Package 'bigmds'

May 7, 2026

Title Multidimensional Scaling for Big Data

Version 3.0.0

Description MDS is a statistic tool for reduction of dimensionality, using as input a distance matrix of dimensions n × n. When n is large, classical algorithms suffer from computational problems and MDS configuration can not be obtained.
With this package, we address these problems by means of six algorithms, being two of them original proposals:
- Landmark MDS proposed by De Silva V. and JB. Tenenbaum (2004).
- Interpolation MDS proposed by Delicado P. and C. Pachón-García (2021) <doi:10.48550/arXiv.2007.11919> (original proposal).
- Reduced MDS proposed by Paradis E (2018).
- Pivot MDS proposed by Brandes U. and C. Pich (2007)
- Divide-and-conquer MDS proposed by Delicado P. and C. Pachón-García (2021) <doi:10.48550/arXiv.2007.11919> (original proposal).
- Fast MDS, proposed by Yang, T., J. Liu, L. McMillan and W. Wang (2006).

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Depends R (>= 3.0.2)

Suggests testthat

Imports pracma, svd, corpcor, parallel, stats

URL https://github.com/pachoning/bigmds

BugReports https://github.com/pachoning/bigmds/issues

NeedsCompilation no

Author Cristian Pachón García [aut, cre] (ORCID: <https://orcid.org/0000-0001-9518-4874>), Pedro Delicado [aut] (ORCID: <https://orcid.org/0000-0003-3933-4852>)

Maintainer Cristian Pachón García <cc.pachon@gmail.com>

Repository CRAN

Date/Publication 2024-01-09 14:30:02 UTC

Contents

`divide_conquer_mds` *Divide-and-conquer MDS*

Description

Roughly speaking, a large data set, `x`, of size n is divided into parts, then classical MDS is performed over every part and, finally, the partial configurations are combined so that all the points lie on the same coordinate system with the aim to obtain a global MDS configuration.

Usage

```
divide_conquer_mds(x, l, c_points, r, n_cores)
```

Arguments

`x`	A matrix with n points (rows) and k variables (columns).
`l`	The size for which classical MDS can be computed efficiently (using `cmdscale` function). It means that if $\bar{l}$ is the limit size for which classical MDS is applicable, then `l`$\leq \bar{l}$.
`c_points`	Number of points used to align the MDS solutions obtained by the division of `x` into p data subsets. Recommended value: `5·r`.
`r`	Number of principal coordinates to be extracted.
`n_cores`	Number of cores wanted to use to run the algorithm.

Details

The divide-and-conquer MDS starts dividing the n points into p partitions: the first partition contains `l` points and the others contain `l-c_points` points. Therefore, $p = 1+$`(n-l)/(l-c_points)`. The partitions are created at random.

Once the partitions are created, `c_points` different random points are taken from the first partition and concatenated to the other partitions After that, classical MDS is applied to each partition, with target low dimensional configuration `r`.

Since all the partitions share `c_points` points with the first one, Procrustes can be applied in order to align all the configurations. Finally, all the configurations are concatenated in order to obtain a global MDS configuration.

Value

Returns a list containing the following elements:

points A matrix that consists of n points (rows) and `r` variables (columns) corresponding to the principal coordinates. Since a dimensionality reduction is performed, `r`$<<k$

eigen The first `r` largest eigenvalues: $\bar{\lambda}_i, i \in \{1, \ldots, r\}$, where $\bar{\lambda}_i = 1/p \sum_{j=1}^{p} \lambda_i^j / n_j$, being λ_i^j the $i-th$ eigenvalue from partition j and n_j the size of the partition j.

References

Delicado P. and C. Pachón-García (2021). *Multidimensional Scaling for Big Data*. https://arxiv.org/abs/2007.11919.

Borg, I. and P. Groenen (2005). *Modern Multidimensional Scaling: Theory and Applications*. Springer.

Examples

```
set.seed(42)
x <- matrix(data = rnorm(4 * 10000), nrow = 10000) %*% diag(c(9, 4, 1, 1))
mds <- divide_conquer_mds(x = x, l = 200, c_points = 5 * 2, r = 2, n_cores = 1)
head(mds$points)
mds$eigen
```

`fast_mds` *Fast MDS*

Description

Fast MDS uses recursive programming in combination with a divide and conquer strategy in order to obtain an MDS configuration for a given large data set `x`.

Usage

```
fast_mds(x, l, s_points, r, n_cores)
```

Arguments

`x`	A matrix with n individuals (rows) and k variables (columns).
`l`	The size for which classical MDS can be computed efficiently (using `cmdscale` function). It means that if $\bar{l}$ is the limit size for which classical MDS is applicable, then `l`$\leq \bar{l}$.
`s_points`	Number of points used to align the MDS solutions obtained by the division of `x` into p submatrices. Recommended value: `5`$\cdot$`r`.
`r`	Number of principal coordinates to be extracted.
`n_cores`	Number of cores wanted to use to run the algorithm.

Details

Fast MDS randomly divides the whole sample data set, x, of size n into p =l/s_points data subsets, where l $\leq \bar{l}$ being $\bar{l}$ the limit size for which classical MDS is applicable. Each one of the p data subsets has size $\tilde{n} = n/p$. If $\tilde{n} \leq$ l then classical MDS is applied to each data subset. Otherwise, fast MDS is recursively applied. In either case, a final MDS configuration is obtained for each data subset.

In order to align all the partial solutions, a small subset of size s_points is randomly selected from each data subset. They are joined to form an alignment set, over which classical MDS is performed giving rise to an alignment configuration. Every data subset shares s_points points with the alignment set. Therefore every MDS configuration can be aligned with the alignment configuration using a Procrustes transformation.

Value

Returns a list containing the following elements:

points A matrix that consists of n individuals (rows) and r variables (columns) corresponding to the principal coordinates. Since we are performing a dimensionality reduction, r$<< k$

eigen The first r largest eigenvalues: $\bar{\lambda}_i, i \in \{1, \ldots, r\}$, where $\bar{\lambda}_i = 1/p \sum_{j=1}^{p} \lambda_i^j / n_j$, being λ_i^j the $i - th$ eigenvalue from partition j and n_j the size of the partition j.

References

Delicado P. and C. Pachón-García (2021). *Multidimensional Scaling for Big Data*. https://arxiv.org/abs/2007.11919.

Yang, T., J. Liu, L. McMillan and W.Wang (2006). *A fast approximation to multidimensional scaling*. In Proceedings of the ECCV Workshop on Computation Intensive Methods for Computer Vision (CIMCV).

Borg, I. and P. Groenen (2005). *Modern Multidimensional Scaling: Theory and Applications*. Springer.

Examples

```
set.seed(42)
x <- matrix(data = rnorm(4 * 10000), nrow = 10000) %*% diag(c(9, 4, 1, 1))
mds <- fast_mds(x = x, l = 200, s_points = 5 * 2, r = 2, n_cores = 1)
head(mds$points)
mds$eigen
```

---

interpolation_mds *Interpolation MDS*

Description

Given that the size of the data set is too large, this algorithm consists of taking a random sample from it of size $l \leq \bar{l}$, being $\bar{l}$ the limit size for which classical MDS is applicable, to perform classical MDS to it, and to extend the obtained results to the rest of the data set by using Gower's interpolation formula, which allows to add a new set of points to an existing MDS configuration.

Usage

```
interpolation_mds(x, l, r, n_cores)
```

Arguments

x	A matrix with n individuals (rows) and k variables (columns).
l	The size for which classical MDS can be computed efficiently (using cmdscale function). It means that if $\bar{l}$ is the limit size for which classical MDS is applicable, then $l \leq \bar{l}$.
r	Number of principal coordinates to be extracted.
n_cores	Number of cores wanted to use to run the algorithm.

Details

Gower's interpolation formula is the central piece of this algorithm since it allows to add a new set of points to an existing MDS configuration so that the new one has the same coordinate system.

Given the matrix x with n points (rows) and and k variables (columns), a first data subsets (based on a random sample) of size l is taken and it is used to compute a MDS configuration.

The remaining part of x is divided into $p = (n-l)/l$ data subsets (randomly). For every data subset, it is obtained a MDS configuration by means of *Gower's interpolation formula* and the first MDS configuration obtained previously. Every MDS configuration is appended to the existing one so that, at the end of the process, a global MDS configuration for x is obtained.

This method is similar to `landmark_mds()` and `reduced_mds()`.

Value

Returns a list containing the following elements:

points A matrix that consists of n individuals (rows) and r variables (columns) corresponding to the principal coordinates. Since we are performing a dimensionality reduction, $r << k$

eigen The first r largest eigenvalues: $\lambda_i, i \in \{1, \ldots, r\}$, where each λ_i is obtained from applying classical MDS to the first data subset.

References

Delicado P. and C. Pachón-García (2021). *Multidimensional Scaling for Big Data*. https://arxiv.org/abs/2007.11919.

Borg, I. and P. Groenen (2005). *Modern Multidimensional Scaling: Theory and Applications*. Springer.

Gower JC. (1968). *Adding a point to vector diagrams in multivariate analysis*. Biometrika.

Examples

```
set.seed(42)
x <- matrix(data = rnorm(4 * 10000), nrow = 10000) %*% diag(c(9, 4, 1, 1))
mds <- interpolation_mds(x = x, l = 200, r = 2, n_cores = 1)
head(mds$points)
mds$eigen
```

`landmark_mds` *Landmark MDS*

Description

Landmark MDS (LMDS) algorithm applies first classical MDS to a subset of the data (*landmark points*) and then the remaining individuals are projected onto the landmark low dimensional configuration using a distance-based triangulation procedure.

Usage

```
landmark_mds(x, num_landmarks, r)
```

Arguments

`x`	A matrix with n points (rows) and k variables (columns).
`num_landmarks`	Number of landmark points to obtain an initial MDS configuration. It is equivalent to `l` parameter used in `interpolation_mds()`, `divide_conquer_mds()` and `fast_mds()`. Therefore, it is the size for which classical MDS can be computed efficiently (using `cmdscale` function). It means that if $\bar{l}$ is the limit size for which classical MDS is applicable, then `l`$\leq \bar{l}$.
`r`	Number of principal coordinates to be extracted.

Details

LMDS applies first classical MDS to a subset of the data (*landmark points*). Then, it uses a distance-based triangulation procedure to project the non-landmark individuals. This distance-based triangulation procedure coincides with *Gower's interpolation formula*.

This method is similar to `interpolation_mds()` and `reduced_mds()`.

Value

Returns a list containing the following elements:

points A matrix that consists of n points (rows) and `r` variables (columns) corresponding to the principal coordinates. Since a dimensionality reduction is performed, `r`$<< k$

eigen The first `r` largest eigenvalues: $\lambda_i, i \in \{1, \ldots, r\}$, where each λ_i is obtained from applying classical MDS to the first data subset.

References

Delicado P. and C. Pachón-García (2021). *Multidimensional Scaling for Big Data*. https://arxiv.org/abs/2007.11919.

Borg, I. and P. Groenen (2005). *Modern Multidimensional Scaling: Theory and Applications*. Springer.

De Silva V. and JB. Tenenbaum (2004). *Sparse multidimensional scaling using landmark points*. Technical Report, Stanford University.

Gower JC. (1968). *Adding a point to vector diagrams in multivariate analysis*. Biometrika.

Examples

```
set.seed(42)
x <- matrix(data = rnorm(4 * 10000), nrow = 10000) %*% diag(c(9, 4, 1, 1))
mds <- landmark_mds(x = x, num_landmarks = 200, r = 2)
head(mds$points)
mds$eigen
```

`pivot_mds` *Pivot MDS*

Description

Pivot MDS, introduced in the literature of graph layout algorithms, is similar to Landmark MDS (`landmark_mds()`) but it uses the distance information between landmark and non-landmark points to improve the initial low dimensional configuration, as more relations than just those between landmark points are taken into account.

Usage

```
pivot_mds(x, num_pivots, r)
```

Arguments

`x`	A matrix with n individuals (rows) and k variables (columns).
`num_pivots`	Number of pivot points to obtain an initial MDS configuration. It is equivalent to `l` parameter used in `interpolation_mds()`, `divide_conquer_mds()` and `fast_mds()`. Therefore, it is the size for which classical MDS can be computed efficiently (using `cmdscale` function). It means that if $\bar{l}$ is the limit size for which classical MDS is applicable, then `l`$\leq \bar{l}$.
`r`	Number of principal coordinates to be extracted.

Value

Returns a list containing the following elements:

points A matrix that consists of n individuals (rows) and `r` variables (columns) corresponding to the principal coordinates. Since we are performing a dimensionality reduction, `r`$<<k$

eigen The first `r` largest eigenvalues: $\lambda_i, i \in \{1, \ldots, r\}$, where each λ_i is obtained from applying classical MDS to the first data subset.

References

Delicado P. and C. Pachón-García (2021). *Multidimensional Scaling for Big Data*. https://arxiv.org/abs/2007.11919.

Brandes U. and C. Pich (2007). *Eigensolver Methods for Progressive Multidimensional Scaling of Large Data*. Graph Drawing.

Borg, I. and P. Groenen (2005). *Modern Multidimensional Scaling: Theory and Applications*. Springer.

Gower JC. (1968). *Adding a point to vector diagrams in multivariate analysis*. Biometrika.

Examples

```
set.seed(42)
x <- matrix(data = rnorm(4 * 10000), nrow = 10000) %*% diag(c(9, 4, 1, 1))
mds <- pivot_mds(x = x, num_pivots = 200, r = 2)
head(mds$points)
mds$eigen
```

`reduced_mds` *Reduced MDS*

Description

A data subset is selected and classical MDS is performed on it to obtain the corresponding low dimensional configuration.Then the reaming points are projected onto this initial configuration.

Usage

```
reduced_mds(x, l, r, n_cores)
```

Arguments

`x`	A matrix with n individuals (rows) and k variables (columns).
`l`	The size for which classical MDS can be computed efficiently (using `cmdscale` function). It means that if $\bar{l}$ is the limit size for which classical MDS is applicable, then `l`$\leq \bar{l}$.
`r`	Number of principal coordinates to be extracted.
`n_cores`	Number of cores wanted to use to run the algorithm.

Details

Gower's interpolation formula is the central piece of this algorithm since it allows to add a new set of points to an existing MDS configuration so that the new one has the same coordinate system.

Given the matrix `x` with n points (rows) and and k variables (columns), a first data subsets (based on a random sample) of size `l` is taken and it is used to compute a MDS configuration.

The remaining part of `x` is divided into $p = (n-\texttt{l})/\texttt{l}$ data subsets (randomly). For every data point, it is obtained a MDS configuration by means of *Gower's interpolation formula* and the first MDS configuration obtained previously. Every MDS configuration is appended to the existing one so that, at the end of the process, a global MDS configuration for `x` is obtained.

#'This method is similar to `landmark_mds()` and `interpolation_mds()`.

Value

Returns a list containing the following elements:

points A matrix that consists of n individuals (rows) and `r` variables (columns) corresponding to the principal coordinates. Since we are performing a dimensionality reduction, `r`$<<k$

eigen The first `r` largest eigenvalues: $\lambda_i, i \in \{1, \ldots, r\}$, where each λ_i is obtained from applying classical MDS to the first data subset.

References

Delicado P. and C. Pachón-García (2021). *Multidimensional Scaling for Big Data*. https://arxiv.org/abs/2007.11919.

Paradis E. (2018). *Multidimensional Scaling With Very Large Datasets*. Journal of Computational and Graphical Statistics.

Borg, I. and P. Groenen (2005). *Modern Multidimensional Scaling: Theory and Applications*. Springer.

Gower JC. (1968). *Adding a point to vector diagrams in multivariate analysis*. Biometrika.

Examples

```
set.seed(42)
x <- matrix(data = rnorm(4 * 10000), nrow = 10000) %*% diag(c(9, 4, 1, 1))
mds <- reduced_mds(x = x, l = 200, r = 2, n_cores = 1)
head(mds$points)
mds$eigen
```

Index