SVD-MDS

March 3, 2017

Type Package

Title SVDMDS

Version 1.0.0

Author Christophe BECAVIN and Nicolas TCHITCHEK

Maintainer Nicolas TCHITCHEK <nicolas.tchitchek@gmail.com>

Description R implementation of the SVD-MDS algorithm

License GPL-3 | file LICENSE

Depends R (>= 3.1),

Imports Rcpp (>= 0.12.4),
ggplot2,
plyr

biocViews DimensionReduction, Visualization, Transcriptomics

LinkingTo Rcpp

RoxygenNote 5.0.1

R topics documented:

`computeEntourageScore` *Computation of the Entourage Score*

Description

This function is used to compute the Entourage Score between two distance matrices, which quantifies the local quality of a MDS representation.

The Entourage Score corresponds to the normalised number of identical nearest neighbours for each object in the two distance matrices.

Usage

```
computeEntourageScore(dist1, dist2, k = 3)
```

Arguments

dist1	a numeric matrix of the first distance matrix
dist2	a numeric matrix of the second distance matrix
k	a numeric indicating the number of nearest neighbours to compare

Details

This function has been implemented in C++ for fast computations and can be executed from R using this wrapper.

Value

a numeric value of the Entourage Score

---

computeKruskalStress *Computation of the Kruskal Stress*

Description

This function is used to compute the Kruskal Stress between two distance matrices, which quantifies the global quality of a MDS representation.

The Kruskal Stress corresponds to the quantify of information lost during the dimentionality reduction process.

Usage

```
computeKruskalStress(dist1, dist2)
```

Arguments

dist1	a numeric matrix of the first distance matrix
dist2	a numeric matrix of the second distance matrix

Details

This function has been implemented in C++ for fast computations and can be executed from R using this wrapper.

Value

a numeric value the Kruskal Stress

distEuclidean *Computation of a distance matrix using the Euclidean metric*

Description

This function computes the distance matrix from a numeric matrix using the Euclidean metric.

Each row of the input matrix must corresponds to an object and each column must corresponds to an attribute.

Usage

```
distEuclidean(data)
```

Arguments

data	a numeric matrix. Rows must correspond to the particles and columns must correspond to the attributes

Details

This function has been implemented in C++ for fast computations and can be executed from R using this wrapper.

Value

a numeric matrix containing the Euclidean distances betweens all particles

distManhattan *Computation of a distance matrix using the Manhattan metric*

Description

This function computes the distance matrix from a numeric matrix using the Manhattan metric.

Each row of the input matrix must corresponds to an object and each column must corresponds to an attribute.

Usage

```
distManhattan(data)
```

Arguments

data	a numeric matrix. Rows must correspond to the objects and columns must correspond to the attributes

Details

This function has been implemented in C++ for fast computations and can be executed from R using this wrapper.

Value

a numeric matrix containing the Manhattan distances betweens all objects

---

`plotMDS` *Generation of ggplot graphical representations for MDS Reference Maps and MDS Projections.*

Description

This function generates a graphical representation for MDS Reference Maps and MDS Projections. Objects can be shaped and colored using the 'color' and 'shape' parameters. Convex hulls for given sets of objects can also be displayed.

Usage

```
plotMDS(mds, color = NULL, shape = NULL, polygon = NULL, title = "MDS",
  display.legend = TRUE)
```

Arguments

`mds`	a MDS result provided by the 'MDSReferenceMap()', 'MDSProjection()', or 'MDSReferenceMapwithProjection()' functions
`color`	a vector used to color the points
`shape`	a vector used for the shape of the points
`polygon`	a vector used to color the convex hull
`title`	a character specifying the title of the representation
`display.legend`	a logical specifying if the graphic legend must be displayed

---

`setConstants` *Setting of the algorithm constants*

Description

This function is used to set the different constants of the MDSRefMaps algorithm.

Usage

```
setConstants(K = 1, F = 0.1, DELTA_T = 0.001, MASS = 10,
  ACC_THRESHOLD = 0)
```

Arguments

`K`	a numeric value indicating the spring strength between two particles
`F`	a numeric value indicating the friction of the springs
`DELTA_T`	a numeric value indicating the time between two steps of the algorithm
`MASS`	a numeric value indicating the mass of each particle
`ACC_THRESHOLD`	a numeric value indicating the acceleration threshold (a value of 0 disactivate this acceleration threshold)

Details

The 'ACC_THRESHOLD' parameter can be used to contrain the object accelerations.

`SVDMDS` *Construction of a SVD-MDS representation*

Description

This function computes a SVD-MDS representation based on a distance matrix.

Usage

```
SVDMDS(dist, k = 2, metric = "euclidean", max_it = 6 * 10^6,
  stress_sd_th = 10^-4, stack_length = 500, verbose = TRUE)
```

Arguments

`dist`	a numeric matrix with all pairwise distances between objects of the representation
`k`	a numeric value specifying the desired number of dimensions in the resulting Reference Map representation
`metric`	a character indicating the distance metric to use ("euclidean" or "manhattan")
`max_it`	a numeric defining the maximal number of steps the algorithm can perform
`stress_sd_th`	a numeric defining the threshold for the standard deviation of Kruskal Stress
`stack_length`	a numeric defining the length of the Kruskal Stress stack (used to compute the standard deviation of the Kruskal Stress)
`verbose`	a boolean enabling the display of debug information at each step of the algorithm

Value

a list of 3 elements containing the position of the objects ('points' element), the Kruskal Stress ('stress' element), and the Entourage Score ('entourage' element)

Index