Package 'sincell'

December 5, 2025

Type Package

Title R package for the statistical assessment of cell state hierarchies from single-cell RNA-seq data

Version 1.43.0

Date 2015-05-28

Author Miguel Julia <migueljuliamolina@gmail.com>, Amalio Telenti <atelenti@jcvi.org>, Antonio Rausell <antonio.rausell@institutimagine.org>

Maintainer Miguel Julia <migueljuliamolina@gmail.com>, Antonio Rausell<antonio.rausell@institutimagine.org>

Depends R (>= 3.0.2), igraph

Description Cell differentiation processes are achieved through a continuum of hierarchical intermediate cell-states that might be captured by single-cell RNA seq. Existing computational approaches for the assessment of cell-state hierarchies from single-cell data might be formalized under a general workflow composed of i) a metric to assess cell-to-cell similarities (combined or not with a dimensionality reduction step), and ii) a graph-building algorithm (optionally making use of a cells-clustering step). Sincell R package implements a methodological toolbox allowing flexible workflows under such framework. Furthermore, Sincell contributes new algorithms to provide cell-state hierarchies with statistical support while accounting for stochastic factors in single-cell RNA seq. Graphical representations and functional association tests are provided to interpret hierarchies.

License GPL (>= 2)

Encoding UTF-8

URL http://bioconductor.org/

Imports Rcpp (>= 0.11.2), entropy, scatterplot3d, MASS, TSP, ggplot2, reshape2, fields, proxy, parallel, Rtsne, fastICA, cluster, statmod

LinkingTo Rcpp

VignetteBuilder knitr

Suggests BiocStyle, knitr, biomaRt, stringr, monocle

biocViews ImmunoOncology, Sequencing, RNASeq, Clustering, GraphAndNetwork, Visualization, GeneExpression, GeneSetEnrichment, BiomedicalInformatics, CellBiology, FunctionalGenomics, SystemsBiology

git_url https://git.bioconductor.org/packages/sincell

git_branch devel

git_last_commit 1afb2cc

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2025-12-04

Contents

`ExpressionMatrix` *Single-cell expression data for genes differentially expressed in differentiating human skeletal muscle myoblasts cells*

Description

This dataset contains expression profiles from a time-series study of differentiating human skeletal muscle myoblasts (object HSMM in Bioconductor package monocle). Expression values are in FPKM units. Data is part of a publicly available single-cell RNA-seq dataset from Trapnell et al 2014. In this work, authors generated single-cell RNA-seq libraries for differentiating myoblasts at 0, 24, 48 and 72 hours. Original data can be accessed at GEO database accession number GSE52529. Following Trapnell et al 2014 and the vignette of its associated Bioconductor package Monocle, the expression matrix is restricted to genes differentially expressed between cells from times 0 and the ensemble of cells of times 24, 28 and 72 hours of differentiation. Steps to achieve this are reported in monocle's vignette. Those steps produce the matrix ExpressionMatrix represent¡ng the expression profiles of those differentially expressed genes. ExpressionMatrix is provided as part of Sincell package in order to keep the running time of its vignette short.

Usage

```
data(ExpressionMatrix)
ExpressionMatrix
```

Format

Numeric matrix

Source

sincell

`f_distance2vector` *Conversion of the lower triangular matrix of a distance matrix into an array*

Description

Auxiliary function to convert of the lower triangular matrix of a distance matrix into an array

Usage

```
f_distance2vector(distance)
```

Arguments

`distance`	A distance matrix or a symmetric matrix

Value

Array resulting from the concatenation of the rows of the lower triangular matrix of the input symetric matrix. Array length is n*(n-1)/2, where n is the number of rows of the symetric matrix.

Examples

```
## Generate some data
Data <- matrix(rnorm(300),ncol=10,nrow=30)

## Calculate distance matrix and transform its lower triangular matrix into a one
## dimensional array
d <- f_distance2vector(as.matrix(dist(Data)))
```

---

geneset.list *Example of a geneset collection*

Description

a gene set collection provided for illustrative purposes in the vignette

Usage

```
data(geneset.list)
geneset.list
```

Format

List of character arrays

Source

sincell

---

knnalgorithm *Auxiliary function for KNN and IMC algorithms*

Description

Auxiliary function

Usage

```
knnalgorithm(distance, mutual, k)
```

Arguments

distance	distance matrix
mutual	logical specifying if the connection between neighbors must be mutual
k	maximum order of neighbors

Value

An adjacency matrix is returned.

See Also

sc_GraphBuilderObj(), sc_clusterObj()

`pseudoreplicatesbymodel` *Auxiliary function of sc_InSilicoCellsReplicatesObj function used when its parameter method="lognormal-3parameters"*

Description

Auxiliary function implemented in C++ making part of the sc_InSilicoCellsReplicatesObj function

Usage

```
pseudoreplicatesbymodel(rows, colums, alpha, vargenes, meangenes, positive, f, seed)
```

Arguments

`rows`	number of rows in list "expressionmatrix" within Sincell object
`colums`	number of colums in list "expressionmatrix" within Sincell object
`alpha`	Vector containing for each gene in "expressionmatrix" the proportion of cells where expression was detected above a given threshold level (parameter "no_expr" in function sc_InSilicoCellsReplicatesObj())
`vargenes`	Vector containing for each gene in "expressionmatrix" the variance of the expression levels in those cells where expression was detected above a given threshold level (parameter "no_expr" in function sc_InSilicoCellsReplicatesObj())
`meangenes`	Vector containing for each gene in "expressionmatrix" the average of the expression levels in those cells where expression was detected above a given threshold level (parameter "no_expr" in function sc_InSilicoCellsReplicatesObj())
`positive`	Force the new matrix to be positive. 1 for TRUE, 0 for FALSE
`f`	R function rnorm
`seed`	seed integer for random generation

Value

A numeric matrix is returned as described in sc_InSilicoCellsReplicatesObj when method="lognormal-3parameters"

See Also

sc_InSilicoCellsReplicatesObj()

pseudoreplicatesbynoise

Auxiliary function of sc_InSilicoCellsReplicatesObj function used when its parameter method="variance.deciles"

Description

Auxiliary function implemented in C++ making part of the sc_InSilicoCellsReplicatesObj function when its parameter method="variance.deciles".

Usage

```
pseudoreplicatesbynoise(originaldata, rows, colums, deciles,
  lengthdeciles, coorsorted, vargenessorted, positive, seed)
```

Arguments

originaldata	"expressionmatrix" within Sincell object: numeric matrix containing a gene expression matrix gathering the expression levels of each single-cell in the experiment (displayed by columns) for each detected gene (displayed by rows)
rows	number of rows in list "expressionmatrix" within Sincell object
colums	number of colums in list "expressionmatrix" within Sincell object
deciles	array containing the indexes indicating the limits of the deciles based on mean of gene expression
lengthdeciles	length(deciles)
coorsorted	order of permutated indexes
vargenessorted	Vector containing for each gene in "expressionmatrix" the variance of the expression levels. Order of genes corresponds to mean expression levels (increasing order)
positive	Force the new matrix to be positive. 1 for TRUE, 0 for FALSE
seed	seed integer for random generation

Value

A numeric matrix is returned as described in sc_InSilicoCellsReplicatesObj when method="variance.deciles"

See Also

sc_InSilicoCellsReplicatesObj()

pseudoreplicatesbynoise_cv2 *Auxiliary function of sc_InSilicoCellsReplicatesObj function used when its parameter method="cv2.deciles"*

Description

Auxiliary function implemented in C++ making part of the sc_InSilicoCellsReplicatesObj function when its parameter method="cv2.deciles"

Usage

```
pseudoreplicatesbynoise_cv2(originaldata, rows, colums, deciles, lengthdeciles,
  coorsorted, vargenessorted, means, positive, seed)
```

Arguments

originaldata	"expressionmatrix" within Sincell object: numeric matrix containing a gene expression matrix gathering the expression levels of each single-cell in the experiment (displayed by columns) for each detected gene (displayed by rows)
rows	number of rows in list "expressionmatrix" within Sincell object
colums	number of colums in list "expressionmatrix" within Sincell object
deciles	array containing the indexes indicating the limits of the deciles based on mean of gene expression
lengthdeciles	length(deciles)
coorsorted	order of permutated indexes
vargenessorted	Vector containing for each gene in "expressionmatrix" the squared coefficient of variation cv2 of the expression levels. Order of genes corresponds to mean expression levels (increasing order)
means	Vector containing for each gene in "expressionmatrix" the mean of the expression levels. Order of genes corresponds to mean expression levels (increasing order)
positive	Force the new matrix to be positive. 1 for TRUE, 0 for FALSE
seed	seed integer for random generation

Value

A numeric matrix is returned as described in sc_InSilicoCellsReplicatesObj when method="cv2.deciles"

See Also

sc_InSilicoCellsReplicatesObj()

`sc_AssociationOfCellsHierarchyWithAGeneSet` *Association of a cell-state hierarchy with a functional gene set*

Description

First, this function assesses a cell-state hierarchy where only the expression levels of the genes in a given functional gene set are considered. Second, it calculates the similarity of that hierarchy with the one assessed by function sc_GraphBuilderObj() on the initial gene expression matrix. Third it provides an empirical p-value of the observed similarity between the two hierarchies. The hierarchy resulting when considering only the genes in the gene set is assessed with exactly the same parameters used to obtain the reference hierarchy. The similarity between the two hierarchies is computed as the spearman rank correlation between the two graphs of the shortest distance for all pairs of cells. The empirical p-value is calculated from a distribution of similarities resulting from random samplings of gene sets of the same size.

Usage

```
sc_AssociationOfCellsHierarchyWithAGeneSet(SincellObject,GeneSet,
  minimum.geneset.size=50,p.value.assessment=TRUE,
  spearman.rank.threshold=0.5,num_it=1000,
  cores=ifelse(detectCores()>=4, 4, detectCores()))
```

Arguments

`SincellObject`	A SincellObject named list as created by function sc_GraphBuilderObj(), containing in member "cellstateHierarchy" a connected graph representing a cell-state hierarchy.
`GeneSet`	A character vector containing the gene names of a functional gene set. Gene names should be of the same type as those used in the gene expression matrix.
`minimum.geneset.size`	Minimum number of genes from the gene set that should be present in the original gene expression matrix and that have a non-zero variance across cells. If that overlap is is lower than this parameter, the association will not be computed.
`p.value.assessment`	A logical value indicating whether an empirical p-value of the similarity should be calculated.
`spearman.rank.threshold`	The minimum value of the spearman rank correlation that the two hierarchies should have to allow computation of an empirical p-value. This limit is set in order to avoid an extra computation time invested in getting an empirical p-value for a low correlation not worthy of consideration.
`num_it`	Number of subsamplings to perform on the original gene expression matrix data contained in SincellObject[["expressionmatrix"]] to obtain the empirical p-value

cores — Number of threads used to paralyze the computation. Under Unix platforms, by default the function uses all cores up to 4 (to avoid possilbe issues while running on a cluster with the default parameter) detected by the operating system. Under non Unix based platforms, this parameter will be automatically set to 1.

Value

The SincellObject named list provided as input where following list members are added: The similarity between the reference hierarchy and the hierarchy obtained from the gene set, stored in SincellObject[["AssociationOfCellsHierarchyWithAGeneSet"]]; and its empirical p-value, stored in SincellObject[["AssociationOfCellsHierarchyWithAGeneSet.pvalue"]]

Examples

```
## Generate some random data
Data <- matrix(abs(rnorm(3000, sd=2)),ncol=10,nrow=50)
rownames(Data)<-character(dim(Data)[1])

## Generate gene names from index
for (i in 1:dim(Data)[1]){rownames(Data)[i]<-as.character(i)}

## Generate a hypothetical gene list from the first 10 gene names
myGeneSet<-rownames(Data)[1:10]

## Initializing SincellObject named list
mySincellObject <- sc_InitializingSincellObject(Data)

## Assessmet of cell-to-cell distance matrix after dimensionality reduction with
## Principal Component Analysis (PCA)
mySincellObject <- sc_DimensionalityReductionObj(mySincellObject, method="PCA",dim=2)

## Cluster
mySincellObject <- sc_clusterObj (mySincellObject, clust.method="max.distance",
  max.distance=0.5)

## Assessment of cell-state hierarchy
mySincellObject<- sc_GraphBuilderObj(mySincellObject, graph.algorithm="SST",
  graph.using.cells.clustering=TRUE)

## Assessment of association of the hierarchy with a gene set
mySincellObject<-sc_AssociationOfCellsHierarchyWithAGeneSet(mySincellObject,
  myGeneSet, minimum.geneset.size=9,p.value.assessment=TRUE,
  spearman.rank.threshold=0.5,num_it=1000)

## To access the similarity between the reference hierarchy and the hierarchy obtained
## from the gene set
myAssociationOfCellsHierarchyWithGeneSet<-
  mySincellObject[["AssociationOfCellsHierarchyWithAGeneSet"]]
myAssociationOfCellsHierarchyWithGeneSet.pvalue<-
  mySincellObject[["AssociationOfCellsHierarchyWithAGeneSet.pvalue"]]
```

`sc_clusterObj` *Clustering of individual cells based on a metric of choice*

Description

This function calculates a disconnected graph where the connected components are the groups generated by the selected clustering method.

In order to obtain a vector showing each cell corresponding cluster, the easiest way is by using the 'clusters()' function from igraph. For more information, check the examples below or the help page of 'clusters()', i.e. 'help(clusters)'.

Usage

```
sc_clusterObj(SincellObject, clust.method="knn", mutual=TRUE, k=3,
  max.distance=0, shortest.rank.percent=10)
```

Arguments

`SincellObject`	A SincellObject named list as created by function sc_distanceObj() or sc_DimensionalityReductionObj(), containing in member "cell2celldist" a distance matrix representing a cell-to-cell distance matrix assessed on a gene expression matrix with a metric of choice
`clust.method`	If clust.method="max.distance", clusters are defined as subgraphs generated by a maximum pair-wise distance cut-off, that is: from a totally connected graph where all cells are connected to each other, the algorithm only keeps pairs of cells connected by a distance lower than a given threshold. If clust.method="percent", clusters are defined as subgraphs generated by a given rank-percentile of the shortest pair-wise distances, that is; from a totally connected graph where all cells are connected to each other, the algorithm only keeps the top "x" percent of shortest pairwise distances as indicated by "shortest.rank.percent". If clust.method="knn", unsupervised K-Nearest Neighbors (K-NN) clustering is performed: From a totally disconnected graph where none of the cells are connected to each other, the algorithm connects each cell to its "k" nearest neighbors. If parameter "mutual=TRUE", Unsupervised K-Mutual Nearest Neighbours (K-MNN) clustering is performed, that is: only reciprocal k nearest neighbors are connected. If clust.method="k-medoids", clustering around medoids (a more robust version of k-means) is performed with function "pam" from package "cluster" on the distance matrix in mySincellObject[["cell2celldist"]] with a desired number of groups indicated in parameter "num.clusters" Hierarchical agglomerative clustering can be performed by internally calling function "hclust" where the agglomeration method is indicated in parameter "clust.method" as one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC). Clusters are obtained by cutting the tree produced by hclust

with function cutree with a desired number of groups indicated in parameter "num.clusters"

`mutual` If clust.method="knn" and "mutual=TRUE", Unsupervised K-Mutual Nearest Neighbours (K-MNN) clustering is performed, that is: only reciprocal k nearest neighbors are connected.

`k` If clust.method="knn", k is an integer specifying the number of nearest neighbors to consider in K-NN and K-KNN

`max.distance` in max.distance algorithm, select up to which distance the points will be linked

`shortest.rank.percent`
in percent algorithm, select the percent of shortest distances will be represented as links

Value

The SincellObject named list provided as input where following list members are added: "cellsClustering"=cellsClustering,"clust.method"=clust.method,"mutual"=mutual, "k"=k,"max.distance"=max.distance,"shortest.rank.p
where "cellsClustering" contains an igraph graph object (see "igraph" R package documentation) representing the result of the clustering performed with the indicated parameters.

Examples

```
## Generate some random data
Data <- matrix(abs(rnorm(3000, sd=2)),ncol=10,nrow=300)

## Initializing SincellObject named list
mySincellObject <- sc_InitializingSincellObject(Data)

## Assessmet of cell-to-cell distance matrix without dimensionality reduction
mySincellObjectA <- sc_distanceObj(mySincellObject, method="spearman")

## Assessmet of cell-to-cell distance matrix after dimensionality reduction
## with Principal Component Analysis (PCA)
mySincellObjectB <- sc_DimensionalityReductionObj(mySincellObject, method="PCA",dim=2)

## Cluster
mySincellObjectA <- sc_clusterObj (mySincellObjectA, clust.method="max.distance",
  max.distance=0.5)
mySincellObjectA <- sc_clusterObj(mySincellObjectA, clust.method="percent",
  shortest.rank.percent=10)

## To access the igraph object representing the clustering output
cellsClusteringA<-mySincellObjectA[["cellsClustering"]]

## Check each cell its corresponding cluster
clusters(cellsClusteringA)

## Cluster
mySincellObjectB <- sc_clusterObj (mySincellObjectB, clust.method="knn", mutual=FALSE, k=3)
mySincellObjectB <- sc_clusterObj (mySincellObjectB, clust.method="knn", mutual=TRUE, k=3)
```

```
## To access the igraph object representing the clustering output
cellsClusteringB<-mySincellObjectB[["cellsClustering"]]

## Check each cell its corresponding cluster
clusters(cellsClusteringB)
```

---

sc_ComparissonOfGraphs

Comparisson of graphs

Description

Function to assess a distance matrix comparing the graphs from Sincell objects that were generated with function sc_GraphBuilderObj(). The distance between two graphs is assessed as 1 minus their similarity, which is calculated as the spearman rank correlation between the two graphs of the shortest distance for all pairs of cells. Cell-state hierarchies are igraph graph objects (see "igraph" R package documentation) representing a totally connected graph.

Usage

```
sc_ComparissonOfGraphs(cellstateHierarchy1,cellstateHierarchy2, ...,
  graph.names=NULL)
```

Arguments

cellstateHierarchy1	A first cell-state hierarchy as created by function sc_GraphBuilderObj() on a SincellObject.
cellstateHierarchy2	A second cell-state hierarchy as created by function sc_GraphBuilderObj() on a SincellObject.
...	Further cell-state hierarchies
graph.names	A vector of characters indicating the names of the cell-state hierarchies provided as arguments.

Value

A distance matrix comparing the graphs.

Examples

```
## Generate some random data
Data <- matrix(abs(rnorm(3000, sd=2)),ncol=10,nrow=30)

## Initializing SincellObject
mySincellObject <- sc_InitializingSincellObject(Data)

## Assessmet of cell-to-cell distance matrix after dimensionality reduction
```

```
## with Principal Component Analysis (PCA), with Independent Component
## Analysis (ICA), or with non-metric Multidimensional Scaling (nonmetric-MDS)
mySincellObject_PCA <- sc_DimensionalityReductionObj(mySincellObject,
  method="PCA",dim=2)
mySincellObject_ICA <- sc_DimensionalityReductionObj(mySincellObject,
  method="ICA",dim=2)
mySincellObject_classicalMDS <- sc_DimensionalityReductionObj(mySincellObject,
  method="classical-MDS",dim=2)
mySincellObject_nonmetricMDS <- sc_DimensionalityReductionObj(mySincellObject,
  method="nonmetric-MDS",dim=2)

## Assessment of cell-state hierarchy
mySincellObject_PCA<- sc_GraphBuilderObj(mySincellObject_PCA,
  graph.algorithm="SST")
mySincellObject_ICA<- sc_GraphBuilderObj(mySincellObject_ICA,
  graph.algorithm="SST")
mySincellObject_classicalMDS<- sc_GraphBuilderObj(mySincellObject_classicalMDS,
  graph.algorithm="SST")
mySincellObject_nonmetricMDS<- sc_GraphBuilderObj(mySincellObject_nonmetricMDS,
  graph.algorithm="SST")

## Comparisson of hierarchies obtained from different methods
myComparissonOfGraphs<-sc_ComparissonOfGraphs(
  mySincellObject_PCA[["cellstateHierarchy"]],
  mySincellObject_ICA[["cellstateHierarchy"]],
  mySincellObject_classicalMDS[["cellstateHierarchy"]],
  mySincellObject_nonmetricMDS[["cellstateHierarchy"]],
  graph.names=c("PCA","ICA","classicalMDS","nonmetricMDS")
  )
plot(hclust(myComparissonOfGraphs))
```

`sc_DimensionalityReductionObj` *Dimensionality reduction of an expression matrix*

Description

Function to perform a dimensionality reduction upon the original gene expression matrix data through a method of choice, either linear or no-linear, among the following: Principal Component Analysis (PCA), Independent Component Analysis (ICA), t-Distributed Stochastic Neighbor Embedding (tSNE), classical Multidimensional Scaling (MDS) and non-metric Multidimensional Scaling.

Usage

```
sc_DimensionalityReductionObj(SincellObject, method="PCA", dim=2,
MDS.distance="spearman", bins=c(-Inf,0,1,2,Inf),tsne.perplexity=1,tsne.theta=0.25)
```

Arguments

SincellObject	A SincellObject named list as created by function sc_InitializingSincellObject with a named member "expressionmatrix" containing a numeric matrix that represents a gene expression matrix gathering the expression levels of each single-cell in the experiment (displayed by columns) for each detected gene (displayed by rows).
method	Dimensionality reduction algorithm to be used. Options are: Principal Component Analysis (method="PCA"), Independent Component Analysis (method="ICA"; using fastICA() function in fastICA package), t-Distributed Stochastic Neighbor Embedding (method="tSNE"; using Rtsne() function in Rtsne package with parameters tsne.perplexity=1 and tsne.theta=0.25), classical Multidimensional Scaling (method="classical-MDS"; using the cmdscale() function) and non-metric Multidimensional Scaling (method="nonmetric-MDS";using the isoMDS() function in MASS package). if method="PCA" is chosen, the proportion of variance explained by each of the principal axes is plotted. We note that Sincell makes use of the Rtsne implementation of the Barnes-Hut algorithm, which approximates the likelihood. The user should be aware that this is a less accurate version of t-SNE than e.g. the one used as basis of viSNE (Amir,E.D. et al. 2013, Nat Biotechnol 31, 545–552).
dim	Number of dimensions in low-dimensional space to be retained. Default is dim=2.
MDS.distance	Distance method to be used if method="classical-MDS" or method="nonmetric-MDS" is selected. The available distances are the Euclidean distance (method="euclidean"), Manhattan distance (also called L1 distance, method="L1"), distance based on Pearson (method="pearson") or Spearman (method="spearman") correlation coefficients, and distance based on Mutual Information (method="MI"). Intervals used to assess Mutual Information are indicated in the parameter "bins" (see below).
bins	Intervals used to discretize the data in the case that Mutual Information distance (MDS.distance="MI") is selected.
tsne.perplexity	perplexity parameter for tSNE algorithm. We refer the reader to the Frequently Asked Questions in http://homepage.tudelft.nl/19j49/t-SNE.html
tsne.theta	tradeoff between speed and accuracy. We refer the reader to the Frequently Asked Questions in http://homepage.tudelft.nl/19j49/t-SNE.html

Value

A SincellObject named list whose members are: expressionmatrix=SincellObject[["expressionmatrix"]], cellsLowDimensionalSpace=cellsLowDimensionalSpace, cell2celldist=distance,method=method, dim=dim, MDS.distance=MDS.distance, bins=bins, where cellsLowDimensionalSpace contains the coordinates of each cell (by columns) in each low dimensional axis (by rows), and "cell2celldist" contains the numeric matrix representing the cell-to-cell distance matrix assessed in low dimensional space

Examples

```
## Generate some random data
```

```
Data <- matrix(abs(rnorm(3000, sd=2)),ncol=10,nrow=300)

## Initializing SincellObject named list
mySincellObject <- sc_InitializingSincellObject(Data)

## To access the gene expression matrix
expressionmatrix<-mySincellObject[["expressionmatrix"]]

## Dimensionality reduction
# Principal Component Analysis (PCA)
mySincellObject <- sc_DimensionalityReductionObj(mySincellObject, method="PCA",dim=2)
# Independent Component Analysis (ICA)
mySincellObject <- sc_DimensionalityReductionObj(mySincellObject, method="ICA",dim=2)
# t-Distributed Stochastic Neighbor Embedding (t-SNE)
mySincellObject <- sc_DimensionalityReductionObj(mySincellObject, method="tSNE",dim=2)
# Classic Multidimensional Scaling (classic-MDS).
mySincellObject <- sc_DimensionalityReductionObj(mySincellObject, method="classical-MDS",dim=2)
# Non-metric Multidimensional Scaling (nonmetric-MDS).
mySincellObject <- sc_DimensionalityReductionObj(mySincellObject, method="nonmetric-MDS",dim=2)

## To access the coordinates of cells (by columns) in low dimensional space (axes by rows)
cellsLowDimensionalSpace<-mySincellObject[["cellsLowDimensionalSpace"]]

## To access the cell-to-cell distance matrix assessed on low dimensional space
cell2celldist<-mySincellObject[["cell2celldist"]]
```

`sc_distanceObj` *Assessment of a cell-to-cell distance matrix with a metric of choice*

Description

Function to assess a cell-to-cell distance matrix from a gene expression matrix with a metric of choice among the following: Euclidean distance, Mutual Information, L1 distance (Manhattan distance), Pearson correlation or Spearman correlation.

Usage

```
sc_distanceObj(SincellObject, method="euclidean", bins=c(-Inf,0,1,2,Inf))
```

Arguments

`SincellObject`	A SincellObject named list as created by function sc_InitializingSincellObject with a named member "expressionmatrix" containing a numeric matrix that represents a gene expression matrix gathering the expression levels of each single-cell in the experiment (displayed by columns) for each detected gene (displayed by rows).

method	Distance method to be used. The available distances are the Euclidean distance (method="euclidean"), Manhattan distance (also called L1 distance, method="L1"), cosine distance (method="cosine") , distance based on Pearson (method="pearson") or Spearman (method="spearman") correlation coefficients, and distance based on Mutual Information (method="MI"). Intervals used to assess Mutual Information are indicated in the parameter "bins" (see below).
bins	Intervals used to discretize the data in the case that Mutual Information distance (method="MI") is selected.

Value

A SincellObject named list whose members are: expressionmatrix=SincellObject[["expressionmatrix"]], cell2celldist=cell2celldist,method=method,bins=bins, where "cell2celldist" contains the numeric matrix representing the cell-to-cell distance matrix assessed by sc_distanceObj with the indicated parameters

Examples

```
## Generate some random data
Data <- matrix(abs(rnorm(3000, sd=2)),ncol=10,nrow=300)

## Initializing SincellObject named list
mySincellObject <- sc_InitializingSincellObject(Data)

## To access the gene expression matrix
expressionmatrix<-mySincellObject[["expressionmatrix"]]

## Distance
mySincellObject<-sc_distanceObj(mySincellObject)
mySincellObject<-sc_distanceObj(mySincellObject, method="MI",bins=c(-Inf,0,2,4,6,8,Inf))
mySincellObject<-sc_distanceObj(mySincellObject, method="spearman")

## To access the cell-to-cell distance matrix
cell2celldist<-mySincellObject[["cell2celldist"]]
```

---

sc_GraphBuilderObj *Graph building function for assessment of cell-state hierarchies*

Description

Function to build a connected graph from a cell-to-cell distance matrix that will be regarded as a cell-state hierarchy. Three algorithms are available: the Minimum Spanning Tree (MST), the Maximum Similarity Spanning Tree (SST) and the Iterative Mutual Clustering Graph (IMC). Optionally, algorithms in sc_GraphBuilderObj can use a precalculated clustering of cells. In the case of MST, this is used to overlay connections between pairs of cells belonging to the same cluster. In the case of SST, clusters of cells are treated as atomic elements in the graph-building process together with non-clustered cells. By definition, IMC builds a connected graph through iterations on the clustering results produced the K-Mutual Nearest Neighbour (K-MNN) algorithm.

Usage

```
sc_GraphBuilderObj(SincellObject, graph.algorithm="MST",
  graph.using.cells.clustering=FALSE,k=3)
```

Arguments

SincellObject	A SincellObject named list as created by function sc_distanceObj() or sc_DimensionalityReductionObj(), containing in member "cell2celldist" a distance matrix representing a cell-to-cell distance matrix assessed on a gene expression matrix with a metric of choice.
graph.algorithm	Graph building algorithm to be used: the Minimum Spanning Tree (graph.algorithm="MST"), the Maximum Similarity Spanning Tree (graph.algorithm="SST") and the Iterative Mutual Clustering Graph (graph.algorithm="IMC").
graph.using.cells.clustering	If graph.using.cells.clustering=TRUE and graph.algorithm="MST" or graph.algorithm="MST", a precalculated clustering of cells is used. The clustering of cells is taken from SincellObject[["cellsClustering"]] as calculated by function sc_clusterObj().
k	If IMC algorithm is selected, the number of nearest neighbors used in the underlying K-Mutual Nearest Neighbour (K-MNN) algorithm is set to k.

Value

The SincellObject named list provided as input where following list members are added: "cellstateHierarchy"=cellstateHierarchy, "graph.algorithm"=graph.algorithm, "graph.using.cells.clustering"=graph.using.cells.clusteri
The member "cellstateHierarchy" contains an igraph graph object (see "igraph" R package documentation) representing a totally connected graph built with the indicated parameters.

Examples

```
## Generate some data
## Generate some random data
Data <- matrix(abs(rnorm(3000, sd=2)),ncol=10,nrow=300)

## Initializing SincellObject named list
mySincellObject <- sc_InitializingSincellObject(Data)

## Assessmet of cell-to-cell distance matrix after dimensionality reduction with
## Principal Component Analysis (PCA)
mySincellObject <- sc_DimensionalityReductionObj(mySincellObject, method="PCA",dim=2)

## Cluster
mySincellObject <- sc_clusterObj (mySincellObject, clust.method="max.distance",
  max.distance=0.5)

## Assessment of cell-state hierarchy
mySincellObject<- sc_GraphBuilderObj(mySincellObject, graph.algorithm="MST",
  graph.using.cells.clustering=FALSE)
mySincellObject<- sc_GraphBuilderObj(mySincellObject, graph.algorithm="SST",
  graph.using.cells.clustering=TRUE)
mySincellObject<- sc_GraphBuilderObj(mySincellObject, graph.algorithm="IMC")
```

```
## To access the totally connected graph (igraph object)
cellstateHierarchy<-mySincellObject[["cellstateHierarchy"]]
```

`sc_InitializingSincellObject`
Function to initialize a sincell object

Description

Function initializes a named list with a unique member so-called "expressionmatrix" containing the input gene expression matrix. Genes with a variance equal to zero are filtered out from the gene expression matrix at this step.

Usage

```
sc_InitializingSincellObject(BaseData)
```

Arguments

`BaseData`	A numeric matrix representing a gene expression matrix gathering the normalized expression levels of each single-cell in the experiment (displayed by columns) for each detected gene (displayed by rows).

Value

a named list: list(expressionmatrix=BaseData)

Examples

```
## Generate some random data
Data <- matrix(abs(rnorm(3000, sd=2)),ncol=10,nrow=300)

## Initializing SincellObject named list
mySincellObject <- sc_InitializingSincellObject(Data)

## To access the gene expression matrix
expressionmatrix<-mySincellObject[["expressionmatrix"]]
```

sc_InSilicoCellsReplicatesObj *In silico generation of replicates of individual cells*

Description

Function to generate in silico replicates of individual cells under different models of noise. These in silico replicates will be used by function sc_StatisticalSupportByReplacementWithInSilicoCellReplicates() in order to provide statistical support to the connected graph in SincellObject[["cellstateHierarchy"]] assessed by function sc_GraphBuilderObj() representing a cell-state hierarchy.

Usage

```
sc_InSilicoCellsReplicatesObj(SincellObject, method="variance.deciles",
  dispersion.statistic = NULL, multiplier=100, no_expr=0.5,
  LogTransformedData = T, baseLogTransformation=exp(1),
  pseudocounts.added.before.log.transformation=1,
  cores=ifelse(detectCores()>=4, 4, detectCores()))
```

Arguments

SincellObject	A SincellObject named list as created by function sc_GraphBuilderObj(), containing i) in member "cellstateHierarchy" a connected graph representing a cell-state hierarchy; and ii) in member "expressionmatrix" a numeric matrix that represents a gene expression matrix gathering the expression levels of each single-cell in the experiment (displayed by columns) for each detected gene (displayed by rows).
method	Method to generate in silico replicates of individual cells. Options are: i) method="variance.deciles": the mean and variance of all genes in the original gene expression matrix is assessed. Genes are assigned to classes according to the deciles of mean they belong to. Next, for a given gene g, a variance v is randomly chosen from the set of variances within the class of the gene. Then, a random value drawn from a uniform distribution U(0,v) of mean zero and variance v is added to the expression value of a gene g in a cell c. By perturbing in this way all genes in a reference cell c we obtain an in silico replicate c'. Redoing the process N times, N stochastic replicates are generated for each original cell. ii) method= "cv2.deciles": Same as i) but a squared coefficient of variation cv2 is randomly chosen from the set of coefficient of variation values within the class of the gene (defined by deciles of mean). Then, the parameter v for the uniform distribution is assessed by v=cv2*(mean**2). iii) method="lognormal-3parameters": random perturbations of gene expression levels are drawn from a log normal distribution log(x)~N(m,v) (where m is the mean and v the variance of the gene levels across all samples) with a third parameter alpha describing the proportion of cells where transcript expression was detected above a given threshold level (parameter "no_expr"; see Shalek et al.

Nature 2014). NOTICE: This option assumes that the expression data has been log-transformed. You may want to check whether your Sincell object contains a gene expression matrix transformed that way.

iv) method="negative.binomial": random perturbations of gene expression levels are drawn from a negative binomial (NB) distribution NB(m,r), where m is the mean and r is the size (i.e. the dispersion parameter). Under this parameterization, the variance is v=m+(m^2/r), therefore r=m^2/(v-m). For each gene, its mean m is estimated from the expression levels of the expression matrix. There are three alternative ways of defining the variance v for each gene, which are indicated in parameter dispersion.statistic. Some works has found that, for most genes, the variability observed among their expression levels across individual cells was better described by a negative binomial (NB) distribution rather than a lognormal distribution (Grün et al., 2014). Grün and colleagues used NB distribution to model not only technical noise but also true biological gene expression noise. Their assumption was that endogenous mRNA abundance follows a NB as supported by a physical model of bursting expression (Raj et al., 2006). A negative binomial noise model was also adopted in (Zeisel et al., 2015). As pointed out in these works, NB is frequently used to model overdispersed count data and has been previously used for bulk RNA-seq data (Anders and Huber, 2010; Robinson et al., 2010). We recommend this approach only if normalized count data is used (i.e. not length-normalized RPKM/FPKM). Sincell can follow an NB distribution parameterized on the observed gene expression levels to generate random perturbations and produce in silico cell replicates accordingly. If log-transformed normalized counts are used, Sincell would unlog the perturbed data through a NB and afterwards will redo the log trans-formation. Parameters "LogTransformedData", "baseLogTransformation", "pseudocounts.added.before.log.transformation", should be indicated to help Sincell perform de unlog and log in a consistent way with user's transformations.

`dispersion.statistic`

if parameter method=="negative.binomial", there are three alternative ways of defininining the variance v that will be used to parameterize the negative binomial distribution a) dispersion.statistic==NULL ; variance is estimated from the input expression levels of the expression matrix b) is.numeric(dispersion.statistic) ; vector provided by the user of length equal to the number of genes in the input expression matrix. This vector should contain cv2 estimates reflecting e.g. estimated technical noise. Estimates of technical noise for each gene can be obtained by modeling the dependence of the coefficient of variation (cv2) of spike-in molecules as a function of their average expression. For instance, in Brennecke et al. 2013, for each technical gene i (e.g. the spike-ins), the sample mean (m) and sample variance of its normalized counts are estimated. Then, the observed squared coefficients of variation (cv2) are fitted against the sample mean (m) with a generalized linear model of the gamma family with identity link and parameterization cv2=a1/m+ alpha0. Applying the fitted formula to the sample mean expression levels of a gene provides an estimate of cv2 arising from technical noise. Sincell permits the incorporation of a technical cv2 estimate per gene in the assessment of in silico cell replicates based on normalized counts (i.e. following the previously described negative binomial distribution

whose dispersion is parameterized using the estimated technical cv2).

c) dispersion.statistic!="cv2.fitted.to.data" ; alternatively, in the absence of spike-in molecules, Sincell implements the fit described in Brennecke et al. 2013 using the cv2 and m values of all genes in the input expression matrix to provide a surrogate of technical noise estimates. However, this alternative should not be used if the user has previously followed our recommendation in Section 1 of using such an approach to identify highly variable genes in order to decrease the size of the input matrix (http://pklab.med.harvard.edu/scw2014/subpop_tutorial.html; Section "Identifying highly variable genes").

`multiplier` Number of in silico replicates of individual cells to generate for each cell in the original data

`no_expr` Threshold value in gene expression levels of SincellObject[["expressionmatrix"]] under which a gene will be considered as non-expressed. In the case that log-transformed RPKM are used, a recomended value is 0,5.

`LogTransformedData`
T (TRUE) or F (FALSE). Indicating whether the input expression matrix used to assessed hierarchies was previously logtransformed

`baseLogTransformation`
if LogTransformedData==T, the base used for the logtransformation

`pseudocounts.added.before.log.transformation`
if LogTransformedData==T, the number of pseudocounts added to the normalized count data before performing logtransformation

`cores` Number of threads used to paralyze the computation. Under Unix platforms, by default the function uses all cores up to 4 (to avoid possilbe issues while running on a cluster with the default parameter) detected by the operating system. Under non Unix based platforms, this parameter will be automatically set to 1.

Value

The SincellObject named list provided as input where list member "InSilicoCellsReplicates" is added. SincellObject[["InSilicoCellsReplicates"]] contains the concatenation by columns of the original expression matrix together with the matrix containing the expression values per gene (by rows) of the in silico generated cells replicates (by columns).

Examples

```
## Generate some random data
Data <- matrix(abs(rnorm(3000, sd=2)),ncol=10,nrow=30)

## Initializing SincellObject named list
mySincellObject <- sc_InitializingSincellObject(Data)

## Assessmet of cell-to-cell distance matrix after dimensionality reduction
## with Principal Component Analysis (PCA)
mySincellObject <- sc_DimensionalityReductionObj(mySincellObject, method="PCA",dim=2)

## Cluster
mySincellObject <- sc_clusterObj (mySincellObject, clust.method="max.distance",
```

```
  max.distance=0.5)

## Assessment of cell-state hierarchy
mySincellObject<- sc_GraphBuilderObj(mySincellObject, graph.algorithm="SST",
  graph.using.cells.clustering=TRUE)

## In silico generation of replicates of individual cells
mySincellObject <- sc_InSilicoCellsReplicatesObj(mySincellObject,
  method="variance.deciles", multiplier=100, no_expr=0.5)

# To access the in silico generated cells replicates
InSilicoCellsReplicates<-mySincellObject[["InSilicoCellsReplicates"]]
```

`sc_marker2color` *Palette of colors from the expression values of a marker gene*

Description

Function that transforms the expression values of a marker gene into a vector of colors that can be used as a color code for the intensity of expression. First, the function extracts the vector of values form the expression matrix row in SincellObject[["expressionmatrix"]] whose name equals the indicated marker. Then those values are transformed into a color scale in which the minimum value is assigned the color "minimum" and the maximum value the color "maximum". If relative.to.marker=TRUE, the minimum and maximum values are taken from the expression values of the marker. If relative.to.marker=FALSE, the minimum and maximum values are taken from the expression values of the entire expression matrix.

Usage

```
sc_marker2color(SincellObject, marker, color.minimum="green",
  color.maximum="red", relative.to.marker=TRUE)
```

Arguments

`SincellObject`	A SincellObject named list as created by function sc_InitializingSincellObject with a named member "expressionmatrix" containing a numeric matrix that represents a gene expression matrix gathering the expression levels of each single-cell in the experiment (displayed by columns) for each detected gene (displayed by rows).
`marker`	Name of the gene marker. It should correspond to a row name in the expression matrix in SincellObject[["expressionmatrix"]]
`color.minimum`	Color that will be assigned to the minimum expression value
`color.maximum`	Color that will be assigned to the maximum expression value
`relative.to.marker`	Logic indicating whether the minimum and maximum values are taken from the expression values of the marker (relative.to.marker=TRUE) or from the entire expression matrix (relative.to.marker=FALSE)

Value

The function returns an array of colors in hexadecimal format

Examples

```
## Generate some random data
Data <- matrix(abs(rnorm(3000, sd=2)),ncol=10,nrow=30)

## Initializing SincellObject named list
mySincellObject <- sc_InitializingSincellObject(Data)

## Adding gene names to expression matrix
rownames(mySincellObject[["expressionmatrix"]]) <- 1:30

## Getting the color vector coding for the expression values of a marker gene
mymarkerColorCodeA<-sc_marker2color(mySincellObject, marker="3",
  color.minimum="green", color.maximum="red", relative.to.marker=TRUE)
mymarkerColorCodeB<-sc_marker2color(mySincellObject, marker="5",
  color.minimum="yellow", color.maximum="blue", relative.to.marker=FALSE)
```

---

sc_StatisticalSupportByGeneSubsampling

Statistical support of cell-state hierarchies by gene subsampling

---

Description

Function to provide statistical support to the connected graph in SincellObject[["cellstateHierarchy"]] assessed by function sc_GraphBuilderObj() representing a cell-state hierarchy. sc_StatisticalSupportByGeneSubsampling() performs "num_it" times a random subsampling of a given number "num_genes" of genes on the original gene expression matrix data in SincellObject[["expressionmatrix"]]. Then, for each resampling, a new connected graph of cells is assessed by calling sc_GraphBuilderObj() with same parameters as for the original SincellObject[["cellstateHierarchy"]]. In each subsampling, the similarity between the resulting connected graph and the original one is assessed as the spearman rank correlation between the two graphs of the shortest distance for all pairs of cells. The distribution of spearman rank correlation values of all iterations is stored as a vector in SincellObject[["StatisticalSupportbyGeneSubsampling"]] and a summary is printed in the standard output.

Usage

```
sc_StatisticalSupportByGeneSubsampling(SincellObject, num_it=100,
  num_genes=as.integer(nrow(SincellObject[["expressionmatrix"]])*0.5),
  cores=ifelse(detectCores()>=4, 4, detectCores()))
```

Arguments

SincellObject	A SincellObject named list as created by function sc_GraphBuilderObj(), containing in member "cellstateHierarchy" a connected graph representing a cell-state hierarchy.

num_it	Number of subsamplings to perform on the original gene expression matrix data contained in SincellObject[["expressionmatrix"]]
num_genes	Number of genes to sample in each subsampling. Default is fifty percent of the genes in the original gene expression matrix.
cores	Number of threads used to paralyze the computation. Under Unix platforms, by default the function uses all cores up to 4 (to avoid possilbe issues while running on a cluster with the default parameter) detected by the operating system. Under non Unix based platforms, this parameter will be automatically set to 1.

Value

The SincellObject named list provided as input where following list members are added: SincellObject[["StatisticalSupportbyGeneSubsampling"]] representing the vector of spearman rank correlation values of all "num_it" iterations. Each element of SincellObject[["StatisticalSupportbyGeneSubsampling"]] represents the similarity between the connected graph resulting from one subsampling and the original graph, and it is assessed as the spearman rank correlation between the two graphs of the shortest distance for all pairs of cells.

Examples

```
## Generate some random data
Data <- matrix(abs(rnorm(3000, sd=2)),ncol=10,nrow=30)

## Initializing SincellObject named list
mySincellObject <- sc_InitializingSincellObject(Data)

## Assessmet of cell-to-cell distance matrix after dimensionality reduction
## with Principal Component Analysis (PCA)
mySincellObject <- sc_DimensionalityReductionObj(mySincellObject, method="PCA",dim=2)

## Cluster
mySincellObject <- sc_clusterObj (mySincellObject, clust.method="max.distance",
  max.distance=0.5)

## Assessment of cell-state hierarchy
mySincellObject<- sc_GraphBuilderObj(mySincellObject, graph.algorithm="SST",
  graph.using.cells.clustering=TRUE)

## Assessment statistical support by gene subsampling
mySincellObject<- sc_StatisticalSupportByGeneSubsampling(mySincellObject,
  num_it=1000)

## To access the distribution of Spearman rank correlations:
StatisticalSupportbyGeneSubsampling<-
  mySincellObject[["StatisticalSupportbyGeneSubsampling"]]
summary(StatisticalSupportbyGeneSubsampling)
```

`sc_StatisticalSupportByReplacementWithInSilicoCellsReplicates` *Statistical support of cell-state hierarchies by random cell substitution with in silico-generated cell replicate*

Description

Function to provide statistical support to the connected graph in SincellObject[["cellstateHierarchy"]] assessed by function sc_GraphBuilderObj() representing a cell-state hierarchy. sc_StatisticalSupportByReplacementWithInS performs "num_it" times a random replacement of a given fraction "fraction.cells.to.replace" cells on the original gene expression matrix with a randomly selected set of in-silico replicates. Then, for each set of substitutions "num_it", a new connected graph of cells is calculated using the same parameters as for the hierarchy being tested. In each "num_it", the similarity between the resulting connected graph and the original one is assessed as the Spearman rank correlation between the two graphs of the shortest distance for all pairs of cells. The distribution of spearman rank correlation values of all iterations is stored as a vector in SincellObject[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]] and a summary is printed in the standard output.

Usage

```
sc_StatisticalSupportByReplacementWithInSilicoCellsReplicates(SincellObject,
  method="own", num_it=100, fraction.cells.to.replace=0.15,
  cores=ifelse(detectCores()>=4, 4, detectCores()))
```

Arguments

`SincellObject`	A SincellObject named list, with a member "cellstateHierarchy" containing a connected graph representing a cell-state hierarchy, a member "expressionmatrix" containing a numeric matrix that represents a gene expression matrix gathering the expression levels of each single-cell in the experiment (displayed by columns) for each detected gene (displayed by rows) and a member "InSilicoCellsReplicates" containing the in silico cells replicates as generated by function sc_InSilicoCellsReplicatesObj()
`method`	The parameter "method" controls for the maximum order of neighborhood k from which in silico cell replicates will be randomly chosen for substitution. When k=0 (or k="own", default value), a cell will be replaced by a replicate from itself (this is the behavior by default). If k=2, a cell will be replaced by a replicate from itself or from any neighbor of order 2 in the graph. If k="all", a cell will be replaced by a replicate from itself or from any other cell in the graph.
`num_it`	number of iterations in which a random replacement of a given fraction "fraction.cells.to.replace" cells on the original gene expression matrix with a randomly selected set of in-silico replicates is performed
`fraction.cells.to.replace`	fraction of cells on the original gene expression matrix to randomly replace with a randomly selected in-silico replicate

cores — Number of threads used to paralyze the computation. Under Unix platforms, by default the function uses all cores up to 4 (to avoid possilbe issues while running on a cluster with the default parameter) detected by the operating system. Under non Unix based platforms, this parameter will be automatically set to 1.

Value

The SincellObject named list provided as input where following list members are added: SincellObject[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]]. Each element of SincellObject[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]] represents the similarity between the original graph and the graph resulting in each "num_it" iteration from a randon substitution of "fraction.cells.to.replace" with in silico replicates. That similarity is assessed as the spearman rank correlation between the two graphs of the shortest distance for all pairs of cells.

Examples

```
## Generate some random data
Data <- matrix(abs(rnorm(3000, sd=2)),ncol=10,nrow=30)

## Initializing SincellObject named list
mySincellObject <- sc_InitializingSincellObject(Data)

## Assessmet of cell-to-cell distance matrix after dimensionality reduction with
## Principal Component Analysis (PCA)
mySincellObject <- sc_DimensionalityReductionObj(mySincellObject, method="PCA",dim=2)

## Cluster
mySincellObject <- sc_clusterObj (mySincellObject, clust.method="max.distance",
  max.distance=0.5)

## Assessment of cell-state hierarchy
mySincellObject<- sc_GraphBuilderObj(mySincellObject, graph.algorithm="SST",
  graph.using.cells.clustering=TRUE)

## In silico generation of replicates of individual cells
mySincellObject <- sc_InSilicoCellsReplicatesObj(mySincellObject,
  method="variance.deciles", multiplier=100, no_expr=0.5)

## Assessment of statistical support by replacement with in silico cells replicates
mySincellObject<-sc_StatisticalSupportByReplacementWithInSilicoCellsReplicates(
  mySincellObject, method="own", num_it=100, fraction.cells.to.replace=0.15)

## To access the distribution of Spearman rank correlations:
StatisticalSupportByReplacementWithInSilicoCellReplicates<-
  mySincellObject[["StatisticalSupportByReplacementWithInSilicoCellReplicates"]]
summary(StatisticalSupportByReplacementWithInSilicoCellReplicates)
```

`sstalgorithm` *Auxiliary function for SST algorithm within function sc_GraphBuilderObj()*

Description

Auxiliary function implemented in C++ making part of the SST algorithm in function sc_GraphBuilderObj()

Usage

```
sstalgorithm(membership, num_cells, distance)
```

Arguments

`membership`	a numeric array
`num_cells`	total number of cells in the sample
`distance`	a distance matrix

Value

A numeric array of length 3 is returned. The first element of the array is the minimum distance, and the second and third ones are the coordinates.

See Also

sc_GraphBuilderObj()

Index