Package ‘MOMA’

May 30, 2024

Title Multi Omic Master Regulator Analysis

Version 1.16.0

Description This package implements the inference of candidate master regulator proteins from multi-omics’ data (MOMA) algorithm, as well as ancillary analysis and visualization functions.

Depends R (>= 4.0)

License GPL-3

Encoding UTF-8

LazyData true

BugReports https://github.com/califano-lab/MOMA/issues

RoxygenNote 7.1.0

biocViews Software, NetworkEnrichment, NetworkInference, Network, FeatureExtraction, Clustering, FunctionalGenomics, Transcriptomics, SystemsBiology

Imports circlize, cluster, ComplexHeatmap, dplyr, ggplot2, graphics, grid, grDevices, magrittr, methods, MKmisc, MultiAssayExperiment, parallel, qvalue, RColorBrewer, readr, reshape2, rlang, stats, stringr, tibble, tidyr, utils

Suggests BiocStyle, knitr, rmarkdown, testthat, viper

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/MOMA
git_branch RELEASE_3_19
git_last_commit 95f66f63
git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-29

Author Evan Paull [aut], Sunny Jones [aut, cre], Mariano Alvarez [aut]

Maintainer Sunny Jones <sunnyjjones@gmail.com>
Contents

areaEnrich .................................................. 3
associateEvents ............................................. 3
checkGeneMap ............................................... 4
checkList .................................................... 5
checkMAE ..................................................... 5
checkPathways ............................................... 6
clusterRange ............................................... 6
clusterReliability .......................................... 7
cnvScoreStouffer .......................................... 8
conditionalModel .......................................... 8
conditionalP ................................................ 9
empiricalP .................................................. 9
elementary.gbm.mae ......................................... 10
fitCurvePercent ............................................ 10
gbm.pathways .............................................. 11
gene.map .................................................... 11
genomicPlotSmall .......................................... 12
getCoverage ................................................ 12
getDataFrame ............................................... 13
getDiggitEmpiricalQvalues ............................... 14
getEmpiricalQvals ......................................... 14
getPvalsMatrix ............................................. 15
getSubtypeEventTables .................................... 15
integrateFunction .......................................... 16
integrateTZ ................................................ 16
makeCoverageDf ........................................... 17
makeSaturationPlots ....................................... 17
mapEntrez .................................................. 18
mapHugo ..................................................... 19
mapScoresCnvBand ......................................... 19
mergeData ................................................... 20
mergeDataBySubtype ....................................... 21
mergeGenomicSaturation ................................... 21
mergeLists ................................................... 22
Moma-class .................................................. 22
MomaConstructor ............................................ 23
mutSig ....................................................... 24
oncoprintPlot .............................................. 25
pathwayDiggitIntersect .................................... 26
plotEvents ................................................... 26
rea .......................................................... 27
reaNULL ...................................................... 28
sampleNameFilter .......................................... 28
sampleOverlap ............................................. 29
sigInteractorsDIGGIT ...................................... 30
sREA ........................................................ 30
areaEnrich  

stoufferIntegrate  .................................................. 31  
stoufferIntegrateDiggit ............................................. 31  
subsetListInteractions ............................................... 32  
validDiggitInteractions .............................................. 32  
viperGetSigTFS .......................................................... 33  
viperGetTFScores ....................................................... 33  

Index  35  

areaEnrich  

*area.enrich* Compute aREA enrichment between all pairwise combinations of VIPER proteins and gene-level events

Description

areaEnrich Compute aREA enrichment between all pairwise combinations of VIPER proteins and gene-level events

Usage

areaEnrich(events.mat, vipermat, event.type, verbose)

Arguments

- **events.mat**  
  A Binary 0/1 matrix with columns as samples, and rows as proteins
- **vipermat**  
  A VIPER network of inferred activity scores with columns as samples, and rows as proteins
- **event.type**  
  Name of the event type for printing purposes
- **verbose**  
  whether to print extra progress statements

Value

A matrix of enrichment scores with rows as event/gene names and columns as VIPER protein names

associateEvents  

Use 'aREA' to calculate the enrichment between each genomic event - VIPER inferred protein pair.

Description

Requires pre-computed VIPER scores and a binary events matrix. Will use only samples in both event and VIPER matrices.
Usage

associateEvents(

    vipermat,
    events.mat,
    min.events = NA,
    whitelist = NA,
    event.type = c("Amplifications", "Deletions", "Mutations", "Fusions", NA),
    verbose
)

Arguments

vipermat Pre-computed VIPER scores with samples as columns and proteins as rows
events.mat Binary 0/1 events matrix with samples as columns and genes or events as rows
min.events Only compute enrichment if the number of samples with these events is GTE to this
whitelist Only compute associations for events in this list
event.type Name of the event type being analyzed
verbose whether to print extra progress statements

Value

A matrix of aREA scores, dimensions are nrow(events.mat) x nrow(vipermat)

checkGeneMap Check Gene Map

Description

Check Gene Map

Usage

checkGeneMap(gene.loc.mapping)

Arguments

gene.loc.mapping
dataframe with gene names, entrez ids and cytoband locations

Value

nothing
**checkList**

---

**Check List of Assays**

**Description**

Check List of Assays

**Usage**

```r
checkList(assaylist)
```

**Arguments**

- `assaylist` : list of assays (viper, cnv, mut and fusion)

**Value**

updated/filter assaylist obj

---

**checkMAE**

---

**Check MultiAssayExperiment**

**Description**

Check MultiAssayExperiment

**Usage**

```r
checkMAE(mae)
```

**Arguments**

- `mae` : MultiAssayExperiment object

**Value**

updated/filtered MAE
checkPathways  

Check Pathways

Description
Check Pathways

Usage
checkPathways(pathways, x, type)

Arguments
- pathways: A named list of lists. Each named list represents interactions between proteins (keys) and their associated partners
- x: the MAE or Assaylist
- type: whether x is MAE or Assaylist

Value
nothing

clusterRange  

Cluster Range

Description
This function generate an cluster structure with 'k' groups and computes the cluster reliability score where 'k' is a range of values

Usage
clusterRange(
  dis,
  range = c(2, 100),
  step = 1,
  cores = 1,
  method = c("pam", "kmeans"),
  data = NULL
)
clusterReliability

Arguments

- `dis` Distance object
- `range` vector with start and end 'k'
- `step` Integer indicating the incremental number of clusters to add in each iteration
- `cores` Maximum number of CPU cores to use
- `method` Either 'pam' k-medoids or kmeans. Must supply the original data matrix if using kmeans
- `data` Original data matrix

Value

list of cluster reliability scores by 'k', 'clustering' (the vector solution) and 'reliability' as well as 'medoids' labels

Description

This function estimates the cluster membership reliability using aREA

Usage

clusterReliability(
  cluster,
  similarity,
  xlim = NULL,
  method = c("element", "cluster", "global")
)

Arguments

- `cluster` Vector of cluster memberships or list of cluster memberships
- `similarity` Similarity matrix
- `xlim` Optional vector of 2 components indicating the limits for computing AUC
- `method` Character string indicating the method to compute reliability, either by element, by cluster or global

Value

Reliability score for each element
**cnvScoreStouffer**  \hspace{1cm} **Integrate CNV scores**

**Description**
Integrate CNV scores

**Usage**
```r
cnvScoreStouffer(
mapping,           
diggit.interactions, 
cytoband = TRUE,  
from.p = FALSE,    
pos.nes.only = TRUE
)
```

**Arguments**
- `mapping`: a named vector of genomic locations/cytoband IDs. names are the gene names for each—i.e. a many to one mapping from HUGO or entrez IDs to cytoband location
- `diggit.interactions`: list indexed by MR/TF name in Entrez Space each points to a named vector of NES / z-scores associated with entrez IDs for each interacting event.
- `cytoband`: Boolean to use cytoband locations for computing final integrated score
- `from.p`: Boolean, set TRUE if diggit.interaction values are p-values instead of z-scores
- `pos.nes.only`: Boolean, only consider positive DIGGIT association scores when ranking candidate MRs (default=TRUE)

**Value**
A vector of z-scores, named by the Master Regulators in ‘diggit.interactions’

**conditionalModel**  \hspace{1cm} **Implements the conditional Bayes model to combine VIPER scores with diggit and pathway scores**

**Description**
Implements the conditional Bayes model to combine VIPER scores with diggit and pathway scores

**Usage**
```r
conditionalModel(viper.scores, diggit.scores, pathway.scores)
```
**conditionalP**

Arguments

- `viper.scores` numeric Vector
- `diggit.scores` List indexed by type char, with numeric score vectors in [0,R+] for each
- `pathway.scores` List, double indexed by each pathway dataset, then with type char. Each points to a numeric score vectors in [0,R+] for each

Value

a named vector of empirical p-values for each protein/candidate Master Regulator

---

**conditionalP**  
*Get the conditional p-value of a gene*

Description

Get the conditional p-value of a gene

Usage

```r
conditionalP(gene.name, condition.on, x)
```

Arguments

- `gene.name` Character
- `condition.on` named Vector of scores for the distribution we are conditioning ON
- `x` named Vector of scores for the dependent distribution

Value

a numeric p-value between 0 and 1

---

**empiricalP**  
*Get the empirical p-value from a distribution (vector)*

Description

Get the empirical p-value from a distribution (vector)

Usage

```r
empiricalP(gene.name, x)
```
**Arguments**

- **gene.name**: Character
- **x**: named Vector of scores for the distribution

**Value**

- a numeric p-value between 0 and 1

---

**Example.gbm.mae**  
**Glioblastoma (GBM) Example Dataset**

**Description**

MultiAssayExperiment Object containing all the genomic assays needed to run the example code for MOMA

**Usage**

`example.gbm.mae`

**Format**

- **viper**: matrix of viper scores with samples in columns and regulators across the rows
- **mut**: matrix of samples and genes with potential mutations. 0 for no mutation, 1 for presence of some non-silent mutation
- **cnv**: matrix of samples and genes with copy number variant scores

---

**fitCurvePercent**  
**Fit based on fractional overall coverage of genomic events**

**Description**

Fit based on fractional overall coverage of genomic events

**Usage**

`fitCurvePercent(sweep, frac = 0.85)`

**Arguments**

- **sweep**: Numeric vector of genomic coverage values, named by -k- threshold
- **frac**: Fraction of coverage to use as a threshold (default .85 = 85 percent)

**Value**

The -k- integer where coverage is achieved
gbm.pathways

**Glioblastoma (GBM) Pathways**

**Description**
Object containing information about the biological pathways that will be used in the analysis

**Usage**
gbm.pathways

**Format**
A list of lists named "cindy" and "preppi" respectively
- **cindy** list of regulators, each with a set of modulators and p values representing their CINDY inferred association
- **preppi** list of regulators, each with a set of potential binding partners and PREPPi inferred p values for probability of binding

---

gene.map

**Gene Location Mapping**

**Description**
Table used for converting between different forms of gene information. Downloaded from HGNC’s custom download portal using the "Approved Symbol", "NCBI Gene ID", "Chromosome" and "Ensembl Gene ID" curated data options and only those with "Approved" status. Updated December 2019.

**Usage**
gene.map

**Format**
A Data frame with 4 columns
- **Gene.Symbol** Approved Symbol gene name
- **Entrez.IDs** NCBI Gene ID
- **Cytoband** Chromosome location
- **Ensembl** Ensembl gene ID

@source [https://www.genenames.org/download/custom/](https://www.genenames.org/download/custom/)
getCoverage

### genomicPlotSmall

**Make small genomic plot**

**Description**

Make small genomic plot

**Usage**

```r
genomicPlotSmall(input.df, fraction = 0.85, tissue.cluster = NULL)
```

**Arguments**

- `input.df`: tissue.coverage.df with mean, k, fraction and unique events.
- `fraction`: what fraction coverage to use for genomic curve threshold
- `tissue.cluster`: which cluster subsample to look at

**Value**

output .png

---

getCoverage

**Get coverage of interactions**

**Description**

Get coverage of interactions

**Usage**

```r
getCoverage(
    MomaObject,
    cMR.ranking,
    viper.samples,
    topN = 100,
    mutation.filter = NULL,
    verbose = FALSE
)
```

**Arguments**

- `MomaObject`: A numeric vector with cluster membership, names are samples
- `cMR.ranking`: A vector entrez IDs, in order
- `viper.samples`: Calculate the genomic coverage only for these sample
- `topN`: Compute coverage for only the top -N- Master Regulators
- `mutation.filter`: Retain only mutation events in this (positive) list
**getValue**

A list of lists, indexed by sample name, with coverage statistics for each sample

---

**getDescription**

**Helper function to get data frame for bar plot plot.events function**

**Description**

Helper function to get data frame for bar plot plot.events function

**Usage**

```
getDataFrame(
  data,
  highlight.genes,
  genomeBand_2_gene,
  max.muts = 10,
  max.cnv = 5
)
```

**Arguments**

- `data`: data.frame with $type, $id, $Freq per event
- `highlight.genes`: genes to look for in mutations/cnv lists (if looking for specific genes because of prior knowledge)
- `genomeBand_2_gene`: mapping of genomic location IDs to gene name: vector of HUGO gene ids, named by genomic loci
- `max.muts`: maximum number of mutations to get per sample, default is 10
- `max.cnv`: maximum number of cnvs to per sample, default is 5

**Value**

ordered data frame with each genomic event and it’s frequency
getDiggitEmpiricalQvalues

Compute the empirical q-values of each genomic-event/VIPER gene pair

Description
Use against the background distribution of associations with a given set of 'null' VIPER genes (i.e. low activity TFs)

Usage
getDiggitEmpiricalQvalues(vipermat, nes, null.TFs, alternative = "both")

Arguments
- vipermat: viper inferences matrix, samples are columns, rows are TF entrez gene IDs
- nes: scores for each mutation (rows) against each TF (columns)
- null.TFs: low-importance TFs used to calculate null distributions
- alternative: Alternative defaults to 'both': significant p-values can come from both sides of the null distribution

Value
A named list of qvalues for each TF/cMR protein. Each entry contains a vector of q-values for all associated events; names are gene ids

getEmpiricalQvals

Get empirical qvals

Description
Get empirical qvals

Usage
getEmpiricalQvals(test.statistics, null.statistics, alternative = "both")

Arguments
- test.statistics: P-values generated from the test comparisons
- null.statistics: P-values generated under the null (permutation) model
- alternative: Optional: 1 or 2 tails used to generate the p-value
**getPvalsMatrix**

**Value**

A list with both the qvalues and empirical p-values from the supplied test and null stats

**getPvalsMatrix**  
*Utility function*

**Description**

Utility function

**Usage**

`getPvalsMatrix(corrected.scores)`

**Arguments**

- `corrected.scores` - corrected p-values processed by 'qvals' package

**Value**

A matrix of p-values for scores between genes/events (rows) and TFs (columns)

---

**getSubtypeEventTables**  
*Helper function to get subtype specific events*

**Description**

Helper function to get subtype specific events

**Usage**

`getSubtypeEventTables(saturation.data, sample.clustering, checkpoints)`

**Arguments**

- `saturation.data` : genomic saturation object from MOMA. List indexed by cluster then sample then regulator with the number of events associated with each additional regulator
- `sample.clustering` : clustering vector with sample names and cluster designations
- `checkpoints` : from momaObj

**Value**

A table that has counts of how many times a particular event happens in a cluster
**integrateFunction**  
*Numerical integration of functions*

**Description**
Integrates numerically a function over a range using the trapezoid method.

**Usage**
```
integrateFunction(f, xmin, xmax, steps = 100, ...)
```

**Arguments**
- `f`: Function of 1 variable (first argument)
- `xmin`: Number indicating the min x value
- `xmax`: Number indicating the max x value
- `steps`: Integer indicating the number of steps to evaluate
- `...`: Additional arguments for `f`

**Value**
Number

---

**integrateTZ**  
*Integration with trapezoid method*

**Description**
This function integrates over a numerical range using the trapezoid method.

**Usage**
```
integrateTZ(x, y)
```

**Arguments**
- `x`: Numeric vector of x values
- `y`: Numeric vector of y values

**Value**
Number
makeCoverageDf

**Description**

Helper function for making the coverage dataframe

**Usage**

```r
makeCoverageDf(coverage.list, cutoff)
```

**Arguments**

- `coverage.list`: List indexed by sample name, contains mut/fus/amp/del interactions
- `cutoff`: number of regulators to include

**Value**

dataframe with each sample and which events are captured by the checkpoint mrs

makeSaturationPlots

**Description**

Main function to generate the summary plots of the analysis

**Usage**

```r
makeSaturationPlots(
    momaObj,
    clustering.solution = NULL,
    important.genes = NULL,
    fCNV = NULL,
    max.events = 30
)
```

**Arguments**

- `momaObj`: momaObj that has already run the saturationCalculation function
- `clustering.solution`: clustering vector with sample names and cluster designations
- `important.genes`: vector of gene names to prioritize when plotting. Can be general genes of interest, oncogenes, tumor supressors etc
mapEntrez

fCNV : vector of confirmed functional CNVs if calculated. Will filter for only those CNVs

max.events : maximum number of events to plot for the oncoplots

Value

object with both types of summary plot for each subtype

Examples

## Not run:
makeSaturationPlots(momaObj, max.events = 20)

## End(Not run)

mapEntrez

Convert from entrez ids to hugo gene names

Description

Convert from entrez ids to hugo gene names

Usage

mapEntrez(entrez.ids)

Arguments

entrez.ids : vector of entrez ids requires hugo2entrez to be loaded

Value

: vector of hugo gene names

See Also

mapHugo

Examples

mapEntrez(c("29974", "5728"))
mapHugo

Convert from hugo gene names to entrez ids

Description
Convert from hugo gene names to entrez ids

Usage
mapHugo(hugo.ids)

Arguments
hugo.ids : vector of hugo gene names, requires hugo2entrez to be loaded

Value
: vector of entrez ids

See Also
mapEntrez

Examples
mapHugo(c("A1CF","PTEN"))

mapScoresCnvBand

Map scores to cytoband location

Description
Map scores to cytoband location

Usage
mapScoresCnvBand(
  mapping,
  diggit.interactions,
  from.p = FALSE,
  pos.nes.only = TRUE
)
mergeData

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mapping</td>
<td>a named vector of genomic locations/cytoband IDs, names are the gene names</td>
</tr>
<tr>
<td></td>
<td>for each—i.e., a many to one mapping from HUGO or entrez IDs to cytoband</td>
</tr>
<tr>
<td></td>
<td>location</td>
</tr>
<tr>
<td>diggit.interactions</td>
<td>list indexed by MR/TF name in Entrez Space</td>
</tr>
<tr>
<td>from.p</td>
<td>DIGGIT interactions are in p-value format instead of z-score (default=FALSE)</td>
</tr>
<tr>
<td>pos.nes.only</td>
<td>Only consider positive associations with NES scores (default=TRUE)</td>
</tr>
<tr>
<td></td>
<td>each points to a named vector of NES / z-scores associated with entrez IDs</td>
</tr>
<tr>
<td></td>
<td>for each interacting event.</td>
</tr>
</tbody>
</table>

Value

A list of input scores, now named by cytoband location

mergeData

Helper function for mergeDataBySubtype

Description

Helper function for mergeDataBySubtype

Usage

mergeData(coverage.range, topN)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>coverage.range</td>
<td>genomic saturation for a particular subtype</td>
</tr>
<tr>
<td>topN</td>
<td>max number of top regulators to search through</td>
</tr>
</tbody>
</table>

Value

dataframe with coverage data for genomic events
mergeDataBySubtype

Create data frame from coverage data, including number of total events ‘covered’ and unique events

Description

Create data frame from coverage data, including number of total events 'covered' and unique events

Usage

mergeDataBySubtype(genomic.saturation, sample.clustering, topN = 100)

Arguments

- genomic.saturation: data from genomic saturation function
- sample.clustering: clustering vector with sample names and cluster designations
- topN: number of regulators to look through. default is 100

Value

dataframe with coverage data for genomic events

mergeGenomicSaturation

mergeGenomicSaturation Create data frame from coverage data, including number of total events ‘covered’ and unique events

Description

mergeGenomicSaturation Create data frame from coverage data, including number of total events 'covered' and unique events

Usage

mergeGenomicSaturation(coverage.range, topN)

Arguments

- coverage.range: List indexed by sample, then sub-indexed by # of master regulators, then by event type (mut/amp/del/fus). Holds all events by sample
- topN: Maximum number of master regulators to compute coverage

Value

A data frame with summary statistics for genomic saturation at each k
mergeLists  

*Helper function*

**Description**

Helper function

**Usage**

`mergeLists(l1, l2)`

**Arguments**

- `l1` list 1
- `l2` list 2

**Value**

single merged list

---

**Moma-class**

*MOMA Object*

**Description**

Main class encapsulating the input data and logic of the MOMA algorithm

**Fields**

- `viper` matrix of inferred activity score inferred by viper
- `mut` binary mutation matrix 1 for presence of mutation, 0 for not, NA if not determined
- `cnv` matrix of cnv values. Can be binary or a range.
- `fusions` binary matrix of fusion events if applicable
- `pathways` list of pathways/connections to consider as extra evidence in the analysis
- `gene.blacklist` character vector of genes to not include because of high mutation frequency
- `output.folder` character vector of location to save files if desired
- `gene.loc.mapping` data frame of gene names, entrez ids and cytoband locations
- `nes` field for saving Normalized Enrichment Matrices from the associate events step
- `interactions` field for saving the MR-interactions list
- `clustering.results` results from clustering are saved here
- `ranks` results field for ranking of MRs based on event association analysis
- `hypotheses` results field for saving events that have enough occurrences to be considered
MomaConstructor

Methods

Cluster( clus.eval = c("reliability", "silhouette"), use.parallel = FALSE, cores = 1 )
Cluster the samples after applying the MOMA weights to the VIPER scores

makeInteractions( genomic.event.types = c("amp", "del", "mut", "fus"), cindy.only = FALSE )
Make interaction web for significant MRs based on their associated events

Rank( use.cindy = TRUE, genomic.event.types = c("amp", "del", "mut", "fus"), use.parallel = FALSE, cores = 1 )
Rank MRs based on DIGGIT scores and number of associated events

runDIGGIT(fCNV = NULL, cnvthr = 0.5, min.events = 4, verbose = FALSE)
Run DIGGIT association function to get associations for driver genomic events

saturationCalculation( clustering.solution = NULL, cov.fraction = 0.85, topN = 100, verbose = FALSE )
Calculate the number of MRs it takes to represent the desired coverage fraction of events

Description

Create MOMA Object from either a MultiAssayExperiment object or a list of assays. See vignette for more information on how to set up and run the MOMA object

Usage

MomaConstructor(
  x,
  pathways,
  gene.blacklist = NA_character_,
  output.folder = NA_character_,
  gene.loc.mapping = gene.map,
  viperAssay = "viper",
  mutMat = "mut",
  cnvMat = "cnv",
  fusionMat = "fusion"
)
Arguments

- `x`: A MultiAssayExperement object or list object with the following assays: (note: by default assays must have these exact names. Otherwise they can be changed using the viperAssay, mutMat, cnvMat and fusionMat parameters.)
  - `viper`: VIPER protein activity matrix with samples as columns and rows as protein IDs
  - `mut`: An indicator matrix (0/1) of mutation events with samples as columns and genes as rows
  - `cnv`: A matrix of CNV scores (typically SNP6 array scores from TCGA) with samples as columns and genes as rows
  - `fusion`: An indicator matrix (0/1) of fusion events with samples as columns and genes as rows

- `pathways`: A named list of lists. Each named list represents interactions between proteins (keys) and their associated partners

- `gene.blacklist`: A vector of genes to exclude from the analysis

- `output.folder`: Location to store output and intermediate results

- `gene.loc.mapping`: A data.frame of band locations and Entrez IDs

- `viperAssay`: name associated with the viper assay in the assay object

- `mutMat`: name associated with the mutation matrix in the assay object

- `cnvMat`: name associated with the cnv matrix in the assay object

- `fusionMat`: name associated with the fusion matrix in the assay object

Value

an instance of class Moma

Examples

```r
momaObj <- MomaConstructor(example.gbm.mae, gbm.pathways)
```

---

### mutSig

**MutSig Blacklisted genes**

Description

List of genes to not include in the DIGGIT mutation inference because they have been found to be mutated more often than expected by chance given background mutation processes.

Usage

```r
mutSig
```
oncoprintPlot

Format

A character vector of Entrez Gene IDs

Source

https://software.broadinstitute.org/cancer/cga/mutsig

oncoprintPlot Function to plot genomic events in the style of oncoPrint/cBioPortal

Description

Function to plot genomic events in the style of oncoPrint/cBioPortal

Usage

oncoprintPlot(
  summary.vec,
  snpmat.thisClus,
  amps.thisClus,
  dels.thisClus,
  fusions.thisClus,
  important.genes,
  band2gene,
  max.events,
  k
)

Arguments

summary.vec : named vector of the counts, named 'Event name':'Type' where type is 'mut', 'amp', 'del', 'fus'. Mutations are in Entrez ID Amp/Deletion CNV events are in genomic band location
snpmat.thisClus : SNP matrix subset to samples in current cluster
amps.thisClus : CNV matrix subset to samples in current cluster (just amplifications)
dels.thisClus : CNV matrix subset to samples in current cluster (just deletions)
fusions.thisClus : Fusion matrix subset to samples in current cluster
important.genes : well known genes to highlight in the analysis
band2gene : mapping of genomic location IDs to gene name: vector of HUGO gene ids, named by genomic location
max.events : maximum number of events to plot for the oncoplots
k : current cluster number
pathwayDiggitIntersect

*Combine DIGGIT inferences with pathway knowledge*

**Description**

Combine DIGGIT inferences with pathway knowledge

**Usage**

```r
pathwayDiggitIntersect(diggit.int, pathway, pos.nes.only = TRUE, cores = 1)
```

**Arguments**

- `diggit.int`: List of interactions between MRs - Genomic events, inferred by DIGGIT
- `pathway`: - a list indexed by TF/MR entrez ID, contains the named vector of p-values for interactions
- `pos.nes.only`: Only use positive associations between MR activity and presence of events (default = True)
- `cores`: Number of cores to use if parallel is selected

**Value**

numeric vector, zscores for each TF/MR

---

plotEvents

*Plot barchart of genomic events*

**Description**

Plot barchart of genomic events

**Usage**

```r
plotEvents(
    summary.vec,
    highlight.genes = NULL,
    genomeBand_2_gene = NULL,
    samples.total,
    max.muts = 10,
    max.cnv = 5
)
```
Arguments

summary.vec : named vector of the counts, named 'Event name':'Type' where type is 'mut', 'amp', 'del', 'fus'. Mutations are in Entrez ID Amp/Deletion CNV events are in genomic band location

highlight.genes : well known genes to highlight in the analysis in

genomeBand_2_gene : mapping of genomic location IDs to gene name: vector of HUGO gene ids, named by genomic loci

samples.total : number of samples in the subtype, used to calculate percentages

max.muts : maximum number of mutations to get per sample, default is 10

max.cnv : maximum number of cnvs to per sample, default is 5

Value

plot object

Description

This function calculates an Enrichment Score of Association based on how the features rank on the samples sorted by a specific gene

Usage

rea(eset, regulon, minsize = 1, maxsize = Inf, event.type = NA, verbose)

Arguments

eset Numerical matrix

regulon A list with genomic features as its names and samples as its entries, indicating presence of event

minsize The minimum number of events to use when calculating enrichment

maxsize The maximum number of events to use when calculating enrichment

event.type Type of event being analyzed

verbose whether to print extra progress statements

Value

A list containing two elements:

groups Regulon-specific NULL model containing the enrichment scores

ss Direction of the regulon-specific NULL model
### reaNULL

*This function generates the NULL model function, which computes the normalized enrichment score and associated p-value*

**Description**

This function generates the NULL model function, which computes the normalized enrichment score and associated p-value.

**Usage**

```r
reaNULL(regulon, minsize = 1, maxsize = Inf)
```

**Arguments**

- **regulon**: A list with genomic features as its names and samples as its entries.
- **minsize**: Minimum number of event (or size of regulon).
- **maxsize**: Maximum number of event (or size of regulon).

**Value**

A list of functions to compute NES and p-value.

### sampleNameFilter

*Retain TCGA sample ids without the final letter designation ('A/B/C')*

**Description**

Retain TCGA sample ids without the final letter designation ('A/B/C').

**Usage**

```r
sampleNameFilter(input, desired.len = 15)
```

**Arguments**

- **input**: Matrix of expression or protein activity scores. Columns are sample names, rows are genes. Input can also just be an input vector of sample names.
- **desired.len**: Length to reduce strings to. Default is 15 because of TCGA naming conventions.

**Value**

An identical matrix with new (shorter) column names, or a vector with the shortened names.

**Examples**

```r
sample.names <- c("TCGA-14-1825-01A", "TCGA-76-4931-01B", "TCGA-06-5418-01A")
sampleNameFilter(sample.names)
```
sampleOverlap

The core function to compute which sample-specific alterations overlap with genomic events that are explained via DIGGIT.

Description

The core function to compute which sample-specific alterations overlap with genomic events that are explained via DIGGIT.

Usage

```r
sampleOverlap(
  MomaObject,
  viper.samples,
  selected.tfs,
  interaction.map,
  cnv.threshold = 0.5,
  mutation.filter = NULL,
  idx.range = NULL,
  verbose = FALSE
)
```

Arguments

- **MomaObject**: Object reference of momaRunner class
- **viper.samples**: Sample vector to restrict sample-specific analysis to
- **selected.tfs**: Transcription factors being analyzed
- **interaction.map**: List object of events 'covered' by the supplied interactions of type mut/amp/del/fus
- **cnv.threshold**: Numeric absolute value to threshold SNP6 and/or GISTIC or other CNV scores. Above that absolute value is considered a positive event.
- **mutation.filter**: A vector of whitelisted mutation events, in entrez gene IDs
- **idx.range**: Number of tfs to check for genomic saturation calculation, default is 1253
- **verbose**: Output status during the run (default=FALSE)

Value

A list of lists, indexed by sample name, with coverage statistics/data for each sample
sigInteractorsDIGGIT  
Filter interactions from NES (DIGGIT) scores and corresponding background-corrected scores.

Description

Use this version in the Bayes model to rank TFs

Usage

```r
sigInteractorsDIGGIT(
corrected.scores,
nes.scores,
cindy,
p.thresh = 0.05,
cindy.only = TRUE
)
```

Arguments

- **corrected.scores**: A list indexed by the genomic event/gene with corresponding pvals and qvals for each TF
- **nes.scores**: Matrix with tfs as columns, rows are genomic events
- **cindy**: CINDy algorithm output matrix
- **p.thresh**: P-value threshold (default=0.05)
- **cindy.only**: Consider only CINDy validated interactions (default=TRUE)

Value

A list (indexed by VIPER protein) of significant genomic interactions and associated pvals over the background (null TF) model, and NES scores

sREA  
Simple one-tail rank based enrichment analysis sREA (for cluster analysis)

Description

This function performs simple 1-tail rank based enrichment analysis

Usage

```r
sREA(signatures, groups)
```
Arguments

signatures Numeric matrix of signatures
groups List containing the groups as vectors of sample names

Value

Matrix of Normalized Enrichment Zscores

Description

dispatch method for either CNV location corrected or SNV

Usage

stoufferIntegrate(interactions, cytoband.map = NULL)

Arguments

interactions List of MR - Genomic Event interactions, inferred by DIGGIT
cytoband.map Data.frame mapping Entrez.IDs to cytoband locations

Value

Z-scores for each MR

stoufferIntegrateDiggit

Use Stouffer’s method to combine z-scores of DIGGIT interactions for each cMR protein.

Description

This function combines only positively associated DIGGIT scores by default to create a culmulative DIGGIT score for each cMR.

Usage

stoufferIntegrateDiggit(interactions, from.p = FALSE, pos.nes.only = TRUE)
validDiggitInteractions

Arguments

interactions  A list indexed by TF, includes z-scores or p-values for each interacting event
from.p       Integrate p-values or z-scores (default z-scores; from.p = FALSE)
pos.nes.only Use only positive NES scores to rank proteins (default TRUE)

Value

A list indexed by TF, a stouffer integrated z-score

---

subsetListInteractions

*Helper function: subset a list to the set of keys supplied return the names of interactions with positive values, in a list structure*

Description

Helper function: subset a list to the set of keys supplied return the names of interactions with positive values, in a list structure

Usage

subsetListInteractions(int.l, keys)

Arguments

int.l       List of interactions, at each index this is a numeric named vector
keys        Keys used to reduce interactions

Value

Returns a filtered list of interactions in the same format as the input

---

validDiggitInteractions

*Return a set of events 'covered' by specified cMR-event interactions*

Description

Return a set of events 'covered' by specified cMR-event interactions

Usage

validDiggitInteractions(interactions, gene.loc.mapping, selected.tfs)
viperGetSigTFS

**Arguments**

- interactions: List indexed by amp/mut/del/fus from cMRs to interacting events
- gene.loc.mapping: Data.frame mapping entrezIDs to cytoband locations
- selected.tfs: For each event type list, search within only these cMRS

**Value**

- a list of events 'covered' by the supplied interactions of type mut/amp/del/fus

---

**viperGetSigTFS**

*Calculate p-values from pseudo zscores / VIPER aREA scores, threshold*

**Description**

Calculate p-values from pseudo zscores / VIPER aREA scores, threshold

**Usage**

viperGetSigTFS(zscores, fdr.thresh = 0.05)

**Arguments**

- zscores: Vector of normally distributed z-scores representing protein activities.
- fdr.thresh: Threshold for false discovery rate, default is 0.05

**Value**

- Get the names of proteins with significant z-scores, after multi-hypothesis correction

---

**viperGetTFScores**

*Function to normalize TF scores*

**Description**

Function to normalize TF scores

**Usage**

viperGetTFScores(vipermat, fdr.thresh = 0.05)

**Arguments**

- vipermat: - matrix of VIPER scores with columns as samples, rows as protein names
- fdr.thresh: - BH-FDR threshold (default 0.05 FDR rate)
viperGetTFScores

Value

A vector of normalized z-scores, named by TF id
Index

* datasets
  example.gbm.mae, 10
  gbm.pathways, 11
  gene.map, 11
  mutSig, 24

* internal
  areaEnrich, 3
  associateEvents, 3
  checkGeneMap, 4
  checkList, 5
  checkMAE, 5
  checkPathways, 6
  clusterRange, 6
  clusterReliability, 7
  conditionalModel, 8
  conditionalP, 9
  empiricalP, 9
  fitCurvePercent, 10
  genomicPlotSmall, 12
  getCoverage, 12
  getDataFrame, 13
  getDiggitEmpiricalQvalues, 14
  getEmpiricalQvals, 14
  getPvalsMatrix, 15
  getSubtypeEventTables, 15
  integrateFunction, 16
  integrateTZ, 16
  makeCoverageDf, 17
  mergeData, 20
  mergeDataBySubtype, 21
  mergeGenomicSaturation, 21
  mergeLists, 22
  oncoprintPlot, 25
  pathwayDiggitIntersect, 26
  plotEvents, 26
  rea, 27
  reaNULL, 28
  sampleOverlap, 29
  sigInteractorsDIGGIT, 30
  sREA, 30
  subsetListInteractions, 32
  validDiggitInteractions, 32
  viperGetSigTFS, 33
  viperGetTFScores, 33

areaEnrich, 3
associateEvents, 3
checkGeneMap, 4
checkList, 5
checkMAE, 5
checkPathways, 6
clusterRange, 6
clusterReliability, 7
cnvScoreStouffer, 8
conditionalModel, 8
conditionalP, 9
empiricalP, 9
example.gbm.mae, 10
fitCurvePercent, 10
gbm.pathways, 11
gene.map, 11
genomicPlotSmall, 12
getCoverage, 12
getDiggitEmpiricalQvalues, 14
getEmpiricalQvals, 14
getPvalsMatrix, 15
getSubtypeEventTables, 15
integrateFunction, 16
integrateTZ, 16
makeCoverageDf, 17
mergeData, 20
mergeDataBySubtype, 21
mergeGenomicSaturation, 21
mergeLists, 22
oncoprintPlot, 25
pathwayDiggitIntersect, 26
plotEvents, 26
rea, 27
reaNULL, 28
sampleOverlap, 29
sigInteractorsDIGGIT, 30
sREA, 30
subsetListInteractions, 32
validDiggitInteractions, 32
viperGetSigTFS, 33
viperGetTFScores, 33
areaEnrich, 3
associateEvents, 3
checkGeneMap, 4
checkList, 5
checkMAE, 5
checkPathways, 6
clusterRange, 6
clusterReliability, 7
cnvScoreStouffer, 8
conditionalModel, 8
conditionalP, 9
datasets
example.gbm.mae, 10
  gbm.pathways, 11
  gene.map, 11
  mutSig, 24
  internal
  areaEnrich, 3
  associateEvents, 3
  checkGeneMap, 4
  checkList, 5
  checkMAE, 5
  checkPathways, 6
  clusterRange, 6
  clusterReliability, 7
  conditionalModel, 8
  conditionalP, 9
  empiricalP, 9
  fitCurvePercent, 10
  genomicPlotSmall, 12
  getCoverage, 12
  getDataFrame, 13
  getDiggitEmpiricalQvalues, 14
  getEmpiricalQvals, 14
  getPvalsMatrix, 15
  getSubtypeEventTables, 15
  integrateFunction, 16
  integrateTZ, 16
  makeCoverageDf, 17
  mergeData, 20
  mergeDataBySubtype, 21
  mergeGenomicSaturation, 21
  mergeLists, 22
  oncoprintPlot, 25
  pathwayDiggitIntersect, 26
  plotEvents, 26
  rea, 27
  reaNULL, 28
  sampleOverlap, 29
  sigInteractorsDIGGIT, 30

35
mapScoresCnvBand, 19
mergeData, 20
mergeDataBySubtype, 21
mergeGenomicSaturation, 21
mergeLists, 22
Moma (Moma-class), 22
Moma-class, 22
MomaConstructor, 23
mutSig, 24
oncoprintPlot, 25
pathwayDiggitIntersect, 26
plotEvents, 26
rea, 27
reaNULL, 28
sampleNameFilter, 28
sampleOverlap, 29
sigInteractorsDIGGIT, 30
sREA, 30
stoufferIntegrate, 31
stoufferIntegrateDiggit, 31
subsetListInteractions, 32
validDiggitInteractions, 32
viperGetSigTFS, 33
viperGetTFScores, 33