# Package 'bdsvd'

March 26, 2026

**Type** Package

**Title** Block Structure Detection Using Singular Vectors

**Version** 1.2.1

**Maintainer** Jan O. Bauer <j.bauer@vu.nl>

**Description** Provides methods to perform block diagonal covariance matrix detection using singular vectors ('BD-SVD'), which can be extended to inherently sparse principal component analysis ('IS-PCA'). The methods are described in Bauer (2025) <doi:10.1080/10618600.2024.2422985> and Bauer (2026) <doi:10.48550/arXiv.2510.03729>.

**License** GPL (>= 2)

**Imports** irlba, matrixStats, Rcpp, stats

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Suggests** cvCovEst, glasso, mvtnorm, dslabs, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**LinkingTo** Rcpp, RcppArmadillo

**NeedsCompilation** yes

**Author** Jan O. Bauer [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7123-4507>),
Ron Holzapfel [aut]

**Repository** CRAN

**Date/Publication** 2026-03-26 13:30:08 UTC

## Contents

---

bdsvd *Block Detection Using Singular Vectors (BD-SVD).*

---

## Description

Performs BD-SVD iteratively to reveal the block structure. Splits the data matrix into one (i.e., no split) or two submatrices, depending on the structure of the first sparse loading $v$ (which is a sparse approximation of the first right singular vector, i.e., a vector with many zero values) that mirrors the shape of the covariance matrix. This procedure is continued iteratively until the block diagonal structure has been revealed.

The data matrix ordered according to this revealed block diagonal structure can be obtained by `bdsvd.structure`.

## Usage

```
bdsvd(
  X,
  dof.lim,
  anp = "2",
  standardize = TRUE,
  max.iter,
  scores = FALSE,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| `X` | Data matrix of dimension $nxp$ with possibly $p >> n$. |
| `dof.lim` | Interval limits for the number of non-zero components in the sparse loading (degrees of freedom). If $S$ denotes the support of $v$, then the cardinality of the support, $\|S\|$, corresponds to the degrees of freedom. Default is `dof.lim <- c(0, p-1)` which is highly recommended to check for all levels of sparsity. |
| `anp` | Which regularization function should be used for the HBIC.<br>• `"1"`: implements $a_{np} = 1$ which corresponds to the BIC.<br>• `"2"`: implements $a_{np} = 1/2log(np)$ which corresponds to the regularization used by Bauer (2025).<br>• `"3"`: implements $a_{np} = log(log(np))$.<br>• `"4"`: implements $a_{np} = log(log(p))$ which corresponds to the regularization used by Wang et al. (2009) and Wang et al. (2013). |
| `standardize` | Standardize the data to have unit variance. Default is `TRUE`. |

| | |
|---|---|
| max.iter | How many iterations should be performed for computing the sparse loading. Default is 200. |
| scores | Compute scores? |
| verbose | Print out progress as iterations are performed. Default is TRUE. |

## Details

The sparse loadings are computed using the method proposed by Shen & Huang (2008). The corresponding implementation is written in Rcpp/RcppArmadillo for computational efficiency and is based on the R implementation by Baglama, Reichel, and Lewis in ssvd **irlba**. However, the implementation has been adapted to better align with the scope of the **bdsvd** package.

## Value

A list containing the feature names of the submatrices of X. The length of the list equals the number of submatrices.

## References

*Bauer, J.O. (2025). High-dimensional block diagonal covariance structure detection using singular vectors, J. Comput. Graph. Stat., 34(3), 1005–1016*

*Wang, H., B. Li, and C. Leng (2009). Shrinkage tuning parameter selection with a diverging number of parameters, J. R. Stat. Soc. B 71 (3), 671–683.*

*Wang, L., Y. Kim, and R. Li (2013). Calibrating nonconvex penalized regression in ultra-high dimension, Ann. Stat. 41 (5), 2505–2536.*

## See Also

bdsvd.structure, bdsvd.ht, single.bdsvd

## Examples

```
#Replicate the simulation study (c) from Bauer (2025).

## Not run:
p <- 500 #Number of variables
n <- 500 #Number of observations
b <- 10  #Number of blocks
design <- "c" #Simulation design "a", "b", "c", or "d".

#Simulate data matrix X
set.seed(1)
Sigma <- bdsvd.cov.sim(p = p, b = b, design = design)
X <- mvtnorm::rmvnorm(n, mean = rep(0, p), sigma = Sigma)
colnames(X) <- seq_len(p)

bdsvd(X, standardize = FALSE, anp = "4")

## End(Not run)
```

## bdsvd.cov.sim *Covariance Matrix Simulation for BD-SVD*

### Description

This function generates covariance matrices based on the simulation studies described in Bauer (2025).

### Usage

```
bdsvd.cov.sim(p, b, design)
```

### Arguments

| | |
|---|---|
| p | Number of variables. |
| b | Number of blocks. Only required for simulation design "c" and "d". |
| design | Simulation design "a", "b", "c", or "d". |

### Value

A covariance matrix according to the chosen simulation design.

### References

*Bauer, J.O. (2025). High-dimensional block diagonal covariance structure detection using singular vectors, J. Comput. Graph. Stat., 34(3), 1005–1016*

### Examples

```
#The covariance matrix for simulation design (a) is given by
Sigma <- bdsvd.cov.sim(p = 500, b = 500, design = "a")
```

## bdsvd.ht *Hyperparameter Tuning for BD-SVD*

### Description

Finds the number of non-zero elements of the sparse loading according to the high-dimensional Bayesian information criterion (HBIC).

### Usage

```
bdsvd.ht(X, dof.lim, standardize = TRUE, anp = "2", max.iter)
```

## Arguments

| | |
|---|---|
| `X` | Data matrix of dimension $nxp$ with possibly $p >> n$. |
| `dof.lim` | Interval limits for the number of non-zero components in the sparse loading (degrees of freedom). If $S$ denotes the support of $v$, then the cardinality of the support, $\|S\|$, corresponds to the degrees of freedom. Default is `dof.lim <- c(0, p-1)` which is highly recommended to check for all levels of sparsity. |
| `standardize` | Standardize the data to have unit variance. Default is `TRUE`. |
| `anp` | Which regularization function should be used for the HBIC.<br>• `"1"`: implements $a_{np} = 1$ which corresponds to the BIC.<br>• `"2"`: implements $a_{np} = 1/2log(np)$ which corresponds to the regularization used by Bauer (2025).<br>• `"3"`: implements $a_{np} = log(log(np))$.<br>• `"4"`: implements $a_{np} = log(log(p))$ which corresponds to the regularization used by Wang et al. (2009) and Wang et al. (2013). |
| `max.iter` | How many iterations should be performed for computing the sparse loading. Default is `200`. |

## Details

The sparse loadings are computed using the method proposed by Shen & Huang (2008). The corresponding implementation is written in `Rcpp`/`RcppArmadillo` for computational efficiency and is based on the `R` implementation by Baglama, Reichel, and Lewis in `ssvd` **irlba**. However, the implementation has been adapted to better align with the scope of the **bdsvd** package. The computation of the HBIC is outlined in Bauer (2025).

## Value

| | |
|---|---|
| `dof` | The optimal number of nonzero components (degrees of freedom) according to the HBIC. |
| `BIC` | The HBIC for the different numbers of nonzero components. |

## References

*Bauer, J.O. (2025). High-dimensional block diagonal covariance structure detection using singular vectors, J. Comput. Graph. Stat., 34(3), 1005–1016*

*Shen, H. and Huang, J.Z. (2008). Sparse principal component analysis via regularized low rank matrix approximation, J. Multivar. Anal. 99, 1015–1034.*

*Wang, H., B. Li, and C. Leng (2009). Shrinkage tuning parameter selection with a diverging number of parameters, J. R. Stat. Soc. B 71 (3), 671–683.*

*Wang, L., Y. Kim, and R. Li (2013). Calibrating nonconvex penalized regression in ultra-high dimension, Ann. Stat. 41 (5), 2505–2536.*

## See Also

`bdsvd`, `single.bdsvd`

## Examples

```
#Replicate the illustrative example from Bauer (2025).


p <- 300 #Number of variables. In Bauer (2025), p = 3000
n <- 500 #Number of observations
b <- 3   #Number of blocks
design <- "c"

#Simulate data matrix X
set.seed(1)
Sigma <- bdsvd.cov.sim(p = p, b = b, design = design)
X <- mvtnorm::rmvnorm(n, mean = rep(0, p), sigma = Sigma)
colnames(X) <- seq_len(p)

ht <- bdsvd.ht(X)
plot(0:(p-1), ht$BIC[,1], xlab = "|S|", ylab = "HBIC", main = "", type = "l")
single.bdsvd(X, dof = ht$dof, standardize = FALSE)
```

---

# bdsvd.structure *Data Matrix Structure According to the Detected Block Structure.*

---

## Description

Either sorts the data matrix $X$ according to the detected block structure $X_1, ..., X_b$, ordered by the number of variables that the blocks contain. Or returns the detected submatrices each individually in a list object.

## Usage

```
bdsvd.structure(X, block.structure, output = "matrix", block.order)
```

## Arguments

| | |
|---|---|
| `X` | Data matrix of dimension $nxp$ with possibly $p >> n$. |
| `block.structure` | Output of `bdsvd()` or `single.bdsvd()` which identified the block structure. |
| `output` | Should the output be the data matrix ordered according to the blocks (`"matrix"`), or a list containing the submatrices (`"submatrices"`). Default is `"matrix"`. |
| `block.order` | A vector that contains the order of the blocks detected by `bdsvd()` or `single.bdsvd()`. The vector must contain the index of each blocks exactly once. Default is `1:b` where `b` is the total number of blocks. |

## Value

Either the data matrix `X` with columns sorted according to the detected blocks, or a list containing the detected submatrices.

## References

*Bauer, J.O. (2025). High-dimensional block diagonal covariance structure detection using singular vectors, J. Comput. Graph. Stat., 34(3), 1005–1016*

## See Also

bdsvd, single.bdsvd

## Examples

```
#Toying with the illustrative example from Bauer (2025).


p <- 150 #Number of variables. In Bauer (2025), p = 3000.
n <- 500 #Number of observations
b <- 3   #Number of blocks
design <- "c"

#Simulate data matrix X
set.seed(1)
Sigma <- bdsvd.cov.sim(p = p, b = b, design = design)
X <- mvtnorm::rmvnorm(n, mean = rep(0, p), sigma = Sigma)
colnames(X) <- seq_len(p)

#Compute iterative BD-SVD
bdsvd.obj <- bdsvd(X, standardize = FALSE)

#Obtain the data matrix X, sorted by the detected blocks
colnames(bdsvd.structure(X, bdsvd.obj, output = "matrix") )
colnames(bdsvd.structure(X, bdsvd.obj, output = "matrix", block.order = c(2,1,3)) )

#Obtain the detected submatrices X_1, X_2, and X_3
colnames(bdsvd.structure(X, bdsvd.obj, output = "submatrices")[[1]] )
colnames(bdsvd.structure(X, bdsvd.obj, output = "submatrices")[[2]] )
colnames(bdsvd.structure(X, bdsvd.obj, output = "submatrices")[[3]] )
```

---

# cdm.pca *High Dimensional Principal Component Analysis*

---

## Description

Performs a principal component analysis on the given data matrix using the methods of Yata and Aoshima (2009, 2010).

## Usage

```
cdm.pca(X, K = 1, method = "CDM", scale = TRUE, orthogonal = FALSE)
```

## Arguments

| | |
|---|---|
| X | Data matrix of dimension $nxp$ with possibly $p >> n$. |
| K | Number of principal components to be computed. If K is larger than the number of variables $p$ contained in the data matrix, $K = p - 1$ loadings are computed. |
| method | Which method should be used to calculate the eigenvectors (loadings) and eigenvalues. method = "DM" uses the method by Yata and Aoshima (2009) and method = "CDM" uses the method by Yata and Aoshima (2010). |
| scale | Should the variables be scaled to have unit variance before the analysis takes place. Default is TRUE. |
| orthogonal | The estimated eigenvectors (loadings) computed using method = "CDM" (Yata and Aoshima, 2010) are orthogonal in the limit thus only approximately orthogonal in the finite sample case. Should the loadings be orthogonalized. Default is FALSE. |

## Details

This function performs principal component analysis using either the DM approach as described in Yata, K., Aoshima, M. (2009), or the CDM approach (Yata, K., Aoshima, M., 2010) Note that there is also a code implementation of CDM available at 'Aoshima Lab' (https://github.com/Aoshima-Lab/HDLSS-Tools/tree/main/CDM) provided by Makoto Aoshima.

## Value

A list with the following components:

| | |
|---|---|
| v | The first K estimated sparse singular vectors (loadings) if the data matrix X. The eigenvectors are orthogonalized if orthogonal = TRUE. |
| l | The corresponding first estimated eigenvalues of the identified block diagonal covariance matrix. |
| K | The number of sparse singular vectors (loadings) that have been computed. |

## References

*Yata, K., Aoshima, M. (2009). PCA consistency for non-Gaussian data in high dimension, low sample size contex, Commun. Stat. - Theory Methods 38, 2634–2652.*

*Yata, K., Aoshima, M. (2010). Effective PCA for high-dimension, low-sample-size data with singular value decomposition of cross data matrix, J. Multivar. Anal. 101, 2060–2077.*

## Examples

```
#Example: run IS-PCA on a gene expression data set with two tissue types

if (requireNamespace("dslabs", quietly = TRUE)) {
data("tissue_gene_expression", package = "dslabs")

#We only select the two tissue types kidney (6) and liver (7)
Y <- as.numeric(tissue_gene_expression$y)
X <- scale(tissue_gene_expression$x[Y %in% c(6, 7), ], scale = FALSE)
```

```
Y <- Y[Y %in% c(6, 7)]

# Run PCA
pca.obj <- cdm.pca(X, K = 2)
PC <- X %*% pca.obj$v

# Plot the first two principal components
plot(PC, pch = Y-5, xlab = "PC1", ylab = "PC2")
}
```

---

`detect.blocks` *Block Detection*

---

## Description

This function returns the block structure of a matrix.

## Usage

```
detect.blocks(V, threshold = 0)
```

## Arguments

| | |
|---|---|
| `V` | Numeric matrix which either contains the loadings or is a covariance matrix. |
| `threshold` | All absolute values of `V` below the threshold are set to zero. |

## Value

An object of class `blocks` containing the features and columns indices corresponding to each detected block.

## References

*Bauer, J.O. (2025). High-dimensional block diagonal covariance structure detection using singular vectors, J. Comput. Graph. Stat., 34(3), 1005–1016*

## See Also

`bdsvd`, `single.bdsvd`

## Examples

```
#In the first example, we replicate the simulation study for the ad hoc procedure
#Est_0.1 from Bauer (2025). In the second example, we manually compute the first step
#of BD-SVD, which can be done using the bdsvd() and/or single.bdsvd(), for constructed
#sparse loadings

#Example 1: Replicate the simulation study (a) from Bauer (2025) for the ad hoc
```

```
#procedure Est_0.1.

## Not run:
p <- 500 #Number of variables
n <- 125 #Number of observations
b <- 500 #Number of blocks
design <- "a"

#Simulate data matrix X
set.seed(1)
Sigma <- bdsvd.cov.sim(p = p, b = b, design = design)
X <- mvtnorm::rmvnorm(n, mean=rep(0, p), sigma=Sigma)
colnames(X) <- 1:p

#Perform the ad hoc procedure
detect.blocks(cvCovEst::scadEst(dat = X, lambda = 0.2), threshold = 0)

## End(Not run)

#Example 2: Manually compute the first step of BD-SVD
#for some loadings V that mirror the two blocks
#("A", "B") and c("C", "D").

V <- matrix(c(1,0,
              1,0,
              0,1,
              0,1), 4, 2, byrow = TRUE)

rownames(V) <- c("A", "B", "C", "D")
detected.blocks <- detect.blocks(V)

#Variables in block one with corresponding column index:
detected.blocks[[1]]$features
detected.blocks[[1]]$block.columns

#Variables in block two with corresponding column index:
detected.blocks[[2]]$features
detected.blocks[[2]]$block.columns
```

---

`ispca` *Inherently Sparse Principal Component Analysis (IS-PCA).*

---

## Description

Performs IS-PCA

## Usage

```
ispca(
```

```
  X,
  K = 1,
  block.structure,
  anp = "2",
  covariance = TRUE,
  method = "CDM",
  orthogonal = FALSE,
  standardize = FALSE,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| X | Data matrix of dimension $nxp$ with possibly $p >> n$. |
| K | Number of singular vectors (loadings) to be computed for each identified submatrix. This means that $b*K$ components are calculated if $b$ blocks are identified by BD-SVD (see Bauer (2025) and bdsvd for details). If K is larger than the number of variables $p_i$ contained in a submatrix, $K = p_i$ loadings are computed. |
| block.structure | Underlying block structure. This parameter is optional as otherwise the functions runs bdsvd() to identify the underlying block structure. When supplied, it must be a 'bdsvd', 'blocks', or 'ispca' object. E.g., pass the result of bdsvd(), single.bdsvd(), ispca(), or detect.blocks(). An identified block structure by any other method can be supplied using detect.blocks() (see example below). |
| anp | Which regularization function should be used for the HBIC.<br>• "1": implements $a_{np} = 1$ which corresponds to the BIC.<br>• "2": implements $a_{np} = 1/2log(np)$ which corresponds to the regularization used by Bauer (2025).<br>• "3": implements $a_{np} = log(log(np))$.<br>• "4": implements $a_{np} = log(log(p))$ which corresponds to the regularization used by Wang et al. (2009) and Wang et al. (2013). |
| covariance | Perform IS-PCA on the covariance (TRUE) or correlation matrix (FALSE). Default is TRUE. |
| method | Which method should be used to calculate the eigenvectors (loadings) and eigenvalues. method = "DM" uses the method by Yata and Aoshima (2009), method = "CDM" uses the method by Yata and Aoshima (2010), and method = "PCA" uses svd. The methods by Yata and Aoshima (2009, 2010) are consistent in HDLSS settings. Default is method = "CDM". |
| orthogonal | The estimated eigenvectors (loadings) computed using method = "CDM" (Yata and Aoshima, 2010) are orthogonal in the limit thus only approximately orthogonal in the finite sample case. Should the loadings be orthogonalized? Default is FALSE. |
| standardize | Standardize the data for block detection using BD-SVD. Default is TRUE. Note: this does not affect the parameter covariance. PCA can be computed on the covariance matrix regardless. |
| verbose | Print out progress for bdsvd as iterations are performed. Default is TRUE. |

## Details

This function performs inherently sparse principal component analysis (IS-PCA) as introduced in Bauer (2026).

## Value

A list with the following components:

**v** The first estimated eigenvectors of the identified block diagonal covariance matrix (i.e., the loadings) as an object of type `matrix`. The eigenvectors are orthogonalized if `orthogonal = TRUE`.

**l** The corresponding first estimated eigenvalues of the identified block diagonal covariance matrix.

**exp.var** The explained variance of the first estimated eigenvalues `l`.

**X.b** The detected submatrices using `bdsvd` as a `list` object.

**block.structure** Either the block structure detected by `bdsvd()` or the user-supplied `block.structure`, depending on the input.

## References

*Bauer, J.O. (2025). High-dimensional block diagonal covariance structure detection using singular vectors, J. Comput. Graph. Stat., 34(3), 1005–1016*

*Bauer, J.O. (2026). Beyond regularization: inherently sparse principal component analysis. Stat. Comp.*

*Yata, K., Aoshima, M. (2009). PCA consistency for non-Gaussian data in high dimension, low sample size contex, Commun. Stat. - Theory Methods 38, 2634–2652.*

*Yata, K., Aoshima, M. (2010). Effective PCA for high-dimension, low-sample-size data with singular value decomposition of cross data matrix, J. Multivar. Anal. 101, 2060–2077.*

## See Also

`bdsvd`, `bdsvd.structure`

## Examples

```
#Example 1: run IS-PCA on a gene expression data set with two tissue types

if (requireNamespace("dslabs", quietly = TRUE)) {
data("tissue_gene_expression", package = "dslabs")

#We only select the two tissue types kidney (6) and liver (7)
Y <- as.numeric(tissue_gene_expression$y)
X <- scale(tissue_gene_expression$x[Y %in% c(6, 7), ], scale = FALSE)
Y <- Y[Y %in% c(6, 7)]

# Run IS-PCA
ispca.obj <- ispca(X = X, anp = "1")
vhat <- ispca.obj$v[, 1:2]
ispc <- X %*% vhat
```

```
# Percentage of non-zero components in the first two loadings
round(colSums(vhat != 0)/ncol(X), 2)

# Plot the first two principal components
plot(ispc, pch = Y-5, xlab = "PC1", ylab = "PC2", main = "IS-PCA")

# Compare to CDM-PCA (see cdm.pca(...))
pca.obj <- cdm.pca(X = X, K = 2)
pc <- X %*% pca.obj$v

par(mfrow = c(1, 2))
plot(ispc, pch = Y-5, xlab = "PC1", ylab = "PC2", main = "IS-PCA")
plot(pc, pch = Y-5, xlab = "PC1", ylab = "PC2", main = "PCA")
par(mfrow = c(1, 1))
}


#Example 2: submit a block structure which was identified by any other approach. This can be
#done by transforming the block structure to an object of type 'blocks' using detect.blocks():

if (requireNamespace("glasso", quietly = TRUE)) {
#Simulate a data matrix X with a block diagonal population covariance matrix.
set.seed(1)
n <- 100
p <- 4
Sigma <- bdsvd.cov.sim(p = p, b = 2, design = "c")

X <-  matrix(rnorm(n * p), n, p) %*% chol(Sigma)
S <- cov(X)

#Identify the block structure using glasso()
S.block <- glasso::glasso(S, 0.2)$w

#S.blocks is a block diagonal matrix:
print(S.block != 0)
#We know extract the block information to an object of class 'blocks' using detect.blocks()
block.structure <- detect.blocks(S.block)
class(block.structure)

#The block.structure of class 'blocks' can now be supplied to ispca()
ispca(X, block.structure = block.structure, verbose = FALSE)
}
```

---

prmats *Principal (Sub)Matrices*

---

## Description

Identifies the principal (sub)matrices

## Usage

```
prmats(X, block.structure, rule = "cumvar", value)
```

## Arguments

| | |
|---|---|
| `X` | Data matrix of dimension $nxp$ with possibly $p >> n$. |
| `block.structure` | Underlying block structure. Must be a 'bdsvd', 'blocks', or 'ispca' object. E.g., pass the result of `bdsvd()`, `single.bdsvd()`, `ispca()`, or `detect.blocks()`. An identified block structure by any other method can be supplied using `detect.blocks()` (see example below). |
| `rule` | Which rule should be used to choose principal submatrices. `rule = "cumvar"` selects the smallest number of principal submatrices (ordered by explained variance) whose cumulative share is at least `value` (a proportion in (0, 1]). `rule = "enrich"` selects all principal submatrices that explain $\times$`value` more than they should on average (see `value`). |
| `value` | Numeric parameter used by `rule`. If `rule = "cumvar"`, `value` is the target cumulative proportion of explained variance (must be in `(0, 1]`). Default is `0.8`. If `rule = "enrich"`, `value` is the factor necessary to be selected compared to the equal-share baseline. E.g., if a submatrix should on average explain 10% of the total explained variance and if `value = 2`, this submatrix is only selected if it explains at least $2x10\%$ = 20% of the total explained variance. Default is `2`. |

## Details

This function selects the principal (sub)matrices as described in Bauer (2026).

## Value

A named list with the following components:

**prmats** List of submatrices ordered by explained variance (rule = 'cumvar') or by factor (rule = 'enrich'). Each element `prmats[[b]]` is a named list with:

- **expl.var** Proportion of total variance explained by block $b$.
- **avg.var** Average variance of the variables in block $b$.
- **factor** Enrichment factor `expl.var / avg.var` (see `value` argument of the function).
- **feature.names** Column names (variables) that belong to block $b$.
- **p.b** Number of variables in block $b$.

**X.pr** The data matrix of the kept submatrices/variables.

## Access

Submatrices can be accessed with list indexing, e.g., `res$prmats[[1]]$feature.names` gives the variable names of the first submatrix.

## References

*Bauer, J.O. (2025). High-dimensional block diagonal covariance structure detection using singular vectors, J. Comput. Graph. Stat., 34(3), 1005–1016*

*Bauer, J.O. (2026). Beyond regularization: inherently sparse principal component analysis. Stat. Comp.*

## See Also

bdsvd, ispca

## Examples

```
#Example: principal submatrices of a gene expression data set with two tissue types

if (requireNamespace("dslabs", quietly = TRUE)) {
data("tissue_gene_expression", package = "dslabs")

#We only select the two tissue types kidney (6) and liver (7)
Y <- as.numeric(tissue_gene_expression$y)
X <- scale(tissue_gene_expression$x[Y %in% c(6, 7), ], scale = FALSE)
Y <- Y[Y %in% c(6, 7)]


#First: run IS-PCA (or submit a identified block structure using bdsvd(...) or detect.blocks(...))

ispca.obj <- ispca(X = X, anp = "1")


#Second: extract the submatrices that explain at least 80% (default value) of the total variance

res <- prmats(X, block.structure = ispca.obj)
res

#One submatix is selected which contains 236 variables (out of 500) and explains
#81.67% of the total variance
length(res$prmats)
res$prmats[[1]]$p.b
round(res$prmats[[1]]$expl.var * 100, 2)


#Alternatively: extract the submatrices that explain five times more of the total variance
#than they should on average ('factor')

res <- prmats(X, block.structure = ispca.obj, rule = "enrich", value = 1.5)
res

#The highest 'factor' is 1.73
res <- prmats(X, block.structure = ispca.obj, rule = "enrich", value = 2)


}
```

`single.bdsvd` *Single Iteration of Block Detection Using Singular Vectors (BD-SVD).*

## Description

Performs a single iteration of BD-SVD: splits the data matrix into one (i.e., no split) or two submatrices, depending on the structure of the first sparse loading $v$ (which is a sparse approximation of the first right singular vector, i.e., a vector with many zero values) that mirrors the shape of the covariance matrix.

## Usage

```
single.bdsvd(X, dof, standardize = TRUE, max.iter)
```

## Arguments

| | |
|---|---|
| `X` | Data matrix of dimension $nxp$ with possibly $p >> n$. |
| `dof` | Number of non-zero components in the sparse loading (degrees of freedom). If $S$ denotes the support of $v$, then the cardinality of the support, $\|S\|$, corresponds to the degrees of freedom. |
| `standardize` | Standardize the data to have unit variance. Default is `TRUE`. |
| `max.iter` | How many iterations should be performed for computing the sparse loading. Default is `200`. |

## Details

The sparse loadings are computed using the method proposed by Shen & Huang (2008). The corresponding implementation is written in `Rcpp`/`RcppArmadillo` for computational efficiency and is based on the `R` implementation by Baglama, Reichel, and Lewis in `ssvd` **irlba**. However, the implementation has been adapted to better align with the scope of the **bdsvd** package.

## Value

A list containing the feature names of the submatrices of `X`. It is either of length one (no split) or length two (split into two submatrices).

## References

*Bauer, J.O. (2025). High-dimensional block diagonal covariance structure detection using singular vectors, J. Comput. Graph. Stat., 34(3), 1005–1016*

*Shen, H. and Huang, J.Z. (2008). Sparse principal component analysis via regularized low rank matrix approximation, J. Multivar. Anal. 99, 1015–1034.*

## See Also

bdsvd, bdsvd.ht

## Examples

```
#Replicate the illustrative example from Bauer (2025).

## Not run:

p <- 300 #Number of variables. In Bauer (2025), p = 3000.
n <- 500 #Number of observations
b <- 3   #Number of blocks
design <- "c"

#Simulate data matrix X
set.seed(1)
Sigma <- bdsvd.cov.sim(p = p, b = b, design = design)
X <- mvtnorm::rmvnorm(n, mean = rep(0, p), sigma = Sigma)
colnames(X) <- 1:p

ht <- bdsvd.ht(X)
plot(0:(p-1), ht$BIC[,1], xlab = "|S|", ylab = "HBIC", main = "", type = "l")
single.bdsvd(X, dof = ht$dof, standardize = FALSE)


## End(Not run)
```

# Index