Package 'chroGPS'

April 9, 2015

Type Package

Title chroGPS: visualizing the epigenome

Version 1.10.0

Date 2014-09-26

Author Oscar Reina, David Rossell

Maintainer Oscar Reina <oscar.reina@irbbarcelona.org>

Description We provide intuitive maps to visualize the association between genetic elements, with emphasis on epigenetics. The approach is based on Multi-Dimensional Scaling. We provide several sensible distance metrics, and adjustment procedures to remove systematic biases typically observed when merging data obtained under different technologies or genetic backgrounds.

License GPL (>=2.14)

Depends R (>= 2.13.0), IRanges, methods, Biobase, MASS, graphics, stats, changepoint

Imports graphics, cluster, DPpackage, ICSNP

Enhances parallel, XML, rgl

Collate adjustPeaks.R distGPS.R domainDist.R mds-class.R mds.R procrustesAdj.R clusGPS.R geneSetGPS.R getmodEncode.R gff2RDList.R gps2xgmml.R

LazyLoad yes

biocViews Visualization, Clustering

R topics documented:

`addVar` *Plot vector of a quantitative variable over a MDS map.*

Description

Given a quantitative variable as a numeric vector with one element for each point on a MDS map, calculate and plot the weight vector corresponding to that variable.

Usage

```
addVar(mds1, z, plot = TRUE, label = "z", pos = 3, ...)
```

Arguments

`mds1`	An object of class `mds` with the MDS object.
`z`	Numeric vector with the quantitative variable, one element for each point.
`plot`	Set to TRUE to calculate and draw the resulting vector on the MDS.
`label`	Something to be printed on the tip of the vector arrow, usually the name of the given variable.
`pos`	Graphical position where the label is drawn respect to the vector arrow tip.
`...`	Additional parameters given to the generic function `plot`.

Value

A named list with the vector components.

Examples

```
# Not run
# See chroGPS-manual.pdf for examples.
```

`adjustPeaks` *Adjust peak width so that samples obtained under different conditions become comparable.*

Description

Peaks obtained under different conditions (e.g. chip-chip, chip-seq, mnase-seq) are typically not comparable in terms of their width. `adjustPeaks` modifies the mean and SD of the peak width distribution for each condition, so that they become equivalent to the condition with widest peaks. See details.

Usage

```
adjustPeaks(x, adjust, sampleid, logscale = TRUE)
```

Arguments

`x`	`RangedDataList` indicating the binding sites for each sample/experiment.
`adjust`	Vector indicating the adjustment factor, i.e. the condition under which each sample has been obtained.
`sampleid`	Vector containing the sample identifier. `sampleid` should take the same value for samples obtained under different conditions, as this is used to detect the samples to be used for Procrustes adjustment.
`logscale`	If set to `TRUE` the mean and SD are matched for log width, otherwise the original widths are used. Working in log scale can help reduce the effect of outliers (e.g. an usually long binding site).

Details

In a sense, the peak calling resolution is decreased so that they become comparable to the less precise technology (notice that there is no reliable way to increase the precision given by a low-resolution technology).

Value

`RangedDataList` object with adjusted widths.

Methods

signature(x='RangedDataList') Each element in `x` contains the binding sites for a different sample. The start, end and chrosomome of each binding sites should be accessed via `start`, `end` and `space`.

See Also

`procrustesAdj` for an alternative, more general, adjustment based on Procrustes. `distGPS` for computing distances, `mds` to create MDS-oriented objects.

Examples

```
#See examples in help(procrustesAdj)
```

`boostMDS` *Improve goodness-of-fit of a given MDS solution in terms of R-square.*

Description

Given a distance matrix and a valid MDS representation for it, improve the R-square correlation between observed and approximated distances until converged is reached for a given threshold.

Usage

```
boostMDS(D, Y, rate = 0.01, maxit = 50, tol = 0.001, samplesize,
verbose = TRUE, scale = FALSE, seed = 149, plt = FALSE, mc.cores = 1)
```

Arguments

`D`	Distance matrix.
`Y`	Matrix with points from a valid MDS solution for the distances in D.
`rate`	Grid step rate, start with 0.1 which usually is a good compromise, try also 0.01, 1, 10.
`maxit`	Maximum number of iterations.
`tol`	Tolerace for R-square convergence.
`samplesize`	When there are over 100 points to represent, the gradiend descent step size is determined using a fraction `samplesize` of the original data points. By default 0.01 with a minimum of 100 points, which typically gives very stable results. Setting large `samplesize` can significantly increase the computational cost.
`verbose`	Give details of the gains in R-square and step size.
`scale`	Whether to scale the MDS coordinates in the output MDS.
`seed`	A random seed to be used in the resampling process if samplesize < 1.
`plt`	Whether to plot the intermediate solutions or not.
`mc.cores`	Number of cores to use in parallelized grid step size search.

Value

The function returns a matrix with the coordinates of a valid MDS solution for distance matrix D where the R-square correlation has been improved. However, have in mind that an MDS solution with better R-square does not necessarily mean the solution is easier to interpret. As with any MDS approach, a balance must be found between pure 'technical' goodness-of-fit and usefulness of the delivered solution in terms of answering the original hypothesis.

References

boostMDS is based on hitMDS (High-Throughput Multidimensional Scaling, see see http://dig.ipk-gatersleben.de/hitmds/hitmds.html for details)

Examples

```
# Not run, see also chroGPS-manual.pdf file for examples
#data(geneSample)
#d = distGPS(geneSample,uniqueRows=TRUE)
#m = mds(d,type=isoMDS)
#m
#plot(m)
#m = boostMDS(d@d,m@points)
#plot(m)
```

---

clusGPS — *Computation of cluster density estimates for cluster contour representation and correct-classification rates (cluster robustness). A pre-computed clustering of elements used in the map has to be given as an input, which is useful to explore results using different clustering algorithms and methodologies (top-down, bottom-up, etc).*

Description

After performing a pre-merging step so that all clusters have a minimum size, semiparametric bayesian density is estimated using a Dirichlet process mixture of normals. This is used both to compute bayesian mis-classification posterior probabilities (correct classification rates) and to estimate probability contours which can be visualized on the MDS map.

The functions `contour2dDP` and `plotContour` functions can be used to compute bayesian density estimates for a given set of elements (points) from a pre-generated 2D MDS object. These functions are used internally by clusGPS to draw cluster contours but are also useful to visualize other type of contours over the map (ie genes from a given Gene Ontology term, having a specific epigenetic mark of interest, etc).

The S4 accessors `clusNames`,`tabClusters` and `clusterID` retrieve information stored within a `clusGPS` object.

Usage

```
clusGPS(d, m, h, sel=NULL, id=NULL, grid, ngrid=1000, densgrid=FALSE, preMerge=TRUE, type = "hclust", m
"average", samplesize = 1, p.adjust = TRUE, k, mc.cores = 1,
set.seed = 149, verbose=TRUE, minpoints=70,...)
contour2dDP(x, ngrid, grid = NULL, probContour = 0.5, xlim, ylim,
    labels = "", labcex = 0.01, col = colors()[393], lwd = 4,
    lty = 1, contour.type = "single", contour.fill = FALSE,
minpoints=100, ...)
clusNames(clus)
tabClusters(clus,name)
clusterID(clus,name)
```

Arguments

d	Object of class distGPS with the pairwise observed dissimilarities between elements.
m	(Optional). Object of class mds with a MDS object generated from the distances in d. Only MDS type "boostMDS" is available. The mds function performs an optimization of the approximated distances in m in order to improve R-square correlation between them and the observed dissimilarities en d, maximizing goodness of fit.
h	(Optional). Object of class hclust with a pre-calculated clustering for the elements in d.
sel	(Optional). Logical vector indicating which elements from d will be used for performing hierarchical clustering with average linkage. This is useful if we want to focus on a given set of points only (i.e. those from a big cluster which we want to study in more detail).
id	(Optional). Label of the cluster which we want to further subdivide, ignoring points from all other clusters. Deprecated, use parameter sel specified above.
grid	Matrix of dimension ngrid*nvar giving the diagonal points of the grid where the density estimate is evaluated. The default value is NULL: grid dimensions are chosen according to the range of the data, and granularity is automatically determined according to data density, in order to provide a more accurate estimation in high density areas, where more resolution is needed.
ngrid	Number of grid points where the density estimate is evaluated. This argument is ignored if a grid is specified. The default value is 1000. Higher values are recommended if data presents very high density areas.
densgrid	Set to true to generate grid points from the quantile distribution of the data using the grid size defined by ngrid. This is useful if the data presents areas of very different density, ranging from very sparse to extremely dense areas, optimizing grid granularity where is necessary, therefore improving resolution of density estimation and reducing computation time.
preMerge	If TRUE will perform a first pre-merging step so that any cluster smaller than minpoints gets merged with its closest cluster based on their centroid distances. This is performed until no clusters < minpoints exist.
type	Type of clustering to be performed. Currently only "hclust" (Agglomerative Nesting) is supported, but any other clustering type can be used by providing a pre-calculated object h. This variable is to become deprecated, since clusGPS will only work with a precomputed clustering.
method	Clustering method. See hclust for details. This variable is to become deprecated, since clusGPS will only work with a precomputed clustering.
samplesize	Proportion of elements to sample for computing clustering and density estimation. This is useful to generate density contours from a subset of the data, speeding up computation.
p.adjust	Set to TRUE to adjust the bayesian posterior probabilities of mis-classification.
k	Integer vector indicating the number of clusters on which density estimation will be computed for mis-classification or contour calculation.

mc.cores	Number of cores to be used for parallel computation with the parallel package.
set.seed	If samplesize<1, random seed to be used to perform random sampling of the data.
verbose	Set to TRUE to output clustering process information.
minpoints	If preMerge is FALSE, then the algorithm will ignore clusters with fewer than minpoints elements. This is useful if the clustering method used tends to generate many very small clusters of limited use and difficult interpretation and for which density estimates may not be correctly computed. The default method is to preMerge clusters since this ensures density estimation is available for all clusters and helps interpreting the map, since no elements are ignored.
x	Numeric matrix indicating coordinates of the points for which a probability contour is calculated in contour2dDP.
probContour	Numeric matrix indicating coordinates of the points for which a probability contour is calculated in contour2dDP.
contour.type	For contour2dDP, type of contour, either 'single' (surrounding the points within the given probContour probability) or 'multiple' to generate terrain-like density contour lines.
contour.fill	Deprecated.
xlim,ylim,labels,labcex,col,lwd,lty	Graphical parameters given to contour2dDP.
clus	A valid clusGPS object from which we want to extract information.
name	Character indicating a valid name within a clusGPS object, from which we want to extract information.
...	Additional parameters.

Value

The function clusGPS returns an object of class clusGPS. See help for clusGPS-methods for details. contour2dDP returns a DPdensity object with density contour information which can be plotted as 2D contours with our plotContour function, as well as with the plot function from the DPpackage package.

Methods

signature(d='distGPS',m='mds') Hierarchical clustering is performed for the elements whose pairwise distances are given in d. For each cluster partition given in k, cluster identity for each element is returned, and semiparametric bayesian density estimation is computed using the point density information from m.

plot signature(m = "clusGPS"): S4 plot method for clusGPS objects.

clusNames signature(m = "clusGPS"): Retrieves names of the clustering configurations stored in clusGPS objects, one for each distance threshold indicated in k, that get automatically named accordingly.

tabClusters signature(m = "clusGPS"): Returns a table with the number of elements in each of the clusters found for an existing clustering configuration with name name within the clusGPS object.

clusterID `signature(m = "clusGPS")`: Returns a vector of cluster assignments for all the elements in an existing clustering configuration `name` within the `clusGPS` object.

Author(s)

Oscar Reina

Examples

```
# Not run
# data(s2)
# # Computing distances
# d <- distGPS(s2.tab,metric=tanimoto,uniqueRows=TRUE)
# # Creating MDS object
# mds1 <- mds(d,type=isoMDS)
# mds1
# plot(mds1)
# Precomputing clustering
# h <- hclust(as.dist(d@d),method=average)
# # Calculating densities (contours and probabilities), takes a while
# clus <- clusGPS(d,mds1,preMerge=TRUE,k=max(cutree(h,h=0.5)))
# # clus contains information for contours and probabilities
# plot(clus,type=contours,k=125,lwd=3,probContour=.75)
# plot(clus,type=stats,k=125,ylim=c(0,1))
# plot(clus,type=avgstat)
# plot(clus,type=density,k=3,ask=TRUE,xlim=range(mds1@points),ylim=range(mds1@points))
```

`clusGPS-class` *Class* `"clusGPS"`

Description

Agglomerative Nesting for a distGPS object. Contains probability contours and bayesian posterior probability of mis-classification for the clusters evaluated.

Details

Parameters for the S4 plot method for `mds` objects.

Object of class `"mds"` with a 2D or 3D Multidimensional Scaling to be plotted.

`drawlabels`**:** TRUE to use rownames of the MDS points as text labels.

`labels`**:** Alternative character vector giving the text labels for the MDS points.

`plantar`**:** If a 3D MDS is used, set plantar to TRUE to plot projected views of the MDS using XY, YZ and XZ axis decomposition.

`point.cex`**:** Size of the points / spheres for the MDS plot.

`text.cex`**:** Size of text labels for the MDS points.

`text.pos`**:** Alignment position of the text labels respective to its points (1,2,3,4).

`point.col`: Color for the MDS points / spheres.

`text.col`: Color for the MDS text labels.

`point.pch`: PCH type for the MDS points.

`type.3d`: Use 'p' for points, 's' for spheres.

`radius`: Radius for the spheres on a 3D MDS plot. Automatically generated from point.cex and the number of points in the MDS.

`app`: Appearance of the 3D spheres on a 3D MDS plot, can be 'fill', 'lines', 'grid'.

`alpha`: Number between 0 and 1 with the level of transparency to be used on spheres on a 3D MDS.

`scalecol`: Set to TRUE to use a color scale for points, for instance to color points (genes) according to their expression level on a chroGPS-genes MDS plot.

`scale`: Scale to use to generate scale colors (for instance normalized gene expression for each element (gene) on chroGPS-genes MDS).

`palette`: Color palette to be used for scale colors.

Objects from the Class

Objects can be created by calls of the form `new("clusGPS", ...)`.

Slots

`h`: Object of class `"hclust"` with Agglomerative Nesting or user-provided cluster object.

`clus`: Object of class `"list"` with probability contour and bayesian posterior probability of mis-classification information for the clusters evaluated.

`adjusted`: Object of class `"logical"` indicating if bayesian posterior probabilities of mis-classification are adjusted for multiple testing.

Author(s)

Oscar Reina

Examples

```
showClass("clusGPS")
```

`distGPS` *Compute matrix with pairwise distances between objects. Several GPS metrics are available.*

Description

The function computes pairwise distances between invididuals (e.g. samples or genes) according to a user-specified metric. Several metrics are available. The precise definition of each metric depends on the class of the first argument (see details section).

Usage

```
distGPS(x, metric=tanimoto, weights, uniqueRows=FALSE, genomelength=NULL, mc.cores=1)
```

Arguments

x	Object for which we want to compute distances
metric	Desired distance metric. Valid options for chroGPS-factors map are 'tanimoto', 'avgdist', 'chisquare' and 'chi' (see details). For chroGPS-genes maps, metrics 'wtanimoto', 'euclidean' and 'manhattan' are also available.
weights	For signature(x='matrix'), an unnamed numeric vector with weights applied to every sample (column) in the original data. The typical example is when we have a sample (epigenetic factor) with several replicates available (biological or technical replicate, different antibody, etc.), and we want to treat them together (for instance giving a 1/nreplicates weight to each one). If not supplied, each replicate is considered as an individual sample (using 1 as weight for every sample).
uniqueRows	If set to TRUE and x is a matrix or data.frame, duplicated rows are removed prior to distance calculation. This can save substantial computing time and memory. Notice however that the dimension of the distance matrix is equal to the number of unique rows in x, instead of nrow.. (x).
genomelength	For 'chi' and 'chisquare' metrics, numeric value indicating the length of the genome. If not given the function uses the minimum length necessary to fit the total length of the result.
mc.cores	If mc.cores>1 and parallel package is loaded, computations are performed in parallel with mc.cores processors when possible.

Details

For RangedDataList objects, distances are defined as follows.

Let a1 and a2 be two RangedData objects. Define as n1 the number of a1 intervals overlapping with some interval in a2. Define n2 analogously. The Tanimoto distance between a1 and a2 is defined as (n1+n2)/(nrow(z1)+nrow(z2)). The average distance between a1 and a2 is defined as .5*(n1/nrow(z1) + n2/nrow(z2)). The wtanimoto distance in chroGPS-genes weights each epigenetic factor (table columns) according to its frequency (table rows). The chi-square distance is defined as the usual chi-square distance on a binary matrix B which is automatically computed by distGPS. The binary matrix B is the matrix with length(x) rows and number of columns equal to the genome length, where B[i,j]==1 indicates that element i has a binding site at base pair j. The chi distance is simply defined as the square root of the chi-square distance. Finally, euclidean and manhattan metrics have the same definition than in the base R function dist.

When choosing a metric one should consider the effect of outliers, i.e. samples with large distance to all other samples. Tanimoto and Average Distance take values between 0 and 1, and therefore outlying distances have a limited effect. Chi-square and Chi distances are not limited between 0 and 1, i.e. some distances may be much larger than others. The Chi metric is slightly more robust to outliers than the Chi-square metric.

For matrix or data.frame objects, x must be a matrix with 0's and 1's (or FALSE and TRUE). The usual definitions are used for Tanimoto (which is equivalent to Jaccard's index), Chi-square and Chi.

Average overlap between rows `i` and `j` is simply the average between the proportion of elements in `i` also in `j` and the proportion of elements in `j` also in `i`.

Value

Object of class `distGPS`, with matrix of pairwise dissimilarities (distances) between objects.

Methods

distGPS:

Each element in `x` is assumed to indicate the binding sites for a different sample, e.g. epigenetic factor. Typically `space(x)` indicates the chromosome, `start(x)` the start position and `end(x)` the end position (in bp). Strand information is ignored.

signature(x='RangedDataList'), **signature(x='matrix')** Rows in `x` contain individuals for which we want to compute distances. Columns in `x` contain the variables, and should only contain either 0's and 1's or `FALSE` and `TRUE`.

splitDistGPS:

This is a set of internal classes and functions to be used in the parallel computation of Multidimensional Scaling.

uniqueCount:

This function collapses a chroGPS-genes matrix or data frame so that elements with the same combination of variables are aggregated into a single entry. Elements become then identified by their unique pattern and a frequency count is also returned.

See Also

`mds` to create MDS-oriented objects, `procrustesAdj` for Procrustes adjustment.

Examples

```
x <- rbind(c(rep(0,15),rep(1,5)),c(rep(0,15),rep(1,5)),c(rep(0,19),1),c(rep(1,5),rep(0,15)))
rownames(x) <- letters[1:4]
d <- distGPS(x,metric=tanimoto)
du <- distGPS(x,metric=tanimoto,uniqueRows=TRUE)
mds1 <- mds(d)
mds1
plot(mds1)
d <- distGPS(x,metric=chisquare)
mds1 <- mds(d)
mds1
plot(mds1)
```

distGPS-class *Class* "distGPS"

Description

Pairwise distances between elements. Function distGPS creates objects of this class. splitDistGPS in an private class used internally for parallel Multidimensional Scaling.

Objects from the Class

Objects can be created by calls of the form new("distGPS", ...).

Slots

d: Object of class "matrix" with pairwise dissimilarities (distances) between elements.

metric: Object of class "character" indicating the metric type used for calculating distances. See function distGPS.

type: Object of class "character", deprecated.

Author(s)

Oscar Reina

Examples

```
showClass("distGPS")
```

domainDist *Overview of intra and inter-domain distances.*

Description

Given a distance of pairwise distances or dissimilarities between elements, return intra and inter-group sets of distances based on a given group definition. This is useful to get an insight on domain robustness for functional related genes or factors.

Usage

```
domainDist(d, gps=factors, domain, type=intra, col=white, avg=FALSE,
plot=TRUE, ...)
```

Arguments

d	Distance/Dissimilarities matrix, usually the slot d on a distGPS object, but any distance matrix can be given as input.
gps	'factors' for a chroGPS-factors distance matrix, 'genes' for a chroGPS-genes one.
domain	Character vector with group identity for each element d. It can be a functional domain classification (i.e. 'Activation', 'Repression', etc), given for each factor on a chroGPS-factors map or for each gene in a chroGPS-genes map. However, any classification of interest can be used (pathways, gene ontology, etc.)
type	Intradomain ('intra') or Interdomain ('inter') distance overview.
col	Character vector with colors to be passed to plot.
avg	TRUE to return also the average inter or intra domain distance.
plot	TRUE to generate inter/intra domain boxplots.
...	Additional parameters given to the generic function plot.

Value

List of inter or intra domain distances.

Examples

```
# Not run
# data(s2)
# d <- distGPS(s2,metric=avgdist,mc.cores=1)
# d.intra <- domainDist(as.matrix(d),domain=s2names$Color,type=intra,plot=TRUE)
# d.inter <- domainDist(as.matrix(d),domain=s2names$Color,type=inter,plot=TRUE)
```

---

geneSetGPS *Highlight point (gene) position over a Multi-dimensional Scaling plot.*

Description

Given a list of genes of interest, the function highlights their position over a Multi-dimensional Scaling plot.

Usage

```
geneSetGPS(x, m, genes, uniqueCount = TRUE, ...)
```

Arguments

x	Matrix or data frame of observations x variables (tipically genes x epigenetic factors), with gene identifiers as rownames.
m	Object of class mds with a valid Multidimensional Scaling representation for the elements in x.
genes	Character vector containing gene identifiers, matching those on rownames(x).
uniqueCount	Set to FALSE if the MDS has been generated directly from the data in x, otherwise set to TRUE to match gene identifiers with their unique pattern of observed variables.
...	Additional parameters given to the generic function plot.

Value

Matrix with coordinates on the given input MDS object for the genes selected.

Author(s)

Oscar Reina

Examples

```
# Not run
# data(s2)
# d <- distGPS(s2.tab,metric=tanimoto,uniqueRows=TRUE)
# mds1 <- mds(d)
# set.seed(149)
# sampleGenes <- rownames(s2.tab)[sample(1:nrow(s2.tab),10,rep=FALSE)]
# pts <- geneSetGPS(s2.tab,mds1,genes=sampleGenes,uniqueCount=TRUE)
# plot(mds1)
# points(getPoints(pts),col=red,cex=3)
```

---

getURL *Retrieve file from URL.*

Description

A function that can be used to retrieve any file of interest from the internet, in our case, modEncode binding site information GFF files into the working directory. See also help for function gff2RDList.

Usage

```
getURL(urls, filenames, extension=.gff3, method=internal)
```

Arguments

urls	Character vector with one or more target URLs to download.
filenames	Character vector with the filename for each URL target.
extension	If desired, an extension to append to filenames.
method	Either 'internal' to use the system's default or 'wget' if it is installed.

Value

Message indicating the path to downloaded file(s).

Examples

```
# Not run
#getURL(http://www.google.com/index.html,index,.html)
```

---

gff2RDList *Retrieve binding site information from GFF3 files.*

Description

An auxiliary function to retrieve binding site information from GFF3 format files (for instance those downloaded from modEncode, see function getURL).

Usage

```
gff2RDList(filenames,listnames,dir,quote=NULL,chrprefix=)
```

Arguments

filenames	GFF3 filenames to read.
listnames	Names for each read filename, will be used as names of the returned RangedDataList. If not given, filenames will be used as listnames.
dir	Directory where the GFF3 files are located.
quote	Quote character used in the GFF3 files around fields, if any.
chrprefix	Prefix to be placed before the chromosome names if desired, for instance 'chr'.

Value

A list with Enriched and Depleted binding sites, each one is an object of class RangedDataList with the RangedData objects containing the respective enriched or depleted binding sites from each GFF3 file.

Examples

```
# Not run
#getURL(http://intermine.modencode.org/release-30/features.do?type=submission&action=export&format=gff3&submis
#test <- gff2RDList(test.gff3,dir=getwd())
#test
#test$Enriched[[1]]
#test$Depleted[[1]]
```

---

gps2xgmml *Export an 'mds' object to Cytoscape .xgmml format*

Description

gps2xgmml creates a .xgmml file for visualizing MDS results in Cytoscape. Two-dimensional MDS maps can be visualized in Cytoscape as usual. For three-dimensional maps Cytoscape's 3D Renderer (http://wiki.cytoscape.org/Cytoscape_3/3D_Renderer) is required.

Usage

```
gps2xgmml(x, fname=out.xgmml, names.arg, fontSize=4, col=col2hex(steelblue), cex)
```

Arguments

x	Object of class mds
fname	Name of output file
names.arg	Names for each point. If missing, they're taken from x.
fontSize	Font size
col	Fill colour(s) for the plotting symbols. Should be given in hexadecimal, e.g. as returned by function col2hex from gplots. Tips: col2hex('steelblue') looks nice in 2D/3D plots, col2hex('steelblue') looks nice in 2D plots and a bit faded on 3D plots.
cex	Expansion factor for plotting symbols. By default, cex=12 for 2D plots and cex=100 for 3D plots.

Details

The .xgmml file contains the map co-ordinates in 2 or 3 dimensions, depending on the number of dimensions stored in the input mds object. To visualize properly a file with 3D co-ordinates, you need to install Cytoscape's 3D Renderer (http://wiki.cytoscape.org/Cytoscape_3/3D_Renderer) and start Cytoscape following the instructions provided therein.

An .xgmml file with 3D co-ordinates can still be visualized in regular Cytoscape but the z-axis will be ignored.

Value

Generates an .xgmml file that can be opened in Cytoscape (File -> Import -> Network).

Examples

```
#See help(mds) for an example
```

`mds` *Metric and non-metric Multidimensional Scaling for a distGPS object.*

Description

Generation of Multidimensional Scaling objects for the dissimilarities between elements given as an input in a `distGPS` object. Metric and non-metric algorithms are available, as well as an optimization algorithm for improving r-square correlation between observed and approximated distances. The MDS calculation for a given distance matrix can be splitted into smaller individual tasks and run in parallel, greatly improving CPU time and system memory usage. The S4 accessor functions `getR2, getStress, getPoints` retrieve R-square correlation, stress and points stored within a `mds` object respectively. The function `is.adj` is useful to know if a certain chroGPS MDS map has been adjusted by Procrustes or not (see help for `procrustesAdj` for details.)

Usage

```
mds(d, m = NULL, k = 2, type = "classic", add = FALSE, cor.method = "pearson", splitMDS = FALSE, split = 0
getR2(m)
getStress(m)
getPoints(m)
```

Arguments

`d`	Object of class `distGPS` with the pairwise observed dissimilarities between elements, a distance matrix.
`m`	(Optional). Object of class `mds` with a MDS object generated from the distances in `d`. Only MDS type "boostMDS" is available. The mds function performs an optimization of the approximated distances in `m` in order to improve r-square correlation between them and the observed dissimilarities en `d`, maximizing goodness of fit.
`k`	Dimensionality of the reconstructed space, typically set to 2 or 3.
`type`	Set to `"classic"` to perform classical MDS (uses function `cmdscale` from package `stats`). Set to `"isoMDS"` to use Kruskal's non-metric MDS (uses function `isoMDS` from package `MASS`) Set to "boostMDS" to perform r-square optimization of a pre-computed input MDS for that distance matrix.
`add`	Logical indicating if an additive constant c* should be computed, and added to the non-diagonal dissimilarities such that all n-1 eigenvalues are non-negative in `cmdscale`.
`cor.method`	A character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman", can be abbreviated.

splitMDS	Set to TRUE to perform computation of the MDS in parallel (see parameters below).
split	Proportion of elements to include in each (but last) distance submatrix.
overlap	Proportion of elements to be used as common anchor points between two adjacent distance submatrixes. These points will be used as spatial references to stitch each two adjacent MDS objects by Procrustes.
stepSize	Size for the quadratic search step to be used for R-square optimization if boostMDS is called, see specific help function for details.
reshuffle	Set to TRUE to perform random resampling of the input distance matrix before splitting it for parallel computation. This is often necessary to sufficiently capture the inherent variability of the data in each distance submatrix so that the stitching process can work properly, as the original data may present an arbitrary sorting of some kind. If a previous resampling of the data has been performed, this is not necessary.
set.seed	Random seed to perform the resampling.
mc.cores	Number of cores to be passed to the mclapply function from the parallel package, used to perform the parallel MDS computations.
...	Additional parameters passed to cmdscale, isoMDS or boostMDS, see each individual help file for details.

Value

The function returns a mds object. See help ("mds-Class") for details.

Methods

mds signature(d = "distGPS", m = "missing"): Creates a mds object with points in a k-dimensional space approximating the pairwise distances in d.

mds signature(d = "distGPS", m = "mds"): For the observed dissimilarities in d and a valid spatial representation of them in m, the function returns a mds object with an optimized representation of d in terms of R-square. The MDS stress measure is also returned. See help for boostMDS for details.

plot signature(m = "mds"): S4 plot method for mds objects.

Author(s)

Oscar Reina

See Also

See functions cmdscale, isoMDS from package MASS.

Examples

```
x <- rbind(c(rep(0,15),rep(1,5)),c(rep(0,15),rep(1,5)),c(rep(0,19),1),c(rep(1,5),rep(0,15)))
rownames(x) <- letters[1:4]
d <- distGPS(x,metric=tanimoto,uniqueRows=TRUE)
mds1 <- mds(d)
mds1
plot(mds1)
#gps2xgmml(mds1, fname=chroGPS_factors.xgmml, fontSize=4,col=col2hex(red), cex=8)
```

`mds-class` *Class* `"mds"`

Description

Multidimensional Scaling. Function `mds` creates object of this class.

Details

Parameters for the S4 plot method for `mds` objects.

Object of class `"mds"` with a 2D or 3D Multidimensional Scaling to be plotted.

`drawlabels`: TRUE to use rownames of the MDS points as text labels.

`labels`: Alternative character vector giving the text labels for the MDS points.

`plantar`: If a 3D MDS is used, set plantar to TRUE to plot projected views of the MDS using XY, YZ and XZ axis decomposition.

`point.cex`: Size of the points / spheres for the MDS plot.

`text.cex`: Size of text labels for the MDS points.

`text.pos`: Alignment position of the text labels respective to its points (1,2,3,4).

`point.col`: Color for the MDS points / spheres.

`text.col`: Color for the MDS text labels.

`point.pch`: PCH type for the MDS points.

`type.3d`: Use 'p' for points, 's' for spheres.

`radius`: Radius for the spheres on a 3D MDS plot. Automatically generated from point.cex and the number of points in the MDS.

`app`: Appearance of the 3D spheres on a 3D MDS plot, can be 'fill', 'lines', 'grid'.

`alpha`: Number between 0 and 1 with the level of transparency to be used on spheres on a 3D MDS.

`scalecol`: Set to TRUE to use a color scale for points, for instance to color points (genes) according to their expression level on a chroGPS-genes MDS plot.

`scale`: Scale to use to generate scale colors (for instance normalized gene expression for each element (gene) on chroGPS-genes MDS).

`palette`: Color palette to be used for scale colors.

`xlim`: Graphical limit for the X axis.

`ylim`: Graphical limit for the Y axis.

`zlim`: Graphical limit for the Z axis for 3D plots.

Objects from the Class

Objects can be created by calls of the form `new("mds", ...)`.

Slots

`points`: Object of class `"matrix"` with coordinates in the approximated space.

`Type`: Object of class `"character"` with the type of MDS (classicMDS, isoMDS).

`Adj`: Object of class `"logical"` indicating if the MDS object has been adjusted by Procrustes or not.

`R.square`: Object of class `"numeric"` with the percentage of variability from the original dissimilarities captured in the approximated distances.

`stress`: Object of class `"numeric"` with the stress for the returned MDS configuration.

Author(s)

Oscar Reina

See Also

`cmdscale` from package base. `isoMDS` from package MASS.

Examples

```
showClass("mds")
```

---

mergeClusters *Unsupervised cluster merging based on their observed overlap with automatic changepoint detection.*

Description

The function uses contour density estimation as computed by the `clusGPS` function to merge significantly overlapping clusters in an unsupervised manner. In each step, clusters with highest overlap are merged, their individual density estimates are updated in a computational feasible manner, and the process continues until the maximum overlap between any given pair of clusters drops swiftly, as detected by the `cpt.mean` function in the `changepoint` package.

Usage

```
mergeClusters(clus, clus.method = "unweighted", cpt.method = "mean", logscale = TRUE, brake = rep(1, len
```

Arguments

clus	A clusGPS object from which we want to merge clusters with a significant overlap when visualized on a chroGPS MDS map. This is quite useful when a clustering method (i.e. hierarchical clustering with average linkage) tends to return a high number of overlapping clusters.
clus.method	Currently only 'unweighted' method is supported, that is, cluster overlap is computed based on spatial location of contours, but the computed overlaps are not weighted for cluster size.
cpt.method	Use 'mean' for using cpt.mean function in changepoint package for computing overlap changepoint. Use 'var' for cpt.var. See specific function help for details.
logscale	Defaults to TRUE. Whether to use decimal or log scale values for computing overlap changepoint.
brake	(Optional). By default, the function returns the clusters from the optimal merging step as detected by the changepoint functions (brake=1). By using smaller values (0, -1, -2, ...) or bigger ones (2, 3, 4, ...) the algorithm can be forced to return the result from any previous or later merging step respectively.
plt	Set to TRUE to visualize maximum cluster overlap for each merging step and changepoint detection (optimal merging step).
mc.cores	Numbers of cores to use in parallel computation.

Value

A clusGPS object where significantly overlapping clusters are merged, highly improving visualization, cluster robustness and further study of the epigenetic configuration of the chroGPS map.

Author(s)

Oscar Reina.

References

Changepoint package from Killick et al, 2012.

See Also

See documentation for package changepoint, clusGPS for epigenetic cluster generation.

Examples

```
# Not run
# data(s2)
# # Computing distances
# d <- distGPS(s2.tab,metric=tanimoto,uniqueRows=TRUE)
# # Creating MDS object
# mds1 <- mds(d,type=isoMDS)
# mds1
# plot(mds1)
```

```
# Precomputing clustering
# h <- hclust(as.dist(d@d),method=average)
# # Calculating densities (contours and probabilities), takes a while
# clus <- clusGPS(d,mds1,preMerge=TRUE,k=300) # Generating a high number of clusters
# clus <- mergeClusters(clus)
```

`procrustesAdj` *Use Procrustes to adjust an MDS map containing samples obtained under different conditions, e.g. technology or genetic backgrounds.*

Description

The function adjusts a previous `mds` to take into account that samples were obtained under different conditions, e.g. technological or genetic. Pairwise adjustments are performed by identifying samples present in both conditions and using Procrustes. When there are more than two conditions, sequential pairwise adjustments are applied (in the order that maximizes the number of common samples in each pairwise adjustment).

Usage

```
procrustesAdj(mds1, d, adjust, sampleid)
```

Arguments

`mds1`	Object of class `mds` with a Multi-dimensional scaling analysis on a distance matrix, typically obtained by a previous call to `mds`.
`d`	Object of class `distGPS` with the matrix used to create the Multidimensional Scaling object usually through a call to `mds`.
`adjust`	Vector indicating the adjustment factor, i.e. the condition under which each sample has been obtained.
`sampleid`	Vector containing the sample identifier. `sampleid` should take the same value for samples obtained under different conditions, as this is used to detect the samples to be used for Procrustes adjustment.

Details

We implement the Procrustes adjustment as follows. First we identify common samples, i.e. those obtained both under conditions A and B. Second, we use Procrustes to estimate the shift, scale and rotation that best matches the position of the samples in B to those in A. If only 1 sample was obtained under both conditions, only the shift is estimated. Last, we apply the estimated shift, scale and rotation to all B samples. That is, the Procruses parameters are estimated using common samples only, which are then applied to all samples to perform the adjustment.

Notice that the R square of the adjusted `mds` is typically improved after Procrustes adjustment, since distances between samples obtained under different conditions are set to `NA` and therefore MDS needs to approximate distances between less points.

When several replicates are available for a given `sampleid` under the same condition (`adjust`), the average position of all replicates is used.

Value

Adjusted mds object. Have in mind that only original distances between samples obtained under the same condition should be conserved, as the adjusted distances manipulated by Procrustes no longer correlate with the distances between their points in the adjusted MDS.

Methods

signature(x='mds') x is a mds object with the results of an MDS analysis.

See Also

distGPS for computing distances, mds to create MDS-oriented objects.

Examples

```
st1 <- runif(100,1,1000); st2 <- runif(100,500,1500)  #Peak starts
st3 <- runif(100,1000,2000); st4 <- runif(100,1500,2000)
#cond1: more precise technology
cond1 <- RangedDataList(s1=RangedData(IRanges(st1,st1+100)),s2=RangedData(IRanges(st2,st2+100)),s3=RangedData(I
#cond2: less precise
cond2 <- RangedDataList(s1=RangedData(IRanges(st1-200,st1+300)),s2=RangedData(IRanges(st2-200,st2+300)),s5=Rang
x <- c(cond1,cond2)
d <- distGPS(x,metric=tanimoto)  #compute distances
mds1 <- mds(d)  #MDS
#Adjust via Procrustes
mds2 <- procrustesAdj(mds1,d,adjust=rep(c(seq,chip),each=3),sampleid=names(x))
plot(mds1)
plot(mds2)
#Adjust via peak width
xadj <- adjustPeaks(x,adjust=rep(c(seq,chip),each=3),sampleid=names(x))
dadj <- distGPS(xadj)
mds3 <- mds(dadj)
plot(mds3)
```

---

profileClusters *Compute enrichment/depletion ratio for the observed epigenetic profiles in epigenetic clusters.*

Description

The function computes the ratio between the proportion of epigenetic mark presence in the clusters given as input and that observed for all elements. Results are returned as a numerical matrix, easily visualized in the shape of a classical heatmap.

Usage

```
profileClusters(x, uniqueCount = TRUE, weights, clus, i, minpoints, merged = FALSE, log2 = TRUE, plt = FA
```

Arguments

x	Genes * Factors matrix or data frame used for generating epigene clusters, indicating 1 for binding of factor j in gene i, 0 otherwise.
uniqueCount	Logical value to indicate if clusters come from epigenes (identical rows in x are merged into a single one) or genes (every row in x is mantained). See help for uniqueCount for details.
weights	Named vector analog to that on distGPS. Names are used as unique column names (i.e. epigenetic factors) so that enrichment profiles for replicates of the same epigenetic factor can be merged into a single element by computing its average enrichment (arithmetic mean).
clus	clusGPS object with epigenetic clusters generated from pairwise distances from x, as generated by the clusGPS function. See help for distGPS and clusGPS for details.
i	Clustering entry from which cluster profiling is to be computed.
minpoints	(Optional). Ignore clusters with fewer than minpoints, deprecated.
merged	(Optional). If clusters provided have been previously merged or not, deprecated.
log2	Logical to indicate if enrichment/depletion proportions are returned in log2 scale. Defaults to TRUE.
plt	Deprecated.

Value

A numerical matrix with the enrichment/depletion profile of the epigenetic marks for each cluster provided in the clusGPS object. Easy to visualize for instance with a heatmap plot.

Author(s)

Oscar Reina.

See Also

distGPS for computing pairwise distances between epigenetic elements. clusGPS for computing epigenetic clusters.

Examples

```
# Not run
# data(s2)
# # Computing distances
# d <- distGPS(s2.tab,metric=tanimoto,uniqueRows=TRUE)
# # Creating MDS object
# mds1 <- mds(d,type=isoMDS)
# mds1
# plot(mds1)
# Precomputing clustering
# h <- hclust(as.dist(d@d),method=average)
# # Calculating densities (contours and probabilities), takes a while
```

```
# clus <- clusGPS(d,mds1,preMerge=TRUE,k=max(cutree(h,h=0.5)))
# Computing cluster profiles
# p1 <- profileClusters(s2.tab, uniqueCount = TRUE, clus, i=125, minpoints=30, log2 = TRUE, plt = FALSE)
# Requires gplots
# heatmap.2(p1,col=redblue(100))
```

---

s2 *Sample binding site and related data from S2 and BG3 cell lines in Drosophila melanogaster.*

Description

chroGPS example dataset including ChIP-CHIP (modEncode) and ChIP-Seq (NCBI GEO GSE19325) data for Drosophila melanogaster S2 and BG3 cell lines as well as S2 wildtype gene expression values coming from Affymetrix Drosophila2 arrays. The object `toydists` stores precomputed `distGPS` objects (called `d, d2, d3`) for the epigenetic factors used in the dynamic vignette that comes with the package.

Usage

```
data(s2)
```

Source

http://www.modencode.org http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE19325

References

http://www.modencode.org http://www.ncbi.nlm.nih.gov/geo/

Examples

```
data(s2)
class(s2)
s2
s2names$Factor
data(s2Seq)
s2Seq
# See vignette examples for several uses of these datasets.
```

splitDistGPS-class *Class* "splitDistGPS"

Description

Set of pairwise distances between elements. This is an internal class to be used with the parallel version of mds, and should not be used on its own.

Objects from the Class

Objects from this class are used internally for parallel Multidimensional Scaling. See mds for details.

Slots

d: List of distGPS objects.

size: Object of class "numeric" indicating the size of the individual distGPS objects in the list. See function mds.

o: Object of class "numeric" with the overlap (anchor points) between adjacent distGPS objects. See function mds.

shuffle: Object of class "numeric", deprecated.

Author(s)

Oscar Reina

Examples

```
showClass("splitDistGPS")
```

Index