Package 'EESPCA'

January 20, 2025

Type Package

Title Eigenvectors from Eigenvalues Sparse Principal Component Analysis (EESPCA)

Version 0.7.0

Author H. Robert Frost

Maintainer H. Robert Frost <rob.frost@dartmouth.edu>

Description Contains logic for computing sparse principal components via the EESPCA method, which is based on an approximation of the eigenvector/eigenvalue identity. Includes logic to support execution of the TPower and rifle sparse PCA methods, as well as logic to estimate the sparsity parameters used by EESPCA, TPower and rifle via cross-validation to minimize the out-of-sample reconstruction error. H. Robert Frost (2021) <doi:10.1080/10618600.2021.1987254>.

Depends R (>= 3.6.0), rifle (>= 1.0.0), MASS, PMA

License GPL (>= 2)

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2022-06-15 20:10:02 UTC

Contents

`EESPCA-package` *Eigenvectors*

Description

Implementation of Eigenvectors from Eigenvalues Sparse Principal Component Analysis (EESPCA).

Details

Package:	EESPCA
Type:	Package
Version:	0.7.0
Date:	2021
License:	GPL-2

Note

This work was supported by the National Institutes of Health grants K01LM012426, R21CA253408, P20GM130454,and P30CA023108.

Author(s)

H. Robert Frost

References

- Frost, H. R. (2022). Eigenvectors from Eigenvalues Sparse Principal Component Analysis (EESPCA). Journal of Computational and Graphical Statistics.

`computeApproxNormSquaredEigenvector` *Approximates the normed squared eigenvector loadings*

Description

Approximates the normed squared eigenvector loadings using a simplified version of the formula associating normed squared eigenvector loadings with the eigenvalues of the full matrix and sub-matrices.

Usage

```
computeApproxNormSquaredEigenvector(cov.X, v1, lambda1, max.iter=5,
    lambda.diff.threshold=1e-6, trace=FALSE)
```

Arguments

`cov.X`	Covariance matrix.
`v1`	Principal eigenvector of `cov.X`, i.e, the loadings of the first PC.
`lambda1`	Largest eigenvalue of `cov.X`.
`max.iter`	Maximum number of iterations for power iteration method when computing sub-matrix eigenvalues. See description `powerIteration`.
`lambda.diff.threshold`	Threshold for exiting the power iteration calculation. See description `powerIteration`.
`trace`	True if debugging messages should be displayed during execution.

Value

Vector of approximate normed squared eigenvector loadings.

See Also

`eespca`,`powerIteration`

Examples

```
set.seed(1)
# Simulate 10x5 MVN data matrix
X=matrix(rnorm(50), nrow=10)
# Estimate covariance matrix
cov.X = cov(X)
# Compute eigenvectors/values
eigen.out = eigen(cov.X)
v1 = eigen.out$vectors[,1]
lambda1 = eigen.out$values[1]
# Print true squared loadings
v1^2
```

```
# Compute approximate normed squared eigenvector loadings
computeApproxNormSquaredEigenvector(cov.X=cov.X, v1=v1,
    lambda1=lambda1)
```

---

computeResidualMatrix *Calculates the residual matrix from the reduced rank reconstruction*

Description

Utility function for computing the residual matrix formed by subtracting from X a reduced rank approximation of matrix X generated from the top k principal components contained in matrix V.

Usage

```
computeResidualMatrix(X,V,center=TRUE)
```

Arguments

X	An n-by-p data matrix whose top k principal components are contained in the p-by-k matrix V.
V	A p-by-k matrix containing the loadings for the top k principal components of X.
center	If true (the default), X will be mean-centered before the residual matrix is computed. If the PCs in V were computed via SVD on a mean-centered matrix or via eigen-decomposition of the sample covariance matrix, this should be set to true.

Value

Residual matrix.

Examples

```
set.seed(1)
# Simulate 10x5 MVN data matrix
X=matrix(rnorm(50), nrow=10)
# Perform PCA
prcomp.out = prcomp(X)
# Get rank 2 residual matrix
computeResidualMatrix(X=X, V=prcomp.out$rotation[,1:2])
```

`eespca` *Eigenvectors from Eigenvalues Sparse Principal Component Analysis (EESPCA)*

Description

Computes the first sparse principal component of the specified data matrix using the Eigenvectors from Eigenvalues Sparse Principal Component Analysis (EESPCA) method.

Usage

```
eespca(X, max.iter=20, sparse.threshold, lambda.diff.threshold=1e-6,
    compute.sparse.lambda=FALSE, sub.mat.max.iter=5, trace=FALSE)
```

Arguments

`X`	An n-by-p data matrix for which the first sparse PC will be computed.
`max.iter`	Maximum number of iterations for power iteration method. See `powerIteration`.
`sparse.threshold`	Threshold on loadings used to induce sparsity. Loadings below this value are set to 0. If not specified, defaults to `1/sqrt(p)`.
`lambda.diff.threshold`	Threshold for exiting the power iteration calculation. If the absolute relative difference in lambda is less than this threshold between subsequent iterations, the power iteration method is terminated. See `powerIteration`.
`compute.sparse.lambda`	If true, the sparse loadings will be used to compute the sparse eigenvalue.
`sub.mat.max.iter`	Maximum iterations for computation of sub-matrix eigenvalues using the power iteration method. To maximize performance, set to 1. Uses the same lambda.diff.threshold.
`trace`	True if debugging messages should be displayed during execution.

Value

A `list` with the following elements:

- "v1": The first non-sparse PC as calculated via power iteration.
- "lambda1": The variance of the first non-sparse PC as calculated via power iteration.
- "v1.sparse": First sparse PC.
- "lambda1.sparse": Variance of the first sparse PC. NA if compute.sparse.lambda is FALSE.
- "ratio": Vector of ratios of the sparse to non-sparse PC loadings.

References

- Frost, H. R. (2021). Eigenvectors from Eigenvalues Sparse Principal Component Analysis (EESPCA). arXiv e-prints. https://arxiv.org/abs/2006.01924

See Also

eespcaForK,computeApproxNormSquaredEigenvector, powerIteration

Examples

```
set.seed(1)
# Simulate 10x5 MVN data matrix
X=matrix(rnorm(50), nrow=10)
# Compute first sparse PC loadings using default threshold
eespca(X=X)
```

---

eespcaCV *Cross-validation for Eigenvectors from Eigenvalues Sparse Principal Component Analysis (EESPCA)*

Description

Performs cross-validation of EESPCA to determine the optimal sparsity threshold. Selection is based on the minimization of reconstruction error. Based on the cross-validation approach of Witten et al. as implemented by the SPC.cv method in the PMA package.

Usage

```
eespcaCV(X, max.iter=20, sparse.threshold.values, nfolds=5,
    lambda.diff.threshold=1e-6, compute.sparse.lambda=FALSE,
    sub.mat.max.iter=5, trace=FALSE)
```

Arguments

X	See description for eespca
max.iter	See description for eespca
sparse.threshold.values	Vector of threshold values to evaluate via cross-validation. See description for eespca for details.
nfolds	Number of cross-validation folds.
lambda.diff.threshold	See description for eespca
compute.sparse.lambda	See description for eespca
sub.mat.max.iter	See description for eespca
trace	See description for eespca

Value

A `list` with the following elements:

- "cv": The mean of the out-of-sample reconstruction error computed for each threshold.
- "cv.error": The standard deviations of the means of the out-of-sample reconstruction error computed for each threshold.
- "best.sparsity": Threshold value with the lowest mean reconstruction error.
- "best.sparsity.1se": Threshold value whose mean reconstruction error is within 1 standard error of the lowest.
- "nonzerovs": Mean number of nonzero values for each threshold.
- "sparse.threshold.values": Tested threshold values.
- "nfolds": Number of cross-validation folds.

References

- Frost, H. R. (2021). Eigenvectors from Eigenvalues Sparse Principal Component Analysis (EESPCA). arXiv e-prints. https://arxiv.org/abs/2006.01924
- Witten, D. M., Tibshirani, R., and Hastie, T. (2009). A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. Biostatistics, 10(3), 515-534.

See Also

`eespca`, `PMA{SPC.cv}`

Examples

```
 set.seed(1)
 # Simulate 10x5 MVN data matrix
 X=matrix(rnorm(50), nrow=10)
 # Generate range of threshold values to evaluate
 default.threshold = 1/sqrt(5)
threshold.values = seq(from=.5*default.threshold, to=1.5*default.threshold, length.out=10)
 # Use 5-fold cross-validation to estimate optimal sparsity threshold
 eespcaCV(X=X, sparse.threshold.values=threshold.values)
```

`eespcaForK` *Multi-PC version of Eigenvectors from Eigenvalues Sparse Principal Component Analysis (EESPCA)*

Description

Computes multiple sparse principal components of the specified data matrix via sequential application of the Eigenvectors from Eigenvalues Sparse Principal Component Analysis (EESPCA) algorithm. After computing the first sparse PC via the `eespca` function, subsequent sparse PCs are computing by repeatedly applying `eespca` to the residual matrix formed by subtracting the reconstruction of `X` from the original `X`. Multiple sparse PCs are not guaranteed to be orthogonal.

Note that the accuracy of the sparse approximation declines substantially for PCs with very small variances. To avoid this issue, `k` should not be set higher than the number of statistically significant PCs according to a Tracey-Widom test.

Usage

```
eespcaForK(X, k=2, max.iter=20, sparse.threshold, lambda.diff.threshold=1e-6,
     compute.sparse.lambda=FALSE, sub.mat.max.iter=5, trace=FALSE)
```

Arguments

`X`	An n-by-p data matrix for which the first k sparse PCs will be computed.
`k`	The number of sparse PCs to compute. The specified k must be 2 or greater (for k=1, use the `eespca` method). A check is made that k is not greater than the maximum theoretical rank of X but, for performance reasons, a check is NOT made that k is less than or equal to the actual rank of X.
`max.iter`	See description for `eespca`
`sparse.threshold`	See description for `eespca`
`lambda.diff.threshold`	See description for `eespca`
`compute.sparse.lambda`	See description for `eespca`
`sub.mat.max.iter`	See description for `eespca`
`trace`	See description for `eespca`

Value

A `list` with the following elements:

- "V": Matrix of sparse loadings for the first k PCs.
- "lambdas": Vector of variances of the first k sparse PCs.

References

- Frost, H. R. (2021). Eigenvectors from Eigenvalues Sparse Principal Component Analysis (EESPCA). arXiv e-prints. https://arxiv.org/abs/2006.01924

See Also

`eespca`

Examples

```
set.seed(1)
# Simulate 10x5 MVN data matrix
X=matrix(rnorm(50), nrow=10)
# Get first two sparse PCs
eespcaForK(X=X, sparse.threshold=1/sqrt(5), k=2)
```

`powerIteration` *Power iteration method for calculating principal eigenvector and eigenvalue.*

Description

Computes the principal eigenvector and eigenvalue of the specified matrix using the power iteration method. Includes support for truncating the estimated eigenvector on each iteration to retain just the k eigenvector loadings with the largest absolute values with all other values set to 0, i.e., the the TPower method by Yuan & Zhang.

Usage

```
powerIteration(X, k, v1.init, max.iter=10, lambda.diff.threshold=1e-6, trace=FALSE)
```

Arguments

`X`	Matrix for which the largest eigenvector and eigenvalue will be computed.
`k`	If specified, the estimated eigenvector is truncated on each iteration to retain only the k loadings with the largest absolute values, all other loadings are set to 0. Must be an integer between 1 and ncol(X).
`v1.init`	If specified, the power iteration calculation will be initialized using this vector, otherwise, the calculation will be initialized using a unit vector with equal values.
`max.iter`	Maximum number of iterations for power iteration method.
`lambda.diff.threshold`	Threshold for exiting the power iteration calculation. If the absolute relative difference in computed eigenvalue is less than this threshold between subsequent iterations, the power iteration method is terminated.
`trace`	True if debugging messages should be displayed during execution.

Value

A `list` with the following elements:

- "v1": The principal eigenvector of `X`.
- "lambda": The largest eigenvalue of `X`.
- "num.iter": Number of iterations of the power iteration method before termination.

References

- Yuan, X.-T. and Zhang, T. (2013). Truncated power method for sparse eigenvalue problems. J. Mach. Learn. Res., 14(1), 899-925.

See Also

eespca

Examples

```
set.seed(1)
# Simulate 10x5 MVN data matrix
X=matrix(rnorm(50), nrow=10)
# Compute sample covariance matrix
cov.X = cov(X)
# Use power iteration to get first PC loadings using default initial vector
powerIteration(X=cov.X)
```

---

reconstruct *Calculates the reduced rank reconstruction*

Description

Utility function for computing the reduced rank reconstruction of X using the PC loadings in V.

Usage

```
reconstruct(X,V,center=TRUE)
```

Arguments

X	An n-by-p data matrix whose top k principal components are contained the p-by-k matrix V.
V	A p-by-k matrix containing the loadings for the top k principal components of X.
center	If true (the default), X will be mean-centered before the reconstruction is computed. If the PCs in V were computed via SVD on a mean-centered matrix or via eigen-decomposition of the sample covariance matrix, this should be set to true.

Value

Reduced rank reconstruction of X.

Examples

```
set.seed(1)
# Simulate 10x5 MVN data matrix
X=matrix(rnorm(50), nrow=10)
# Perform PCA
prcomp.out = prcomp(X)
# Get rank 2 reconstruction
reconstruct(X, prcomp.out$rotation[,1:2])
```

`reconstructionError` *Calculates the reduced rank reconstruction error*

Description

Utility function for computing the squared Frobenius norm of the residual matrix formed by subtracting from `X` a reduced rank approximation of matrix `X` generated from the top k principal components contained in matrix `V`.

Usage

```
reconstructionError(X,V,center=TRUE)
```

Arguments

`X`	An n-by-p data matrix whose top k principal components are contained the p-by-k matrix `V`.
`V`	A p-by-k matrix containing the loadings for the top k principal components of `X`.
`center`	If true (the default), `X` will be mean-centered before the reconstruction error is computed. If the PCs in `V` were computed via SVD on a mean-centered matrix or via eigen-decomposition of the sample covariance matrix, this should be set to true.

Value

The squared Frobenius norm of the residual matrix.

Examples

```
set.seed(1)
# Simulate 10x5 MVN data matrix
X=matrix(rnorm(50), nrow=10)
# Perform PCA
prcomp.out = prcomp(X)
# Get rank 2 reconstruction error, which will be the minimum since the first 2 PCs are used
reconstructionError(X, prcomp.out$rotation[,1:2])
# Use all PCs to get approximately 0 reconstruction error
reconstructionError(X, prcomp.out$rotation)
```

`rifleInit` *Computes the initial eigenvector for the rifle method of Tan et al.*

Description

Computes the initial eigenvector for the rifle method of Tan et al. (as implemented by the `rifle` method in the rifle R package) using the `initial.convex` method from the rifle package with lambda=sqrt(log(p)/n) and K=1.

Usage

```
rifleInit(X)
```

Arguments

`X`	n-by-p data matrix to be evaluated via PCA.

Value

Initial eigenvector to use with rifle method.

References

- Tan, K. M., Wang, Z., Liu, H., and Zhang, T. (2018). Sparse generalized eigenvalue problem: optimal statistical rates via truncated rayleigh flow. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 80(5), 1057-1086.

See Also

`riflePCACV`, `rifle{rifle}`, `rifle{initial.convex}`

Examples

```
set.seed(1)
# Simulate 10x5 MVN data matrix
X=matrix(rnorm(50), nrow=10)
# Compute initial eigenvector to use with rifle method
v1.init = rifleInit(X)
# Use with rifle method to get first PC loadings with 2 non-zero elements
rifle(A=cov(X), B=diag(5), init=v1.init, k=2)
```

`riflePCACV` *Sparsity parameter selection via cross-validation for rifle method of Tan et al.*

Description

Sparsity parameter selection for PCA-based rifle (as implemented by the `rifle` method in the rifle package) using the cross-validation approach of Witten et al. as implemented by the `SPC.cv` method in the PMA package.

Usage

```
riflePCACV(X, k.values, nfolds=5)
```

Arguments

`X`	n-by-p data matrix being evaluated via PCA.
`k.values`	Set of truncation parameter values to evaluate via cross-validation. Values must be between 1 and p.
`nfolds`	Number of folds for cross-validation

Value

k value that generated the smallest cross-validation error.

References

- Tan, K. M., Wang, Z., Liu, H., and Zhang, T. (2018). Sparse generalized eigenvalue problem: optimal statistical rates via truncated rayleigh flow. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 80(5), 1057-1086.
- Witten, D. M., Tibshirani, R., and Hastie, T. (2009). A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. Biostatistics, 10(3), 515-534.

See Also

`rifleInit`, `rifle{rifle}`, `PMA{SPC.cv}`

Examples

```
set.seed(1)
# Simulate 10x5 MVN data matrix
X=matrix(rnorm(50), nrow=10)
# Generate range of k values to evaluate
k.values = 1:5
# Use 5-fold cross-validation to estimate optimal k value
riflePCACV(X=X, k.values=k.values)
```

tpower *Implementation of the Yuan and Zhang TPower method.*

Description

Implements the TPower method by Yuan and Zhang. Specifically, it computes the sparse principal eigenvector using power iteration method where the estimated eigenvector is truncated on each iteration to retain just the k eigenvector loadings with the largest absolute values with all other values set to 0.

Usage

```
tpower(X, k, v1.init, max.iter=10, lambda.diff.threshold=1e-6, trace=FALSE)
```

Arguments

X	Matrix for which the largest eigenvector and eigenvalue will be computed.
k	Must be an integer between 1 and ncol(X). The estimated eigenvector is truncated on each iteration to retain only the k loadings with the largest absolute values, all other loadings are set to 0.
v1.init	If specified, the power iteration calculation will be initialized using this vector, otherwise, the calculation will be initialized using a unit vector with equal values.
max.iter	Maximum number of iterations for power iteration method.
lambda.diff.threshold	Threshold for exiting the power iteration calculation. If the absolute relative difference in computed eigenvalues is less than this threshold between subsequent iterations, the power iteration method is terminated.
trace	True if debugging messages should be displayed during execution.

Value

The estimated sparse principal eigenvector.

References

- Yuan, X.-T. and Zhang, T. (2013). Truncated power method for sparse eigenvalue problems. J. Mach. Learn. Res., 14(1), 899-925.

See Also

powerIteration,tpowerPCACV

Examples

```
set.seed(1)
# Simulate 10x5 MVN data matrix
X=matrix(rnorm(50), nrow=10)
# Compute first sparse PC loadings with 2 non-zero elements
tpower(X=cov(X), k=2)
```

`tpowerPCACV` *Sparsity parameter selection for the Yuan and Zhang TPower method using cross-validation.*

Description

Sparsity parameter selection for PCA-based TPower using the cross-validation approach of Witten et al. as implemented by the `SPC.cv` method in the PMA package.

Usage

```
tpowerPCACV(X, k.values, nfolds=5)
```

Arguments

`X`	n-by-p data matrix being evaluated via PCA.
`k.values`	Set of truncation parameter values to evaluate via cross-validation. Values must be between 1 and p.
`nfolds`	Number of folds for cross-validation

Value

k value that generated the smallest cross-validation error.

References

- Yuan, X.-T. and Zhang, T. (2013). Truncated power method for sparse eigenvalue problems. J. Mach. Learn. Res., 14(1), 899-925.
- Witten, D. M., Tibshirani, R., and Hastie, T. (2009). A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. Biostatistics, 10(3), 515-534.

See Also

`tpower`,`PMA{SPC.cv}`

Examples

```
set.seed(1)
# Simulate 10x5 MVN data matrix
X=matrix(rnorm(50), nrow=10)
# Generate range of k values to evaluate
k.values = 1:5
# Use 5-fold cross-validation to estimate optimal k value
tpowerPCACV(X=X, k.values=k.values)
```

Index