Package 'bigmds'

March 29, 2021

Title Multidimensional Scaling for Big Data

Version 1.0.0

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 three algorithms:
- Divide and Conquer MDS developed by Delicado and Pachon-Garcia, (2020) <arXiv:2007.11919>.
- Fast MDS, which is an implementation of Tynia, Y., L. Jinze, M. Leonard, and W. Wei, (2006).
- MDS based on Gower interpolation, which uses Gower interpolation formula as described in Gower, J.C. and D.J, Hand (1995, ISBN: 978-0-412-71630-0).
The main idea of these methods is based on partitioning the dataset into small pieces, where classical methods can work. In order to align all the solutions, it is used Procrustes formula as described in Borg, I. and Groenen, P. (2005, ISBN : 978-0-387-25150-9).

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.1

Imports stats, pdist

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

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

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

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2021-03-29 13:50:05 UTC

R topics documented:

`divide_conquer_mds` *Divide and Conquer MDS*

Description

Performs *Multidimensional Scaling* for big datasets using a Divide and Conquer strategy. This method can compute a MDS configuration even when the dataset is so large that classical MDS methods (`cmdscale`) can not be run due to computational problems.

Usage

```
divide_conquer_mds(x, l, tie, k, dist_fn = stats::dist, ...)
```

Arguments

`x`	A matrix with n individuals (rows) and q variables (columns).
`l`	The largest value which allows classical MDS to be computed efficiently, i.e, the largest value which makes `cmdscale()` be run without any computational issues.
`tie`	Number of points used to align the MDS solutions obtained by the division of `x` into p submatrices. Recommended value: `2·k`.
`k`	Number of principal coordinates to be extracted.
`dist_fn`	Distance function to be used for obtaining a MDS configuration.
`...`	Further arguments passed to `dist_fn` function.

Details

In order to obtain a MDS configuration for the entire matrix `x`, it is needed to break the dataset into p submatrices (*Divide and Conquer strategy*).

In order to obtain p, `tie` and `l` are taken into account: `p=n/(l-tie)`. This allows to use `cmdscale` function in every submatrix.

Taking into account that given a MDS solution, any rotation is another (valid) MDS solution, it is needed a way to obtain the same coordinate system for all the partitions.

To achieve such a common coordinate system, the algorithm starts by taking the first partition and calculating a MDS configuration as well as a subsample of size `tie` (from the partition, not from its MDS configuration). These `tie` points will be used in order to force the other partitions to have the same coordinate system as the first one.

Given a partition, the `tie` points are appended to it. After that, a MDS configuration is obtained. Therefore, for these `tie` points there are two MDS solutions. In order to aligned them, Procrustes parameters are obtained. These parameters are applied to the MDS configuration of the partition.

Value

Returns a list containing the following elements:

points A matrix that consists of n individuals (rows) and k variables (columns) corresponding to the MDS coordinates.

eigen The first k eigenvalues.

References

Delicado P. and C. Pachon-Garcia (2020). *Multidimensional Scaling for Big Data*. https://arxiv.org/abs/2007.11919

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

Examples

```
set.seed(42)
x <- matrix(data = rnorm(4*10000), nrow = 10000) %*% diag(c(15, 10, 1, 1))
mds <- divide_conquer_mds(x = x, l = 200, tie = 2*2, k = 2, dist_fn = stats::dist)
head(cbind(mds$points, x[, 1:2]))
var(x)
var(mds$points)
```

`fast_mds` *Fast MDS*

Description

Performs *Multidimensional Scaling* for big datasets using a recursive algorithm. This method can compute a MDS configuration even when the dataset is so large that classical MDS methods (`cmdscale`) can not be run due to computational problems.

Usage

```
fast_mds(x, l, s, k, dist_fn = stats::dist, ...)
```

Arguments

`x`	A matrix with n individuals (rows) and q variables (columns).
`l`	The largest value which allows classical MDS to be computed efficiently, i.e, the largest value which makes `cmdscale()` be run without any computational issues.
`s`	Number of points used to align the MDS solutions obtained by the division of `x` into p submatrices. Recommended value: $2 \cdot$ `k`.
`k`	Number of principal coordinates to be extracted.
`dist_fn`	Distance function to be used for obtaining a MDS configuration.
`...`	Further arguments passed to `dist_fn` function.

Details

In order to obtain a MDS configuration for the entire matrix x, it is partitioned into p submatrices, where p=l/s.

For every partition cmdscale is applied if the number of observations is less than l. Otherwise, fast_mds is called. Notice that in this part is where this algorithm becomes a recursive one.

Once every submatrix has its own MDS configuration, s (random) points are taken from every partition of x. These points are put into a matrix M. Notice that M has s·p rows and q columns.

After that, a MDS configuration for M is obtained. So, there are 2 configurations for the s points: one from performing MDS over every partition and another one from M. This allows to compute Procrustes (alignment method) so that all the MDS solutions share the same coordinate system.

Value

Returns a list containing the following elements:

points A matrix that consists of n individuals (rows) and k variables (columns) corresponding to the MDS coordinates.

eigen The first k eigenvalues.

References

Tynia, Y., L. Jinze, M. Leonard, and W. Wei (2006). *A fast approximation to multidimensional scaling*. Proceedings of the ECCV Workshop on Computation Intensive Methods for Computer Vision (CIMCV). http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.79.2445

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

Examples

```
set.seed(42)
x <- matrix(data = rnorm(4*10000), nrow = 10000) %*% diag(c(15, 10, 1, 1))
mds <- fast_mds(x = x, l = 200, s = 2*2, k = 2, dist_fn = stats::dist)
head(cbind(mds$points, x[, 1:2]))
var(x)
var(mds$points)
```

---

gower_interpolation_mds

MDS based on Gower interpolation formula

Description

Performs *Multidimensional Scaling* for big datasets using Gower interpolation formula. This method can compute a MDS configuration even when the dataset is so large that classical MDS methods (cmdscale) can not be run due to computational problems.

Usage

```
gower_interpolation_mds(x, l, k, dist_fn = stats::dist, ...)
```

Arguments

x	A matrix with n individuals (rows) and q variables (columns).
l	The largest value which allows classical MDS to be computed efficiently, i.e, the largest value which makes `cmdscale()` be run without any computational issues.
k	Number of principal coordinates to be extracted.
dist_fn	Distance function to be used for obtaining a MDS configuration.
...	Further arguments passed to `dist_fn` function.

Details

Gower 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 individuals (rows) and q variables (columns), a submatrix based on a random sample of `l` individuals is taken and it is used to compute a MDS configuration.

The remaining part of `x` is divided into p=(n-`l`)/`l` submatrices. For every submatrix, it is obtained a MDS configuration by means of *Gower interpolation formula* and the first (random) submatrix. Every MDS configuration is appended to the existing one so that, at the end of the process, a MDS configuration for `x` is built.

Value

Returns a list containing the following elements:

points A matrix that consists of n individuals (rows) and `k` variables (columns) corresponding to the MDS coordinates.

eigen The first `k` eigenvalues.

References

Gower, J.C. and D.J, Hand (1995). *Biplots*. Volume 54. CRC Press.

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

Examples

```
set.seed(42)
x <- matrix(data = rnorm(4*10000), nrow = 10000) %*% diag(c(15, 10, 1, 1))
mds <- gower_interpolation_mds(x = x, l = 200, k = 2, dist_fn = stats::dist)
head(cbind(mds$points, x[, 1:2]))
var(x)
var(mds$points)
```

Index