# Package ‘attract’

April 14, 2017

**Type** Package  
**Title** Methods to Find the Gene Expression Modules that Represent the Drivers of Kauffman’s Attractor Landscape  
**Version** 1.26.0  
**Date** 2016-10-13  
**Author** Jessica Mar  
**Maintainer** Samuel Zimmerman <sezimmer@einstein.yu.edu>  

**Description**  
This package contains the functions to find the gene expression modules that represent the drivers of Kauffman’s attractor landscape. The modules are the core attractor pathways that discriminate between different cell types of groups of interest. Each pathway has a set of synexpression groups, which show transcriptionally-coordinated changes in gene expression.

**License** LGPL (>= 2.0)

**Collate**  
classdef.R findAttractorsStep.R removeFlatGenesStep.R  
findSynexprsStep.R findFuncEnrichAndCorr.R

**Depends**  
R (>= 3.3.1), methods, AnnotationDbi

**Imports**  
Biobase, limma, cluster, GOstats, graphics, stats, reactome.db, KEGGREST, org.Hs.eg.db, utils

**Suggests** illuminaHumanv1.db  
biocViews KEGG, Reactome, GeneExpression, Pathways, GeneSetEnrichment, Microarray, RNASeq

**NeedsCompilation** no

## R topics documented:

- attract-package .................................................. 2  
- AttractorModuleSet-class ...................................... 3  
- buildCustomIncidenceMatrix .................................. 4  
- calcFuncSynexprs ............................................... 5  
- calcInform ...................................................... 6  
- calcModfstat .................................................... 7  
- calcRss .......................................................... 8  
- exprs.dat ..................................................... 9
Methods to find the Gene Expression Modules that Represent the Drivers of Kauffman’s Attractor Landscape

Description

This package contains functions used to determine the gene expression modules that represent the drivers of Kauffman’s attractor landscape.

Details

| Package: | attract |
| Type:    | Package |
| Version: | 1.25.2   |
| Date:    | 2016-10-13 |
| License: | LazyLoad: yes |

The method can be summarized in the following key steps: (1) Determine core KEGG or reactome pathways that discriminate the most strongly between celltypes or experimental groups of interest (see findAttractors). (2) Find the different synexpression groups that are present within a core attractor pathway (see findSynexprs). (3) Find sets of genes that show highly similar profiles to the synexpression groups within an attractor pathway module (see findCorrPartners). (4) Test for functional enrichment for each of the synexpression groups to detect any potentially shared biological themes (see calcFuncSynexprs).

Author(s)

Jessica Mar <jess@jimmy.harvard.edu>

References

Examples

## Not run:
data(subset.loring.eset)
attractor.states <- findAttractors(subset.loring.eset, "celltype", nperm=10, annotation="illuminaHumanv1.db")
remove.these.genes <- removeFlatGenes(subset.loring.eset, "celltype", contrasts=NULL, limma.cutoff=0.05)
mapk.syn <- findSynexprs("04010", attractor.states, remove.these.genes)
mapk.cor <- findCorrPartners(mapk.syn, subset.loring.eset, remove.these.genes)
mapk.func <- calcFuncSynexprs(mapk.syn, attractor.states, "CC", annotation="illuminaHumanv1.db")

## End(Not run)

---

### Description

This is a class representation for storing the output of the `findAttractors` function.

### Objects from the Class

Objects are output by the function `findAttractors`. Objects can also be created by using `new("AttractorModuleSet", ...)`.

### Slots

- **eSet**: ExpressionSet which primarily stores the expression data and the phenotype/sample data sets.
- **cellTypeTag**: character string of the tag which stores the group membership information for the samples. Must be a column name of the data frame pData(eset).
- **incidenceMatrix**: incidence matrix used as input to GSEAlm.
- **rankedPathways**: Data frame of significantly enriched pathways, ranked first by significance and then by size.

### Methods

No methods have yet been defined with class "AttractorModuleSet" in the signature.

### Note

This class is better describe in the vignette.

### Author(s)

Jessica Mar <jess@jimmy.harvard.edu>

### Examples

## Not run:
new.attractmodule <- new("AttractorModuleSet", eSet=new("ExpressionSet"), cellTypeTag=character(1), incidenceMatrix=matrix(0))

## End(Not run)
buildCustomIncidenceMatrix

This function builds an incidence matrix for custom gene sets.

Description

This function builds an incidence matrix for custom gene sets.

Usage

buildCustomIncidenceMatrix(geneSetFrame, geneNames, databaseGeneFormat, expressionSetGeneFormat, geneSetNames)

Arguments

geneSetFrame  a dataframe where rows are gene sets and columns are genes.
geneNames     a vector of all the genes in the geneSetFrame dataframe
databaseGeneFormat  a character string specifying the type of identifier for a gene in a database (KEGG, reactome, MsigDB) gene set. The default value is NULL. (ex. SYMBOL, ENTREZID, REFSEQ, ENSEMBL)
expressionSetGeneFormat  a character string specifying the type of identifier for a gene in your expression data set. The default value is NULL. (ex. SYMBOL, ENTREZID, REFSEQ, ENSEMBL)
geneSetNames  a vector of the name of the custom gene sets.

Details

This function creates an incidence matrix from a dataframe where the rows are the names of gene sets and the columns are genes.

Value

A matrix object with 0 and 1 entries where 1 denotes membership of a gene in a custom gene set, 0 denotes non-membership.

Author(s)

Jessica Mar

References

Mar, J., C. Wells, and J. Quackenbush, Identifying the Gene Expression Modules that Represent the Drivers of Kauffman’s Attractor Landscape. to appear, 2010.
calcFuncSynexprs  

Functional enrichment analysis for a set of synexpression groups.

Description

This function performs functional enrichment for a given set of synexpression groups.

Usage

calcFuncSynexprs(mySynExpressionSet, myAttractorModuleSet, ontology = "BP", min.pvalue = 0.05, min.pwaysize = 5, annotation = "illuminaHumanv2.db", analysis = "microarray", expressionSetGeneFormat = NULL, ...)

Arguments

- mySynExpressionSet: SynExpressionSet object.
- myAttractorModuleSet: AttractorModuleSet object.
- ontology: character string specifying which GO ontology to use, either "MF", "BP", or "CC"; defaults to "BP".
- min.pvalue: numeric value specifying adjusted P-value cut-off to use, categories with P-values <= min.pvalue will be reported.
- min.pwaysize: integer specifying minimum size of the pathway or category to consider for enrichment analysis.
- annotation: character string specifying the annotation package that corresponds to the chip platform the data was generated from.
- analysis: a character string specifying what type of experiment you performed, microarray or RNAseq.
- expressionSetGeneFormat: a character string specifying the type of identifier for a gene in your expression data set. The default value is NULL. (ex. SYMBOL, ENTREZID, REFSEQ, ENSEMBL)
- ... additional arguments.

Details

This function performs a functional enrichment analysis on each synexpression group using the hyperGTest from the GOstats package. P-values are adjusted using the Benjamini-Hochberg correction method. Results are returned only if they satisfy the minimum P-value level, as specified by the min.pvalue argument.

Value

A list object.

Author(s)

Jessica Mar
calcInform

Function calculates the informativeness metric (average MSS) for a set of cluster assignments.

Description

Function calculates the informativeness metric (average MSS) for a set of cluster assignments.

Usage

calcInform(exprs.dat, cl, class.vector)

Arguments

exprs.dat  a matrix of gene expression values.
cl  a vector of cluster assignments.
class.vector  a vector specifying the group membership of the samples.

Details

This function is also called internally by findSynexprs.

Value

A numeric value representing the average MSS value (informativeness metric) for a set of cluster assignments. For an informative cluster, the RSS values should be very small relative to those produced by the informativeness metric (the MSS values).

Author(s)

Jessica Mar

References

Examples

```r
## Not run:
library(cluster)
data(subset.loring.eset)
clustObj <- agnes(as.dist(1-t(cor(exprs(subset.loring.eset)))))
cinform.vals <- NULL
for( i in 1:10 ){
cinform.vals <- c(cinform.vals, calcInform(exprs(subset.loring.eset), cutree(clustObj,i), pData(subset.loring.eset)$celltype))
}
k <- (1:10)[cinform.vals==max(cinform.vals)] # gives the optimal number of clusters
## End(Not run)
```

Description

Function calculates a modified F-statistic for a set of cluster assignments.

Usage

```r
calcModfstat(exprs.dat, cl, class.vector)
```

Arguments

- `exprs.dat`: a matrix of gene expression values.
- `cl`: a vector of cluster assignments.
- `class.vector`: a vector specifying group membership of the samples.

Details

This function is called internally by `findSynexprs`.

Value

A modified F-statistic (average MSS/average RSS) value for a set of cluster assignments.

Author(s)

Jessica Mar

Examples

```r
## Not run:
library(cluster)
data(subset.loring.eset)
clustObj <- agnes(as.dist(1-t(cor(exprs(subset.loring.eset)))))
cfmod.vals <- NULL
for( i in 1:10 ){
cfmod.vals <- c(cfmod.vals, calcModfstat(exprs(subset.loring.eset), cutree(clustObj,i), pData(subset.loring.eset)$celltype))
}
```
calcRss

Function calculates the average RSS for a set of cluster assignments.

Description
Function calculates the average RSS for a set of cluster assignments.

Usage
calcRss(exprs.dat, cl, class.vector)

Arguments
exprs.dat a matrix of gene expression values.
cl a vector of cluster assignments.
class.vector a vector specifying the group membership of the samples.

Details
This function is called internally by findSynexprs. For an informative cluster, the RSS values should be very small relative to those produced by the informativeness metric (the MSS values).

Value
A numeric value representing the average RSS value for this set of cluster assignments.

Author(s)
Jessica Mar

Examples

## Not run:
library(cluster)
data(subset.loring.eset)
clustObj <- agnes(as.dist(1-t(cor(exprs(subset.loring.eset)))))
crss.vals <- NULL
for( i in 1:10 ){
crss.vals <- c(crss.vals, calcRss(exprs(subset.loring.eset), cutree(clustObj,i), pData(subset.loring.eset)$celltype))
}
# The RSS values are expected to be smaller than the informativeness metric values in the presence of genuine clusters.

## End(Not run)
**Description**

This is a matrix object containing published gene expression data from Mueller et al. (NCBI GEO accession id GSE11508). The data set contains 11044 probes for 68 samples. From the original data set, we have selected four cell lines giving a total of 68 samples - embryonic stem cells (12 samples), neural progenitors (31 samples), neural stem cells (8 samples) and teratoma-differentiated cells (17 samples). The lines have also been restricted based on Illumina BeadChip platform, and only those collected using the WG-6 version have been used.

We also applied a quality filter to the original gene expression data where a probe was retained if it passed a 0.99 detection score in 75

**Usage**

```r
data(exprs.dat)
```

**Format**

A matrix with normalized log2 expression intensities for 11044 probes on 68 samples (representing 4 different cell types).

**Value**

A matrix object containing published gene expression data from Mueller et al. (NCBI GEO accession id GSE11508). The data set contains 11044 probes for 68 samples.

**References**


**See Also**

`samp.info, loring.eset`

**Examples**

```r
data(exprs.dat)
```
filterDataSet

This function filters our lowly expressed genes in RNAseq data.

Description

This function filters our lowly expressed genes in RNAseq data.

Usage

filterDataSet(data, filterPerc = 0.75)

Arguments

data A dataset with genes as rows and samples as columns.
filterPerc a number specifying the percent of expression values that are not equal to 0 for a gene.

Details

This function removes any genes in a dataset that have an expression value of 0 for a specified percentage of samples.

Value

A data frame is returned.

Author(s)

Jessica Mar

Examples

data(exprs.dat)
exprs.filtered.dat <- filterDataSet(exprs.dat)

findAttractors

Infers the set of cell-lineage specific gene expression modules using GSEAIm and KEGG.

Description

The function infers a set of KEGG pathways that correspond to the cell-lineage specific gene expression modules, as determined using GSEA. These pathways represent those that show the greatest discrimination between the different cell types or tissues in the expression data set supplied.

Usage

findAttractors(myEset, cellTypeTag, min.pwaysize = 5, annotation = "illuminaHumanv2.db", database = ...)
findAttractors

Arguments

myEset ExpressionSet object.
cellTypeTag character string of the variable name which stores the cell-lineages or experimental groups of interest for the samples in the data set (this string should be one of the column names of pData(myEset)).
min.pwaysize integer specifying the minimum size of the KEGG or reactome pathways to consider in the analysis.
annotation character string specifying the annotation package that corresponds to the chip platform or organism (for RNAseq data) the data was generated from.
database a character string specifying what pathway database you would like to use.
analysis a character string specifying what type of experiment you performed, microarray or RNAseq.
databaseGeneFormat a character string specifying the type of identifier for a gene in a database (KEGG, REACTOME, MsigDB) gene set. The default value is NULL. (ex. SYMBOL, ENTREZID, REFSEQ, ENSEMBL)
expressionSetGeneFormat a character string specifying the type of identifier for a gene in your expression data set. The default value is NULL. (ex. SYMBOL, ENTREZID, REFSEQ, ENSEMBL)

Details

This function subsets the expression data so that only those genes with annotations in KEGG or reactome are used for the downstream gene set enrichment analysis. This subset is stored in the eSet slot of the AttractoModuleSet output object.

The GSEAIm algorithm finds the KEGG or reactome pathway modules which discriminate between the celotypes or experimental groups of interest. It also ranks the results of the GSEAIm step by significance of these pathway modules, as stored in rankedPathways.

The output object of the findAttractors function also contains the incidence matrix that was built for the KEGG or reactome pathways, stored in the slot incidenceMatrix and the character string denoting which column of the sample data represents the cell type or experimental groups of interest, as stored in the slot cellTypeTag.

Value

An AttractorModuleSet object.

Author(s)

Jessica Mar

References

Mar, J., C. Wells, and J. Quackenbush, Identifying the
findCorrPartners

Gene Expression Modules that Represent the Drivers of Kauffman’s Attractor Landscape. to appear, 2010.

Examples

data(subset.loring.eset)
attractor.states <- findAttractors(subset.loring.eset, "celltype", annotation="illuminaHumanv1.db", database="KEGG", analysis="microarray", databaseGeneFormat=NULL, expressionSetGeneFormat=NULL)
MSigDBpath <- system.file("extdata","c4.cgn.v5.0.entrez.gmt",package="attract")
attractor.states.cutsom <- findAttractors(subset.loring.eset, "celltype", annotation="illuminaHumanv1.db", database=MSigDBpath, analysis="microarray", databaseGeneFormat="ENTREZID", expressionSetGeneFormat="PROBEID")

findCorrPartners

Determines Genes with Highly Correlated Expression Profiles to a Synexpression Group

Description

This function finds genes with expression profiles highly correlated to a synexpression group.

Usage

findCorrPartners(mySynExpressionSet, myEset, removeGenes = NULL, cor.cutoff = 0.85, ...)

Arguments

mySynExpressionSet
  SynExpressionSet object.
myEset
  ExpressionSet object.
removeGenes
  vector of probes that specify those genes who demonstrate little variability across the different celltypes and thus should be removed from downstream analysis.
cor.cutoff
  numeric value specifying the correlation cut-off.
...
  additional arguments.

Details

Genes with highly correlated profiles to the synexpression groups (e.g. $R > 0.85$) are also likely to be integral in maintaining cell type-specific differences, however due to their lack of inclusion in resources like KEGG, would not have been picked up by the first GSEA step using findAttractors.

Value

A SynExpressionSet object which stores the genes that are highly correlated with the synexpression group provided, and their average expression profile.

Author(s)

Jessica Mar
findSynexprs

Examples

data(subset.loring.eset)
attractor.states <- findAttractors(subset.loring.eset, "celltype", annotation="illuminaHumanv1.db")
remove.these.genes <- removeFlatGenes(subset.loring.eset, "celltype", contrasts=NULL, limma.cutoff=0.05)
mapk.syn <- findSynexprs("04010", attractor.states, remove.these.genes)
mapk.cor <- findCorrPartners(mapk.syn, subset.loring.eset, remove.these.genes)

findSynexprs This function finds the synexpression groups present within a core attractor pathway module.

Description
This function takes the modules that were inferred from the GSEA step using (findAttractors) and finds a set of transcriptionally coherent set of genes associated with a particular core attractor pathway, i.e. the synexpression groups.

Usage
findSynexprs(myIDs, myDataSet, cellTypeTag, removeGenes = NULL, min.clustersize = 5, ...)

Arguments
myIDs either a single character string or vector of character strings denoting the KEGG or reactome IDs of the pathway modules to be analyzed. It may also be a character codevector of gene names of a pathway if defining a custom pathway.

myDataSet AttractorModuleSet object, output of the findAttractors step. This could also be an ExpressionSet object if using a custom pathway.

cellTypeTag character string of the variable name which stores the cell-lineages or experimental groups of interest for the samples in the data set (this string should be one of the column names of pData(myEset)).

removeGenes vector of gene names that specify those genes who demonstrate little variability across the different celltypes and thus should be removed from downstream analysis.

min.clustersize integer specifying the minimum number of genes that must be present in clusters that are inferred.

... additional arguments.

Details
This function performs a hierarchichal cluster analysis of the genes in a core attractor pathway module, and uses an informativeness metric to determine the number of optimal clusters (synexpression groups) that describe the data.

Value
If a single KEGG or reactome ID is specified in pwayIds, then a SynExpressionSet object is returned. If a multiple KEGG or reactome IDs are specified, then an environment object is returned where the keys are labeled "pwayIDsynexprs" (e.g. for MAPK KEGGID = 04010, the key is pway04010synexprs). The value associated with each key is a SynExpressionSet object.
Author(s)
Jessica Mar

References
Mar, J., C. Wells, and J. Quackenbush, Identifying the Gene Expression Modules that Represent the Drivers of Kauffman’s Attractor Landscape. to appear, 2010.

Examples
data(subset.loring.eset)
attractor.states <- findAttractors(subset.loring.eset, "celltype", annotation="illuminaHumanv1.db")
remove.these.genes <- removeFlatGenes(subset.loring.eset, "celltype", contrasts=NULL, limma.cutoff=0.05)
mapk.syn <- findSynexprs("04010", attractor.states, "celltype", remove.these.genes)
top5.syn <- findSynexprs(attractor.states@rankedPathways[1:5,1], attractor.states, "celltype", removeGenes=mapk.syn)
vec.geneid <- c("GI_17999531-S","GI_17978503-A")
custom.syn <- findSynexprs(vec.geneid, subset.loring.eset, "celltype", removeGenes=remove.these.genes)

loring.eset
An ExpressionSet Object containing published data from Müller et al.

Description
This is an ExpressionSet object containing the published data from Müller et al. (NCBI GEO accession id GSE11508). The expression data set contains 11044 probes for 68 samples.

Usage
data(loring.eset)

Format
An ExpressionSet object.

Value
An ExpressionSet object containing the published data from Müller et al. (NCBI GEO accession id GSE11508). The expression data set contains 11044 probes for 68 samples.

References

See Also
exprs.dat, samp.info

Examples
data(loring.eset)
exprs.dat <- exprs(loring.eset) # gene expression matrix
Description

This function plots the average expression profile for a specific synexpression group.

Usage

plotsynexprs(mySynExpressionSet, tickMarks, tickLabels, vertLines, index=1, ...)

Arguments

mySynExpressionSet

SynExpressionSet object.

tickMarks
numeric vector of specifying the location of the tick marks along the x-axis. There should be one tick for each cell type or group.

tickLabels
character vector specifying the labels to be appear underneath the tick marks on the x-axis. These should correspond to the cell type or group names.

vertLines
numeric vector specifying the location of the vertical lines that indicate the cell type or group-specific regions along the x-axis.

index
numeric value specifying which synexpression group should be plotted.

... additional arguments.

Details

Generic plotting parameters can be passed to this function to create a more sophisticated plot, e.g col="blue", main="Synexpression Group 1".

Value

A plot showing the average expression profile for the synexpression group specified.

Author(s)

Jessica Mar

Examples

data(subset.loring.eset)
attractor.states <- findAttractors(subset.loring.eset, "celltype", nperm=10, annotation="illuminaHumanv1.db")
remove.these.genes <- removeFlatGenes(subset.loring.eset, "celltype", contrasts=NULL, limma.cutoff=0.05)
mapk.syn <- findSynexprs("04010", attractor.states, remove.these.genes)
par(mfrow=c(2,2))
pretty.col <- rainbow(3)
for( i in 1:3 ){
  plotsynexprs(mapk.syn, tickMarks=c(6, 28, 47, 60), tickLabels=c("ESC", "PRO", "NSC", "TER"), vertLines=c(12.5, 43.5, 51.5), index=i, main=paste("Synexpression Group ", i, sep=""), col=pretty.col[i])
}
removeFlatGenes

Flags a set of genes which demonstrates little variation across the cell-types or experimental groups of interest.

Description

This function uses a linear model set up in limma to assess the degree of association between celltype and a gene’s expression profile. In this way, we can flag those genes whose profiles show very little change across different celltype groups, or in other words are "flat".

Usage

```r
removeFlatGenes(eSet, cellTypeTag, contrasts = NULL, limma.cutoff = 0.05, ...)
```

Arguments

- `eSet` ExpressionSet object.
- `cellTypeTag` character string of the variable name which stores the cell-lineages or experimental groups of interest for the samples in the data set (this string should be one of the column names of pData(myEset)).
- `contrasts` optional vector of contrasts that specify the comparisons of interest. By default, all comparisons between the different groups are generated.
- `limma.cutoff` numeric specifying the P-value cutoff. Genes with P-values greater than this value are considered "flat" and will be included in the set of flat genes.
- `...` additional arguments.

Details

Flat genes are removed from the analysis after the core attractor pathway modules are first inferred (i.e. the `findAttractors` step).

Value

A vector with gene names (as defined in the eset) of those genes with expression profiles that hardly vary across different celltype or experimental groups.

Author(s)

Jessica Mar

References

limma package.

Examples

```r
data(subset.loring.eset)
remove.these.genes <- removeFlatGenes(subset.loring.eset, "celltype", contrasts=NULL, limma.cutoff=0.05)
```
**samp.info**

$samp.info$ Contains the Sample Information for the Mueller data set.

**Description**

This is sample information data frame for the samples in the Mueller data set (NCBI GEO accession id GSE11508). The data frame contains the cell type groups for the 68 samples.

**Usage**

data(samp.info)

**Format**

A data.frame object with one column of sample IDs (these are the column IDs of the exprs.dat expression matrix object) and second column indicating which cell type each sample represents.

- **ChipID**: A vector of sample IDs.
- **celltype**: A vector denoting the cell type a sample represents.

**Value**

A sample data frame for the samples in the Mueller data set (NCBI GEO accession id GSE11508). The data frame contains the cell type groups for the 68 samples.

**References**


**See Also**

exprs.dat, loring.eset

**Examples**

data(samp.info)

---

**subset.loring.eset**

An ExpressionSet Object containing published data from Müller et al.

**Description**

This is an ExpressionSet object containing a subset of the published data from Müller et al. (NCBI GEO accession id GSE11508). The expression data set contains 5522 probes for 68 samples. This ExpressionSet object was created specifically to demonstrate the functions in this package. If you’re looking for the full Müller data set, see loring.eset.

**Usage**

data(subset.loring.eset)
SynExpressionSet-class

Format

An ExpressionSet object.

Value

An ExpressionSet object containing a subset of the published data from Müller et al. (NCBI GEO accession id GSE11508). The expression data set contains 5522 probes for 68 samples.

References


See Also

exprs.dat, samp.info, loring.eset

Examples

data(subset.loring.eset)
subset.exprs.dat <- exprs(subset.loring.eset) # gene expression matrix

SynExpressionSet-class

Class SynExpressionSet

Description

This is a class representation for storing synexpression group information.

Objects from the Class

Objects are output by the function findSynexprs. Objects can also be created by using new("SynExpressionSet", ...)

Slots

groups: A list object denoting the probes or gene IDs (rnaseq) belonging to each synexpression group.

profiles: A matrix of average expression profiles for each synexpression group, each profile is stored as a row.

Methods

No methods have yet been defined with class "SynExpressionSet" in the signature.

Note

This class is described in more detail in the vignette.

Author(s)

Jessica Mar <jess@jimmy.harvard.edu>
Examples

new.synexpressionset <- new("SynExpressionSet", groups=list(), profiles=matrix(0))
Index