Package 'msPCA'

August 25, 2026

Type Package

Title Sparse Principal Component Analysis with Multiple Principal Components

Version 0.5.1

Date 2026-08-25

Description Implements an algorithm for computing multiple sparse principal components of a dataset. The method is based on Cory-Wright and Pauphilet ``Sparse PCA with Multiple Components'' (2026) <doi:10.1287/opre.2023.0598>. The algorithm uses an iterative deflation heuristic with a truncated power method applied at each iteration to compute sparse principal components with controlled sparsity.

License MIT + file LICENSE

URL https://jeanpauphilet.github.io/msPCA/

BugReports https://github.com/jeanpauphilet/msPCA/issues

Depends R (>= 3.5)

Imports Rcpp (>= 1.0.11)

Suggests covr, datasets, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

LinkingTo Rcpp, RcppEigen

Encoding UTF-8

LazyData true

LazyDataCompression xz

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Ryan Cory-Wright [aut, cph] (ORCID: <https://orcid.org/0000-0002-4485-0619>), Jean Pauphilet [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-6352-0984>)

Maintainer Jean Pauphilet <jpauphilet@london.edu>

Repository CRAN

Date/Publication 2026-08-25 10:40:08 UTC

Contents

`feasibility_violation_off` *Feasibility Violation*

Description

Computes the feasibility violation defined as $\sum_{t>s} |u_t^\top u_s|$ if orthogonality constraints are enforced (`feasibilityConstraintType = 0`) and $\sum_{t>s} |u_t^\top C u_s| / \mathrm{tr}(C)$ if zero-correlation constraints are enforced (`feasibilityConstraintType = 1`).

Usage

```
feasibility_violation_off(C, U, feasibilityConstraintType)
```

Arguments

`C`	A matrix. The correlation or covariance matrix (p x p).
`U`	A matrix. Each column corresponds to a p-dimensional PC.
`feasibilityConstraintType`	An integer. Type of feasibility constraints to be enforced. 0: orthogonality constraints; 1: uncorrelatedness constraints.

Details

In the zero-correlation case the pairwise terms are normalized by the total variance $\mathrm{tr}(C)$. Because the PCs are unit-norm, $|u_t^\top C u_s|$ is homogeneous of degree one in `C`, so the unnormalized quantity depends on the units of the data; dividing by $\mathrm{tr}(C)$ makes it invariant to a rescaling of `C`. This matches the definition used internally by `mspca()` against `feasibilityTolerance`.

Value

A float.

Examples

```
TestMat <- cor(mtcars)
mspcares <- mspca(TestMat, r = 2, ks = c(4, 4), verbose = FALSE)
feasibility_violation_off(TestMat, mspcares$x_best, 0)
```

`fraction_variance_explained`

Fraction of Variance Explained

Description

Computes the sum of marginal component variances divided by $tr(\Sigma)$. This equals the variance explained by projection onto the span of U when the columns of U are orthonormal; otherwise it is the normalized sparse-PCA objective.

Usage

```
fraction_variance_explained(C, U)
```

Arguments

`C`	A matrix. The correlation or covariance matrix (p x p).
`U`	A matrix. The matrix containing the r PCs (p x r).

Value

A float.

Examples

```
TestMat <- cor(mtcars)
mspcares <- mspca(TestMat, r = 2, ks = c(4, 4), verbose = FALSE)
fraction_variance_explained(TestMat, mspcares$x_best)
```

fraction_variance_explained_perPC *Fraction of Variance Explained Per PC*

Description

Computes the fraction of variance explained (variance explained normalized by the trace of the covariance/correlation matrix) by each PC.

Usage

```
fraction_variance_explained_perPC(C, U)
```

Arguments

C	A matrix. The correlation or covariance matrix (p x p).
U	A matrix. The matrix containing the r PCs (p x r).

Value

A numeric vector of length `ncol(U)`.

mspca *Multiple Sparse PCA*

Description

Returns multiple sparse principal components of a dataset using an iterative deflation heuristic. As in the `elasticnet` package, the data is passed as a single argument `M` whose interpretation is set by `type`: `"Sigma"` (the default) treats `M` as a covariance/correlation matrix (p x p) and `"X"` treats `M` as a raw data matrix (n observations x p variables). With `type = "X"` the algorithm operates on the data directly via the products $X^\top(X\beta)$ and never forms the p x p matrix, which is substantially more scalable when $n \ll p$.

Usage

```
mspca(
  M,
  r,
  ks,
  type = c("Sigma", "X"),
  feasibilityConstraintType = 0,
  verbose = TRUE,
  maxIter = 200,
  feasibilityTolerance = 1e-04,
```

```
  stallingTolerance = 1e-08,
  timeLimitTPM = 20,
  maxRestartTPM = 30,
  minRestartTPM = 20,
  center = TRUE,
  scale = TRUE,
  divisor = c("n-1", "n"),
  checkPSD = TRUE,
  symTolerance = 1e-08,
  psdTolerance = 1e-08
)
```

Arguments

`M`	A matrix. The data, interpreted according to `type`: a covariance/ correlation matrix (p x p) when `type = "Sigma"`, or a raw data matrix (n x p) when `type = "X"`.
`r`	An integer. Number of principal components (PCs) to be computed. Must be a single whole number greater than or equal to 1.
`ks`	An integer vector. Target sparsity of each PC. Every element must be a whole number greater than or equal to 1 (a sparsity level of zero or less does not define a component and is rejected). `length(ks)` must not exceed `r`; if it is shorter than `r`, a warning is issued and the algorithm is run for `length(ks)` PCs.
`type`	(optional) Either "Sigma" (default; `M` is a covariance/correlation matrix) or "X" (`M` is a raw data matrix).
`feasibilityConstraintType`	(optional) An integer. Type of feasibility constraints to be enforced. 0: orthogonality constraints; 1: uncorrelatedness constraints. Must be exactly 0 or 1; fractional values are rejected rather than rounded. Default 0.
`verbose`	(optional) A Boolean. Controls console output. `TRUE`/`FALSE` or the numbers 0/1; any other value is an error. Default TRUE.
`maxIter`	(optional) An integer. Maximum number of iterations of the algorithm. Must be a whole number greater than or equal to 1. Default 200.
`feasibilityTolerance`	(optional) A float. Tolerance for constraint violation (orthogonality/uncorrelatedness, according to `feasibilityConstraintType`). Under uncorrelatedness the violation is normalized by the total variance `tr(Sigma)`. Must be non-negative; `Inf` is allowed and accepts any solution as feasible. Default 1e-4.
`stallingTolerance`	(optional) A float. Controls the objective improvement below which the algorithm is considered to have stalled. Must be non-negative; `Inf` is allowed. Default 1e-8.
`timeLimitTPM`	(optional) An integer. Maximum time in seconds for the truncated power method (inner iteration). Must be a finite, non-negative whole number; `Inf` is not accepted, so use a large finite value to effectively disable the limit. Default 20.

`maxRestartTPM` (optional) An integer. Number of random restarts of the truncated power method (inner iteration) for the first outer iteration. Must be a whole number greater than or equal to 0; zero means no random restarts. Default 30.

`minRestartTPM` (optional) An integer. Number of random restarts of the truncated power method (inner iteration) for outer iterations >= 2. Must be a whole number greater than or equal to 0. Default 20.

`center` (optional, type = "X") A Boolean. Center the columns of `M` before computing the covariance. `TRUE/FALSE` or the numbers 0/1. Default TRUE.

`scale` (optional, type = "X") A Boolean. Scale the columns of `M` to unit variance, i.e. operate on the correlation matrix. `TRUE/FALSE` or the numbers 0/1. Default TRUE.

`divisor` (optional, type = "X") Either "n-1" (default, sample covariance, matches `cov/cor`) or "n" (population covariance). Default "n-1".

`checkPSD` (optional, type = "Sigma") A Boolean. Verify that `M` is positive semidefinite. `TRUE/FALSE` or the numbers 0/1; any other value is an error. Default TRUE.

`symTolerance` (optional, type = "Sigma") A float. Tolerance for the symmetry check on `M`. Must be non-negative. Default 1e-8.

`psdTolerance` (optional, type = "Sigma") A float. Tolerance (on the smallest eigenvalue) for the PSD check on `M`. Must be non-negative. Default 1e-8.

Value

An object of class `"mspca"` (a list) with fields: `x_best` (p x r matrix of sparse PC loadings), `objective_value`, `feasibility_violation`, `runtime`, `variance_explained` (per-PC explained variance), `total_variance` (trace of the covariance matrix), `feasibilityConstraintType` (the value used to fit, reused as the default for all diagnostics), and `nonredundancy`. With `type = "X"` it additionally records `inputType`, `center`, `scale`, `divisor`, `nObs`, and `p`. Use `print()` to display the sparse loadings and `summary()` for a full per-PC breakdown.

`nonredundancy` is a list of two r x r matrices, `orthogonality` ($|u_t^\top u_s|$) and `uncorrelatedness` ($|u_t^\top \Sigma u_s| / \mathrm{tr}(\Sigma)$), computed once at fit time from the sorted loadings. Only the strict upper triangle is populated; the diagonal and lower triangle are `NA`. Both are stored regardless of which constraint was enforced, so a summary under either definition is available without the covariance matrix and without refitting.

The uncorrelatedness terms are normalized by the total variance $\mathrm{tr}(\Sigma)$. The loadings being unit-norm, $|u_t^\top \Sigma u_s|$ scales linearly with Σ, so normalization makes the uncorrelatedness terms scale invariant.

Note that `feasibility_violation` (returned by the solver) is the quantity compared against `feasibilityTolerance` during the fit.

Examples

```
# From a covariance/correlation matrix (the default type):
TestMat <- cor(mtcars)
res <- mspca(TestMat, r = 2, ks = c(4, 4), verbose = FALSE)
print(res)
summary(res)
```

```
# Equivalent call from the raw data matrix:
res_X <- mspca(as.matrix(mtcars), r = 2, ks = c(4, 4), type = "X", verbose = FALSE)
print(res_X)
```

`print.mspca` *Print an mspca Object*

Description

S3 print method for objects of class `"mspca"` returned by `mspca()`. Displays the sparse loading matrix (restricted to the union of non-zero rows) together with the percentage of variance explained and the number of non-zero loadings per component.

Usage

```
## S3 method for class 'mspca'
print(x, C = NULL, digits = NULL, ...)
```

Arguments

`x`	An object of class `"mspca"`, as returned by `mspca()`.
`C`	(optional) A numeric matrix (p x p). The covariance or correlation matrix used when fitting. Not required: `mspca()` stores the per-PC variance figures and the variable names on the fitted object, for both input types. Supplying `C` recomputes the variance figures from it and takes the row labels from its dimnames.
`digits`	An integer or `NULL`. Number of significant digits for display. When `NULL` (the default), `getOption("digits")` is used, so the output respects `options(digits = ...)`.
`...`	Further arguments required by the `print()` generic; not used by this method.

Details

When the model was fit from a covariance/correlation matrix (`type = "Sigma"`), pass that matrix as `C` so that per-PC variance figures can be computed; when it was fit from a raw data matrix (`type = "X"`), `C` may be omitted because the figures are stored inside the object.

Value

Invisibly returns `x`.

Examples

```
TestMat <- cor(mtcars)
res <- mspca(TestMat, r = 2, ks = c(4, 4), verbose = FALSE)
print(res)
```

snp500 *Market-deflated correlation matrix of S&P 500 daily returns*

Description

The empirical correlation matrix of daily log-returns for 423 S&P 500 constituents over January 2010 to December 2019 (2,515 trading days), after removing the leading ("market") factor.

Usage

```
snp500
```

Format

A symmetric numeric matrix with 423 rows and 423 columns. Row and column names are ticker symbols, in the same order.

Details

S&P 500 returns are dominated by a market factor that loads positively on virtually every stock and accounts for a disproportionate share of total variance. To expose cross-sectional structure – sector and style effects – the leading eigenvector v_1 of the empirical correlation matrix Σ is removed by projection,

$$P^\top \Sigma P, \qquad P = I - v_1 v_1^\top,$$

and it is this deflated matrix that is stored in `snp500`. The result is symmetric positive semidefinite with rank $p - 1 = 422$; the zero eigenvalue is the deflated market direction.

Only the deflated correlation matrix is distributed, not the underlying returns. It is the only object the S&P 500 case study needs, and at 423 x 423 it is small enough to ship.

Source

Derived from the "S&P500 daily update dataset", released under CC0 1.0 (public domain dedication): https://www.kaggle.com/datasets/yash16jr/s-and-p500-daily-update-dataset. The derivation is scripted in `data-raw/snp500.R`. Note that the upstream dataset is updated daily; this matrix is the archival record for the fixed window above.

See Also

`vignette("case-study-snp500", package = "msPCA")` for the analysis built on this matrix.

Examples

```
data(snp500)
dim(snp500)
round(snp500[1:4, 1:4], 3)

# Rank deficient by exactly one: the market factor has been removed.
```

```
ev <- eigen(snp500, symmetric = TRUE, only.values = TRUE)$values
sum(ev > 1e-8)
```

---

summary.mspca *Summarize an mspca Object*

Description

S3 summary method for objects of class "mspca" returned by mspca(). Returns (and prints) a per-PC summary table (number of non-zero loadings, variance explained, FVE, and cumulative FVE) together with the pairwise non-redundancy (feasibility) violation matrix and the total solver runtime.

Usage

```
## S3 method for class 'mspca'
summary(object, C = NULL, feasibilityConstraintType = NULL, digits = NULL, ...)
```

Arguments

object	An object of class "mspca", as returned by mspca().
C	(optional) A numeric matrix (p x p). The covariance or correlation matrix used when fitting. mspca() stores every figure this method reports, so C is ignored for objects that carry those stored diagnostics; it is needed only to summarize an object that does not. It will be removed in a future release.
feasibilityConstraintType	(optional) An integer or NULL. Type of constraint used to compute the violations reported in the summary. 0 for orthogonality; 1 for zero pairwise correlation. Must be exactly 0 or 1; fractional values are rejected rather than rounded. When NULL (the default) the type stored in object at fit time is used.
digits	An integer or NULL. Number of significant digits for display. When NULL (the default), getOption("digits") is used.
...	Further arguments required by the summary() generic; not used by this method.

Details

The violations are reported under the constraint type that was actually enforced when the object was fitted, which mspca() records in object$feasibilityConstraintType. The printed output always names the definition in use. Passing feasibilityConstraintType explicitly overrides this in order to inspect the solution under the other definition; a warning is emitted when the requested type differs from the fitted one, since the resulting figure does not describe a constraint the solver enforced.

Value

Invisibly returns a list of class "summary.mspca" with fields:

table Data frame with columns PC, nonzero, variance, fve, cumulative_fve, and max_violation (the largest violation of that PC against any other PC).

feasibility_mat r x r matrix of pairwise violations ($|u_i^\top u_j|$ or $|u_i^\top \Sigma u_j| / \mathrm{tr}(\Sigma)$). Diagonal and lower triangle are NA.

feasibility Scalar total violation (sum of the upper triangle of feasibility_mat). Strictly off-diagonal, and therefore not comparable with object$feasibility_violation; see mspca().

feasibility_perPC Named vector of per-PC maximum violations.

feasibilityConstraintType The constraint type the reported violations refer to.

fittedConstraintType The constraint type enforced at fit time, or NA for objects that do not record it.

runtime Solver runtime in seconds (if stored in the object).

r Number of sparse PCs.

inputType "Sigma" or "X".

See Also

mspca() for the stored nonredundancy matrices.

Examples

```
TestMat <- cor(mtcars)
res <- mspca(TestMat, r = 2, ks = c(4, 4), verbose = FALSE)
summary(res)

# Fitting under uncorrelatedness: the summary follows the fit automatically.
res_u <- mspca(TestMat, r = 2, ks = c(4, 4),
               feasibilityConstraintType = 1, verbose = FALSE)
summary(res_u)

# The other set of scores is stored too, and needs no refit:
res_u$nonredundancy$orthogonality
```

---

tpm *Truncated Power Method*

Description

Returns the leading sparse principal component of a dataset using the truncated power method. As in mspca(), the data is passed as a single argument M whose interpretation is set by type: "Sigma" (default) for a covariance/correlation matrix (p x p) or "X" for a raw data matrix (n x p). See mspca() for the raw-data preprocessing controls.

Usage

```
tpm(
  M,
  k,
  type = c("Sigma", "X"),
  maxIter = 200,
  verbose = TRUE,
  timeLimit = 10,
  center = TRUE,
  scale = TRUE,
  divisor = c("n-1", "n"),
  checkPSD = TRUE,
  symTolerance = 1e-08,
  psdTolerance = 1e-08
)
```

Arguments

M	A matrix. The data, interpreted according to type: a covariance/ correlation matrix (p x p) when type = "Sigma", or a raw data matrix (n x p) when type = "X".
k	An integer. Target sparsity of the PC. Must be a single whole number greater than or equal to 1.
type	(optional) Either "Sigma" (default; M is a covariance/correlation matrix) or "X" (M is a raw data matrix).
maxIter	(optional) An integer. Maximum number of iterations of the algorithm. Must be a whole number greater than or equal to 1. Default 200.
verbose	(optional) A Boolean. Controls console output. TRUE/FALSE or the numbers 0/1; any other value is an error. Default TRUE.
timeLimit	(optional) An integer. Maximum time in seconds. Must be a finite, non-negative whole number; Inf is not accepted. Default 10.
center	(optional, type = "X") A Boolean. Center the columns of M. TRUE/FALSE or the numbers 0/1. Default TRUE.
scale	(optional, type = "X") A Boolean. Scale the columns of M to unit variance. TRUE/FALSE or the numbers 0/1. Default TRUE.
divisor	(optional, type = "X") Either "n-1" (default) or "n".
checkPSD	(optional, type = "Sigma") A Boolean. Verify M is PSD. TRUE/FALSE or the numbers 0/1; any other value is an error. Default TRUE.
symTolerance	(optional, type = "Sigma") A float. Symmetry-check tolerance. Must be non-negative. Default 1e-8.
psdTolerance	(optional, type = "Sigma") A float. PSD-check tolerance. Must be non-negative. Default 1e-8.

Value

An object of class `"tpm"` (a list) with fields: `x_best` (p x 1 matrix containing the sparse PC loading), `objective_value`, and `runtime`. With `type = "X"` it additionally records `inputType`, `center`, `scale`, `divisor`, `nObs`, and `p`.

References

Yuan, X. T., & Zhang, T. (2013). Truncated power method for sparse eigenvalue problems. *The Journal of Machine Learning Research*, **14**(1), 899–925.

Examples

```
TestMat <- cor(mtcars)
tpm(TestMat, 4)
```

---

variance_explained_perPC

Variance Explained Per PC

Description

Computes the variance explained by each PC.

Usage

```
variance_explained_perPC(C, U)
```

Arguments

`C`	A matrix. The correlation or covariance matrix (p x p).
`U`	A matrix. The matrix containing the r PCs (p x r).

Value

A numeric vector of length `ncol(U)`.

Index