Package: blox (via r-universe)

October 27, 2026

Type Package

Title Block Diagonal Matrix Approximation

Version 0.0.2

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

Description Finds the best block diagonal matrix approximation of a symmetric matrix. This can be exploited for divisive hierarchical clustering using singular vectors, named HC-SVD. The method is described in Bauer (202X) <doi:10.48550/arXiv.2308.06820>.

License GPL (>= 2)

Imports matrixStats, Rcpp, RcppArmadillo

Encoding UTF-8

RoxygenNote 7.3.2

Suggests bdsvd, corrplot, irlba, knitr, psych, testthat (>= 3.0.0)

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

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

Repository https://cran.r-universe.dev

Date/Publication 2026-09-16 17:08:44 UTC

RemoteUrl https://github.com/cran/blox

RemoteRef HEAD

RemoteSha 35c15a90ecc258abc73bf768fefd45486b4192b8

Contents

bd.approx 2
hc.beta 5
hcsvd 7
is.ultrametric 10

Index **12**

`bd.approx` *Two Block Diagonal Matrix Approximation.*

Description

Finds the best two block diagonal matrix approximation of a similarity matrix according to some distance (linkage) function as described in Bauer (202X). By default, candidate splits are determined by sparse approximations of the first eigenvectors of the similarity matrix. Alternatively, all unique two-way partitions can be evaluated by exhaustive search.

Usage

```
bd.approx(
  S,
  linkage = "average",
  q = 1,
  h.power = 2,
  balance,
  max.iter,
  search = "hcsvd",
  exhaustive.max.p = 20
)
```

Arguments

`S`	A scaled pxp similarity matrix. For example, this may be a correlation matrix.
`linkage`	The linkage function to be used. This should be one of `"average"`, `"ncut"`, `"rcut"`, `"RV"` (for RV-coefficient), or `"single"`.
`q`	Number of sparse eigenvectors to be used whenever the HC-SVD search is applied. This should be either a numeric value between zero and one to indicate percentages, or `"Kaiser"`. For a numerical value between zero and one, the number of sparse eigenvectors is determined as the corresponding share of the total number of eigenvectors, capped at $p - 1$. E.g., `q = 1` uses all $p - 1$ available sparse eigenvectors. The Kaiser rule uses the number of eigenvalues of the Hadamard-transformed similarity matrix that are at least one, plus two additional eigenvectors, capped at $p - 1$.
`h.power`	`h`-th Hadamard power of `S` used whenever sparse-eigenvector candidate generation is applied. This should be a positive integer. Larger powers shrink similarities with magnitude smaller than one towards zero and can increase the contrast between within- and between-block similarities, although their effect depends on the similarity structure and clustering objective.
`balance`	Minimum proportion of the smaller block when splitting into two. Must be a numeric value in $(0, 0.5]$. For example, `balance = 0.5` enforces an exact 50:50 split, while `balance = 0.2` allows splits as unbalanced as 20:80, with more balanced splits such as 30:70 or 40:60 also permitted. If an exact split is not possible (e.g., `balance = 0.5` when $p = 9$), the closest integer partition is used (e.g., 4 and 5 per block).

max.iter	How many iterations should be performed for computing the sparse eigenvectors when the HC-SVD search is used. Default is 500.
search	Search strategy. "hcsvd" uses sparse-eigenvector candidate generation, "exhaustive" evaluates all unique two-way partitions for the selected linkage function only, and "auto" uses exhaustive search when p is at most exhaustive.max.p and HC-SVD otherwise.
exhaustive.max.p	Largest problem size for which exhaustive search is used when search = "auto". Default is 20. The exhaustive C++ implementation is restricted to at most 30 objects.

Details

The sparse eigenvectors are computed using the method of Shen and Huang (2008). The method is implemented by Baglama, Reichel, and Lewis in ssvd (irlba). Here, we use a Rcpp/RcppArmadillo implementation based on ssvd with slight modifications to suit our method and for faster performance.

Value

A list containing:

B	The best two block diagonal matrix approximation.
BD	The best two block diagonal matrix approximation permuted to a block diagonal shape: $BD = PBt(P)$.
P	The permutation matrix P: $BD = PBt(P)$.
clustering	The clustering vector as an integer vector of length p, which gives for each component the number 1 or 2 of the cluster/split to which it belongs.
split	A list containing the two splits.
distance	The approximation error (distance) according to the selected linkage function.
search	The search strategy used for the split.
n.candidates	The number of candidate evaluations performed.

References

Bauer, J.O. (202X). Divisive hierarchical clustering using block diagonal matrix approximations. Working paper.

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

Examples

```
#We give a trivial example for a block diagonal matrix perturbed by
#noise, for adapting clustering objectives of spectral clustering,
#and for balanced clustering.
```

```
### TOY EXAMPLE

A <- matrix(c(2,1,1,3), 2, 2)  # 2x2 block
B <- matrix(c(5,4,4,6), 2, 2)  # 2x2 block

# Create a 5x5 zero matrix and insert blocks at right positions.
M <- matrix(0, 4, 4)
M[1:2, 1:2] <- A
M[3:4, 3:4] <- B

M.tilde <- M + matrix(rnorm(4^2, 0, 0.2), 4, 4)

#Construct a similaritiy matrix with same block structure
S <- cov2cor(t(M.tilde) %*% M.tilde)
bd <- bd.approx(S)

#Block diagonal approximation:
bd$B

#We can also permute the block diagonal shape:
S2 <- S[c(1, 3, 2, 4), c(1, 3, 2, 4)]
bd2 <- bd.approx(S2)

#bd2$B gives us again the block diagonal approximation
bd2$B

#And bd2$BD gives us the block diagonal approximation permuted to
#block diagonal shape
bd2$BD


### ADAPTING CLUSTERING OBJECTIVES

#We will use the USArrests example (see ?hcsvd).
data("USArrests")
USArrests["Maryland", "UrbanPop"] <- 76.6
D <- as.matrix(dist(USArrests))
S <- 1 - D / max(D)

#We compute k = 2 clusters adapting the objective of spectral clustering
#with the ratio cut.
bd.approx(S, linkage = "rcut")


### BALANCED CLUSTERING

#We can also enforce balanced clustering, such as two clusters of equal
#size (50:50). We will do this for the USArrests example from above.
bd.approx(S, linkage = "rcut", balance = 0.5)
```

`hc.beta` *Compute Revelle's Beta for all worst split-halves using HC-SVD.*

Description

Performs HC-SVD to reveal the hierarchical variable structure using average linkage as described in Bauer (202X). For a data matrix comprising p items, this means that $p-1$ splits are identified. The obtained structure aligns with the structure according to the worst split-half reliability and is thus used to compute a hierarchy of all Revelle's beta as described in Bauer (2026).

Usage

```
hc.beta(R, splits = NULL, n.splits = NULL, is.corr = TRUE, verbose = TRUE)
```

Arguments

`R`	A correlation matrix of dimension pxp or a data matrix of dimension nxp can be provided. If a data matrix is supplied, it must be indicated by setting `is.corr = FALSE`, and the correlation matrix will then be calculated as `cor(X)`.
`splits`	An object containing the splits identified by HC-SVD. This can either be the result of `hcsvd` (for all splits) or `bd.approx` (for a single split). If omitted, `hc.beta` will internally call `hcsvd(R)` and compute Revelle's beta for all $p-1$ splits.
`n.splits`	Number of splits for which Revelle's beta is computed. If `splits` is from `hcsvd`, the default is all $p-1$ splits. If `splits` is from `bd.approx`, only a single split is available and `n.splits` is set to 1.
`is.corr`	Is the supplied object a correlation matrix. Default is `TRUE` and this parameter must be set to `FALSE` if a data matrix instead of a correlation matrix is supplied.
`verbose`	Print out progress as $p-1$ iterations for divisive hierarchical clustering are performed. Default is `TRUE`.

Details

Supplementary details are in Bauer (2026).

Value

A list with `n.splits` components. Each split is a list of four components:

`split`	The split number.
`beta`	Revelle's beta for this split.
`A`	One of the two sub-scales that has been split.
`B`	One of the two sub-scales that has been split.
`beta.alpha`	Computes the ratio of Revelle's beta and Cronbach's alpha.

References

Bauer, J.O. (2026). Revelle's beta: The wait is over—computation becomes possible. Psychometrika.

Bauer, J.O. (202X). Divisive hierarchical clustering using block diagonal matrix approximations. Working paper.

See Also

bd.approx hcsvd

Examples

```
#We compute the worst split-half reliabilities on a correlation matrix.


#Load the correlation matrix Bechtoldt from the psych
#package (see ?Bechtoldt for more information).
if (requireNamespace("psych", quietly = TRUE)) {
  data("Bechtoldt", package = "psych")
}
R <- Bechtoldt


### RUN HC-SVD FOR HIERARCHICAL VARIABLE CLUSTERING

#Compute HC-SVD (with average linkage).
hcsvd.obj <- hcsvd(R)

#The object of type hclust with corresponding dendrogram can be obtained
#directly from hcsvd(...):
hc.div <- hcsvd.obj$hclust
plot(hc.div, axes = FALSE, ylab = "", main = "Revelle's Beta Splits")


### COMPUTE REVELLE'S BETA FOR ALL IDENTIFIED SPLITS

#Compute Revelle's beta
betas <- hc.beta(R = R)

#Alternatively, you can submit the object obtained from hcsvd(). Thus,
#the hiearchy needs not to be computed again using hcsvd().
betas <- hc.beta(R = R, splits = hcsvd.obj)

#Visualize the splits, e.g., as
splits <- sapply(betas, `[[`, "split")
beta.values <- sapply(betas, `[[`, "beta")

plot(splits, beta.values,
  type = "b",
  xlab = "Split",
  ylab = "Revelle's Beta",
```

```
  main = "Revelle's Beta Across Splits",
  pch = 19)

#Visualize the ratio of Revelle's beta and Cronbach's alpha
beta.alpha <- sapply(betas, `[[`, "beta.alpha")
plot(splits, beta.values,
  type = "b",
  xlab = "Split",
  ylab = "Beta/Alpha",
  main = "Ratio of Beta and Alpha Across Splits",
  pch = 19)


### COMPUTE REVELLE'S BETA FOR THE FIRST IDENTIFIED SPLIT

#The first split can be identified using bd.approx()
#This is computationally faster, as only the first split
#is identified
hc.beta(R = R, splits = bd.approx(R))
```

---

hcsvd *Hierarchical Clustering Using Singular Vectors (HC-SVD).*

Description

Performs HC-SVD to reveal the hierarchical structure as described in Bauer (202X). This divisive approach iteratively splits each cluster into two subclusters. By default, candidate splits are determined by sparse approximations of the first eigenvectors of the similarity matrix. For small clusters, an exhaustive search over all unique two-way partitions can alternatively be used. The selected split is the one that yields the best block-diagonal approximation of the similarity matrix according to a specified linkage function. The procedure continues until each object is assigned to its own cluster.

Usage

```
hcsvd(
  S,
  linkage = "average",
  q = 1,
  h.power = 2,
  max.iter,
  verbose = TRUE,
  search = "hcsvd",
  exhaustive.max.p = 20
)
```

Arguments

S	A scaled pxp similarity matrix. For example, this may be a correlation matrix.
linkage	The linkage function to be used. This should be one of "average", "single", or "RV" (for RV-coefficient). Note that the RV-coefficient might not yield an ultrametric distance.
q	Number of sparse eigenvectors to be used whenever the HC-SVD search is applied. This should be either a numeric value between zero and one to indicate percentages, or "Kaiser". For a numerical value between zero and one, the number of sparse eigenvectors is determined as the corresponding share of the total number of eigenvectors, capped at $p - 1$. E.g., q = 1 uses all $p - 1$ available sparse eigenvectors. The Kaiser rule uses the number of eigenvalues of the Hadamard-transformed similarity matrix that are at least one, plus two additional eigenvectors, capped at $p - 1$.
h.power	h-th Hadamard power of S used whenever sparse-eigenvector candidate generation is applied. This should be a positive integer. Larger powers shrink similarities with magnitude smaller than one towards zero and can increase the contrast between within- and between-block similarities, although their effect depends on the similarity structure and clustering objective.
max.iter	How many iterations should be performed for computing the sparse eigenvectors when the HC-SVD search is used. Default is 500.
verbose	Print out progress as $p - 1$ iterations for divisive hierarchical clustering are performed. Default is TRUE.
search	Search strategy. "hcsvd" uses sparse-eigenvector candidate generation, "exhaustive" evaluates all unique two-way partitions for the selected linkage function only, and "auto" uses exhaustive search whenever the current cluster contains at most exhaustive.max.p objects and HC-SVD otherwise.
exhaustive.max.p	Largest cluster size for which exhaustive search is used when search = "auto". Default is 20. The exhaustive C++ implementation is restricted to at most 30 objects.

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 which is the base for the **blox** package.

Supplementary details are in hc.beta and in Bauer (2026).

Value

A list containing:

hclust	The clustering structure identified by HC-SVD as an object of type hclust. For linkage = "RV", dendrogram heights are monotonically adjusted if required because the RV-coefficient need not yield an ultrametric distance.

`dist.matrix` The distance matrix of the HC-SVD structure as an object of class `dist`. This matrix need not be ultrametric for `linkage = "RV"`.

`u.sim` The corresponding similarity matrix of S, calculated as `1-dist.matrix`.

`q.p` A vector of length $p - 1$ containing the ratio q_i/p_i of the number of sparse eigenvectors used relative to the total number of eigenvectors for the split of each cluster. Since HC-SVD uses at most $p_i - 1$ sparse eigenvectors, this ratio is always smaller than one. The ratio is set to `NA` when exhaustive search is used or when the cluster contains only two objects, since no sparse eigenvectors are computed in these cases.

`search` A vector indicating whether `"hcsvd"`, `"exhaustive"`, or the unique two-object split was used at each recursive step.

References

Bauer, J.O. (2026). Revelle's beta: The wait is over—computation becomes possible. Psychometrika.

Bauer, J.O. (202X). Divisive hierarchical clustering using block diagonal matrix approximations. Working paper.

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**}

Examples

```
#We give one example for variable clustering directly on a correlation matrix,
#and we replicate the USArrest example in Bauer (202X) for observation clustering.
#More elaborate code alongside a different example for variable clustering can be
#found in the corresponding supplementary material of that manuscripts.


### VARIABLE CLUSTERING

#Load the correlation matrix Bechtoldt from the psych
#package (see ?Bechtoldt for more information).
if (requireNamespace("psych", quietly = TRUE)) {
  data("Bechtoldt", package = "psych")
}

#Compute HC-SVD (with average linkage).
hcsvd.obj <- hcsvd(Bechtoldt)

#The object of type hclust with corresponding dendrogram can be obtained
#directly from hcsvd(...):
hc.div <- hcsvd.obj$hclust
plot(hc.div, ylab = "")
```

```
#The dendrogram can also be obtained from the ultrametric distance matrix:
plot(hclust(hcsvd.obj$dist.matrix), main = "HC-SVD", sub = "", xlab = "")


### OBSERVATION CLUSTERING

#Correct for the known transcription error
data("USArrests")
USArrests["Maryland", "UrbanPop"] <- 76.6

#The distance matrix is scaled (divided by max(D)) to later allow a
#transformation to a matrix S that fulfills the properties of a similarity
#matrix.
D <- as.matrix(dist(USArrests))
D <- D / max(D)
S <- 1 - D

#Compute HC-SVD (with average linkage).
hcsvd.obj <- hcsvd(S)

#The object of type hclust with corresponding dendrogram can be obtained
#directly from hcsvd(...):
hc.div <- hcsvd.obj$hclust
plot(hc.div, ylab = "")

#The dendrogram can also be obtained from the ultrametric distance matrix:
plot(hclust(hcsvd.obj$dist.matrix), main = "HC-SVD", sub = "", xlab = "")
```

`is.ultrametric` *Ultrametric Distance Property*

Description

This function checks the ultrametric property of a distance matrix obtained by HC-SVD.

Usage

```
is.ultrametric(hcsvd.obj)
```

Arguments

`hcsvd.obj`	An object of type hcsvd(...)

Value

Returns TRUE if the ultrametric property is fulfilled. Otherwise FALSE.

References

Bauer, J.O. (202X). Divisive hierarchical clustering using block diagonal matrix approximations. Working paper.

See Also

hcsvd

Examples

```
#Load the correlation matrix Bechtoldt from the psych
#package (see ?Bechtoldt for more information).
if (requireNamespace("psych", quietly = TRUE)) {
  data("Bechtoldt", package = "psych")
}

#Compute HC-SVD (with average linkage).
hcsvd.obj <- hcsvd(Bechtoldt)

#Check the ultrametric property
is.ultrametric(hcsvd.obj)
```

Index

bd.approx, 2, *5, 6*
bdsvd, *9*

hc.beta, 5, *8*
hcsvd, *5, 6*, 7, *11*

is.ultrametric, 10

ssvd, *3*, *8*