Package 'OMICsPCA'

October 15, 2023

Type Package

Title An R package for quantitative integration and analysis of multiple omics assays from heterogeneous samples

Version 1.18.0

Date 21/08/2018

Author Subhadeep Das [aut, cre], Dr. Sucheta Tripathy [ctb]

Maintainer Subhadeep Das <subhadeep1024@gmail.com>

Depends R (>= 3.5.0), OMICsPCAdata

Suggests knitr, RUnit, BiocGenerics

Description OMICsPCA is an analysis pipeline designed to integrate multi OMICs experiments done on various subjects (e.g. Cell lines, individuals), treatments (e.g. disease/control) or time points and to analyse such integrated data from various various angles and perspectives. In it's core OMICsPCA uses Principal Component Analysis (PCA) to integrate multiomics experiments from various sources and thus has ability to over data insufficiency issues by using the ingegrated data as representatives. OMICsPCA can be used in various application including analysis of overall distribution of OMICs assays across various samples /individuals /time points; grouping assays by user-defined conditions; identification of source of variation, similarity/dissimilarity between assays, variables or individuals.

License GPL-3

Encoding UTF-8

biocViews ImmunoOncology, MultipleComparison, PrincipalComponent, DataRepresentation, Workflow, Visualization, DimensionReduction, Clustering, BiologicalQuestion, EpigeneticsWorkflow, Transcription, GeneticVariability, GUI, BiomedicalInformatics, Epigenetics, FunctionalGenomics, SingleCell

Imports HelloRanges, fpc, stats, MultiAssayExperiment, pdftools, methods, grDevices, utils,clValid, NbClust, cowplot, rmarkdown, kableExtra, rtracklayer, IRanges, GenomeInfoDb, reshape2, ggplot2, factoextra, rgl, corrplot, MASS, graphics, FactoMineR, PerformanceAnalytics, tidyr, data.table, cluster, magick

VignetteBuilder knitr

LazyData TRUE

RoxygenNote 6.0.1

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

git_branch RELEASE_3_17

git_last_commit 141de51

git_last_commit_date 2023-04-25

Date/Publication 2023-10-15

R topics documented:

`analyse_individuals` *Quick analysis and visualization of the individuals/annotations/rows (e.g. Tss, gene) from integrated Assay.*

Description

This function displays scatterplot of the individuals from integrated assays. This allows the display of scatterplots both in 2D and 3D.

Usage

```
analyse_individuals(name,Assay, choice, PC, group,
groupinfo = NULL, ...)
```

Arguments

name	Name of the "PCA" object containing the assay data
Assay	Name of an Assay
choice	1 = 2D scatterplot on selected PCs through PC argument. 2 = 3D scatterplot on selected PCs through PC argument for details of the choices see the vignette
PC	A vector of numbers corresponding to principal components
group	A vector of group names. Required only if choice = 2
groupinfo	output of create_group, or similar object containing group information.
...	additional arguments allowed to base function "plot3d" of package "rgl"

Value

Displays various plots and tables as per the combination of input arguments

Author(s)

Subhadeep Das <subhadeep1024@gmail.com>

Examples

```
PCAlist <- integrate_variables(Assays = c("H2az","H3k4me1",
"H3k9ac"), name = multi_assay,
groups = c("WE","RE"), groupinfo = groupinfo_ext,
scale.unit = FALSE, graph = FALSE)

analyse_individuals(name = PCAlist,
Assay = "H3k9ac", groupinfo = groupinfo_ext,
choice = 1, PC = c(1,2))
```

---

analyse_integrated_individuals

Quick analysis and visualization of the individuals/annotations/rows (e.g. Tss, gene)

Description

This function works similarly as "analyse_individuals()". The only extra argument needs to be supplied here is "start_end", (if choice = 1.) returned by "integrate_pca()"

Usage

```
analyse_integrated_individuals(
name, choice = 1, geom = "point",
PC = c(1,2), groupinfo = NULL, ...)
```

Arguments

name	Name of the "PCA" object containing the integrated PCA.
choice	1 = 2D scatterplot on selected PCs through PC argument. 2 = 3D scatterplot on selected PCs through PC argument for details of the choices see the vignette
geom	Used when choice = 1. Avalilabe options are : "point", "text", c("point","text"). default = "point"
PC	A vector of numbers corresponding to principal components
groupinfo	same as integrate_variables()
...	additional arguments allowed to base function "plot3d" of package "rgl"

Value

Displays various plots and tables as per the combination of input arguments

Author(s)

Subhadeep Das <subhadeep1024@gmail.com>

Examples

```
exclude <- list(0,c(1,9))

int_PCA <- integrate_pca(Assays = c("H2az",
"H3k9ac"), name = multi_assay, mergetype = 2,
exclude = exclude,
groupinfo = groupinfo_ext, graph = FALSE)


name = int_PCA$int_PCA

analyse_integrated_individuals(
name = name,
choice = 2, PC = c(1,2,3),
col = c("RED", "BLACK","GREEN"),
groupinfo = groupinfo_ext)
```

analyse_integrated_variables *Quick analysis and visualization of the integrated Assays by integrate() function.*

Description

This function is designed for a quick analysis and visualization of the data integrated by "integrateAssays()"" function. It works similarly as analyse_variables. In addition to that it takes an additinal argument "start_end" returned by "integrateAssays()".

Usage

```
analyse_integrated_variables(start_end = start_end, Assay = "all",
name, choice, title = NULL, PC = 1,
var_type = "contrib", ...)
```

Arguments

start_end	a list as returned by "integrate_pca()"
Assay	Name of an Assay. default = "all"
name	Name of the "PCA" object returned by "integrate_pca()"
choice	1 = variance barplot 2 = Loadings of cell lines on selected PCs 3 = hCorrelation matrix 4 = Squarred loadings of Cell lines on a PC 5 = Contribution of Cell lines on a PC for details of the choices see the vignette
title	title of plots
PC	The PC on which the plots will be drawn. used in choice 4 and 5. default = 1
var_type	Used when choice = 2. available options are: "coord", "cos2" and "contrib"
...	additional arguments allowed to base functions "used in various choices. See vignettes for details.

Value

Displays various plots and tables as per the combination of input arguments

Author(s)

Subhadeep Das <subhadeep1024@gmail.com>

Examples

```
exclude <- list(0,c(1,9))

int_PCA <- integrate_pca(Assays = c("H2az",
"H3k9ac"), name = multi_assay, mergetype = 2,
exclude = exclude, graph = FALSE)

start_end = int_PCA$start_end

name = int_PCA$int_PCA


analyse_integrated_variables(start_end = start_end, name = name,
choice = 1, title = "variance barplot", Assay = 1, addlabels = TRUE)
```

---

analyse_variables *Quick analysis and visualization of the integrated Assays by integrate() function.*

Description

This function is designed for a quick analysis and visualization of the data integrated by integrate() function. It takes 3 compulsory arguments name, Assay and choice. The type of analysis should be selected through the "choice" argument. This function acts as a wrapper around a collection of functions of package "factoextra" and "corrplot" and thus can take additional arguments specific and non-conflicting to such functions.

Usage

```
analyse_variables(name, Assay, choice, title = NULL, PC = 1,
var_type = "contrib", ...)
```

Arguments

name	Name of the "PCA" object
Assay	Name of an Assay.
choice	1 = variance barplot 2 = Loadings of cell lines on selected PCs 3 = hCorrelation matrix 4 = Squarred loadings of Cell lines on a PC 5 = Contribution of Cell lines on a PC for details of the choices see the vignette
title	Title of the plot. default = NULL
PC	used in choice = 4 and 5. Indicates the PC on which choice 4 and 5 will depend. default = 1

var_type	used in choice = 2. available options are: "coord", "cos2" and "contrib". default = "contrib"
...	extra arguments passed through the functions imported from various packages. See vignettes for details.

Value

Displays various plots and tables as per the combination of input arguments

Author(s)

Subhadeep Das <subhadeep1024@gmail.com>

Examples

```
PCAlist <- integrate_variables(Assays = c("H2az","H3k4me1",
"H3k9ac"), name = multi_assay,
groups = c("WE","RE"), groupinfo = groupinfo_ext,
scale.unit = FALSE, graph = FALSE)

analyse_variables(name = PCAlist, Assay = "H2az", choice = 1,
title = "variance barplot", addlabels = TRUE)
```

---

chart_correlation *Pairwise correlation, scatter plot and histogram on selected groups.*

Description

This function creates pairwise correlation plots and tables; scatter plots and histograms on selected groups. The type of results should be passed through the argument 'choice'. This is a wrapper on various functions and thus can take additional and non-conflicting arguments specific to them.

Usage

```
chart_correlation(name, Assay, groups, choice,
groupinfo = NULL,
...)
```

Arguments

name	Name of the "MultiAssayExperiment" object containing the assay data
Assay	Name of an assay
groups	Name of all or subset of groups

choice
: "table" = correlation table

  "scatter" = scatterplot of each pair of columns/variables of selected groups

  "hist" = histogram of each column/variables

  "all" = all of 1,2,3 together

groupinfo
: output of create_group or similar object.

...
: additional arguments for base functions "chart.Correlation" and "pairs"

Value

Displays various plots and tables as per the combination of input arguments

Author(s)

Subhadeep Das <subhadeep1024@gmail.com>

Examples

```
chart_correlation(name = multi_assay,
Assay = "H2az", groupinfo = groupinfo_ext,
groups = "WE", choice = "table")
```

---

cluster *Cluster data points*

Description

Clustering of data points using various algorithms

Usage

```
cluster(name, n = NULL, graph = NULL, choice, title = NULL, ...)
```

Arguments

name
: name of the dataframe or matrix object containing the coordinates of data points. The output of "extract()" may be directly put here.

n
: Number of clusters

graph
: logical. Plots the clusterplot on first 2 dimensions if set TRUE

choice
: Clustering algorithm to use. Available choices are: "density", "kmeans", "pam"

title
: Title of the plot

...
: additional non-conflicting arguments to cluster functions

Value

returns a list containing the cluster and plot information

Author(s)

Subhadeep Das

References

Martin Ester, Hans-Peter Kriegel, Joerg Sander, Xiaowei Xu (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. Institute for Computer Science, University of Munich. Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96).

Forgy, E. W. (1965). Cluster analysis of multivariate data: efficiency vs interpretability of classifications. Biometrics, 21, 768–769.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm AS 136: A K-means clustering algorithm. Applied Statistics, 28, 100–108. doi: 10.2307/2346830.

Lloyd, S. P. (1957, 1982). Least squares quantization in PCM. Technical Note, Bell Laboratories. Published in 1982 in IEEE Transactions on Information Theory, 28, 128–137.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, eds L. M. Le Cam & J. Neyman, 1, pp. 281–297. Berkeley, CA: University of California Press.

Examples

```
exclude <- list(0,c(1,9))

int_PCA <- integrate_pca(Assays = c("H2az",
"H3k9ac"),
groupinfo = groupinfo,
name = multi_assay, mergetype = 2,
exclude = exclude, graph = FALSE)

name = int_PCA$int_PCA

data <- extract(name = name, PC = c(1:4),
groups = c("WE","RE"), integrated = TRUE,
rand = 300, groupinfo = groupinfo_ext)

clusters <- cluster(name = data, n = 2,
choice = "kmeans",
title = "kmeans on 2 clusters")
```

`cluster_boxplot` *Comparison of clusters by boxplot*

Description

boxplot of the assay values in each cluster

Usage

```
cluster_boxplot(name, Assay, clusterobject, clustercolumn, choice = NULL)
```

Arguments

`name`	
`Assay`	name of the assay which is to be compared
`clusterobject`	a cluster object returned by the cluster() function
`clustercolumn`	the column in the clusterobject containing the group membership information
`choice`	type of graph

Value

a ggplot object

Author(s)

Subhadeep Das

Examples

```
bp <- cluster_boxplot(name = multi_assay,
Assay = "H2az", clusterobject = clustered_data,
clustercolumn = 5)


bp
```

`cluster_parameters` *Detection of algorithm and number of clusters*

Description

Detection of appropriate clustering algorithm and cluster number for given data using "clvalid" and "NbClust" in background for cluster validation.

Usage

```
cluster_parameters(name, comparisonAlgorithm = "clValid",
optimal = FALSE, n = 2:6,
clusteringMethods = c("kmeans", "pam"),
validationMethods = c("internal", "stability"),
distance = "euclidean", ...)
```

Arguments

`name`	dataframe returned by "extract()".
`optimal`	logical. If TRUE, returns a dataframe of optimal results
`n`	a vector of numbers corresponding to the number of clusters to be tested or validated
`comparisonAlgorithm`	2 choices available: "clValid" or "NbClust"
`clusteringMethods`	a vector of single or multiple names of clustering algorithms. available choices are: 1) if comparisonAlgorithm = "clValid" : "hierarchical", "kmeans", "diana", "fanny", "som", "model", "sota", "pam", "clara" and "agnes" 2) if comparisonAlgorithm = "NbClust" : "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median", "centroid", "kmeans".
`validationMethods`	name of the method to validate clusters. Available options (one or more): 1) if comparisonAlgorithm = "clValid" then one of: "kl", "ch", "hartigan", "ccc", "scott", "marriot", "trcovw", "tracew", "friedman", "rubin", "cindex", "db", "silhouette", "duda", "pseudot2", "beale", "ratkowsky", "ball", "ptbiserial", "gap", "frey", "mcclain", "gamma", "gplus", "tau", "dunn", "hubert", "sdindex", "dindex", "sdbw", "all" (all indices except GAP, Gamma, Gplus and Tau), "alllong" (all indices with Gap, Gamma, Gplus and Tau included). 2) if comparisonAlgorithm = "NbClust" then one or more of: "internal", "stability", and "biological"
`distance`	metric used to calculate distance matrix. options: 1) for "clValid" :

"euclidean", "correlation", and "manhattan".

2) for "NbClust" :

This must be one of: "euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski"

... additional non-conflicting arguments to "clValid" or "Nbclust"

Value

1) for "clValid"

an object of class "clValid" (optimal = FALSE) or a dataframe of optimal values (optimal = TRUE)

2) for "NbClust"

a list of :

All.index, All.CriticalValues, Best.nc and Best.partition.

See the help pages of "clValid" (?clValid) and "NbClust" (?NbClust) for more details.

Author(s)

Subhadeep Das

References

Brock, G., Pihur, V., Datta, S. and Datta, S. (2008) clValid: An R Package for Cluster Validation Journal of Statistical Software 25(4) http://www.jstatsoft.org/v25/i04

Charrad M., Ghazzali N., Boiteau V., Niknafs A. (2014). "NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set.", "Journal of Statistical Software, 61(6), 1-36.", "URL http://www.jstatsoft.org/v61/i06/".

Examples

```
exclude <- list(0,c(1,9))

int_PCA <- integrate_pca(Assays = c("H2az",
"H3k9ac"),
groupinfo = groupinfo,
name = multi_assay, mergetype = 2,
exclude = exclude, graph = FALSE)

name = int_PCA$int_PCA

data <- extract(name = name, PC = c(1:4),
groups = c("WE","RE"), integrated = TRUE, rand = 600,
groupinfo = groupinfo_ext)

#### Using "clValid" ####

clusterstats <- cluster_parameters(name = data,
optimal = FALSE, n = 2:4, comparisonAlgorithm = "clValid",
distance = "euclidean", clusteringMethods = c("kmeans"),
```

```
validationMethods = c("internal"))
```

`create_group` *Subsets an Assay dataframe into smaller groups*

Description

This function subsests an user defined Assay into smaller groups according to user supplied instructions.

Usage

```
create_group(name, group_names = NULL,
                grouping_factor = NULL,
                comparison = NULL,
                condition = NULL
                )
```

Arguments

`name`	Name of the "MultiAssayExperiment"" object containing the assay data
`group_names`	A vector containing the user defined names of the groups to be created
`grouping_factor`	name of the dataframe on which grouping will be done
`comparison`	A vector of comparison symbols such as >, <, ==, >=, <=,%in%, etc
`condition`	A vector of conditions corresponding to 'comparison'. "condition"" should be a vector or range of digits (e.g. c(1,3,7,9) or 1:5) if%in% is chosen as comparison. Otherwise, a single digit should be chosen.

Value

returns the group membership of individuals as a dataframe

Author(s)

Subhadeep Das <subhadeep1024@gmail.com>

Examples

```
groupinfo <- create_group(name = multi_assay,
group_names = c("WE","RE","NE","IntE"),
grouping_factor = "CAGE",
comparison = c(">=","%in%","==","%in%"),
condition = c("25","1:5","0","6:24"))

head(groupinfo)
```

descriptor *Distribution of Assays as percentage of cell lines*

Description

descriptor() produces panels of boxplots displaying the distribution of various factors(e.g. ChIP-seq of H3k9ac histone modification) in percentage of cell lines.

choice = 1 output is collection of boxplots explaining the distribution of percentage of cell lines having or overlapping with the input factors (e.g. H3k9ac).

choice = 2 displays the distribution of percentage of cell lines having or overlapping with increasing number of factors for the selected group (e.g. WE or RE etc). If this option is selected, then a group name should be provided through choice2group.

Usage

```
descriptor(name, factors, groups,
choice = 1, choice2group = NULL, title = NULL,
groupinfo = NULL)
```

Arguments

name	Name of the "MultiAssayExperiment" object containing the assay data
factors	A vector containing the name of the Assays
groups	Name of all or subset of groups
choice	Numerical value between 1 and 2. See Description and vignettes for details of plots generated by this 2 choices
choice2group	Used when choice is 2. This argument allows a name of group.
title	Title of the plot
groupinfo	output of create_group() or similar object

Value

Displays various plots as per the combination of various input arguments.

Author(s)

Subhadeep Das <subhadeep1024@gmail.com>

Examples

```
descriptor(name = multi_assay,
factors = c(
"H2az","H3k9ac","H3k4me1"),
groups = c("WE","RE","IntE","NE"),
choice = 1,
title = "Distribution of percentage of cell types overlapping
```

```
with various factors",
groupinfo = groupinfo_ext)
```

---

extract *extraction of projected cooordinates from "PCA" object.*

Description

funtion to extract projected coordinates of individuals on principal components.

Usage

```
extract(name, groups, integrated = TRUE, Assay = NULL,
rand = NULL, PC = c(1,2,3,4), groupinfo = NULL)
```

Arguments

name	name of the OMICs object.
groups	one or more groups created by or supplied through "create_group()"
integrated	whether to extract coordinates from integrated or individual assay.
Assay	name of the assay if integrated = FALSE
rand	number of rows to be extracted randomly.
PC	principal component numbers. The projected informaton on this PCs will be extracted.
groupinfo	same as integrate_variables()

Value

a data frame of the extracted value.

Author(s)

Subhadeep Das

Examples

```
exclude <- list(0,c(1,9))

int_PCA <- integrate_pca(Assays = c("H2az",
"H3k9ac"),
groupinfo = groupinfo,
name = multi_assay, mergetype = 2,
exclude = exclude, graph = FALSE)

name = int_PCA$int_PCA
```

```
data <- extract(name = name, PC = c(1:4),
groups = c("WE","RE"), integrated = TRUE, rand = 600,
groupinfo = groupinfo_ext)
```

`extract_assay` *Data extraction from "MultiAssayExperiment" object*

Description

Extraction of individuals according to user supplied group and Assay

Usage

```
extract_assay(name, Assay, groups, groupinfo = NULL,
addgroupnames = TRUE)
```

Arguments

`name`	name of the "OMICs" object from which data is to be retrieved.
`Assay`	name of the "Assay" from which data is to be retrieved.
`groups`	name of the "groups" from which data is to be retrieved.
`groupinfo`	output of create_group() or similar object
`addgroupnames`	logical argument. If "TRUE", adds the group membership info of returned individuals

Value

a data frame containing the intended information

Author(s)

Subhadeep Das

Examples

```
x <- extract_assay(name = multi_assay, Assay = "H2az",
groups = c("WE","RE"), groupinfo = groupinfo_ext)

x
```

integrate_pca *Integration of multiple Assays into linear combinations by PCA.*

Description

This function integrates multiple assays read by InputOMICs() into linear combinations using PCA. Following earlier analyses, user may want to exclude some variables/columns/cell lines which should be supplied through the "exclude" argument (see vignettes for more explanation)

PCA is done by the function PCA() from package "FactoMineR". Additional arguments may be supplied to PCA() through "..." .

Usage

```
integrate_pca(Assays, name, exclude, mergetype = 1,
groupinfo = NULL, ...)
```

Arguments

Assays	A vector of names of an Assays read by "InputOMICs()"
name	Name of the "MultiAssayExperiment" object containing the assay data
exclude	A list that indicates which columns to exclude. See vignettes for details
mergetype	2 options : 1 = subset of groups should be entered by user and PCA is run on this subset to integrate Assays into PCs 2 = PCA is run on all groups
groupinfo	same as plot_density()
...	aditional arguments supplied to function "PCA()" from package "FactoMineR"

Details

The "mergetype" argument lets user decide whether to combine all groups or a subset of groups created by "create_group". If mergetype is set to 1, "integrateAssays" asks for the name of groups which should be a charatcer or charagter vector (see vignettes for more details).

Value

"integrateAssays()" returns a list containing the start and end column of each Assay as ordered in the "Assays" argument.

Author(s)

Subhadeep Das <subhadeep1024@gmail.com>

Examples

```
int_PCA <- integrate_pca(Assays = c("H2az",
"H3k9ac"), name = multi_assay, mergetype = 2,
exclude = list(0,c(1,9)), graph = FALSE,
groupinfo = groupinfo_ext)
```

`integrate_variables` *Integration of an experiment/ Assay done on many Cell lines/ time points into linear combinations by PCA.*

Description

If an assay (e.g. ChIP-seq for the histone modification H3k9ac) is done on multiple cells/conditions/treatment/time, it might be, sometimes, necessary to integrate or combine them. Such combination may be done by several techniques like Principal Component Analysis (PCA) or Factor Analysis (FA). This function integrates an assay experimented on multiple cells or conditions into many linear combinations using PCA.

Usage

```
integrate_variables(Assays, name, groups, groupinfo = NULL, ...)
```

Arguments

`Assays`	A vector of names of an assays
`name`	Name of the "MultiAssayExperiment" object containing the assay data
`groups`	Name of all or subset of groups
`groupinfo`	output of create_group or similar object containing group information
`...`	Additional arguments supplied to "PCA()" imported from "FactoMineR"

Value

an object of class "PCA"

Author(s)

Subhadeep Das <subhadeep1024@gmail.com>

Examples

```
PCAlist <- integrate_variables(Assays = c("H2az","H3k4me1",
"H3k9ac"), name = multi_assay,
groups = c("WE","RE"), groupinfo = groupinfo_ext,
scale.unit = FALSE, graph = FALSE)
```

intersect *support function of prepareDataset*

Description

intersects a factor file and an annotation file

Usage

```
intersect(fact, anno, ...)
```

Arguments

fact	Full path of the directory containing only the bed (.bed) files corresponding to several cell lines of the specified factor.
anno	Full path of the annotation file (e.g. gene, TSS, exon etc.)
...	additional arguments received from "prepareDataset()" and suppliet to "R_bedtools_intersect()"

Value

A list containing the intersections of the files (paths of which are supplied throgh the arguments).

Author(s)

Subhadeep Das

Examples

```
anno <- system.file("extdata/annotation2/TSS_groups.bed",
package = "OMICsPCAdata")

fact <- system.file("extdata/factors2/demofactor",
package = "OMICsPCAdata")

Cells <- intersect(fact = fact, anno = anno)

Cells
```

merge_cells *Storage of intersected factors from multiple cell types as columns into a dataframe*

Description

This function takes the output list of the function "intersect" and stores the intersected value of the factor and annotation files into a dataframe. The values corrsponding to the entries in annotation file represents the corresponding value of in each cell line. If an entry is not found in a factor file, its corresponding value will be 0.

Usage

```
merge_cells(list, Cells)
```

Arguments

list	Full path of the file containing name of the Annotations (e.g. gene) in a single column
Cells	output list of function "integrate()"

Value

A dataframe wich rows corresponding to each entry in annotation file and columns corresponding to the intersected value with the corresponding annotation in each cell line.

Author(s)

Subhadeep Das

Examples

```
anno <- system.file("extdata/annotation2/TSS_groups.bed",
package = "OMICsPCAdata")

fact <- system.file("extdata/factors2/demofactor",
package = "OMICsPCAdata")

Cells <- intersect(fact = fact, anno = anno)

list <- system.file("extdata/annotation2/TSS_list",
package = "OMICsPCAdata")

merged_Cells <- merge_cells(list = list, Cells = Cells)

head(merged_Cells)
```

plot_density *Visualization of Density of various groups on Principal components*

Description

This function displays density of individuals

Usage

```
plot_density(name, Assay, PC = 1, groups,
groupinfo = NULL, ...)
```

Arguments

name	Name of the "PCA" object containing the assay data
Assay	Name of an Assay
PC	A number corresponding to the principal component on which density is to be calculated
groups	A vector of names or subset of names of groups
groupinfo	same as integrate_variables()
...	additional arguments of base function "geom_density" from package "ggplot2"

Value

a "gg" "ggplot" object

Author(s)

Subhadeep Das <subhadeep1024@gmail.com>

Examples

```
PCAlist <- integrate_variables(Assays = c("H2az","H3k4me1",
"H3k9ac"), name = multi_assay,
groups = c("WE","RE"), groupinfo = groupinfo_ext,
scale.unit = FALSE, graph = FALSE)

densityplot <- plot_density(name = PCAlist,
Assay = "H2az", groupinfo = groupinfo_ext,
PC = 1, groups = c("WE","RE"),
adjust = 1)
```

plot_density_3D *Visualization of 3D Density of various groups on Principal components*

Description

This function displays density of individuals from various groups created by create_group(). This wraps "persp" from package "graphics" and thus takes additional graphical input shown in example.

Usage

```
plot_density_3D(name, Assay, group, PC1 = 1,
PC2 = 2, static = FALSE,
gridsize = 100, groupinfo = NULL, ...)
```

Arguments

name	Name of the "PCA" object containing the assay data
Assay	Name of an Assay
group	Names of a group
PC1, PC2	Numbers corresponding to the principal components on which density is to be calculated
static	Logical if TRUE a static plot is generated. default = FALSE
gridsize	A number used in kernel smoothing. default is 100
groupinfo	Same as integrate_variables()
...	additional arguments passed to base function "persp" from package "graphics"

Details

2D density is calculated using the "kde2d"" function from package "MASS"" which use kernel density estimation (KDE) to calculate density of 2D data. If the variance on either or both of the PCs are 0, the KDE can't be calculated.

Value

Displays 3D density plots of PCs

Author(s)

Subhadeep Das <subhadeep1024@gmail.com>

Examples

```
PCAlist <- integrate_variables(Assays = c("H2az"),
name = multi_assay,
groups = c("WE","RE"), groupinfo = groupinfo_ext,
scale.unit = FALSE, graph = FALSE)

plot_density_3D(name = PCAlist, Assay = "H2az",
group = "WE", PC1 = 1, PC2 = 2, grid_size = 100,
static = FALSE, groupinfo = groupinfo_ext)
```

---

plot_integrated_density

Visualization of Density of various groups on Principal components of integrated assays

---

Description

This function works similarly as "plot_density()". See the man page of "plot_density()" for details.

Usage

```
plot_integrated_density(name, PC = 1, groups,
groupinfo = NULL, ...)
```

Arguments

name	Name of the integrated "PCA"" object
PC	A number corresponding to the principal component on which density is to be calculated. default = 1
groups	A vector of names or subset of names of groups
groupinfo	same as plot_density()
...	additional arguments allowed to base function "geom_density" of package "ggplot2"

Value

a "gg" "ggplot" object

Author(s)

Subhadeep Das <subhadeep1024@gmail.com>

Examples

```
exclude <- list(0,c(1,9))

int_PCA <- integrate_pca(Assays = c("H2az",
"H3k9ac"), name = multi_assay, mergetype = 2,
exclude = exclude, groupinfo = groupinfo_ext,
,graph = FALSE)

name = int_PCA$int_PCA

densityplot <- plot_integrated_density(name = name, PC = 1,
groups = c("WE","RE","IntE","NE"), groupinfo = groupinfo_ext)

# additional graphical functions (e.g. xlim, ylim, theme) may be
#added with densityplot (see section VIII. Density analysis)

densityplot
```

`plot_integrated_density_3D` *Visualization of 3D Density of various groups on Principal components of multiple Assays*

Description

works similarly as "plot_density_3D" . See vignettes and man page of "plot_density_3D" for more details.

Usage

```
plot_integrated_density_3D(name, PC1 = 1, PC2 = 2, group,
gridsize = 100, static = FALSE, groupinfo = NULL, ...)
```

Arguments

`name`	Name of the integrated "PCA" object
`PC1, PC2`	Numbers corresponding to the principal components on which density is to be calculated
`group`	Names of a group
`gridsize`	A number used in kernel smoothing. default is 100
`static`	Logical if TRUE a static plot is generated. default = FALSE
`groupinfo`	same as integrate_variables()
`...`	additional arguments allowed to base function "persp" of package "graphics"

Details

2D density is calculated using the "kde2d"" function from package "MASS"" which use kernel density estimation (KDE) to calculate density of 2D data. If the variance on either or both of the PCs are 0, the KDE can't be calculated.

Value

Displays 3D density plots.

Author(s)

Subhadeep Das <subhadeep1024@gmail.com>

Examples

```
exclude <- list(0,c(1,9))

int_PCA <- integrate_pca(Assays = c("H2az",
"H3k9ac"), name = multi_assay, mergetype = 2,
exclude = exclude, groupinfo = groupinfo_ext,
,graph = FALSE)

name = int_PCA$int_PCA


plot_integrated_density_3D(name = name, PC1 = 1, PC2 = 2,
group = c("WE","RE"), gridsize = 100, static = FALSE,
groupinfo = groupinfo_ext)
```

---

prepare_dataset *Intersects and merge data from multiple experiments into a dataframe*

Description

This function intersects feature files (e.g. ChIP-seq, RNA seq) with annotation files(e.g.exons, genes) and represents the intersected values of each annotation from all cell lines in a data frame.

Usage

```
prepare_dataset(factdir, annofile, annolist, ...)
```

Arguments

factdir	Full path of directory containing (only) feature (e.g ChIP-seq, CAGE) files in bed format
annofile	Full path of the annotation file (e.g. TSS, exon, gene) in bed format
annolist	Full path of the file containing name of the annotations (e.g. name of exons)
...	additional arguments to pass to "R_bedtools_intersect()" through "intersect()"

Details

If an annotation (e.g. gene) does not present in an assay file (e.g ChIP-seq of H3k9ac in the Cell Gm12878), this function puts a 0 as its value.

Value

a dataframe of intersected values.

Author(s)

Subhadeep Das <subhadeep1024@gmail.com>

Examples

```
anno <- system.file("extdata/annotation2/TSS_groups.bed",
                    package = "OMICsPCAdata")

list <- system.file("extdata/annotation2/TSS_list",
                    package = "OMICsPCAdata")

fact <- system.file("extdata/factors2/demofactor",
                    package = "OMICsPCAdata")

prepare_dataset(factdir = fact, annofile = anno, annolist = list)
```

Index