Package 'TimiRGeN'

May 30, 2023

Type Package

Title Time sensitive microRNA-mRNA integration, analysis and network generation tool

Version 1.10.0

Description TimiRGeN (Time Incorporated miR-mRNA Generation of Networks) is a novel R package which functionally analyses and integrates time course miRNA-mRNA differential expression data. This tool can generate small networks within R or export results into cytoscape or pathvisio for more detailed network construction and hypothesis generation. This tool is created for researchers that wish to dive deep into time series multiomic datasets. TimiRGeN goes further than many other tools in terms of data reduction. Here, potentially hundreds-of-thousands of potential miRNA-mRNA interactions can be whittled down into a handful of high confidence miRNA-mRNA interactions affecting a signalling pathway, across a time course.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 4.1), Mfuzz, MultiAssayExperiment

Imports biomaRt, clusterProfiler, dplyr (>= 0.8.4), FreqProf, gtools (>= 3.8.1), gplots, ggdendro, gghighlight, ggplot2, graphics, grDevices, igraph (>= 1.2.4.2), RCy3, readxl, reshape2, rWikiPathways, scales, stats, tidyr (>= 1.0.2), stringr (>= 1.4.0)

Suggests BiocManager, kableExtra, knitr (>= 1.27), org.Hs.eg.db, org.Mm.eg.db, testthat, rmarkdown

VignetteBuilder knitr

biocViews Clustering, miRNA, Network, Pathways, Software, TimeCourse, Visualization

URL https://github.com/Krutik6/TimiRGeN/

BugReports https://github.com/Krutik6/TimiRGeN/issues

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

git_branch RELEASE_3_17

git_last_commit 74072d3

git_last_commit_date 2023-04-25

Date/Publication 2023-05-29

Author Krutik Patel [aut, cre]

Maintainer Krutik Patel <krutik.patel@newcastle.ac.uk>

R topics documented:

`addIds` *addIds*

Description

Adds entrez or ensembl IDs to the nested dataframes within a list(c) or list of lists (s). The IDs are created by the getIdsMir and getIdsMrna functions, both are needed for addIds.

Usage

```
addIds(MAE, method, filtered_genelist, miR_IDs, mRNA_IDs)
```

Arguments

`MAE`	MultiAssayExperiment to store the output of addIds. It is recommended to use the MAE which stores results from significantVals.
`method`	Either "c" or "s", respectively for combined or separated analysis.
`filtered_genelist`	A list of nested dataframes if 'c' or a list of lists with nested dataframes if 's'. This will be found as metadata within the MAE object used in the significantVals function.
`miR_IDs`	miR_ensembl or miR_entrez. Use a getIDsMir function to acquire this. This will be stored as an assay in the MAE used in a getIdsMir function.
`mRNA_IDs`	mRNA_ensembl or mRNA_entrez. Use a getIDsMrna function to acquire this. This will be stored as an assay in the MAE used in a getIdsMrna function.

Value

List of dataframes with entrezIDs/ ensembl IDs and gene names as columns which will be stored as metadata in the input MAE.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:50,]

mRNA <- mm_mRNA[1:100,]

Data <- startObject(miR = mm_miR, mRNA = mm_mRNA)

Data <- getIdsMir(Data, assay(Data, 1), orgDB = org.Mm.eg.db, 'mmu')

Data <- getIdsMrna(Data, assay(Data, 2), mirror = 'useast',
                   species = 'mmusculus', orgDB = org.Mm.eg.db)

Data <- combineGenes(MAE = Data, miR_data = assay(Data, 1),
                     mRNA_data = assay(Data, 2))

Data <- genesList(MAE = Data, method = 'c', genetic_data = assay(Data, 9),
                  timeString = 'D')

Data <- significantVals(MAE = Data, method = 'c',
                        geneList = metadata(Data)[[1]],
                        maxVal = 0.05, stringVal = "adjPVal")

Data <- addIds(MAE = Data, method = "c",
               filtered_genelist = metadata(Data)[[2]],
               miR_IDs = assay(Data, 3), mRNA_IDs = assay(Data, 7))
```

---

addPrefix *addPrefix*

Description

Adds 'miR_' or 'mRNA_' genetypes to colnames of dataframes. Resulting colnames should be in the following naming system: 'genetype_timepoint.resulttype'. This function is essential for separate analysis of miR-mRNA DE data. If using the combined analysis, there is no need to use addPrefix.

Usage

```
addPrefix(MAE, gene_df, prefixString = '')
```

Arguments

MAE	MultiAssayExperiment to store output of addPrefix. It is recommended to use the MAE object which stores output from startObject.
gene_df	Dataframe of mRNA or miR results from differential expression analysis. Will be stored as an assay within the MAE used in the startObject function.
prefixString	Prefix to be added e.g. "miR" or "mRNA".

Value

Dataframe which has a specific prefix infront of each column name. Will be stored as an assay in the input MAE.

Examples

```
data(mm_miR)

data(mm_mRNA)

MAE <- startObject(miR = mm_miR, mRNA = mm_mRNA)

MAE <- addPrefix(MAE = MAE, gene_df = assay(MAE, 1),
                 prefixString = "miR")

MAE <- addPrefix(MAE = MAE, gene_df = assay(MAE, 2),
                 prefixString = "mRNA")
```

---

clusterCheck *clusterCheck*

Description

clusterCheck creates a PCA plot using functions from Mfuzz. This will help to indicate if an appropriate number of clusters have been created. The closer the circles are to one another the more likely that they should belong to the same cluster. Read more about Mfuzz here https://bioconductor.org/packages/release/bioc/html/

Usage

```
clusterCheck(Clusters, W)
```

Arguments

Clusters	A large list of clusters, statistics and phenodata. This will be stored as metadata within the MAE used in the createClusters function.
W	TRUE or FALSE. Should the plot be shown in a new window? Default is FALSE.

Value

A PCAplot showing distance of clusters.

Examples

```
MAE <- MultiAssayExperiment()

metadata(MAE)[["e_list"]] <- e_list_mouse

metadata(MAE)[["w_list"]] <- w_list_mouse[1:10]

MAE <- wikiMatrix(MAE, ID_list = metadata(MAE)[[1]],
                  wp_list = metadata(MAE)[[2]])

MAE <- turnPercent(MAE = MAE,
                   wikiMatrix = assay(MAE, 1))

MAE <- createClusters(MAE, method = "c",
                    percentMatrix = assay(MAE, 2),
                    noClusters = 2, variance = 0.99)

clusterCheck(Clusters = metadata(MAE)[[3]], W = FALSE)
```

`clusterList` *clusterList*

Description

clusterList will transform clusters created by createClusters2 into lists based on which genes associate most to each cluster. Genes which associate with a cluster are determined by the fitCluster parameter in the function.

Usage

```
clusterList(MAE, clusterData, fitCluster, miR_IDs, mRNA_IDs)
```

Arguments

`MAE`	MultiAssayExperiment which will store the results from createClusters.
`clusterData`	A dataframe which contains cluster-pathway fit scores and is stored as an assay within the MAE used in the createClusters2 function.
`fitCluster`	Integer from 0-1. How well should genes fit into a cluster? Default is 0.5.
`miR_IDs`	miR_ensembl or miR_entrez. Use a getIDsMir function to acquire this. This will be stored as an assay in the MAE used in a getIdsMir function.
`mRNA_IDs`	mRNA_ensembl or mRNA_entrez. Use a getIDsMrna function to acquire this. This will be stored as an assay in the MAE used in a getIdsMrna function.

Value

A list containing the genes which fit to each cluster.

Examples

```
library(org.Hs.eg.db)

data(long_data)

miRNA <- long_data[c(1:105),]

mRNA <- long_data[-c(1:105),]

MAE <- startObject(miR = miRNA, mRNA = mRNA)

MAE <- getIdsMir(MAE, assay(MAE, 1), orgDB = org.Hs.eg.db, 'hsa')

MAE <- getIdsMrna(MAE, assay(MAE, 2), mirror = 'useast', 'hsapiens',
                  orgDB = org.Hs.eg.db)

MAE <- combineGenes(MAE, miR_data = assay(MAE, 1),
                    mRNA_data = assay(MAE, 2))

MAE <- createClusters2(MAE = MAE, genetic_data = assay(MAE, 9),
                       noClusters =2)

MAE <- clusterList(MAE = MAE, clusterData = assay(MAE, 11), fitCluster = 0.5,
                   miR_IDs = assay(MAE, 3),
                   mRNA_IDs = assay(MAE, 7))
```

---

combineGenes *combineGenes*

Description

Combines miR and mRNA data into one dataframe. Input columns should be written as : timepoint.DifferentialExpressionResultType e.g. D1.log2fc or H6.adjPval. Column names should be the same for miR and mRNA data. If a more detailed explanation of column nomenclature is needed please read the vignette. combineGenes is essential for combined analysis of miR-mRNA data. If using separate analysis, there is no need to use combineGenes.

Usage

```
combineGenes(MAE, miR_data, mRNA_data)
```

Arguments

MAE	Input MAE which stores results from combineGenes. It is recommended to use the MAE which was used in startObject.
miR_data	microRNA dataframe. Rows should be genes, columns are DE results and time point. This should be the stored as an assay within the MAE used in the startObject function.
mRNA_data	mRNA dataframe. Rows should be genes, columns are DE results and time point. This should be the stored as an assay within the MAE used in the startObject function.

Value

A dataframe with combined miR and mRNA data. Will be stored as an assay in the input MAE.

Examples

```
library(org.Mm.eg.db)

data(mm_miR)

data(mm_mRNA)

MAE <- startObject(miR = mm_miR, mRNA = mm_mRNA)

MAE <- combineGenes(MAE = MAE, miR_data = assay(MAE, 1),
                    mRNA_data = assay(MAE, 2))
```

---

createClusters *createClusters*

Description

Creates soft clusters to assess changes in gene abundance during the time course in many pathways. createClusters will create 3 data files. 1) Clusters will contain cluster logistics information and will be stored as metadata, 2) MfuzzData will contain fuzzy clustering information and will be stored as an experiment, 3) ClusterData will contain cluster-pathway fit information and will be stored as an assay. This function may take some time as it downloads pathway information.

Usage

```
createClusters(MAE, method, percentMatrix, noClusters,
dataString = '', variance)
```

Arguments

MAE	MultiAssayExperiment which will store the results from createClusters. It is recommended to use the MAE object which stores the output of by turnPercent.
method	Either "c" or "s", respectively for combined or separated analysis.
percentMatrix	A matrix containing wikipathway-data information. It is output from the turnPercent function and will be stored as an assay within the MAE used in the turnPercent function.
noClusters	Number of clusters to create, the default is 5.
dataString	Only for use in "s" analysis. Insert the prefix string e.g. "mRNA" or "miR". The string added should be the same as the prefixString added during the addPrefix function.
variance	Numeric value from 0-1 to control strictness of filtering. Higher variance means more pathways will be excluded from the analysis.

Value

3 new objects in the input MAE. Clusters(metadata): A list to be used as the input in checkClusters and quickFuzz. MfuzzData(ExperimentList): An ExpressionSet object to be used as input for quickFuzz. ClusterData(assay): An assay to be used as input for returnCluster.

Examples

```
MAE <- MultiAssayExperiment()

metadata(MAE)[["e_list"]] <- e_list_mouse

metadata(MAE)[["w_list"]] <- w_list_mouse[1:10]

MAE <- wikiMatrix(MAE, ID_list = metadata(MAE)[[1]],
                  wp_list = metadata(MAE)[[2]])

MAE <- turnPercent(MAE = MAE,
                   wikiMatrix = assay(MAE, 1))

MAE <- createClusters(MAE, method = "c",
                    percentMatrix = assay(MAE, 2),
                    noClusters = 2, variance = 0.99)
```

---

createClusters2 *createClusters2*

Description

Creates clusters from a dataframe of mRNAs and miRs. This function should primarily be used when analysing data that has not gone through pair-wise DE. This function will create clusters from longitudinal temporal patterns. createClusters2 will create 3 data files. 1) Clusters will contain cluster logistics information and will be stored as metadata, 2) MfuzzData will contain fuzzy clustering

information and will be stored as an experiment, 3) ClusterData will contain cluster-pathway fit information and will be stored as an assay.

Usage

```
createClusters2(MAE, genetic_data, noClusters)
```

Arguments

MAE	MultiAssayExperiment which will store the results from createClusters2.
genetic_data	A dataframe with miR and mRNA information together. This is the output from the combineGenes function and will be stored as an assay within the MAE used in the combineGenes function.
noClusters	How many clusters should be generated? Default is 5.

Value

3 new objects in the input MAE. Clusters(metadata): A list to be used as the input in checkClusters and quickFuzz. MfuzzData(ExperimentList): An ExpressionSet object to be used as input for quickFuzz. ClusterData(assay): An assay to be used as input for returnCluster.

Examples

```
data(long_data)
miRNA <- long_data[c(1:105),]
mRNA <- long_data[-c(1:105),]

MAE <- startObject(miR = miRNA, mRNA = mRNA)

MAE <- combineGenes(MAE, miR_data = assay(MAE, 1),
                    mRNA_data = assay(MAE, 2))

MAE <- createClusters2(MAE = MAE, genetic_data = assay(MAE, 3))
```

---

cytoMake *cytoMake*

Description

Creates a cytoscape network based on the output of matrixFilter. Requires cytoscapePing() to be used. Make sure Cytoscape is open first. Must use Cytoscape version 3.7 or later.

Usage

```
cytoMake(interactionData, titleString = '', collectionString = '')
```

Arguments

`interactionData`
: Dataframe which contains filtered miR-mRNA interactions. This is output from matrixFilter and should be found as an assay within the MAE used in the matrixFilter function.

`titleString`
: Title of the network. Enter a string which cytoscape will see as the graph name. Default is "Network".

`collectionString`
: Title of the collection of networks. Enter string which cytoscape will see as the collection name. Many differently titled graphs can be added to a single collection. Default is "miR-mRNA interactions".

Value

A network of filtered miR-mRNA interactions specific for a pathway of interest. It will be visible in cytoscape version 3.7 or later.

Examples

```
## Not run:
Filt_df <- data.frame(row.names = c("mmu-miR-320-3p:Acss1",
                                    "mmu-miR-27a-3p:Odc1"),
                      avecor = c(-0.9191653, 0.7826041),
                      miR = c("mmu-miR-320-3p", "mmu-miR-27a-3p"),
                      mRNA = c("Acss1", "Acss1"),
                      miR_Entrez = c(NA, NA),
                      mRNA_Entrez = c(68738, 18263),
                      TargetScan = c(1, 0),
                      miRDB = c(0, 0),
                      Predicted_Interactions = c(1, 0),
                      miRTarBase = c(0, 1),
                      Pred_Fun = c(1, 1))

RCy3::cytoscapePing()

cytoMake(interactionData = Filt_df, titleString = 'test' ,
         collectionString = 'collectiontest')

## End(Not run)
```

---

dataMiningMatrix *dataMiningMatrix*

Description

Mines out predicted/ functional interactions which correspond between miR-mRNA interactions found in Targetscans, miRDB, miRTarBase and the interactions from the miR-mRNA correlation matrix. If a database cannot be downloaded, dataMiningMatrix can be used regardless, but it is recommended to download all three databases.

Usage

```
dataMiningMatrix(MAE, corrTable, targetscan , mirdb, mirtarbase)
```

Arguments

MAE	MultiAssayExperiment which will store the output of dataMiningMatrix. It is recommended to use the MAE object which stores output from the mirMrnaInt function.
corrTable	Correlation matrix of interactions between the mRNAs from a pathway of interest and miRNA data. This is created by the mirMrnaInt function and should be stored as an assay within the MAE used in the mirMrnaInt function.
targetscan	Species specific miR-mRNA interactions predicted by targetscans. This is the output from the dloadTargetscan function. It should be stored as an assay within the MAE used in the dloadTargetscan function. If this data cannot be downloaded, dataMiningMatrix can be run without it.
mirdb	Species specific miR-mRNA interactions predicted by miRDB. This is the output from the dloadMirdb function. It should stored as an assay within the MAE used in the dloadMirdb function. If this data cannot be downloaded, dataMiningMatrix can be run without it.
mirtarbase	Species specific miR-mRNA interactions which are functionally curated by mirtarbase. This is the output from the dloadMirtarbase function. It should be stored as an assay within the MAE used in the dloadMirtarbase function. If this data cannot be downloaded, dataMiningMatrix can be run without it.

Value

A matrix which cross references the occurrences of miR-mRNA interactions between databases and the given data. Output will be stored as an assay in the input MAE.

---

diffExpressRes *diffExpressRes*

Description

diffExpressRes will produce a dataframe which contains data for only one result type, along with an ID of choice. It is recommended to use this function on a DE results which represents abundance such as log2fc or average expression, as this data will be averaged and correlated later in the analysis. This is to be used for miR and mRNA data individually.

Usage

```
diffExpressRes(MAE, df, dataType = '', genes_ID, idColumn = '',
name = '')
```

Arguments

MAE	MultiAssayExperiment to store the output of diffExpressRes within it. This function is to be used after pathways of interest have been identified by enrichWiki or returnCluster. It is recommended to store all diffExpressRes results in the MAE used in enrichWiki and/ or returnCluster.
df	mRNA or miR dataframe (rownames as genes and DE results as columns). These will be found as assays in the MAE object used within the startObject function.
dataType	Column name to take an average from e.g. "Log2FC", "AveExp". This string should be found consistently in the column names of your input data. It is recommended to use a DE result value which represents abundance, rather than confidence.
genes_ID	Dataframe that was created from a getIds function e.g. mRNA_ensembl or miR_entrez. Use the same ID type for miR and mRNA data. These dataframes will be found as assays within the MAE which stores results from the getIds functions.
idColumn	Name of column to use as the merge point. If Column names in getIds results have not been changed, it should be "GENENAME". Default has been left as "GENENAME".
name	New name of the assay. Should be a unique string. Remember each assay in a MAE must have a unique name.

Value

Dataframe with only a single result type from DE (e.g. Log2FC) and an ID type e.g. entrezIDs. Output will be stored as an assay in the input MAE.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:50,]

mRNA <- mm_mRNA[1:100,]

MAE <- startObject(miR = miR, mRNA = mRNA)

MAE <- getIdsMir(MAE, assay(MAE, 1), orgDB = org.Mm.eg.db, 'mmu')

MAE <- getIdsMrna(MAE, assay(MAE, 2), "useast", 'mmusculus', orgDB = org.Mm.eg.db)

MAE <- diffExpressRes(MAE, df = assay(MAE, 2), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 7),
                      idColumn = 'GENENAME',
                      name = "mRNA_log2fc")
```

dloadGmt *dloadGmt*

Description

Downloads the most up-to-date versions of the mouse or human wikipathway information databases. Output will be stored as three distinct dataframes within the input MAE 1) path_gene, 2) path_names, 3) path_data.

Usage

```
dloadGmt(MAE, species = "")
```

Arguments

MAE	MultiAssayExperiment to store downloaded GMT data in. It might be useful to start a new MAE for dloadGmt using MultiAssayExperiment(). This is so the MAE objects used in this analysis do not get too large.
species	Full species names e.g. retrieve "Homo sapiens" or "Mus musculus" data.

Value

3 dataframes. 1) path_gene, 2) path_names, 3) path_data. All of which will be stored as assays in the input MAE.

Examples

```
MAE <- MultiAssayExperiment()

MAE <- dloadGmt(MAE, species = "Homo sapiens")
```

dloadMirdb *dloadMirdb*

Description

Downloads most recent version (6.0) of predicted targets from the mirdb database http://mirdb.org/download.html. This will take some time. miR-mRNA interactions from the species of interest will be extracted. Species of interests associated org.db package must be loaded beforehand.

Usage

```
dloadMirdb(MAE, species, orgDB)
```

Arguments

MAE	MultiAssayExperiment which will store downloaded mirDB data. It is recommended to use the MAE which was used in the mirMrnaInt function.
species	Species of interest e.g. "hsa" or "mmu".
orgDB	Organism db package specific for species of interest.

Value

A dataframe of predicted, species specific mRNA-miR interactions. Will be stored as an assay in the input MAE.

Examples

```
## Not run:

library(org.Mm.eg.db)

MAE <- MultiAssayExperiment()

MAE <-dloadMirdb(MAE, 'mmu', org.Mm.eg.db)


## End(Not run)
```

---

dloadMirtarbase *dloadMirtarbase*

Description

Downloads most recent version (8.0) of functional targets from the miRTarBase database http://mirtarbase.cuhk.edu.cn/php/do
. Species specific miR-mRNA interactions which do not have 'weak' evidence are used.

Usage

```
dloadMirtarbase(MAE, species)
```

Arguments

MAE	MultiAssayExperiment which will store the downloaded mirtarbase data. It is recommended to use the MAE which was used in the mirMrnaInt function.
species	Species of interest e.g. "hsa" or "mmu".

Value

Dataframe of species specific miR-mRNA interactions with strong functional evidence. Output will be stored as an assay in the input MAE.

Examples

```
MAE <- MultiAssayExperiment()

MAE <- dloadMirtarbase(MAE, "mmu")
```

---

dloadTargetscan *dloadTargetscan*

Description

Downloads most recent version (7.2) of predicted targets from the targetscan database http://www.targetscan.org/cgi-bin/targetscan/data_download.vert72.cgi. miR-mRNA interactions from the species of interest will be extracted.

Usage

```
dloadTargetscan(MAE, species)
```

Arguments

MAE	MultiAssayExperiment which will store the downloaded targetscan data. It is recommended to use the MAE which was used in the mirMrnaInt function.
species	Species of interest e.g "hsa" or "mmu."

Value

Dataframe of species specific predicted mRNA-miR interactions. Output will be stored as an assay in the input MAE.

Examples

```
## Not run:
MAE <- MultiAssayExperiment()

MAE <-dloadTargetscan(MAE, "mmu")

## End(Not run)
```

eNames *eNames*

Description

Extracts the gene IDs from nested dataframes created from the addIds function.

Usage

```
eNames(MAE, method = '', gene_IDs)
```

Arguments

MAE	MultiAssayExperiment which will store output of eNames. It is recommended to use the MAE which stores the output of addIds.
method	Either 'c' or 's', respectively for combined or separated analysis.
gene_IDs	List of DE data and associated gene IDs. This is the output from the addIds function, this should be found as metadata in the MAE used in the addIds function.

Value

A single list of entrez/ ensembl IDs for each time point. Output will be stored as metadata in the input MAE.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:50,]

mRNA <- mm_mRNA[1:100,]

Data <- startObject(miR = mm_miR, mRNA = mm_mRNA)

Data <- getIdsMir(Data, assay(Data, 1), org.Mm.eg.db, 'mmu')

Data <- getIdsMrna(Data, assay(Data, 2), mirror = 'useast', 'mmusculus', org.Mm.eg.db)

Data <- combineGenes(MAE = Data, miR_data = assay(Data, 1),
                     mRNA_data = assay(Data, 2))

Data <- genesList(MAE = Data, method = 'c', genetic_data = assay(Data, 9),
                  timeString = 'D')

Data <- significantVals(MAE = Data, method = 'c',
                        geneList = metadata(Data)[[1]],
                        maxVal = 0.05, stringVal = "adjPVal")
```

```
Data <- addIds(MAE = Data, method = "c",
               filtered_genelist = metadata(Data)[[2]],
               miR_IDs = assay(Data, 3), mRNA_IDs = assay(Data, 7))

Data <- eNames(MAE = Data, method = "c", gene_IDs = metadata(Data)[[3]])
```

---

enrichWiki *enrichWiki*

Description

Finds which wikipathways are enriched within the data. This function uses gene set enrichment analysis from clusterProfiler to find enriched signalling pathways. Each time point is analysed individually. In the case of separated TimiRGeN analysis, each gene type and time point are analysed individually.

Usage

```
enrichWiki(MAE, method = '', ID_list, orgDB, path_gene, path_name,
                  ID = '', universe, pvalcutoff, qvaluecutoff,
                  padjustmethod)
```

Arguments

MAE	MultiAssayExperiment which will store the output from enrichWiki. It is recommended to use the MAE object which stores the output from the dloadGmt function.
method	Either 'c' or 's', respectively for combined or separated analysis.
ID_list	List of ensembl or entrez IDs for each sample. This is the output from eNames function. This will be found as metadata within the MAE used in the eNames function.
orgDB	DB package of the species being analysed. e.g. org.Mm.eg.db if mouse miR-mRNA data is being looked into.
path_gene	Dataframe containing pathway ID - gene ID information. This is output from either dloadGmt or gmtEnsembl. It will be stored as an assay within the MAE used in dloadGmt or gmtEnsembl.
path_name	Dataframe containing pathway ID - pathway names information. This is output from dloadGmt. It will be stored as an assay within the MAE used in dloadGmt.
ID	Either "ENTREZID" or "ENSEMBL". This should be the same as the ID type used for ID_list. dloadGmt loads data as entrez gene IDs and gmtEnsembl converts this to ensembl gene IDs.
universe	A column of gene IDs to be used as the background for gene set enrichment. IDs should be stored as characters. It is recommended to use all genes found within the wikipathways of the species being analysed as background i.e. path_gene$gene or universe = assay(MAE, i)[[2]]/ MAE[[i]][2]. To add a unique universe, create a list of gene IDs (entrezID or ensembl) which are classed as characters.

pvalcutoff	Default is 0.05. P value cut-off point.
qvaluecutoff	Default is 0.2. q value cut-off point.
padjustmethod	Default is 'BH'. This sets the pvalue adjustment method. Look into the enricher function from clusterProfiler for more info.

Value

A large list which identifies which wikipathways are most enriched at each time point of the input data. Output will be stored as metadata in the input MAE.

Examples

```
library(org.Mm.eg.db)

MAE <- MultiAssayExperiment()

metadata(MAE)[["e_list"]] <- e_list_mouse

MAE <- dloadGmt(MAE, species = "Mus musculus")

MAE <- enrichWiki(MAE = MAE, method = 'c', ID_list = metadata(MAE)[[1]],
                  orgDB = org.Mm.eg.db, path_gene = assay(MAE, 1),
                  path_name = assay(MAE, 2), ID = "ENTREZID",
                  universe = assay(MAE, 1)[[2]])
```

---

e_list_mouse *e_list_mouse*

Description

List of entrezIDs of the significantly DE genes from the mouse fibrosis dataset. To make examples run faster this data is used, instead of re-running previous functions throughout the examples of TimiRGeN. e_list_mouse is the output of the eNames function when the combined analysis is performed on mm_miR and mm_RNA and when entrezIDs are used as the gene IDs.

Usage

```
data("e_list_mouse")
```

Format

The format is: List of 5: D1 D2 D3 D7 D14 Each list contains significantly differentially expressed entrezgene IDs, specific for each time point.

Details

List of entrezgeneIDs per time point (5) from combined analysis of mm_miR and mm_mRNA. Used to speed up examples for building and checks.

Source

From using eNames during combined analysis of mm_miR and mm_mRNA..

References

NA

Examples

```
data(e_list_mouse)
```

`genesList` *genesList*

Description

Produces a list of nested dataframes. The list will depend on the type of analysis that is to be conducted. For combined analysis method = "c", and for separated analysis method = "s".

In combined analysis colnames should be 'timepoint.resulttype'. genesList will make new dataframes separated at 'timepoint.'.

In separated analysis colnames should be 'genetype_timepoint.resulttype'. genesList will make separate lists for each 'genetype_', and these lists will have dataframes which have been made by separating at 'timepoint.'.

Make sure to follow colname nomenclature carefully. Please refer to the vignette for more details on the nomenclature.

Usage

```
genesList(MAE, method, genetic_data, timeString, miR_data, mRNA_data)
```

Arguments

`MAE`	MultiAssayExperiment which will store the output from genesList. It is recommended to use the MAE which stores output from combineGenes (combined analysis) or addPrefix (separated analysis).
`method`	Either "c" or "s", respectively for combined or separated analysis.
`genetic_data`	If "c", this should be a dataframe with miR and mRNA information together. This is the output from the combineGenes function and will be stored as an assay within the MAE used in the combineGenes function.
`timeString`	If "c", this should be a common string representing 'timepoints' e.g. for H.1, H.10, H.20, timeString = "H".
`miR_data`	If "s", a dataframe of microRNA data. Rownames are genes and colnames are: genetype_timepoint.resulttype. Column names should be the same in mRNA and miR data. miR_data is from the addPrefix function, and will be stored as an assay within the MAE used in addPrefix.

mRNA_data — If "s", a dataframe of mRNA data. Rownames are genes and colnames are: genetype_timepoint.resulttype. Column names should be the same in mRNA and miR data. mRNA_data is from the addPrefix function, and will be stored as an assay within the MAE used in addPrefix.

Value

A list of dataframes separated by features in the column names. Output will be stored as metadata in the input MAE.

Examples

```
miR <- mm_miR[1:50,]

mRNA <- mm_mRNA[1:100,]

MAE <- startObject(miR = mm_miR, mRNA = mm_mRNA)

# For separated analysis

MAE <- addPrefix(MAE = MAE, gene_df = assay(MAE, 1),
                 prefixString = "miR")

MAE <- addPrefix(MAE = MAE, gene_df = assay(MAE, 2),
                 prefixString = "mRNA")

MAE <- genesList(MAE, method = "s", miR_data = assay(MAE, 3),
                 mRNA_data = assay(MAE, 4))

# For combined analysis

MAE <- combineGenes(MAE, miR_data = assay(MAE, 1),
                    mRNA_data = assay(MAE, 2))

MAE <- genesList(MAE, method = 'c', genetic_data = assay(MAE, 3),
                 timeString = 'D')
```

---

getIdsMir *getIdsMir*

Description

getIdsMir will produce ensembl and entrez ID data for microRNAs. It will also produce adjusted ensembl and entrez for IDs that are specific to microRNAs that share an ID. They will be stored as 4 individual assays in a MAE. org.Mm.eg.db must be loaded prior to using this function.

Usage

```
getIdsMir(MAE, miR, orgDB, miRPrefix)
```

Arguments

MAE	MultiAssayExperiment to store the output of getIdsMir. It is recommended to use the MAE which contains output from startObject.
miR	A Dataframe. Rownames are genes and columns are results of DE. This should be found as an assay within the MAE used in the startObject function. Please read vignette for nomenclature guidance.
orgDB	org.xx.eg.db package which corresponds to the species being analysed.
miRPrefix	microRNA prefix for the species being analysed e.g. 'mmu', 'hsa', 'rno' ect.

Value

4 dataframes consisting of either entrez or ensembl ID information. 2 of these will be adjusted for shared IDs. Output will be stored as assays in the input MAE.

Examples

```
library(org.Mm.eg.db)

data(mm_miR)

# Make sure miRNA gene name nomenclature is correct for TimiRGeN analysis!

miR <- mm_miR[1:100,]

MAE <- startObject(miR = miR, mRNA = NULL)

MAE <- getIdsMir(MAE, assay(MAE, 1), orgDB = org.Mm.eg.db, miRPrefix = 'mmu')
```

---

getIdsMrna *getIdsMrna*

Description

getIdsMrna will produce ensembl and entrez ID dataframes for mRNAs. These will be stored as 2 individual assays within a MAE.

This function will attempt to use biomaRt in the first instance. If server issues occur, it will instead use clusterProfiler. Generally biomaRt has more annotation IDs, but the results between these methods will vary.

Usage

```
getIdsMrna(MAE, mRNA, mirror, species, orgDB)
```

Arguments

MAE	MultiAssayExperiment to store the output of getIdsMrna. It is recommended to use the MAE which contains output from startObject.
mRNA	Dataframe. Rownames are genes and columns are results of DE. This should be found as an assay within the MAE used in the startObject function. Please read vignette for nomenclature guidance.
mirror	String to identify which biomaRt server is best. This is based on location. Either 'useast', 'uswest', 'asia' or 'www'. Default is 'www'.
species	Species of interest. E.g. mmusculus or hsapiens.
orgDB	org.xx.eg.db package which corresponds to the species being analysed.

Value

2 new dataframes in the MAE. One with entrez information and the other with ensembl gene ID information.

Examples

```
library(org.Mm.eg.db)
data(mm_mRNA)

mRNA <- mm_mRNA[1:20,]

MAE <- startObject(miR = NULL, mRNA = mRNA)

MAE <- getIdsMrna(MAE = MAE, mRNA = assay(MAE, 2), mirror = 'useast',
                      species = 'mmusculus', orgDB = org.Mm.eg.db)
```

---

gmtEnsembl *gmtEnsembl*

Description

Change entrez IDs in path_gene and path_data into ensembl IDs. Will create two new dataframes with ensembl IDs and wikipathway information.

Usage

```
gmtEnsembl(MAE, path_gene, path_data, orgDB)
```

Arguments

MAE	MultiAssayExperiment which will store the output of gmtEnsembl. It is recommended to use the same MAE object which contains output from dloadGmt.
path_gene	Dataframe with wikipathway IDs and entrezgene IDs. path_gene is from the dloadGmt function. It will be stored as an assay within the MAE used in the dloadGmt function.

path_data	Dataframe with wikipathway IDs, wikipathway names and entrezgene IDs. path_data is from the dloadGmt function. It will be stored as an assay within the MAE used in the dloadGmt function.
orgDB	Load the appropriate db package e.g. org.Hs.eg.db if human wikipathways are being used.

Value

2 dataframes. One containing wikipathway IDs and ensembl gene IDs, and the other containing wikipathway IDs, ensembl gene IDs and wikipathway names. Output will be stored as assays in the input MAE.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:50,]

mRNA <- mm_mRNA[1:100,]

MAE <- startObject(miR = mm_miR, mRNA = mm_mRNA)

MAE <- dloadGmt(MAE, species = "Mus musculus")

MAE <- gmtEnsembl(MAE = MAE, assay(MAE, 3),
                  assay(MAE, 5), org.Mm.eg.db)
```

---

hs_miR *Human microRNA data set*

Description

Differential expression from Human breast cancer cells (MCF-7). Normoxic conditions contrasted against 16H, 32H and 48H under hypoxia. Data was put though limma for DE. hs_miR consists DE results from 189 microRNAs.

Column names are in the following format: timepoint(hours).DEresult (logFC or adjPVal).

The miR names are not using in TimiRGeN friendly nomenclature. Gene names must be changed before use in TimiRGeN analysis. This must be changed before using TimiRGeN. miR gene name changing is explained in section 3.1 of the vignette.

Usage

```
data("hs_miR")
```

Format

A data frame of miRNAs put through differential expression with 189 observations on the following 6 variables.

`H16.logFC` a numeric vector containing log2FC values of Hypoxia_16_hours/ Normoxic conditions.

`H16.adjPVal` a numeric vector containing adjusted P values values of Hypoxia_16_hours/ Normoxic conditions.

`H32.logFC` a numeric vector containing log2FC values of Hypoxia_32_hours/ Normoxic conditions.

`H32.adjPVal` a numeric vector containing adjusted P values values of Hypoxia_32_hours/ Normoxic conditions

`H48.logFC` a numeric vector containing log2FC values of Hypoxia_48_hours/ Normoxic conditions.

`H48.adjPVal` a numeric vector containing adjusted P values values of Hypoxia_48_hours/ Normoxic conditions

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE47534

References

C. Camps, H. K. Saini, D. R. Mole, H. Choudhry, M. Reczko, J. A. Guerra-Assunção, Y.-M.Tian, F. M. Buffa, A. L. Harris, A. G. Hatzigeorgiou, et al., "Integrated analysis of microrna and mrna expression and association with hif binding reveals the complexity of microrna expression regulation under hypoxia," Molecular cancer, vol. 13, no. 1, p. 28, 2014.. <https://molecular-cancer.biomedcentral.com/articles/10.1186/1476-4598-13-28>

Examples

```
data(hs_miR)
## maybe str(hs_miR) ; plot(hs_miR) ...
```

---

hs_mRNA *Human mRNA data set*

Description

Differential expression from Human breast cancer cells (MCF-7). Normoxic conditions contrasted against 16H, 32H and 48H under hypoxia mRNA data. Data was put though limma for DE. hs_mRNA only has DE results from 2000 mRNA genes for speed and size optimisation.

Column names are in the following format:timepoint(hours).DEresult (logFC or adjPVal).

Usage

```
data("hs_mRNA")
```

Format

A data frame of mRNAs put through differential expression with 2000 observations on the following 6 variables.

`H16.logFC` a numeric vector containing log2FC values of Hypoxia_16_hours/ Normoxic conditions.

`H16.adjPVal` a numeric vector containing adjusted P values values of Hypoxia_16_hours/ Normoxic conditions.

`H32.logFC` a numeric vector containing log2FC values of Hypoxia_32_hours/ Normoxic conditions.

`H32.adjPVal` a numeric vector containing adjusted P values values of Hypoxia_32_hours/ Normoxic conditions

`H48.logFC` a numeric vector containing log2FC values of Hypoxia_48_hours/ Normoxic conditions.

`H48.adjPVal` a numeric vector containing adjusted P values values of Hypoxia_48_hours/ Normoxic conditions

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE47534

References

C. Camps, H. K. Saini, D. R. Mole, H. Choudhry, M. Reczko, J. A. Guerra-Assunção, Y.-M.Tian, F. M. Buffa, A. L. Harris, A. G. Hatzigeorgiou, et al., "Integrated analysis of microrna and mrna expression and association with hif binding reveals the complexity of microrna expression regulation under hypoxia," Molecular cancer, vol. 13, no. 1, p. 28, 2014.. <https://molecular-cancer.biomedcentral.com/articles/10.1186/1476-4598-13-28>

Examples

```
data(hs_mRNA)
## maybe str(hs_mRNA) ; plot(hs_mRNA) ...
```

`hs_probes` *hs_probes*

Description

Gene names and ensemble genes IDs stored as characters. This is used as a demonstration in using a "user made" universe for overenrichment analysis. This is only used in section 3 of the vignette.

Usage

```
data("hs_probes")
```

Format

A data frame with 16801 observations on the following 2 variables.

`external_gene_name` a character vector of gene names of probes.

`ensembl_gene_id` a character vector of ensemble gene IDs of probes.

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GPL6884 https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GPL8

References

Camps C, Saini HK, Mole DR, Choudhry H et al. Integrated analysis of microRNA and mRNA expression and association with HIF binding reveals the complexity of microRNA expression regulation under hypoxia. Mol Cancer 2014 Feb 11;13:28. PMID: 24517586

Examples

```
data(hs_probes)
```

`linearRegr` *linearRegr*

Description

Creates a linear model between a mRNA or miRNA of choice and a selection of it's filtered binding partners. The model is based on the users design.

Usage

```
linearRegr(mreg, colselect, colpair, alterpairs)
```

Arguments

`mreg`	Matrix generated from the multiReg function. This should be found as an assay within the MAE used during the multiReg function.
`colselect`	Column from mreg which contain the gene of interest. Default is 2.
`colpair`	Column of binding parter of the gene of interest. Default is 3. If NANs are found, test alternative formulas.
`alterpairs`	Column(s) of other binding partners of the gene of interest. This can include any numer of columns. If NANs are found, test alternative formulas. e.g. = 4:7, c(4, 6, 8), 4.

Value

A linear regression model which represents miRNA-mRNA interaction(s) which can be further explored.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:100,]

mRNA <- mm_mRNA[1:200,]

MAE <- startObject(miR = miR, mRNA = mRNA)

MAE <- getIdsMir(MAE, assay(MAE, 1), orgDB = org.Mm.eg.db, 'mmu')

MAE <- getIdsMrna(MAE, assay(MAE, 2), "useast", 'mmusculus', orgDB = org.Mm.eg.db)


MAE <- diffExpressRes(MAE, df = assay(MAE, 1), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 3),
                       idColumn = 'GENENAME',
                       name = "miRNA_log2fc")

MAE <- diffExpressRes(MAE, df = assay(MAE, 2), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 7),
                      idColumn = 'GENENAME',
                      name = "mRNA_log2fc")

Filt_df <- data.frame(row.names = c("mmu-miR-145a-3p:Adamts15",
                                    "mmu-miR-146a-5p:Acy1"),
                      corr = c(-0.9191653, -0.7826041),
                      miR = c("mmu-miR-145a-3p", "mmu-miR-146a-5p"),
                      mRNA = c("Adamts15", "Acy1"),
                      miR_Entrez = c(387163, NA),
                      mRNA_Entrez = c(235130, 109652),
                      TargetScan = c(1, 0),
                      miRDB = c(0, 0),
                      Predicted_Interactions = c(1, 0),
                      miRTarBase = c(0, 1),
                      Pred_Fun = c(1, 1))

MAE <- matrixFilter(MAE, miningMatrix = Filt_df, negativeOnly = FALSE,
                    threshold = 1, predictedOnly = FALSE)

MAE <- multiReg(MAE = MAE, gene_interest = "Adamts15",
                mRNAreg =TRUE, filt_df=MAE[[11]], miRNA_exp=MAE[[9]],
                mRNA_exp=MAE[[10]])

model1 <- linearRegr(mreg = MAE[[12]], colselect =2, colpair =3)
summary(model1$regression)
```

---

long_data *Data from a longer time series (10 time points)*

Description

Averaged raw count data of miRNAs (105) and mRNAs (1041). Time based differential expression was performed with DESeq2 (using LTR method). Differentially expressed genes were filtered from average raw counts. miRNA and mRNA data was combined into one dataframe. This is only example data used in section 5 of the vignette.

Usage

```
data("long_data")
```

Format

A data frame with 1146 observations on the following 10 variables.

`H0.Counts` a numeric vector of Raw counts at 0 hours.

`H1.Counts` a numeric vector of Raw counts at 1 hours.

`H2.Counts` a numeric vector of Raw counts at 2 hours.

`H3.Counts` a numeric vector of Raw counts at 3 hours.

`H4.Counts` a numeric vector of Raw counts at 4 hours.

`H5.Counts` a numeric vector of Raw counts at 5 hours.

`H6.Counts` a numeric vector of Raw counts at 6 hours.

`H8.Counts` a numeric vector of Raw counts at 8 hours.

`H12.Counts` a numeric vector of Raw counts at 12 hours.

`H24.Counts` a numeric vector of Raw counts at 24 hours.

References

Baran-Gale J, Purvis JE, Sethupathy P. An integrative transcriptomics approach identifies miR-503 as a candidate master regulator of the estrogen response in MCF-7 breast cancer cells. RNA 2016 Oct;22(10):1592-603. PMID: 27539783

Examples

```
data(long_data)
miRNA <- long_data[c(1:105),]
mRNA <- long_data[-c(1:105),]
```

makeDynamic *makeDynamic*

Description

Produces a dataframe that can be imported into pathvisio to show how changes in genes expression levels over the time course. Follow instructions found in the vignette which show how to save this file and further instructions found in the issues section of the TimiRGeN gihub https://github.com/Krutik6/TimiRGeN/issues/2

Usage

```
makeDynamic(MAE, miR_expression, mRNA_expression, miR_IDs_adj,
                  dataType = '')
```

Arguments

MAE	MultiAssayExpreriment to store the output of makeDynamic. It is recommended to use the same MAE which stores output from matrixFilter.
miR_expression	Dataframe containing abundance values (e.g. log2fc or average expression) from miR specific differential expression , along with gene IDs. This is the output from diffExpressRes. Output of diffExpressRes should be stored as an assay within the MAE used in diffExpressRes.
mRNA_expression	Dataframe containing abundance values (log2fc or average expression) from mRNA specific differential expression, along with gene IDs. This is the output from diffExpressRes. Output of diffExpressRes should be stored as an assay within the MAE used in diffExpressRes.
miR_IDs_adj	Dataframe which contains adjusted gene IDs from miR data. Either miR_adjusted_entrez or miR_adjusted_ensembl. Should be found as an assay in the MAE used a getIdsMir function.
dataType	String which represents the gene ID used in this analysis. Either "En" (ensembl data) or "L" (entrez data).

Value

miR and mRNA dynamic data that can be saved and be used in pathvisio to display dynamic behaviour of miRs and mRNAs of interest over the time series in a signalling pathway of interest. Output will be stored as an assay in the input MAE.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:50,]

mRNA <- mm_mRNA[1:100,]
```

```
MAE <- startObject(miR = miR, mRNA = mRNA)

MAE <- getIdsMir(MAE, assay(MAE, 1), orgDB = org.Mm.eg.db, 'mmu')

MAE <- getIdsMrna(MAE, assay(MAE, 2), "useast", 'mmusculus', orgDB = org.Mm.eg.db)

MAE <- diffExpressRes(MAE, df = assay(MAE, 1), dataType = 'Log2FC',
               genes_ID = assay(MAE, 3),
               idColumn = 'GENENAME',
               name = "miR_express")

MAE <- diffExpressRes(MAE, df = assay(MAE, 2), dataType = 'Log2FC',
               genes_ID = assay(MAE, 7),
               idColumn = 'GENENAME',
               name = 'mRNA_express')

MAE <- makeDynamic(MAE, miR_expression = assay(MAE, 9),
                   mRNA_expression = assay(MAE, 10),
                   miR_IDs_adj = assay(MAE, 5),
                   dataType = "L")
```

---

makeMapp *makeMapp*

Description

Creates a dataframe which can be imported into pathvisio by use of the the MAPP plugin. This will add the filtered miRs to the wikipathway of interest on pathvisio. Follow instructions found in the vignette which show how to save this file and further instructions found in the issues section of the TimiRGeN gihub https://github.com/Krutik6/TimiRGeN/issues/2.

Usage

```
makeMapp(MAE, filt_df, miR_IDs_adj, dataType = '')
```

Arguments

MAE	MultiAssayExperiment to store the output of makeMapp. It is recommended to use the same MAE which stores the output from matrixFilter.
filt_df	Dataframe of mined miR-mRNA interactions. This is output of matrixFilter. It should be stored as an assay in the MAE used in the matrixFilter function.
miR_IDs_adj	Dataframes with adjusted gene IDs to account for -5p and -3p specific miRs. miR_adjusted_entrez or miR_adjusted_ensembl. Should be found as assays within the MAE used a getIdsMir function.
dataType	String which represents the gene ID used in this analysis. Either "En" (ensembl data) or "L" (entrez data).

Value

A dataframe containing microRNAs and adjusted gene IDs which can be saved as a text file to be imported into pathvisio via the MAPPapp. Output will be saved as an assay in the input MAE.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:50,]

mRNA <- mm_mRNA[1:100,]

MAE <- startObject(miR = mm_miR, mRNA = mm_mRNA)

MAE <- getIdsMir(MAE, assay(MAE, 1), orgDB = org.Mm.eg.db, 'mmu')

MAE <- getIdsMrna(MAE, assay(MAE, 2), "useast", 'mmusculus', orgDB = org.Mm.eg.db)

Filt_df <- data.frame(row.names = c("mmu-miR-320-3p:Acss1",
                                    "mmu-miR-27a-3p:Odc1"),
                      avecor = c(-0.9191653, 0.7826041),
                      miR = c("mmu-miR-320-3p", "mmu-miR-27a-3p"),
                      mRNA = c("Acss1", "Odc1"),
                      miR_Entrez = c(NA, NA),
                      mRNA_Entrez = c(68738, 18263),
                      TargetScan = c(1, 0),
                      miRDB = c(0, 0),
                      Predicted_Interactions = c(1, 0),
                      miRTarBase = c(0, 1),
                      Pred_Fun = c(1, 1))

MAE <- makeMapp(MAE, filt_df = Filt_df, miR_IDs_adj = assay(MAE, 5),
                dataType = 'L')
```

---

makeNet *makeNet*

Description

Creates an igraph object from filtered miR-mRNA interactions. Resulting list can be used to display an internal R miR-mRNA interaction network.

Usage

```
makeNet(MAE, filt_df)
```

Arguments

MAE	MultiAssayExperiment to store output from makeNet. It is recommended to use the same MAE which stores output from matrixFilter.
filt_df	Filtered miR-mRNA interactions produced by the matrixFilter function. This should be stored as an assay within the MAE used in the matrixFilter function.

Value

A list of igraph data which represent miR-mRNA interactions filtered from the input data, wikipathway of choice and database mining, This list is input for quickNet. Output will be stored as metadata in the input MAE.

Examples

```
Filt_df <- data.frame(row.names = c("mmu-miR-320-3p:Acss1",
                                    "mmu-miR-27a-3p:Odc1"),
                      corr = c(-0.9191653, 0.7826041),
                      miR = c("mmu-miR-320-3p", "mmu-miR-27a-3p"),
                      mRNA = c("Acss1", "Acss1"),
                      miR_Entrez = c(NA, NA),
                      mRNA_Entrez = c(68738, 18263),
                      TargetScan = c(1, 0),
                      miRDB = c(0, 0),
                      Predicted_Interactions = c(1, 0),
                      miRTarBase = c(0, 1),
                      Pred_Fun = c(1, 1))

MAE <- MultiAssayExperiment()

MAE <- makeNet(MAE, Filt_df)
```

---

matrixFilter *matrixFilter*

Description

Filters out miR-mRNA interactions based on how many times an interaction has been predicted and/ or validated. miR-mRNA interactions can also be filtered by correlations of expression values (log2fc or ave exp). Negatively correlating miR-mRNA interactions can be filtered for, and degree of correlation is also a filterable parameter.

Usage

```
matrixFilter(MAE, miningMatrix, negativeOnly, predictedOnly,
                    threshold, maxCor)
```

Arguments

MAE	MultiAssayExperiment to store the output of matrixFilter. It is recommended to use the same MAE which stores the results from dataMiningMatrix.
miningMatrix	A large correlation matrix which has miR-mRNA validation information from targetscans, mirdb and mirtarbase. This is output from dataMiningMatrix, and should be stored as an assay within the MAE used in the dataMiningMatrix function.
negativeOnly	TRUE or FALSE. Should only negatively correlating miR-mRNA interactions be retrieved? Default is TRUE.
predictedOnly	TRUE or FALSE. Should only predicted interactions should be retrieved? Default is TRUE.
threshold	Integer from 0 to 3. How many databases should a miR-mRNA interaction be found in? If predictedOnly = TRUE, then maximum threshold is 2.
maxCor	Number from -1 to 1. What is the highest average correlation that is allowed? Default is -0.5. The lower the maxCor, the stricter the filtering.

Value

Filtered miR-mRNA interactions that are specific for a signalling pathway of interest and the input data. Output will be stored as an assay in the input MAE.

Examples

```
Int_matrix <- data.frame(row.names = c("mmu-miR-320-3p:Acss1",
                                       "mmu-miR-27a-3p:Odc1"),
                         corr = c(-0.9191653, 0.7826041),
                         miR = c("mmu-miR-320-3p", "mmu-miR-27a-3p"),
                         mRNA = c("Acss1", "Odc1"),
                         miR_Entrez = c(NA, NA),
                         mRNA_Entrez = c(68738, 18263),
                         TargetScan = c(1, 0),
                         miRDB = c(0, 0),
                         Predicted_Interactions = c(1, 0),
                         miRTarBase = c(0, 1),
                         Pred_Fun = c(1, 1))

MAE <- MultiAssayExperiment()

MAE <- matrixFilter(MAE, miningMatrix = Int_matrix, negativeOnly = TRUE,
                         threshold = 1, predictedOnly = FALSE)
```

---

mirMrnaInt *mirMrnaInt*

Description

Create a correlation matrix of all the potential miR-mRNA interactions which could arise between the input miR data and the mRNAs found from the wikiMrna function. The time series DE data will be averaged from the dataframe created by diffExpressRes of miR data and the dataframe created by wikiMrna. This will show miR-mRNA correlations over the time course.

Usage

```
mirMrnaInt(MAE, miR_express, GenesofInterest, maxInt, corMeth)
```

Arguments

MAE	MultiAssayExperiment which will store the output of mirMrnaInt. It is recommended to begin a new MAE using MultiAssayExperiment() here so the MAE objects do not get too large.
miR_express	Dataframe from using the diffExpressRes function on miR data. Rownames should be miR gene names and columns should include DE results displaying abundance e.g. log2fc or average expression. These dataframes should also have gene IDs. This should be stored as an assay within the MAE used in the diffExpressRes function.
GenesofInterest	Dataframe including mRNAs found in both the input data and the pathway of interest, as well as gene IDs. This is the output from wikiMrna. This should be found as an assay within the MAE which was used in the wikiMrna function. Make sure the same ID type is used in the inputs for miR_express and GenesofInterest.
maxInt	Integer. Should be equal to number of samples in both mRNA and miR data e.g. number of different time points. In the example it is 5 because there are 5 time points.
corMeth	Add string : "pearson", "spearman" or "kendall". Default is "pearson".

Value

A large correlation matrix which contains averaged miR-mRNA time series information for every possible miR-mRNA interaction between the genes of interest and all the miRs. Output will be stored as an assay in the input MAE.

Examples

```
G <- data.frame(row.names = c("Acaa1a", "Acadm", "Acss1", "Adh1"),
                "D1.Log2FC" = c("-1.2944593","-2.0267432","-2.1934942",
                                "-2.1095853"),
                "D2.Log2FC" = c("-1.1962396","-2.1345451","-1.7699232",
                                "-1.0961674"),
                "D3.Log2FC" = c("0.2738496","-1.9991046","-1.7637549",
                                "-1.6572653"),
                "D7.Log2FC" = c("-0.51765245","-2.20689661","-0.68479699",
                                "-2.06512466"),
```

```
                  "D14.Log2FC" = c("-0.4510294","-1.1523849","-0.4297012",
                                   "-1.1017597"),
                  "ID" = c("113868","11364","68738","11522"))

MIR <- data.frame(row.names = c("mmu-miR-101a-3p", "mmu-miR-101a-5p",
                                "mmu-miR-101c", "mmu-miR-106a-5p"),
                  "D1.Log2FC" = c("-0.0039141722","-0.4328659746",
                                  "-0.0038897133", "-0.4161749123"),
                  "D2.Log2FC" = c("-0.210605345","-0.600422732",
                                  "-0.210574742", "-0.530311376"),
                  "D3.Log2FC" = c("-0.315070839","-0.745367163",
                                  "-0.315012148", "-0.559274530"),
                  "D5.Log2FC" = c("-0.41087763","-0.63952382",
                                  "-0.41087876", "-1.03618015"),
                  "D14.Log2FC" = c("-0.39466968","-0.60122678",
                                   "-0.39461099", "-0.41889698"),
                  "ID" = c("387143","387143","100628572","723829"))

MAE <- MultiAssayExperiment()

MAE <- mirMrnaInt(MAE, miR_express = MIR, GenesofInterest = G,
                      maxInt = 5, corMeth = "pearson")
```

`miRTarBase` *miRTarBase dataset*

Description

miRTarBase dataset - downloaded from http://mirtarbase.cuhk.edu.cn/php/download.php on 7/03/21. miRNA-mRNA interactions labelled as "weak" have been removed.

Usage

```
data("miRTarBase")
```

Format

A data frame with 13315 observations on the following 9 variables.

`miRTarBase.ID` Column of characters describing the ID of each miRNA-mRNA interaction.

`miRNA` Column of characters describing the ID of each miRNA from an interaction.

`Species..miRNA.` Column of characters describing the name of the species from each miRNA of an interaction.

`Target.Gene` Column of characters describing the ID of each mRNA from an interaction.

`Target.Gene..Entrez.ID.` Column of characters describing the entrezgene ID of each miRNA from an interaction.

`Species..Target.Gene.` Column of characters describing the name of the species from each mRNA of an interaction.

`Experiments` Column of characters describing the experiment used to capture each miNRA-mRNA interaction.

`Support.Type` Column of characters describing the strength of each experiment.

`References..PMID.` Column of characters describing the publication ID from each miRNA-mRNA interaction.

Source

http://mirtarbase.cuhk.edu.cn/php/download.php

References

Huang HY, Lin YC, Li J, Huang KY, Shrestha S, Hong HC, Tang Y, Chen YG, Jin CN, Yu Y, Xu JT, Li YM, Cai XX, Zhou ZY, Chen XH, Pei YY, Hu L, Su JJ, Cui SD, Wang F, Xie YY, Ding SY, Luo MF, Chou CH, Chang NW, Chen KW, Cheng YH, Wan XH, Hsu WL, Lee TY, Wei FX, Huang HD* "miRTarBase 2020: updates to the experimentally validated microRNA-target interaction database" Nucleic Acids Research 2020 Jan 8;48(D1):D148-D154.

Examples

```
data(miRTarBase)
```

---

mm_miR *Mouse microRNA data*

Description

Mouse Fibrosis microRNA data from differential expression analysis. Folic Acid was injected into mouse kidneys to induce fibrosis and nephropathy. Measurements were taken prior to (0 days) and 1, 2, 3, 7 and 14 after injection. The 0 time point was contrasted over other time points. Data was put through limma for analysis. mm_miR consists of DE results from 278 mouse miRs.

The miR names are using TimiRGeN friendly nomenclature. This data is an example of how miR names should be named before input into TimiRGeN.

Column names are in the following format: timepoint(days).DEresult (Log2FC or adjPVal).

Usage

```
data("mm_miR")
```

Format

A data frame of miRNAs put through differential expression with 278 observations in the following 10 variables.

`D1.Log2FC` a numeric vector containing log2FC values of D1_fibrosis/ D0_fibrosis.

`D1.adjPVal` a numeric vector containing adjusted P values of D1_fibrosis/ D0_fibrosis.

`D2.Log2FC` a numeric vector containing log2FC values of D2_fibrosis/ D0_fibrosis.

`D2.adjPVal` a numeric vector containing adjusted P values of D2_fibrosis/ D0_fibrosis.

`D3.Log2FC` a numeric vector containing log2FC values of D3_fibrosis/ D0_fibrosis.

`D3.adjPVal` a numeric vector containing adjusted P values of D3_fibrosis/ D0_fibrosis.

`D7.Log2FC` a numeric vector containing log2FC values of D7_fibrosis/ D0_fibrosis.

`D7.adjPVal` a numeric vector containing adjusted P values of D7_fibrosis/ D0_fibrosis.

`D14.Log2FC` a numeric vector containing log2FC values of D14_fibrosis/ D0_fibrosis.

`D14.adjPVal` a numeric vector containing adjusted P values of D14_fibrosis/ D0_fibrosis.

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE613287

References

M. Pavkovic, L. Pantano, C. V. Gerlach, S. Brutus, S. A. Boswell, R. A. Everley, J. V. Shah, S. H. Sui, and V. S. Vaidya, "Multi omics a nalysis of fibrotic kidneys in two mouse models," Scientific data, vol. 6, no. 1, p. 92, 2019. <https://www.nature.com/articles/s41597-019-0095-5>

Examples

```
data(mm_miR)
## maybe str(mm_miR) ; plot(mm_miR) ...
```

---

mm_mRNA *Mouse microRNA data*

Description

Mouse Fibrosis mRNA data from differential expression analysis. Folic Acid was injected into mouse kidneys to induce fibrosis and nephropathy. Measurements were taken prior to and 1, 2, 3, 7 and 14 days after injection. The 0 time point was contrasted over other time points. Data was put through limma for analysis. mm_mRNA only has DE results from 2000 mRNA genes for speed and size optimisation.

Column names are in the following format: timepoint(days).DEresult (Log2FC or adjPVal).

Usage

```
data("mm_mRNA")
```

Format

A data frame of mRNAs put through differential expression with 2000 observations in the following 10 variables.

`D1.Log2FC` a numeric vector containing log2FC values of D1_fibrosis/ D0_fibrosis.

`D1.adjPVal` a numeric vector containing adjusted P values of D1_fibrosis/ D0_fibrosis.

`D2.Log2FC` a numeric vector containing log2FC values of D2_fibrosis/ D0_fibrosis.

`D2.adjPVal` a numeric vector containing adjusted P values of D2_fibrosis/ D0_fibrosis.

`D3.Log2FC` a numeric vector containing log2FC values of D3_fibrosis/ D0_fibrosis.

`D3.adjPVal` a numeric vector containing adjusted P values of D3_fibrosis/ D0_fibrosis.

`D7.Log2FC` a numeric vector containing log2FC values of D7_fibrosis/ D0_fibrosis.

`D7.adjPVal` a numeric vector containing adjusted P values of D7_fibrosis/ D0_fibrosis.

`D14.Log2FC` a numeric vector containing log2FC values of D14_fibrosis/ D0_fibrosis.

`D14.adjPVal` a numeric vector containing adjusted P values of D14_fibrosis/ D0_fibrosis.

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE613287

References

M. Pavkovic, L. Pantano, C. V. Gerlach, S. Brutus, S. A. Boswell, R. A. Everley, J. V. Shah, S. H. Sui, and V. S. Vaidya, "Multi omics a nalysis of fibrotic kidneys in two mouse models," Scientific data, vol. 6, no. 1, p. 92, 2019. <https://www.nature.com/articles/s41597-019-0095-5>

Examples

```
data(mm_mRNA)
## maybe str(mm_mRNA) ; plot(mm_mRNA) ...
```

`multiReg` *multiReg*

Description

Creates a new matrix containing the gene of interest and each binding partner that it interacts with.

Usage

```
multiReg(MAE, gene_interest, mRNAreg, filt_df, miRNA_exp, mRNA_exp)
```

Arguments

MAE	Input MAE which stores results from multiReg It is recommended to use the MAE which was used in matrixFilter.
gene_interest	Name of gene of interest. Full string required with no spelling mistakes e.g. "mmu-miR-140-5p", "Ffg1", "hsa-miR-29a-5p". If gene of interest is an mRNA, mRNAreg must be TRUE. Otherwise, if the gene of interest is a miRNA, then mRNAreg must be FALSE.
mRNAreg	TRUE or FALSE. Is the gene of interest a mRNA? Default is TRUE.
filt_df	Dataframe from the matrixFilter function.
miRNA_exp	miRNA data from using the diffExpressRes function on miRNA data.
mRNA_exp	mRNA data from using the diffExpressRes function on miRNA data

Value

A matrix which contains the gene of interest and all binding partners. Their values (Log2FC or ave exp) for each time point are also produced.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:100,]

mRNA <- mm_mRNA[1:200,]

MAE <- startObject(miR = miR, mRNA = mRNA)

MAE <- getIdsMir(MAE, assay(MAE, 1), orgDB = org.Mm.eg.db, 'mmu')

MAE <- getIdsMrna(MAE, assay(MAE, 2), "useast", 'mmusculus', )

MAE <- diffExpressRes(MAE, df = assay(MAE, 1), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 3),
                       idColumn = 'GENENAME',
                       name = "miRNA_log2fc")

MAE <- diffExpressRes(MAE, df = assay(MAE, 2), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 7),
                      idColumn = 'GENENAME',
                      name = "mRNA_log2fc")

Filt_df <- data.frame(row.names = c("mmu-miR-145a-3p:Adamts15",
                                    "mmu-miR-146a-5p:Acy1"),
                      corr = c(-0.9191653, 0.7826041),
                      miR = c("mmu-miR-145a-3p", "mmu-miR-146a-5p"),
                      mRNA = c("Adamts15", "Acy1"),
                      miR_Entrez = c(387163, NA),
                      mRNA_Entrez = c(235130, 109652),
                      TargetScan = c(1, 0),
```

```
                    miRDB = c(0, 0),
                    Predicted_Interactions = c(1, 0),
                    miRTarBase = c(0, 1),
                    Pred_Fun = c(1, 1))

MAE <- matrixFilter(MAE, miningMatrix = Filt_df, negativeOnly = FALSE,
                    threshold = 1, predictedOnly = FALSE)

MAE <- multiReg(MAE = MAE, gene_interest = "Adamts15",
                mRNAreg =TRUE, filt_df=MAE[[11]], miRNA_exp=MAE[[9]],
                mRNA_exp=MAE[[10]])
```

---

quickBar *quickBar*

Description

Creates a bar plot which compares the confidence levels for each wikipathway association to the filtered input data. The number of genes in common between a pathway and the input data are taken into account to generate confidence scores. This function is used specifically for a single time point at a time, or if using "s" analysis, the function is used for a single time point within a single gene type (miR or mRNA).

Usage

```
quickBar(X, Y, N)
```

Arguments

X	Dataframes within a list including count information, confidence scores and wikipathway information. This is the output from the enrichWiki function. It will be stored as metadata within the MAE used in the enrichWiki function. Data can be retrieved using [[i]][[j]] on the output of enrichWiki.
Y	String which is associated to the nested dataframe selected for X. This is the output from the enrichWiki function. It will be stored as metadata within the MAE used in the enrichWiki function. Data can be retrieved using names([[i]][j]) on the output of enrichWiki.
N	Integer representing number of pathways to display. Default is 15.

Value

Bar plot showing which pathways are most enriched for genes found at each time point ("c") or at each time point within a gentype ("s").

Examples

```
library(org.Mm.eg.db)

MAE <- MultiAssayExperiment()

metadata(MAE)[["e_list"]] <- e_list_mouse

MAE <- dloadGmt(MAE, species = "Mus musculus")

MAE <- enrichWiki(MAE = MAE, method = 'c', ID_list = metadata(MAE)[[1]],
                  orgDB = org.Mm.eg.db, path_gene = assay(MAE, 1),
                  path_name = assay(MAE, 2), ID = "ENTREZID",
                  universe = assay(MAE, 1)[[2]])

q <- quickBar(X = metadata(MAE)[[2]][[1]], Y = names(metadata(MAE)[[2]][1]))

# to view bar plot enter plot(q)
```

---

quickCrossCorr *quickCrossCorr*

Description

Plots a cross-correlation plot to compare the miRNA and mRNA of a selected pair. This is a useful test of the similarities between the the two time series. It tracks movement of two time series relative to one another to determine how well they match and at which point the best match occurs.

Usage

```
quickCrossCorr(filt_df, pair, miRNA_exp, mRNA_exp, scale,
Interpolation, timecourse)
```

Arguments

filt_df	Dataframe from the matrixFilter function.
pair	Interger representing the pair to be explored.
miRNA_exp	miRNA data from using the diffExpressRes function on miRNA data.
mRNA_exp	mRNA data from using the diffExpressRes function on miRNA data
scale	TRUE or FALSE. Should data be scales. Default is FALSE. If the correlation is based on Log2FC values scale should be TRUE.
Interpolation	TRUE or FALSE. Should the whole time course be interpolated over by a smooth spline? Default is FALSE. This is most useful for longer and regular time courses.
timecourse	If Iterpolation is TRUE, how many time points should be interpolated over?

Value

A cross correlation plot.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:50,]

mRNA <- mm_mRNA[1:100,]

MAE <- startObject(miR = miR, mRNA = mRNA)

MAE <- getIdsMir(MAE, assay(MAE, 1), orgDB = org.Mm.eg.db, 'mmu')

MAE <- getIdsMrna(MAE, assay(MAE, 2), "useast", 'mmusculus', orgDB = org.Mm.eg.db)

MAE <- diffExpressRes(MAE, df = assay(MAE, 1), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 3),
                       idColumn = 'GENENAME',
                       name = "miRNA_log2fc")

MAE <- diffExpressRes(MAE, df = assay(MAE, 2), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 7),
                      idColumn = 'GENENAME',
                      name = "mRNA_log2fc")

Filt_df <- data.frame(row.names = c("mmu-miR-145a-3p:Adamts15",
                                    "mmu-miR-146a-5p:Acy1"),
                      corr = c(-0.9191653, 0.7826041),
                      miR = c("mmu-miR-145a-3p", "mmu-miR-146a-5p"),
                      mRNA = c("Adamts15", "Acy1"),
                      miR_Entrez = c(387163, NA),
                      mRNA_Entrez = c(235130, 109652),
                      TargetScan = c(1, 0),
                      miRDB = c(0, 0),
                      Predicted_Interactions = c(1, 0),
                      miRTarBase = c(0, 1),
                      Pred_Fun = c(1, 1))

MAE <- matrixFilter(MAE, miningMatrix = Filt_df, negativeOnly = FALSE,
                    threshold = 1, predictedOnly = FALSE)

quickCrossCorr(filt_df=MAE[[11]], pair=1, miRNA_exp=MAE[[9]],
               mRNA_exp=MAE[[10]],scale = FALSE, Interpolation = FALSE)
```

---

quickDendro *quickDendro*

Description

Ceates a dendrogram of the genes from the pathway of interest.

Usage

```
quickDendro(filt_df, miRNA_exp, mRNA_exp, distmeth, hclustmeth,
pathwayname)
```

Arguments

filt_df	Dataframe from the matrixFilter function.
miRNA_exp	miRNA data from using the diffExpressRes function on miRNA data.
mRNA_exp	mRNA data from using the diffExpressRes function on miRNA data.
distmeth	Dist method for hierarchical clustering. Default is "maximum".
hclustmeth	Hclust method for hierarchical clustering. Default is "ward.D".
pathwayname	Character which is the name of pathway of interest. Default is "Pathway".

Value

A dendrogram with the genes on the Y axis and the distances on the X axis.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:50,]

mRNA <- mm_mRNA[1:100,]

MAE <- startObject(miR = miR, mRNA = mRNA)

MAE <- getIdsMir(MAE, assay(MAE, 1), orgDB = org.Mm.eg.db, 'mmu')

MAE <- getIdsMrna(MAE, assay(MAE, 2), "useast", 'mmusculus', orgDB = org.Mm.eg.db)

MAE <- diffExpressRes(MAE, df = assay(MAE, 1), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 3),
                       idColumn = 'GENENAME',
                       name = "miRNA_log2fc")

MAE <- diffExpressRes(MAE, df = assay(MAE, 2), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 7),
                      idColumn = 'GENENAME',
                      name = "mRNA_log2fc")

Filt_df <- data.frame(row.names = c("mmu-miR-145a-3p:Adamts15",
                                    "mmu-miR-146a-5p:Acy1"),
                      corr = c(-0.9191653, 0.7826041),
                      miR = c("mmu-miR-145a-3p", "mmu-miR-146a-5p"),
                      mRNA = c("Adamts15", "Acy1"),
```

```
                        miR_Entrez = c(387163, NA),
                        mRNA_Entrez = c(235130, 109652),
                        TargetScan = c(1, 0),
                        miRDB = c(0, 0),
                        Predicted_Interactions = c(1, 0),
                        miRTarBase = c(0, 1),
                        Pred_Fun = c(1, 1))

MAE <- matrixFilter(MAE, miningMatrix = Filt_df, negativeOnly = FALSE,
                    threshold = 1, predictedOnly = FALSE)

quickDendro(filt_df=MAE[[11]], miRNA_exp=MAE[[9]],
            mRNA_exp=MAE[[10]], pathwayname = "Test")
```

---

quickDMap *quickDMap*

Description

Creates a companion heatmap for the dendrogram made by quickDendro.

Usage

```
quickDMap(filt_df, miRNA_exp, mRNA_exp, distmeth, hclustmeth,
pathwayname)
```

Arguments

filt_df	Dataframe from the matrixFilter function.
miRNA_exp	miRNA data from using the diffExpressRes function on miRNA data.
mRNA_exp	mRNA data from using the diffExpressRes function on miRNA data.
distmeth	Dist method for hierarchical clustering. Default is "maximum".
hclustmeth	Hclust method for hierarchical clustering. Default is "ward.D".
pathwayname	Character which is the name of pathway of interest. Default is "Pathway".

Value

A heatmap with time points as the x axis and genes as the y axis. Gene order will be the same as quickDendro.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:50,]

mRNA <- mm_mRNA[1:100,]
```

```
MAE <- startObject(miR = miR, mRNA = mRNA)

MAE <- getIdsMir(MAE, assay(MAE, 1), orgDB = org.Mm.eg.db, 'mmu')

MAE <- getIdsMrna(MAE, assay(MAE, 2), "useast", 'mmusculus', orgDB = org.Mm.eg.db)

MAE <- diffExpressRes(MAE, df = assay(MAE, 1), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 3),
                       idColumn = 'GENENAME',
                       name = "miRNA_log2fc")

MAE <- diffExpressRes(MAE, df = assay(MAE, 2), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 7),
                      idColumn = 'GENENAME',
                      name = "mRNA_log2fc")

Filt_df <- data.frame(row.names = c("mmu-miR-145a-3p:Adamts15",
                                    "mmu-miR-146a-5p:Acy1"),
                      corr = c(-0.9191653, 0.7826041),
                      miR = c("mmu-miR-145a-3p", "mmu-miR-146a-5p"),
                      mRNA = c("Adamts15", "Acy1"),
                      miR_Entrez = c(387163, NA),
                      mRNA_Entrez = c(235130, 109652),
                      TargetScan = c(1, 0),
                      miRDB = c(0, 0),
                      Predicted_Interactions = c(1, 0),
                      miRTarBase = c(0, 1),
                      Pred_Fun = c(1, 1))

MAE <- matrixFilter(MAE, miningMatrix = Filt_df, negativeOnly = FALSE,
                    threshold = 1, predictedOnly = FALSE)

quickDendro(filt_df=MAE[[11]], miRNA_exp=MAE[[9]],
            mRNA_exp=MAE[[10]], pathwayname = "Test")

quickDMap(filt_df=MAE[[11]], miRNA_exp=MAE[[9]],
          mRNA_exp=MAE[[10]], pathwayname = "Test")
```

---

quickFuzz *quickFuzz*

Description

Plots fuzzy clusters. Each different cluster created will represent a different temporal behaviour. Depending on the data, more or fewer cluster may be appropriate. Use clusterCheck to influence this decision before moving onto quickFuzz. Each line in a cluster represents a pathway. Pathways are divided by colour. The more intense the colour of a line, the stronger they fit a particular cluster / temporal behaviour. Fuzzy clustering is a soft clustering approach where objects are not divided into fixed clusters. Each pathway can exist in each cluster but each pathway will differ on the degree to which they fit to each cluster. Look into the clusterData dataframe created by createClusters to see this. If a cluster peaks interest, continue to analysis of that cluster with the returnCluster function.

Usage

```
quickFuzz(Mfuzzdata, Clusters, W, background, labelcol, axiscol,
axisline, subcol, ylab)
```

Arguments

Mfuzzdata	A large ExpressionSet object which contain fuzzy clustering data. This is output from the createClusters function. The Expressionset object should be stored as an experiment in the MAE used in the createClusters function.
Clusters	A large list containing information about clusters, statistics and phenodata. This is output from the createClusters function. The list should be stored as metadata in the MAE used in the createClusters function.
W	TRUE or FALSE? Should the plot be shown in a new window? Default is TRUE.
background	Plot background colour. Default is black.
labelcol	Plot labels colour. Default is yellow.
axiscol	Plot axis labels colour. Default is white.
axisline	Plot axis line colour. Default is white.
subcol	Plot sub title colour. Default is yellow.
ylab	y axis label. Default is "Genes found in data and pathway".

Value

A plot of different clusters showing how the number of genes found to be significant varies between the input data and wikipathways. These variations are captured as temporal behaviours and are clustered.

Examples

```
MAE <- MultiAssayExperiment()

metadata(MAE)[["e_list"]] <- e_list_mouse

metadata(MAE)[["w_list"]] <- w_list_mouse[1:10]

MAE <- wikiMatrix(MAE, ID_list = metadata(MAE)[[1]],
                  wp_list = metadata(MAE)[[2]])

MAE <- turnPercent(MAE = MAE,
                   wikiMatrix = assay(MAE, 1))

MAE <- createClusters(MAE, method = "c",
                      percentMatrix = assay(MAE, 2),
                      noClusters = 2, variance = 0.99)

quickFuzz(Mfuzzdata = experiments(MAE)[[4]],
          Clusters = metadata(MAE)[[3]], W = FALSE)
```

quickHClust *quickHClust*

Description

Plots all the genes found in a particular cluster. The plots will contain the data (gray) and a smoothed line (red).

Usage

```
quickHClust(filt_df, miRNA_exp, mRNA_exp, distmeth,hclustmeth,
pathwayname, k, cluster)
```

Arguments

filt_df	Dataframe from the matrixFilter function.
miRNA_exp	miRNA data from using the diffExpressRes function on miRNA data.
mRNA_exp	mRNA data from using the diffExpressRes function on miRNA data.
distmeth	Dist method for hierarchical clustering. Default is "maximum".
hclustmeth	Hclust method for hierarchical clustering. Default is "ward.D".
pathwayname	Character which is the name of pathway of interest. Default is "Pathway".
k	Integer. Number of clusters.
cluster	Integer. Which cluster to look into? Default is 1.

Value

Time course plots of each gene found in the cluster of interest, from the pathway of interest.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:50,]

mRNA <- mm_mRNA[1:100,]

MAE <- startObject(miR = miR, mRNA = mRNA)

MAE <- getIdsMir(MAE, assay(MAE, 1), orgDB = org.Mm.eg.db, 'mmu')

MAE <- getIdsMrna(MAE, assay(MAE, 2), "useast", 'mmusculus', orgDB = org.Mm.eg.db)

MAE <- diffExpressRes(MAE, df = assay(MAE, 1), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 3),
                       idColumn = 'GENENAME',
                       name = "miRNA_log2fc")
```

```
MAE <- diffExpressRes(MAE, df = assay(MAE, 2), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 7),
                      idColumn = 'GENENAME',
                      name = "mRNA_log2fc")

Filt_df <- data.frame(row.names = c("mmu-miR-145a-3p:Adamts15",
                                    "mmu-miR-146a-5p:Acy1"),
                      corr = c(-0.9191653, 0.7826041),
                      miR = c("mmu-miR-145a-3p", "mmu-miR-146a-5p"),
                      mRNA = c("Adamts15", "Acy1"),
                      miR_Entrez = c(387163, NA),
                      mRNA_Entrez = c(235130, 109652),
                      TargetScan = c(1, 0),
                      miRDB = c(0, 0),
                      Predicted_Interactions = c(1, 0),
                      miRTarBase = c(0, 1),
                      Pred_Fun = c(1, 1))

MAE <- matrixFilter(MAE, miningMatrix = Filt_df, negativeOnly = FALSE,
                    threshold = 1, predictedOnly = FALSE)

quickHClust(filt_df=MAE[[11]], miRNA_exp=MAE[[9]],
            mRNA_exp=MAE[[10]], pathwayname = "Test", k = 2, cluster = 1)
```

---

quickMap *quickMap*

Description

Generates a heatmap of all miRNA:mRNA binding pairs that have been filtered. Pairs are ranked by decreasing correlation.

Usage

```
quickMap(filt_df, numpairs)
```

Arguments

filt_df	Dataframe from the matrixFilter function.
numpairs	Number of pairs to plot. Must be an integer more than 1.

Value

Heatmap of miRNA-mRNA pairs.

Examples

```
Filt_df <- data.frame(row.names = c("mmu-miR-145a-3p:Adamts15",
                                    "mmu-miR-146a-5p:Acy1"),
                      corr = c(-0.9191653, -0.7826041),
                      miR = c("mmu-miR-145a-3p", "mmu-miR-146a-5p"),
                      mRNA = c("Adamts15", "Acy1"),
                      miR_Entrez = c(387163, NA),
                      mRNA_Entrez = c(235130, 109652),
                      TargetScan = c(1, 0),
                      miRDB = c(0, 0),
                      Predicted_Interactions = c(1, 0),
                      miRTarBase = c(0, 1),
                      Pred_Fun = c(1, 1))

MAE <- MultiAssayExperiment()

MAE <- matrixFilter(MAE, miningMatrix = Filt_df, negativeOnly = FALSE,
                    threshold = 1, predictedOnly = FALSE)

quickMap(filt_df = MAE[[1]], numpairs = 2)
```

---

quickNet *quickNet*

Description

Generates an igraph network representing the miR-mRNA interactions filtered from the input data using pathway analysis and database filtering. Pink nodes are miRs and light blue nodes are mRNAs. Edges are coloured by the average correlations. Note, plot window size may need to be adjusted to accommodate image.

Usage

```
quickNet(net, pathwayname)
```

Arguments

net	List generated by makeNet function. If you have fewer than two interactions the plot will not work. Output from makeNet should be stored as metadata within the MAE used in the makeNet function.
pathwayname	Character which is the name of pathway of interest. Default is "Pathway".

Value

A network depicting filtered miR-mRNA interactions for a specific wikipathway of interest.

Examples

```
Filt_df <- data.frame(row.names = c("mmu-miR-320-3p:Acss1",
                                    "mmu-miR-27a-3p:Odc1"),
                      corr = c(-0.9191653, 0.7826041),
                      miR = c("mmu-miR-320-3p", "mmu-miR-27a-3p"),
                      mRNA = c("Acss1", "Acss1"),
                      miR_Entrez = c(NA, NA),
                      mRNA_Entrez = c(68738, 18263),
                      TargetScan = c(1, 0),
                      miRDB = c(0, 0),
                      Predicted_Interactions = c(1, 0),
                      miRTarBase = c(0, 1),
                      Pred_Fun = c(1, 1))

MAE <- MultiAssayExperiment()

MAE <- makeNet(MAE, Filt_df)

quickNet(metadata(MAE)[[1]])
```

`quickPathwayTC` *quickPathwayTC*

Description

Plots time series data from the pathway of interest. Genes which pass a user defined threshold will be highlighted.

Usage

```
quickPathwayTC(filt_df, miRNA_exp, mRNA_exp, morethan, threshold,
 pathwayname)
```

Arguments

`filt_df`	Dataframe from the matrixFilter function.
`miRNA_exp`	miRNA data from using the diffExpressRes function on miRNA data.
`mRNA_exp`	mRNA data from using the diffExpressRes function on miRNA data.
`morethan`	TRUE or FALSE. Default is TRUE.
`threshold`	Integer that is user defined. Default is 1.
`pathwayname`	Character which is the name of pathway of interest. Default is "Pathway".

Value

Line plot of all the genes of interest from a pathway of interest.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:100,]

mRNA <- mm_mRNA[1:200,]

MAE <- startObject(miR = miR, mRNA = mRNA)

MAE <- getIdsMir(MAE, assay(MAE, 1), orgDB = org.Mm.eg.db, 'mmu')

MAE <- getIdsMrna(MAE, assay(MAE, 2), "useast", 'mmusculus', orgDB = org.Mm.eg.db)

MAE <- diffExpressRes(MAE, df = assay(MAE, 1), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 3),
                       idColumn = 'GENENAME',
                       name = "miRNA_log2fc")

MAE <- diffExpressRes(MAE, df = assay(MAE, 2), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 7),
                      idColumn = 'GENENAME',
                      name = "mRNA_log2fc")

Filt_df <- data.frame(row.names = c("mmu-miR-145a-3p:Adamts15",
                                    "mmu-miR-146a-5p:Acy1"),
                      corr = c(-0.9191653, 0.7826041),
                      miR = c("mmu-miR-145a-3p", "mmu-miR-146a-5p"),
                      mRNA = c("Adamts15", "Acy1"),
                      miR_Entrez = c(387163, NA),
                      mRNA_Entrez = c(235130, 109652),
                      TargetScan = c(1, 0),
                      miRDB = c(0, 0),
                      Predicted_Interactions = c(1, 0),
                      miRTarBase = c(0, 1),
                      Pred_Fun = c(1, 1))

MAE <- matrixFilter(MAE, miningMatrix = Filt_df, negativeOnly = FALSE,
                    threshold = 1, predictedOnly = FALSE)

quickPathwayTC(filt_df=MAE[[11]], miRNA_exp=MAE[[9]],
               mRNA_exp=MAE[[10]], morethan = TRUE, threshold =1,
               pathwayname = "Test")
```

---

quickReg *quickReg*

Description

quickReg will produce a regression plot between a miRNA:mRNA pair. An abline wil be generated. Odds-ratio will be calculated from the coefficient. Confident intervals at 95 coefficient also. This is

a useful method to identify the relationship between a miRNA:mRNA pair over time. The higher the odds-ratio the greater the correlation between the two time series. The smaller the 95 intervals range, the more confident we can be about plotting the abline.

Usage

```
quickReg(reg_df, colselect)
```

Arguments

reg_df	Matrix/assay which which was generated by the multiReg function. Should be in the MAE used during the multiReg function.
colselect	Integer which represents a column within reg_df containing mRNA/ miRNA which the gene of interest (column 2) is to be contrasted agianst.

Value

Regression plot between a miRNA:mRNA pair.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:50,]

mRNA <- mm_mRNA[1:100,]

MAE <- startObject(miR = miR, mRNA = mRNA)

MAE <- getIdsMir(MAE, assay(MAE, 1), orgDB = org.Mm.eg.db, 'mmu')

MAE <- getIdsMrna(MAE, assay(MAE, 2), "useast", 'mmusculus', orgDB = org.Mm.eg.db)

MAE <- diffExpressRes(MAE, df = assay(MAE, 1), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 3),
                       idColumn = 'GENENAME',
                       name = "miRNA_log2fc")

MAE <- diffExpressRes(MAE, df = assay(MAE, 2), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 7),
                      idColumn = 'GENENAME',
                      name = "mRNA_log2fc")

Filt_df <- data.frame(row.names = c("mmu-miR-145a-3p:Adamts15",
                                    "mmu-miR-146a-5p:Acy1"),
                      corr = c(-0.9191653, 0.7826041),
                      miR = c("mmu-miR-145a-3p", "mmu-miR-146a-5p"),
                      mRNA = c("Adamts15", "Acy1"),
                      miR_Entrez = c(387163, NA),
                      mRNA_Entrez = c(235130, 109652),
                      TargetScan = c(1, 0),
                      miRDB = c(0, 0),
```

```
                    Predicted_Interactions = c(1, 0),
                    miRTarBase = c(0, 1),
                    Pred_Fun = c(1, 1))

MAE <- matrixFilter(MAE, miningMatrix = Filt_df, negativeOnly = FALSE,
                  threshold = 1, predictedOnly = FALSE)

MAE <- multiReg(MAE = MAE, gene_interest = "Adamts15",
                mRNAreg =TRUE, filt_df=MAE[[11]], miRNA_exp=MAE[[9]],
                mRNA_exp=MAE[[10]])

model1 <- linearRegr(mreg = MAE[[12]], colselect =2, colpair =3)

summary(model1$regression)

quickReg(reg_df = MAE[[12]], colselect = 3)
```

---

quickTC *quickTC*

Description

Plots miRNA:mRNA pair over timecourse.

Usage

```
quickTC(filt_df, pair, miRNA_exp, mRNA_exp, scale,Interpolation,
timecourse)
```

Arguments

filt_df	Dataframe from the matrixFilter function.
pair	Interger representing the pair to be explored.
miRNA_exp	miRNA data from using the diffExpressRes function on miRNA data.
mRNA_exp	mRNA data from using the diffExpressRes function on miRNA data
scale	TRUE or FALSE. Should data be scales. Default is FALSE.
Interpolation	TRUE or FALSE. Should the whole time course be interpolated over by a smooth spline? Default is FALSE. This is most useful for longer time courses.
timecourse	If Iterpolation is TRUE, how many time points should be interpolated over?

Value

Time course plot of selected pair.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:50,]

mRNA <- mm_mRNA[1:100,]

MAE <- startObject(miR = miR, mRNA = mRNA)

MAE <- getIdsMir(MAE, assay(MAE, 1), orgDB = org.Mm.eg.db, 'mmu')

MAE <- getIdsMrna(MAE, assay(MAE, 2), "useast", 'mmusculus', orgDB = org.Mm.eg.db)

MAE <- diffExpressRes(MAE, df = assay(MAE, 1), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 3),
                       idColumn = 'GENENAME',
                       name = "miRNA_log2fc")

MAE <- diffExpressRes(MAE, df = assay(MAE, 2), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 7),
                      idColumn = 'GENENAME',
                      name = "mRNA_log2fc")

Filt_df <- data.frame(row.names = c("mmu-miR-145a-3p:Adamts15",
                                    "mmu-miR-146a-5p:Acy1"),
                      corr = c(-0.9191653, 0.7826041),
                      miR = c("mmu-miR-145a-3p", "mmu-miR-146a-5p"),
                      mRNA = c("Adamts15", "Acy1"),
                      miR_Entrez = c(387163, NA),
                      mRNA_Entrez = c(235130, 109652),
                      TargetScan = c(1, 0),
                      miRDB = c(0, 0),
                      Predicted_Interactions = c(1, 0),
                      miRTarBase = c(0, 1),
                      Pred_Fun = c(1, 1))

MAE <- matrixFilter(MAE, miningMatrix = Filt_df, negativeOnly = FALSE,
                    threshold = 1, predictedOnly = FALSE)

quickTC(filt_df=MAE[[11]], pair=1, miRNA_exp=MAE[[9]],
        mRNA_exp=MAE[[10]], scale = FALSE)
```

---

quickTCPred *quickTCPred*

Description

Creates a regression plot over the time course, which compares data and a simulation which was predicted using the data. Data is based on the model formula used in the multiReg function and linearRegr function. R.squared and p value are also calculated and pasted into the plot.

Usage

```
quickTCPred(model, reg_df)
```

Arguments

model	Linear model which was generated by the linearRegr function.
reg_df	Matrix/assay which which was generated by the multiReg function. Should be in the MAE used during the multiReg function.

Value

A linear regression plot with data and a simulation.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:50,]

mRNA <- mm_mRNA[1:100,]

MAE <- startObject(miR = miR, mRNA = mRNA)

MAE <- getIdsMir(MAE, assay(MAE, 1), orgDB = org.Mm.eg.db, 'mmu')

MAE <- getIdsMrna(MAE, assay(MAE, 2), "useast", 'mmusculus', orgDB = org.Mm.eg.db)

MAE <- diffExpressRes(MAE, df = assay(MAE, 1), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 3),
                       idColumn = 'GENENAME',
                       name = "miRNA_log2fc")

MAE <- diffExpressRes(MAE, df = assay(MAE, 2), dataType = 'Log2FC',
                      genes_ID = assay(MAE, 7),
                      idColumn = 'GENENAME',
                      name = "mRNA_log2fc")

Filt_df <- data.frame(row.names = c("mmu-miR-145a-3p:Adamts15",
                                    "mmu-miR-146a-5p:Acy1"),
                      corr = c(-0.9191653, 0.7826041),
                      miR = c("mmu-miR-145a-3p", "mmu-miR-146a-5p"),
                      mRNA = c("Adamts15", "Acy1"),
                      miR_Entrez = c(387163, NA),
                      mRNA_Entrez = c(235130, 109652),
                      TargetScan = c(1, 0),
                      miRDB = c(0, 0),
                      Predicted_Interactions = c(1, 0),
                      miRTarBase = c(0, 1),
                      Pred_Fun = c(1, 1))

MAE <- matrixFilter(MAE, miningMatrix = Filt_df, negativeOnly = FALSE,
```

```
                    threshold = 1, predictedOnly = FALSE)

MAE <- multiReg(MAE = MAE, gene_interest = "Adamts15",
                mRNAreg =TRUE, filt_df=MAE[[11]], miRNA_exp=MAE[[9]],
                mRNA_exp=MAE[[10]])

model1 <- linearRegr(mreg = MAE[[12]], colselect =2, colpair =3)

summary(model1$regression)

quickTCPred(model = model1, reg_df = MAE[[12]])
```

---

reduceWiki *reduceWiki*

Description

Returns all gene IDs of a single wikipathway of interest. This function is recommended to be used after a signalling pathway of interest is found.

Usage

```
reduceWiki(MAE, path_data, stringWiki = '')
```

Arguments

MAE	MultiAssayExperiment which will store the results of reduceWiki. It is recommended to use the same MAE which stores the data from dloadGmt/ gmtEnsembl.
path_data	Dataframe with wikipathway IDs, gene IDs and pathway names from either the dloadGmt or gmtEnsembl functions. These will be found as assays within the MAE used in the dloadGmt or gmtEnsembl functions.
stringWiki	Full name of the wikipathway of interest. Make sure to spell this correctly. Each wikipathway can only be added once to the same MAE object.

Value

A dataframe that only contains information about the wikipathway of interest. Output will be stored as an assay in the input MAE.

Examples

```
MAE <- MultiAssayExperiment(list(path_data = data.frame(
                        "wpid" = c(rep("WP571", 6)),
                        "gene" = c(16175, 12370,26419, 19249, 19645, 18479),
                      "name" = c(rep("Fas pathway and Stress induction of HSP regulation",
                        6)))))

MAE <- reduceWiki(MAE, path_data = assay(MAE, 1),
                  stringWiki = 'Fas pathway and Stress induction of HSP regulation')
```

returnCluster *returnCluster*

Description

Retrieves information about which wikipathways fitted best to a specific cluster. This function is to be used after quickFuzz.

Usage

```
returnCluster(MAE, clusterData, whichCluster, fitCluster)
```

Arguments

MAE	MultiAssayExperiment which will store the output from returnCluster. It is recommended to use the same MAE which stores output from the createClusters function.
clusterData	A dataframe which contains cluster-pathway fit scores and is stored as an assay within the MAE used in the createClusters function.
whichCluster	Integer which should corresponds to the cluster of interest.
fitCluster	Integer from 0-1. How well should the pathways fit into the selected cluster? Default is 0.99.

Value

A dataframe that contains information about the pathways that corresponded best with the chosen cluster. Output will be stored as an assay in the input MAE.

Examples

```
MAE <- MultiAssayExperiment()

metadata(MAE)[["e_list"]] <- e_list_mouse

metadata(MAE)[["w_list"]] <- w_list_mouse[1:10]

MAE <- wikiMatrix(MAE, ID_list = metadata(MAE)[[1]],
                  wp_list = metadata(MAE)[[2]])

MAE <- turnPercent(MAE = MAE,
                   wikiMatrix = assay(MAE, 1))

MAE <- createClusters(MAE, method = "c",
                    percentMatrix = assay(MAE, 2),
                    noClusters = 2, variance = 0.99)

MAE <- returnCluster(MAE, clusterData = assay(MAE, 3), whichCluster = 1,
                     fitCluster = 0.5)
```

savePlots *savePlots*

Description

Saves all plots from enrichWiki into the current working directory.

Usage

```
savePlots(largeList, maxInt, fileType = '', width, height)
```

Arguments

largeList	A large list containing GSEA results. This should be stored as metadata within the MAE used in the enrichWiki function.
maxInt	Integer, number of samples in data set.
fileType	Type of file for images to be exported as: "png", "tiff", "svg" or "jpeg".
width	= Width of plots in inches. Default is 22 inches.
height	= Heightof plots in inches. Default is 10 inches.

Value

Saves plots in working directory. Each sample (e.g. time point) will have a separate plot.

significantVals *significantVals*

Description

Filters out genes in each nested dataframe which are not deemed significantly differentially expressed. Each sample will be filtered independently.

Usage

```
significantVals(MAE, method = '', geneList, maxVal, stringVal = '')
```

Arguments

MAE	MultiAssayExperiment to store the output of significantVals. It is recommended to use the MAE used in the genesList.
method	Either "c" or "s", respectively for combined or separated analysis.
geneList	A list of nested dataframes if "c" analysis is used or a list of lists of nested dataframes if "s" is used. This will be the output from them genesList function. The resulting list will be found as metadata, in the MAE used in the genesList function.

maxVal	Numeric value which represents the maximum cut off value for significance e.g. 0.05.
stringVal	Character. Common DE result type which is found in all nested dataframes. This will be used for filtration e.g. pval, adjPval or qval. Make sure the spelling matches the colnames for each sample.

Value

A list of dataframes with only significantly differentially expressed genes. Output will be stored as metadata within the input MAE.

Examples

```
miR <- mm_miR[1:50,]

mRNA <- mm_mRNA[1:100,]

Data <- startObject(miR = mm_miR, mRNA = mm_mRNA)

Data <- combineGenes(MAE = Data, miR_data = assay(Data, 1),
                     mRNA_data = assay(Data, 2))

Data <- genesList(MAE = Data, method = 'c', genetic_data = assay(Data, 3),
                  timeString = 'D')

Data <- significantVals(MAE = Data, method = 'c',
                        geneList = metadata(Data)[[1]],
                        maxVal = 0.05, stringVal = "adjPVal")
```

`startObject` *startObject*

Description

Creates a MultiAssayExperiment (MAE) from miR and mRNA dataframes. MAE's will be the constant object used throughout TimiRGeN. The input dataframes should contain rows as genes, and results from differential expression (DE) as columns. Columns should also indicate the time point related to each sample. Row names and column names must adhere to TimiRGeN friendly nomenclature. Please do read the vignette for a full description of the required nomenclature.

Usage

```
startObject(miR, mRNA)
```

Arguments

miR	microRNA dataframe/ matrix. Rows should be miR gene names which use the TimiRGeN friendly naming system. Columns should be results of DE and time points.
mRNA	mRNA dataframe/ matrix. Rows should be mRNA gene names. Columns should be results of DE and time points.

Value

MultiAssayExperiment containing miR and mRNA data stored as assays. Use assays(MAE, i) or MAE[[i]] to access assays. Use metadata(MAE)[[i]] to access metadata. Use experiments(MAE)[[i]] to access experiments.

Examples

```
data(mm_miR)

data(mm_mRNA)

Data <- startObject(miR = mm_miR, mRNA = mm_mRNA)
```

---

turnPercent *turnPercent*

Description

Genes found in common between the input data and each pathway are normalised by percentages. This is to normalise for pathway size.

Usage

```
turnPercent(MAE, wikiMatrix)
```

Arguments

MAE	MultiAssayExperiment which will store the output from turnPercent. It is recommended to use the MAE which stores output from the wikiMatrix function.
wikiMatrix	Numeric matrix of wikipathways and samples. This should be stored as an assay within the MAE used in the wikiMatrix function.

Value

A percentage matrix which contrasts genes found in pathways and samples. Output will be stored as an assay within the input MAE.

Examples

```
MAE <- MultiAssayExperiment()

metadata(MAE)[["e_list"]] <- e_list_mouse

metadata(MAE)[["w_list"]] <- w_list_mouse[1:10]

MAE <- wikiMatrix(MAE, ID_list = metadata(MAE)[[1]],
                  wp_list = metadata(MAE)[[2]])

MAE <- turnPercent(MAE = MAE, wikiMatrix = assay(MAE, 1))
```

UUO_data *UUO_data*

Description

An alternative mouse kidney fibrosis dataset which is used in section 6 of the vignette to demonstrate meta-analysis with TimiRGeN. UUO_data contains 240 miRNAs and 1874 mRNAs, all of which can also be found in mm_mRNA and mm_miR data files. This subset was loaded into the package for speed. Log2FC and adjusted P values for each gene is found. Differential expression was conducted on a pair-wise manor (D3/D0, D7/D0 and D14/D0). Unlike the folic acid data (mm_miR and mm_mRNA), the UUO data did not include time points D1 and D2.

Usage

```
data("UUO_data")
```

Format

A data frame with 2114 observations on the following 6 variables.

`D3.Log2FC` a numeric vector of log2fc values from D3/D0. Contains miRNA and mRNA data.

`D3.ajdPVal` a numeric vector of adj p values values from D3/D0. Contains miRNA and mRNA data.

`D7.Log2FC` a numeric vector of log2fc values from D7/D0. Contains miRNA and mRNA data.

`D7.ajdPVal` a numeric vector of adj p values values from D7/D0. Contains miRNA and mRNA data.

`D14.Log2FC` a numeric vector of log2fc values from D14/D0. Contains miRNA and mRNA data.

`D14.ajdPVal` a numeric vector of adj p values values from D14/D0. Contains miRNA and mRNA data.

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE118341

References

Pavkovic M, Pantano L, Gerlach CV, Brutus S et al. Multi omics analysis of fibrotic kidneys in two mouse models. Sci Data 2019 Jun 14;6(1):92.

Examples

```
data(UUO_data)
UUO_miRNA <- UUO_data[c(1:240),]
UUO_mRNA <- UUO_data[-c(1:240),]
```

`wikiList` *wikiList*

Description

Provides a list of wikipathways specific for a species, and each gene found to be within each pathway. wikiList will download and process a large amount of data from the wikipathways website so this may take some time to complete.

Usage

```
wikiList(MAE, stringSpecies = '', stringSymbol = '')
```

Arguments

`MAE`	MultiAssayExperiment which will store the results from wikiList. It is recommended to use the same MAE which stores output from the dloadGmt function.
`stringSpecies`	Full species name to decide which wikipathways to download. 'Homo sapiens' to download human pathways or 'Mus musculus' to download mouse pathways.
`stringSymbol`	Type of gene ID to retrieve e.g. 'En' for ensemble gene IDs or 'L' for entrezgene IDs.

Value

List of wikipathways and associated genes saved as as strings. Output will be stored as metadata in the input MAE.

wikiMatrix *wikiMatrix*

Description

Creates a matrix which shows how many genes from the input mRNA and miRNA data are found in each wikipathway, for a specific species.

Usage

```
wikiMatrix(MAE, ID_list, wp_list)
```

Arguments

MAE	MultiAssayExperiment which will store the output of wikiMatrix. It is recommended that the MAE object which stores output from the wikiList function.
ID_list	List of lists of entrez gene IDs or ensembl gene IDs stored as strings. This is the output of the eNames function and is stored as metadata in the MAE used in the eNames function.
wp_list	List of lists containing wikipathways with entrez gene IDs or ensembl IDs as strings. This is the output of the wikiList function and is stored as metadata in the MAE used in the wikiList function.

Value

A matrix showing which genes are found in each time points for each pathway. Output will be stored as an assay in the input MAE.

Examples

```
MAE <- MultiAssayExperiment()

metadata(MAE)[["ID_list"]] <- e_list_mouse

metadata(MAE)[["w_list"]] <- w_list_mouse[1:10]

MAE <- wikiMatrix(MAE, ID_list = metadata(MAE)[[1]],
                  wp_list = metadata(MAE)[[2]])
```

wikiMrna *wikiMrna*

Description

Identify genes that are in common in both the wikipathway of interest and the significantly differentially expressed input mRNAs.

Usage

```
wikiMrna(MAE, mRNA_express, singleWiki, stringWiki='')
```

Arguments

MAE	MultiAssayExperiment which will store the results of wikiMrna. It is recommended to use the same MAE which stores output from the diffExpressRes and reduceWiki functions.
mRNA_express	Dataframe from the diffExpressRes function used on the input mRNA data. This should be found as an assay within the MAE used in the diffExpressRes function.
singleWiki	Dataframe containing information about only one pathway. This is output from the reduceWiki function. This should be found as an assay within the MAE used in the reduceWiki function.
stringWiki	Name of the pathway of interest. Should be the same as the stringWiki parameter from the reduceWiki function.

Value

A dataframe which only contains mRNAs which are found in both the input data and the wikipathway of interest. Output will be stored as an assay in the input MAE.

Examples

```
library(org.Mm.eg.db)

miR <- mm_miR[1:200,]

mRNA <- mm_mRNA[401:600,]

MAE <- startObject(miR = miR, mRNA = mRNA)

MAE <- getIdsMir(MAE, assay(MAE, 1), orgDB = org.Mm.eg.db, 'mmu')

MAE <- getIdsMrna(MAE, assay(MAE, 2), "useast", 'mmusculus', org.Mm.eg.db)

MAE <- dloadGmt(MAE = MAE, species = "Mus musculus")

MAE <- reduceWiki(MAE, path_data = assay(MAE, 11),
```

```
                    stringWiki = 'TGF Beta Signaling Pathway')

MAE <- diffExpressRes(MAE, df = assay(MAE, 2), dataType = 'Log2FC',
                genes_ID = assay(MAE, 7),
                idColumn = 'GENENAME',
                name = "mRNA_log")

MAE <- wikiMrna(MAE, mRNA_express = assay(MAE, 13),
                 singleWiki = assay(MAE, 12),
                 stringWiki = 'TGF Beta Signaling Pathway')
```

`w_list_mouse` *Wikipathways lists for mouse produced by the wikiList function.*

Description

List of entrezIDs associated with each wikipathway made for mouse. This dataset will allow for functions to run faster during CMD check and vignette building. This is used instead of the wikiList function in several instances in TimiRGeN because the wikiList takes a lot of time to download data. For example in the vignette and in several function examples. To reproduce the output enter the following code. >MAE <- MultiAssayExperiment >MAE2 <- wikiList(MAE, stringSpecies = 'Mus musculus', stringSymbol = 'L')

Usage

```
data("w_list_mouse")
```

Format

List of every mouse wikipathway. Each wikipathway will have associated genes stored as entrezIDs.

Details

A Large list of each wikipathway and associated enterzgene IDs.

Source

https://www.wikipathways.org/index.php/Download_Pathways

References

Denise N Slenter et al. "WikiPathways: a multifaceted pathway database b ridging metabolomics to other omics research". In: Nucleic acids research 46.D1 (2017), pp. D661–D667.

Examples

```
data(w_list_mouse) -> w_list_mouse
```

Index