale	A logical indicating whether to scale the columns of dat to have unit variance.
parallel	A logical option indicating whether to run the main cross-validation loop with future_lapply(). This is passed directly to cross_validate().

Value

A list of results containing the following elements:

- estimate - A matrix corresponding to the estimate of the optimal covariance matrix estimator.
- estimator - A character indicating the optimal estimator and corresponding hyperparameters, if any.
- risk_df - A tibble providing the cross-validated risk estimates of each estimator.
- cv_df - A tibble providing each estimators' loss over the folds of the cross-validated procedure.
- args - A named list containing arguments passed to cvCovEst.

Examples

```
cvCovEst(
  dat = mtcars,
  estimators = c(
    linearShrinkLWEst, thresholdingEst, sampleCovEst
  ),
  estimator_params = list(
    thresholdingEst = list(gamma = seq(0.1, 0.3, 0.1))
  ),
  center = TRUE,
  scale = TRUE
)
```

---

cvFrobeniusLoss *Cross-Validation Function for Aggregated Frobenius Loss*

Description

cvFrobeniusLoss() evaluates the aggregated Frobenius loss over a fold object (from '**origami**' (Coyle and Hejazi 2018)).

Usage

```
cvFrobeniusLoss(fold, dat, estimator_funs, estimator_params = NULL)
```

Arguments

fold	A fold object (from make_folds()) over which the estimation procedure is to be performed.
dat	A data.frame containing the full (non-sample-split) data, on which the cross-validated procedure is performed.
estimator_funs	An expression corresponding to a vector of covariance matrix estimator functions to be applied to the training data.
estimator_params	A named list of arguments corresponding to the hyperparameters of covariance matrix estimators, estimator_funs. The name of each list element should be the name of an estimator passed to estimator_funs. Each element of the estimator_params is itself a named list, with names corresponding to an estimators' hyperparameter(s). These hyperparameters may be in the form of a single numeric or a numeric vector. If no hyperparameter is needed for a given estimator, then the estimator need not be listed.

Value

A tibble providing information on estimators, their hyperparameters (if any), and their scaled Frobenius loss evaluated on a given fold.

References

Coyle J, Hejazi N (2018). "origami: A Generalized Framework for Cross-Validation in R." *Journal of Open Source Software*, **3**(21), 512. doi: 10.21105/joss.00512, https://doi.org/10.21105/joss.00512.

Examples

```
library(MASS)
library(origami)
library(rlang)

# generate 10x10 covariance matrix with unit variances and off-diagonal
# elements equal to 0.5
Sigma <- matrix(0.5, nrow = 10, ncol = 10) + diag(0.5, nrow = 10)

# sample 50 observations from multivariate normal with mean = 0, var = Sigma
dat <- mvrnorm(n = 50, mu = rep(0, 10), Sigma = Sigma)

# generate a single fold using MC-cv
resub <- make_folds(dat,
  fold_fun = folds_vfold,
  V = 2
)[[1]]
cvFrobeniusLoss(
  fold = resub,
  dat = dat,
  estimator_funs = rlang::quo(c(
    linearShrinkEst, thresholdingEst, sampleCovEst
```

```
  )),
  estimator_params = list(
    linearShrinkEst = list(alpha = c(0, 1)),
    thresholdingEst = list(gamma = c(0, 1))
  )
)
```

`cvMatrixFrobeniusLoss` *Cross-Validation Function for Matrix Frobenius Loss*

Description

`cvMatrixFrobeniusLoss()` evaluates the matrix Frobenius loss over a `fold` object (from '**origami**' (Coyle and Hejazi 2018)). This loss function is equivalent to that presented in `cvFrobeniusLoss()` in terms of estimator selections, but is more computationally efficient.

Usage

```
cvMatrixFrobeniusLoss(fold, dat, estimator_funs, estimator_params = NULL)
```

Arguments

`fold`	A `fold` object (from `make_folds()`) over which the estimation procedure is to be performed.
`dat`	A `data.frame` containing the full (non-sample-split) data, on which the cross-validated procedure is performed.
`estimator_funs`	An `expression` corresponding to a vector of covariance matrix estimator functions to be applied to the training data.
`estimator_params`	A named `list` of arguments corresponding to the hyperparameters of covariance matrix estimators, `estimator_funs`. The name of each list element should be the name of an estimator passed to `estimator_funs`. Each element of the `estimator_params` is itself a named `list`, with names corresponding to an estimators' hyperparameter(s). These hyperparameters may be in the form of a single `numeric` or a `numeric` vector. If no hyperparameter is needed for a given estimator, then the estimator need not be listed.

Value

A `tibble` providing information on estimators, their hyperparameters (if any), and their matrix Frobenius loss evaluated on a given `fold`.

References

Coyle J, Hejazi N (2018). "origami: A Generalized Framework for Cross-Validation in R." *Journal of Open Source Software*, **3**(21), 512. doi: 10.21105/joss.00512, https://doi.org/10.21105/joss.00512.

Examples

```
library(MASS)
library(origami)
library(rlang)

# generate 10x10 covariance matrix with unit variances and off-diagonal
# elements equal to 0.5
Sigma <- matrix(0.5, nrow = 10, ncol = 10) + diag(0.5, nrow = 10)

# sample 50 observations from multivariate normal with mean = 0, var = Sigma
dat <- mvrnorm(n = 50, mu = rep(0, 10), Sigma = Sigma)

# generate a single fold using MC-cv
resub <- make_folds(dat,
  fold_fun = folds_vfold,
  V = 2
)[[1]]
cvMatrixFrobeniusLoss(
  fold = resub,
  dat = dat,
  estimator_funs = rlang::quo(c(
    linearShrinkEst, thresholdingEst, sampleCovEst
  )),
  estimator_params = list(
    linearShrinkEst = list(alpha = c(0, 1)),
    thresholdingEst = list(gamma = c(0, 1))
  )
)
```

`cvScaledMatrixFrobeniusLoss` *Cross-Validation Function for Scaled Matrix Frobenius Loss*

Description

`cvScaledMatrixFrobeniusLoss()` evaluates the scaled matrix Frobenius loss over a `fold` object (from '**origami**' (Coyle and Hejazi 2018)). The squared error loss computed for each entry of the estimated covariance matrix is scaled by the training set's sample variances of the variable associated with that entry's row and column variables. This loss should be used instead of `cvMatrixFrobeniusLoss()` when a dataset's variables' values are of different magnitudes.

Usage

```
cvScaledMatrixFrobeniusLoss(fold, dat, estimator_funs, estimator_params = NULL)
```

Arguments

`fold`	A `fold` object (from `make_folds()`) over which the estimation procedure is to be performed.

`dat` A `data.frame` containing the full (non-sample-split) data, on which the cross-validated procedure is performed.

`estimator_funs` An `expression` corresponding to a vector of covariance matrix estimator functions to be applied to the training data.

`estimator_params`
A named `list` of arguments corresponding to the hyperparameters of covariance matrix estimators, `estimator_funs`. The name of each list element should be the name of an estimator passed to `estimator_funs`. Each element of the `estimator_params` is itself a named `list`, with names corresponding to an estimators' hyperparameter(s). These hyperparameters may be in the form of a single `numeric` or a `numeric` vector. If no hyperparameter is needed for a given estimator, then the estimator need not be listed.

Value

A `tibble` providing information on estimators, their hyperparameters (if any), and their scaled matrix Frobenius loss evaluated on a given `fold`.

References

Coyle J, Hejazi N (2018). "origami: A Generalized Framework for Cross-Validation in R." *Journal of Open Source Software*, **3**(21), 512. doi: 10.21105/joss.00512, https://doi.org/10.21105/joss.00512.

Examples

```
library(MASS)
library(origami)
library(rlang)

# generate 10x10 covariance matrix with unit variances and off-diagonal
# elements equal to 0.5
Sigma <- matrix(0.5, nrow = 10, ncol = 10) + diag(0.5, nrow = 10)

# sample 50 observations from multivariate normal with mean = 0, var = Sigma
dat <- mvrnorm(n = 50, mu = rep(0, 10), Sigma = Sigma)

# generate a single fold using MC-cv
resub <- make_folds(dat,
  fold_fun = folds_vfold,
  V = 2
)[[1]]
cvScaledMatrixFrobeniusLoss(
  fold = resub,
  dat = dat,
  estimator_funs = rlang::quo(c(
    linearShrinkEst, thresholdingEst, sampleCovEst
  )),
  estimator_params = list(
    linearShrinkEst = list(alpha = c(0, 1)),
    thresholdingEst = list(gamma = c(0, 1))
```

```
  )
)
```

`denseLinearShrinkEst` *Linear Shrinkage Estimator, Dense Target*

Description

`denseLinearShrinkEst()` computes the asymptotically optimal convex combination of the sample covariance matrix and a dense target matrix. This target matrix's diagonal elements are equal to the average of the sample covariance matrix estimate's diagonal elements, and its off-diagonal elements are equal to the average of the sample covariance matrix estimate's off-diagonal elements. For information on this estimator's derivation, see Ledoit and Wolf (2020) and Schäfer and Strimmer (2005).

Usage

```
denseLinearShrinkEst(dat)
```

Arguments

`dat`	A numeric `data.frame`, `matrix`, or similar object.

Value

A `matrix` corresponding to the estimate of the covariance matrix.

References

Ledoit O, Wolf M (2020). "The Power of (Non-)Linear Shrinking: A Review and Guide to Covariance Matrix Estimation." *Journal of Financial Econometrics*. ISSN 1479-8409, doi: 10.1093/jjfinec/nbaa007, nbaa007, https://academic.oup.com/jfec/advance-article-pdf/doi/10.1093/jjfinec/nbaa007/33416890/nbaa007.pdf.

Schäfer J, Strimmer K (2005). "A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics." *Statistical Applications in Genetics and Molecular Biology*, **4**(1). doi: 10.2202/15446115.1175, https://www.degruyter.com/view/journals/sagmb/4/1/article-sagmb.2005.4.1.1175.xml.xml.

Examples

```
denseLinearShrinkEst(dat = mtcars)
```

is.cvCovEst *Check for cvCovEst Class*

Description

`is.cvCovEst()` provides a generic method for checking if input is of class `cvCovEst`.

Usage

```
is.cvCovEst(x)
```

Arguments

`x`	The specific object to test.

Value

A `logical` indicating `TRUE` if `x` inherits from class `cvCovEst`.

Examples

```
cv_dat <- cvCovEst(
  dat = mtcars,
  estimators = c(
    thresholdingEst, sampleCovEst
  ),
  estimator_params = list(
    thresholdingEst = list(gamma = seq(0.1, 0.3, 0.1))
  ),
  center = TRUE,
  scale = TRUE
)

is.cvCovEst(cv_dat)
```

linearShrinkEst *Linear Shrinkage Estimator*

Description

`linearShrinkEst()` computes the linear shrinkage estimate of the covariance matrix for a given value of `alpha`. The linear shrinkage estimator is defined as the convex combination of the sample covariance matrix and the identity matrix. The choice of `alpha` determines the bias-variance trade-off of the estimators in this class: values near 1 are more likely to exhibit high variance but low bias, and values near 0 are more likely to be be very biased but have low variance.

Usage

```
linearShrinkEst(dat, alpha)
```

Arguments

dat	A numeric data.frame, matrix, or similar object.
alpha	A numeric between 0 and 1 defining convex combinations of the sample covariance matrix and the identity. alpha = 1 produces the sample covariance matrix, and alpha = 0 returns the identity.

Value

A matrix corresponding to the estimate of the covariance matrix.

Examples

```
linearShrinkEst(dat = mtcars, alpha = 0.1)
```

---

linearShrinkLWEst *Ledoit-Wolf Linear Shrinkage Estimator*

Description

linearShrinkLWEst() computes an asymptotically optimal convex combination of the sample covariance matrix and the identity matrix. This convex combination effectively shrinks the eigenvalues of the sample covariance matrix towards the identity. This estimator is more accurate than the sample covariance matrix in high-dimensional settings under fairly loose assumptions. For more information, consider reviewing the manuscript by Ledoit and Wolf (2004).

Usage

```
linearShrinkLWEst(dat)
```

Arguments

dat	A numeric data.frame, matrix, or similar object.

Value

A matrix corresponding to the Ledoit-Wolf linear shrinkage estimate of the covariance matrix.

References

Ledoit O, Wolf M (2004). "A well-conditioned estimator for large-dimensional covariance matrices." *Journal of Multivariate Analysis*, **88**(2), 365–411. ISSN 0047-259X, doi: 10.1016/S0047-259X(03)000964, https://www.sciencedirect.com/science/article/pii/S0047259X03000964.

Examples

```
linearShrinkLWEst(dat = mtcars)
```

`nlShrinkLWEst` *Analytical Non-Linear Shrinkage Estimator*

Description

`nlShrinkLWEst()` invokes the analytical estimator presented by Ledoit and Wolf (2018) for applying a nonlinear shrinkage function to the sample eigenvalues of the covariance matrix. The shrinkage function relies on an application of the Hilbert Transform to an estimate of the sample eigenvalues' limiting spectral density. This estimated density is computed with the Epanechnikov kernel using a global bandwidth parameter of `n^(-1/3)`. The resulting shrinkage function pulls eigenvalues towards the nearest mode of their empirical distribution, thus creating a localized shrinkage effect rather than a global one.

We do not recommend that this estimator be employed when the estimand is the correlation matrix. The diagonal entries of the resulting estimate are not guaranteed to be equal to one.

Usage

```
nlShrinkLWEst(dat)
```

Arguments

`dat`	A numeric `data.frame`, `matrix`, or similar object.

Value

A `matrix` corresponding to the estimate of the covariance matrix.

References

Ledoit O, Wolf M (2018). "Analytical nonlinear shrinkage of large-dimensional covariance matrices." Technical Report 264, Department of Economics - University of Zurich. https://EconPapers.repec.org/RePEc:zur:econwp:264.

Examples

```
nlShrinkLWEst(dat = mtcars)
```

plot.cvCovEst *Generic Plot Method for cvCovEst*

Description

The plot method is a generic method for plotting objects of class, "cvCovEst". The method is designed as a tool for diagnostic and exploratory analysis purposes when selecting a covariance matrix estimator using cvCovEst.

Usage

```
## S3 method for class 'cvCovEst'
plot(
  x,
  dat_orig,
  estimator = NULL,
  plot_type = c("summary"),
  stat = c("min"),
  k = NULL,
  leading = TRUE,
  abs_v = TRUE,
  switch_vars = FALSE,
  min_max = FALSE,
  ...
)
```

Arguments

x	An object of class, "cvCovEst". Specifically, this is the standard output of the function cvCovEst.
dat_orig	The numeric data.frame, matrix, or similar object originally passed to cvCovEst.
estimator	A character vector specifying one or more classes of estimators to compare. If NULL, the class of estimator associated with optimal cvCovEst selection is used.
plot_type	A character vector specifying one of four choices of diagnostic plots. Default is "summary". See Details for more about each plotting choice.
stat	A character vector of one or more summary statistics to use when comparing estimators. Default is "min" for minimum cross-validated risk. See Details for more options.
k	A integer indicating the number of leading/trailing eigenvalues to plot. If NULL, will default to the number of columns in dat_orig.
leading	A logical indicating if the leading eigenvalues should be used. Default is TRUE. If FALSE, the trailing eigenvalues are used instead.
abs_v	A logical determining if the absolute value of the matrix entries should be used for plotting the matrix heat map. Default is TRUE.

switch_vars	A logical. If TRUE, the hyperparameters used for the x-axis and factor variables are switched in the plot of the cross-validated risk. Only applies to estimators with more than one hyperparameter. Default is FALSE.
min_max	A logical. If TRUE, only the minimum and maximum values of the factor hyperparameter will be used. Only applies to estimators with more than one hyperparameter. Default is FALSE.
...	Additional arguments passed to the plot method. These are not explicitly used and should be ignored by the user.

Details

This plot method is designed to aide users in understanding the estimation procedure carried out in cvCovEst(). There are currently four different values for plot_type that can be called:

- "eigen" - Plots the eigenvalues associated with the specified estimator and stat arguments in decreasing order.
- "risk" - Plots the cross-validated risk of the specified estimator as a function of the hyperparameter values passed to cvCovEst(). This type of plot is only compatible with estimators which take hyperparameters as arguments.
- "heatmap" - Plots a covariance heat map associated with the specified estimator and stat arguments. Multiple estimators and performance stats may be specified to produce grids of heat maps.
- "summary" - Specifying this plot type will run all of the above plots for the best performing estimator selected by cvCovEst(). These plots are then combined into a single panel along with a table containing the best performing estimator within each class. If the optimal estimator selected by cvCovEst() does not have hyperparameters, then the risk plot is replaced with a table displaying the minimum, first quartile, median, third quartile, and maximum of the cross-validated risk associated with each class of estimator.

The stat argument accepts five values. They each correspond to a summary statistic of the cross-validated risk distribution within a class of estimator. Possible values are:

- "min" - minimum
- "Q1" - first quartile
- "median" - median
- "Q3" - third quartile
- "max" - maximum

Value

A plot object

Examples

```
cv_dat <- cvCovEst(
  dat = mtcars,
  estimators = c(
    thresholdingEst, sampleCovEst
```

```
  ),
  estimator_params = list(
    thresholdingEst = list(gamma = seq(0.1, 0.9, 0.1))
  ),
  center = TRUE,
  scale = TRUE
)

plot(x = cv_dat, dat_orig = mtcars)
```

`poetEst` *POET Estimator*

Description

`poetEst()` implements the Principal Orthogonal complEment Thresholding (POET) estimator, a nonparametric, unobserved-factor-based estimator of the covariance matrix (Fan et al. 2013). The estimator is defined as the sum of the sample covariance matrix' rank-k approximation and its post-thresholding principal orthogonal complement. The hard thresholding function is used here, though others could be used instead.

Usage

```
poetEst(dat, k, lambda)
```

Arguments

`dat`	A numeric `data.frame`, `matrix`, or similar object.
`k`	An `integer` indicating the number of unobserved latent factors. Empirical evidence suggests that the POET estimator is robust to overestimation of this hyperparameter (Fan et al. 2013). In practice, it is therefore preferable to use larger values.
`lambda`	A non-negative `numeric` defining the amount of thresholding applied to each element of sample covariance matrix's orthogonal complement.

Value

A `matrix` corresponding to the estimate of the covariance matrix.

References

Fan J, Liao Y, Mincheva M (2013). "Large covariance estimation by thresholding principal orthogonal complements." *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, **75**(4), 603–680. ISSN 13697412, 14679868, https://www.jstor.org/stable/24772450.

Examples

```
poetEst(dat = mtcars, k = 2L, lambda = 0.1)
```

`robustPoetEst` *Robust POET Estimator for Elliptical Distributions*

Description

`robustPoetEst()` implements the robust version of Principal Orthogonal complEment Thresholding (POET) estimator, a nonparametric, unobserved-factor-based estimator of the covariance matrix when the underlying distribution is elliptical (Fan et al. 2018). The estimator is defined as the sum of the sample covariance matrix's rank-k approximation and its post-thresholding principal orthogonal complement. The rank-k approximation is constructed from the sample covariance matrix, its leading eigenvalues, and its leading eigenvectors. The sample covariance matrix and leading eigenvalues are initially estimated via an M-estimation procedure and the marginal Kendall's tau estimator. The leading eigenvectors are estimated using spatial Kendall's tau estimator. The hard thresholding function is used to regularize the idiosyncratic errors' estimated covariance matrix, though other regularization schemes could be used.

We do not recommend that this estimator be employed when the estimand is the correlation matrix. The diagonal entries of the resulting estimate are not guaranteed to be equal to one.

Usage

```
robustPoetEst(dat, k, lambda, var_est = c("sample", "mad", "huber"))
```

Arguments

`dat`	A numeric `data.frame`, `matrix`, or similar object.
`k`	An `integer` indicating the number of unobserved latent factors. Empirical evidence suggests that the POET estimator is robust to overestimation of this hyperparameter (Fan et al. 2013). In practice, it is therefore preferable to use larger values.
`lambda`	A non-negative `numeric` defining the amount of thresholding applied to each element of sample covariance matrix's orthogonal complement.
`var_est`	A `character` dictating which variance estimator to use. This must be one of the strings `"sample"`, `"mad"`, or `"huber"`. `"sample"` uses sample variances; `"mad"` estimates variances via median absolute deviation; `"huber"` uses an M-estimator for variance under the Huber loss.

Value

A `matrix` corresponding to the estimate of the covariance matrix.

References

Fan J, Liao Y, Mincheva M (2013). "Large covariance estimation by thresholding principal orthogonal complements." *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, **75**(4), 603–680. ISSN 13697412, 14679868, https://www.jstor.org/stable/24772450.

Fan J, Liu H, Wang W (2018). "Large covariance estimation through elliptical factor models." *Ann. Statist.*, **46**(4), 1383–1414. doi: 10.1214/17AOS1588, https://doi.org/10.1214/17-AOS1588.

Examples

```
robustPoetEst(dat = mtcars, k = 2L, lambda = 0.1, var_est = "sample")
```

`sampleCovEst` *Sample Covariance Matrix*

Description

`sampleCovEst()` computes the sample covariance matrix. This function is a simple wrapper around `covar()`.

Usage

```
sampleCovEst(dat)
```

Arguments

`dat`	A numeric `data.frame`, `matrix`, or similar object.

Value

A `matrix` corresponding to the estimate of the covariance matrix.

Examples

```
sampleCovEst(dat = mtcars)
```

`scadEst` *Smoothly Clipped Absolute Deviation Estimator*

Description

`scadEst()` applies the SCAD thresholding function of Fan and Li (2001) to each entry of the sample covariance matrix. This penalized estimator constitutes a compromise between hard and soft thresholding of the sample covariance matrix: it is a linear interpolation between soft thresholding up to `2 * lambda` and hard thresholding after `3.7 * lambda` (Rothman et al. 2009).

Usage

```
scadEst(dat, lambda)
```

Arguments

dat	A numeric data.frame, matrix, or similar object.
lambda	A non-negative numeric defining the degree of thresholding applied to each element of dat's sample covariance matrix.

Value

A matrix corresponding to the estimate of the covariance matrix.

References

Fan J, Li R (2001). "Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties." *Journal of the American Statistical Association*, **96**(456), 1348–1360. doi: 10.1198/016214501753382273, https://doi.org/10.1198/016214501753382273.

Rothman AJ, Levina E, Zhu J (2009). "Generalized Thresholding of Large Covariance Matrices." *Journal of the American Statistical Association*, **104**(485), 177–186. doi: 10.1198/jasa.2009.0101, https://doi.org/10.1198/jasa.2009.0101.

Examples

```
scadEst(dat = mtcars, lambda = 0.2)
```

---

summary.cvCovEst *Generic Summary Method for cvCovEst*

Description

summary() provides summary statistics regarding the performance of cvCovEst() and can be used for diagnostic plotting.

Usage

```
## S3 method for class 'cvCovEst'
summary(
  object,
  summ_fun = c("cvRiskByClass", "bestInClass", "worstInClass", "hyperRisk"),
  ...
)
```

Arguments

object	A named list of class "cvCovEst".
summ_fun	A character vector specifying which summaries to output. See Details for function descriptions.
...	Additional arguments passed to summary()These are not explicitly used and should be ignored by the user.

Details

summary() accepts four different choices for the summ_fun argument. The choices are:

- "cvRiskByClass" - Returns the minimum, first quartile, median, third quartile, and maximum of the cross-validated risk associated with each class of estimator passed to cvCovEst().
- "bestInClass" - Returns the specific hyperparameters, if applicable, of the best performing estimator within each class.
- "worstInClass" - Returns the specific hyperparameters, if applicable, of the worst performing estimator within each class.
- "hyperRisk" - For estimators that take hyperparameters as arguments, this function returns the hyperparameters associated with the minimum, first quartile, median, third quartile, and maximum of the cross-validated risk within each class of estimator. Each class has its own tibble, which are returned as a list.

Value

A named list where each element corresponds to the output of of the requested summaries.

Examples

```
cv_dat <- cvCovEst(
  dat = mtcars,
  estimators = c(
    linearShrinkEst, thresholdingEst, sampleCovEst
  ),
  estimator_params = list(
    linearShrinkEst = list(alpha = seq(0.1, 0.9, 0.1)),
    thresholdingEst = list(gamma = seq(0.1, 0.9, 0.1))
  ),
  center = TRUE,
  scale = TRUE
)

summary(cv_dat)
```

---

taperingEst *Tapering Estimator*

Description

taperingEst() estimates the covariance matrix of a data.frame-like object with ordered variables by gradually shrinking the bands of the sample covariance matrix towards zero. The estimator is defined as the Hadamard product of the sample covariance matrix and a weight matrix. The amount of shrinkage is dictated by the weight matrix and is specified by a hyperparameter k. This estimator is attributed to Cai et al. (2010).

The weight matrix is a Toeplitz matrix with entries defined as follows. Let i and j index the rows and columns of the weight matrix, respectively. If abs(i - j) <= k / 2, then entry i, j in the weight matrix is equal to 1. If k / 2 < abs(i - j) < k, then entry i, j is equal to 2 - 2 * abs(i - j) / k. Otherwise, entry i, j is equal to 0.

Usage

```
taperingEst(dat, k)
```

Arguments

dat	A numeric data.frame, matrix, or similar object.
k	A non-negative, even numeric integer.

Value

A matrix corresponding to the estimate of the covariance matrix.

References

Cai TT, Zhang C, Zhou HH (2010). "Optimal rates of convergence for covariance matrix estimation." *Ann. Statist.*, **38**(4), 2118–2144. doi: 10.1214/09AOS752, https://doi.org/10.1214/09-AOS752.

Examples

```
taperingEst(dat = mtcars, k = 0.1)
```

---

thresholdingEst *Hard Thresholding Estimator*

Description

thresholdingEst() computes the hard thresholding estimate of the covariance matrix for a given value of gamma. The threshold estimator of the covariance matrix applies a hard thresholding operator to each element of the sample covariance matrix. For more information on this estimator, review Bickel and Levina (2008).

Usage

```
thresholdingEst(dat, gamma)
```

Arguments

dat	A numeric data.frame, matrix, or similar object.
gamma	A non-negative numeric defining the degree of hard thresholding applied to each element of dat's sample covariance matrix.

Value

A matrix corresponding to the estimate of the covariance matrix.

References

Bickel PJ, Levina E (2008). "Covariance regularization by thresholding." *Annals of Statistics*, **36**(6), 2577–2604. doi: 10.1214/08AOS600, https://doi.org/10.1214/08-AOS600.

Examples

```
thresholdingEst(dat = mtcars, gamma = 0.2)
```

Index