Package ‘ClusterJudge’

May 29, 2024

Type  Package
Title  Judging Quality of Clustering Methods using Mutual Information
Version  1.26.0
Date  2022-02-06
Author  Adrian Pasculescu
Maintainer  Adrian Pasculescu <a.pasculescu@gmail.com>
Description  ClusterJudge implements the functions, examples and other software published as an algorithm by Gibbons, FD and Roth FP. The article is called `Judging the Quality of Gene Expression-Based Clustering Methods Using Gene Annotation` and it appeared in Genome Research, vol. 12, pp1574-1581 (2002). See package?ClusterJudge for an overview.
Depends  R (>= 3.6), stats, utils, graphics, infotheo, lattice, latticeExtra, htrr, jsonlite
Suggests  yeastExpData, knitr, rmarkdown, devtools, testthat, biomaRt
License  Artistic-2.0
biocViews  Software, StatisticalMethod, Clustering, GeneExpression, GO
LazyLoad  yes
LazyData  yes
VignetteBuilder  knitr
git_url  https://git.bioconductor.org/packages/ClusterJudge
git_branch  RELEASE_3_19
git_last_commit  ff7962e
git_last_commit_date  2024-04-30
Repository  Bioconductor 3.19
Date/Publication  2024-05-29
ClusterJudge-package

Judging Quality of Clustering Methods using Mutual Information

Description

ClusterJudge implements the functions, examples and other software published as an algorithm by Gibbons, FD and Roth FP. The article is called "Judging the Quality of Gene Expression-Based Clustering Methods Using Gene Annotation" and it appeared in Genome Research, vol. 12, pp1574-1581 (2002). See package?ClusterJudge for an overview.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.
ClusterJudge is a way to judge the quality of clustering performed elsewhere on some entities. This judgement is based on some additional entity-attribute information. For example, using Gene Ontology annotated attributes offered by Saccharomyces Genome Database (SGD), it can judge the clusters of Yeast genes (i.e. entities) resulted from experiments related to the mitotic Yeast cell cycle. This is done by evaluating the mutual information between a gene membership in a cluster, and the attributes it possesses.

Author(s)

Adrian Pasculescu

Maintainer: Adrian Pasculescu <a.pasculescu@gmail.com>
**References**


---

**attribute_mut_inf**

*calculates the mutual information between each attribute of the entity.attribute pairs*

**Description**

calculates the mutual information based on the entropy. The mutual information of the pair of attributes A,B is $\text{mi}(A,b) = \text{H}(A) + \text{H}(B) - \text{H}(\text{cbind}(A,B))$ where H is the entropy.

**Usage**

```
attribute_mut_inf(entity.attribute, show.progress = FALSE, alternative.calc = FALSE)
```

**Arguments**

- **entity.attribute**
  data frame or matrix with 2 columns The assumption is that first column represents some 'entities' like gene names or gene ids. And the second column represents 'attributes' of entities (for example Gene Ontology ID 'GO:0007260' which is 'tyrosine phosphorylation of STAT protein')

- **show.progress**
  if set to TRUE will try to show periodically the estimated percentage of completion. Note that calculation of mutual information between Gene Ontology (GO) attributes takes a long time. We already have precalculated the GO mutual information for Yeast

- **alternative.calc**
  logical that if set to TRUE will try to use a direct calculation of the entropy from its definition and bypass the functions defined in the infotheo R package

**Value**

a matrix having as rownames and columnnames the attribute names from the input entity.attribute structure. The values are the mutual informations between the attributes in the region above the main diagonal. All the values below the diagonal are NA or zero.
Warning

Calculation of mutual information can be time consuming when many attributes are present for many entities

Author(s)

Adrian Pasculescu

References


See Also

mi.GO.Yeast consolidate_entity_attribute

Examples

```r
data(mi.GO.Yeast) ### is a precalculated example of mutual information for the Yeast genes
### and their Gene Ontololy (GO) attributes
str(mi.GO.Yeast)

#### the following example will generate the mutual information
#### of 100 sampled pairs of Yeast, GO associations
#### (otherwise it might take take minutes or hours to calculate the mutual information)
data(Yeast.GO.assocs) ###
entity.attribute.sampled <- Yeast.GO.assocs[sample(1:nrow(Yeast.GO.assocs),100),]
mi.GO.Yeast.sampled <- attribute_mut_inf(entity.attribute.sampled, show.progress=TRUE)
```

Description

calculates the sum of mutual information between clusters and each of the attributes For example if we note mutual information between Clusters and each Attribute i as: MI(C,Ai) (where C is the variable that has the cluster ids) then, for all attributes (assuming independence) it records the sum:

\[ \text{MI}(C,A) = \sum \text{MI}(C,A_i) = n\text{E}(C) + \sum \text{E}(A_i) - \sum \text{E}(C,A_i) \]

Then swaps elements between a random pair of clusters and repeats the above calculations. The swapping is repeated 2 times, 4 times, 8 times ... up to 2^12 times Finally a complete shuffle of clustering is applied. The resulting variation of the total mutual information as a fraction of the initial value is plotted against the increasing number of swaps (and the final suffling) A god clustering should have a pronounced decrease of the total mutual information when the number of swaps is increased.
Usage

clusterJudge(clusters, entity.attribute, plot.notes = "", plot.saveRDS.file=NULL)

Arguments

clusters          a named vectors of integers (or a factor). The names (or the levels of the factor) must match some (as many as possible) of the entities of the entity.attribute structure.
entity.attribute  data frame or matrix with 2 columns The assumption is that first column represents some ‘entities’ like gene names or gene ids. And the second column represents ‘attributes’ of entities (for example Gene Ontology ID 'GO:0007260' which is ‘tyrosine phosphorylation of STAT protein’) Usually this is a consolidated entity.attribute where the attributes with very low number of entities or with very low mutual information have been removed (see consolidate_entity_attribute and the definition of Uncertainty on attributes mutual information)
plot.notes        a string that will be added to the plot as explanation of what clustering represents.
plot.saveRDS.file if not NULL must be a string represented a file location where the plot will be saved as an RDS object. The plot can be then retrieved at any time using readRDS function.

Value

a data.frame with the number of swapps between clusters used for randomization and the total mutual information calculated after each of the sets of swaps The last value is for the full shuffle of the clusters. Since the shuffle is using the base sample function without setting a random seed, the last value will vary.

Note

a dot is printed on the console after each of the 12 sets of swaps

Author(s)

Adrian Pasculescu

References


See Also

consolidate_entity_attribute help
Examples

```r
library('yeastExpData')
data(ccyclered)

clusters <- ccyclered$Cluster
### convert from Gene names to the new standard of Saccharomyces Genome Database (SGD) gene ids
cyclered$SGDID <- sub('^S', 'S00', ccyclered$SGDID)
names(clusters) <- cyclered$SGDID
str(clusters)

data(Yeast.GO.assocs)
str(Yeast.GO.assocs)

### get the pre-calculated mutual information between all pairs of Gene Ontology (GO) attributes of Yeast genes

data(mi.GO.Yeast)
str(mi.GO.Yeast)

### consolidate (reduce) the number of attributes by removing attributes with very few entities
### and removing ones very corelated (i.e. with high mutual information)
Yeast.GO.assocs.cons <- consolidate_entity_attribute(entity.attribute = Yeast.GO.assocs
, min.entities.per.attr =3
, mut.inf=mi.GO.Yeast
, U.limit = c(0.8, 0.1, 0.001))

### apply clusterJudge for entity attributes having a very low uncertainty (0.001)
mi.by.swaps<-clusterJudge(clusters, entity.attribute=Yeast.GO.assocs.cons[["0.001"]]
, plot.notes='Yeast clusters judged at uncertainty level 0.001 - Ref: Tavazoie S,& all
, plot.saveRDS.file = 'cj.rds') ### save the plot for later use

p <- readRDS('cj.rds') ### retrieve the previous plot
pdf('cj.pdf'); plot(p); dev.off() ### plot on another device
```

---

### clusterJudge_z_score

**ClusterJudge_z_score** calculates the 'cluster judge' z-score

**Description**

calculates the 'cluster judge' z-score as defined in the reference The z-score is based on shuffling the clusters at random and calculating the total mutual information relative to the entity.attribute table. After the selected number of randomizations the mean MR and standard deviation SDR of the mutual information is used in the definition of the z.score = (MI- MIR)/SDR where MI is the mutual information of the original clustering. The higher the z.score the better the clustering. A box-and-wisker plot is generated that shows how far is the clustering versus random clustering based on the mutual information to the selected entity.attribute
Usage

\[
\text{clusterJudge}_z\text{\_score}(\text{clusters, entity.attribute, nmb.randomizations = 30, plot.saveRDS.file=NULL})
\]

Arguments

- **clusters**: a named vectors of integers (or a factor). The names (or the levels of the factor) must match some (as many as possible) of the rownames of the `entity.attribute` table.
- **entity.attribute**: data frame or matrix with 2 columns. The assumption is that first column represents some ‘entities’ like gene names or gene ids. And the second column represents ‘attributes’ of entities (for example Gene Ontology ID ‘GO:0007260’ which is ‘tyrosine phosphorylation of STAT protein’) Usually this is a consolidated `entity.attribute` where the attributes with very low number of entities or with very low mutual information have been removed (see `consolidate\_entity\_attribute` and the definition of Uncertainty on attributes mutual information).
- **nmb.randomizations**: number of randomization iterations
- **plot.saveRDS.file**: if not NULL must be a string represented a file location where the plot will be saved as an RDS object. The plot can be then retrieved at any time using `readRDS` function.

Value

- a data.frame with the number of randomization shuffles and the total mutual information calculated after each of the shuffles

Note

- a dot is printed on the console after each randomization (shuffling) step

Author(s)

Adrian Pasculescu

References


Examples

```r
library('yeastExpData')
data(ccyclered)

clusters <- ccyclered$Cluster
### convert from Gene names to the new standard of Saccharomyces Genome Database (SGD) gene ids
ccyclered$SGID <- sub('^S','S00',ccyclered$SGID)
```
data(Yeast.GO.assocs)  #### obtain associations and consolidate them at uncertainty level 0.001
Yeast.GO.assocs.cons <- consolidate_entity_attribute(entity.attribute = Yeast.GO.assocs
, min.entities.per.attr = 3
, mut.inf = mi.GO.Yeast
, U.limit = c(0.001))

#### calculate z.scores for the associations consolidated at 0.001 Uncertainty level
z.scores <- clusterJudge_z_score(clusters
, entity.attribute = Yeast.GO.assocs.cons[["0.001"]]
, nmb.randomizations = 30)

**consolidate_entity_attribute**

*removes the redundant attributes based on the mutual information between attributes*

**Description**

Calculates the mutual information between every pair of attributes and the uncertainty level (as defined in the reference ...) Plots the distribution of the uncertainty and the selected levels of uncertainty that are used as filteres.

**Usage**

```r
consolidate_entity_attribute(entity.attribute, min.entities.per.attr, mut.inf=NULL
, U.limit = c(0.8, 0.6, 0.4, 0.2, 0.1, 0.01, 0.001)
, plot.saveRDS.file=NULL)
```

**Arguments**

- **entity.attribute**
  data frame or matrix with 2 columns The assumption is that first column represent some 'entities' like gene names or gene ids. And the second column represents 'attributes' of entities (for example Gene Ontology ID 'GO:0007260' which is 'tyrosine phosphorylation of STAT protein') Usually this is a consolidated entity.attribute where the attributes with very low number of entities or with very low mutual information have been removed (see consolidate_entity_attribute and the definition of Uncertainty on attributes mutual information)

- **min.entities.per.attr**
  a number or NULL

- **mut.inf**
  the mutual information square matrix before applying any filteres based on the uncertainty or NULL

- **U.limit**
  a numerical vector

- **plot.saveRDS.file**
  if not NULL must be a string represented a file location where the plot will be saved as an RDS object. The plot can be then retrieved at any time using readRDS function.
Value

a data frame (if mut.inf argument is null) or a list of data frames. The data frames are the 'consolidated' entity.attribute structures. Where the consolidation of attributes is based on a minimum number of entities per attribute or on the mutual information between attributes.

Author(s)

Adrian Pasculescu

References


Examples

data(Yeast.GO.assocs)
data(mi.GO.Yeast)

Yeast.GO.assocs.cons <- consolidate_entity_attribute(entity.attribute = Yeast.GO.assocs
, min.entities.per.attr =3
, mut.inf=mi.GO.Yeast
 , U.limit = c(0.01, 0.001)
 , plot.saveRDS.file='this_plot.rds') ### save also the plot

str(Yeast.GO.assocs.cons)
pdf(’my_plot.pdf’) ### place the plot saved by consolidate_entity_attribute into a pdf file
plot(readRDS(’this_plot.rds’)) ###
dev.off()

---

**convert_Yeast_SGDId_2_systematic**

*converts the SGD Gene Id into the systematic name of the gene*

---

Description

converts the Saccharomyces Genome Database (SGD) Gene Id into the systematic name of the gene

Usage

convert_Yeast_SGDId_2_systematic(SGD.Ids = c("S000007287", "S000007287", "S000007287"))

Arguments

SGD.Ids a vector of characters. Each element is a string representing a gene.
download_Yeast_GO_mapping

Value

a matrix of characters containing the input SGD gene IDs as first column and the corresponding Yeast systematic gene names as second column.

Note

it requires online access to http://llama.mshri.on.ca/cgi/synergizer/serv where the conversion is performed.

It uses the httr and jsonlite R packages.

Author(s)

Adrian Pasculescu

References

Yeast - Systematic Names Assignment in http://www.yeastgenome.org/help/community

Examples

classified.genes <- convert_Yeast_SGDId_2_systematic(
    SGD.Ids = c("S000007287", "S000007287", "S000007287"))

str(classified.genes)

download_Yeast_GO_mapping

downloads the Gene Ontology attributes mapped to Yeast genes

Description

The Gene Ontology attributes are provided for example by the Saccharomyces Genome Database (SGD) as a zipped file with a wealth of information. The function filters from the file only the minimal information necessary: the Gene Ontology attribute ID (GOID) and the Yeast Gene ID (SGDID) for which the attribute applies.

Usage

download_Yeast_GO_mapping(
    yeast.GO.url = "http://downloads.yeastgenome.org/curation/literature/gene_association.sgd.gaf.gz")

Arguments

yeast.GO.url The web site address where the Yeast genes and their Gene Ontology attribute files is located.
download_Yeast_GO_mapping

Details

Only the unique associations between genes and their attributes are kept. In the original downloaded file there might be redundancies due to the different types of evidences used in the associations.

The downloaded file is expected to be in a tab delimited format with comment lines that start with the exclamation (!) character. The commented lines are ignored. The expected column names can be obtained by looking at the function code.

Value

a data.frame with two columns: SGDID - the ids of the Yeast gene names (these are the 'entities')
GOID - the ids of the Gene Ontology attributes corresponding to the above genes

Note

since the download is time consuming, this package provides also as a dataset called Yeast.GO.assocs containing the associations already downloaded as of on February 2017.

For other species one can use specialized R and Bioconductor packages such as biomaRt (see the commented part in the example section).

Author(s)

Adrian Pasculescu

References


See Also

Yeast.GO.assocs help, ~~~

Examples

Yeast.GO.assocs <- download_Yeast_GO_mapping()

# For other species one can use specialized R and Bioconductor packages such as biomaRt
# as in the following `toy` commented example:
#library(biomaRt)
#rn <- useDataset("rnorvegicus_gene_ensembl", mart=useMart("ensembl"))
### exemplify for a limited set of genes
#rgd.symbol=c("As3mt", "Borcs7", "Cyp17a1", "Wbp11", "Sfxn2", "Arl3")
#entity.attr=getBM(attributes=c('rgd_symbol', 'go_id'), filters='rgd_symbol', values=rgd.symbol, mart=rn)
download_Yeast_Go_terms

*downloads the description of Gene Ontology attributes*

**Description**

The full description of Gene Ontology attributes are provided for example by the Saccharomyces Genome Database (SGD).

**Usage**

```r
download_Yeast_Go_terms(
  url.GO.terms = "http://downloads.yeastgenome.org/curation/literature/go_terms.tab")
```

**Arguments**

- `url.GO.terms` The web site address where the description of Gene Ontology attribute files is located.

**Details**

The file is expected to be in tab delimited format having the following columns: `GOID`, `GO_Term`, `GO_Aspect`, `GO_Term_Definition`

**Value**

A data.frame containing the information provided by the Gene Ontology consortium

**Note**

The function is optional. It only helps user in understanding what is the meaning of the GO attributes for the Yeast examples.

**Author(s)**

Adrian Pasculescu

**References**


**See Also**

`download_Yeast.GO_mapping`

**Examples**

```r
GO.terms <- download_Yeast_Go_terms()
str(GO.terms)
```
entities_attribute_stats

presents basic statistics on the number of entities per attribute

Description

Plots the density distribution of the number of entities per attribute and shows what is the number of attributes proposed to be ignored (and the number of attributes that will be kept)

Usage

entities_attribute_stats(entity.attribute, min.entities.per.attr = NULL, entity.space.name = "Yeast genes", attribute.space.name = "Gene Ontology", plot.saveRDS.file=NULL)

Arguments

entity.attribute
  data frame or matrix with 2 columns The assumption is that first column represents some ‘entities’ like gene names or gene ids. And the second column represents ‘attributes’ of entities (for example Gene Ontology ID ‘GO:007260’ which is ‘tyrosine phosphorylation of STAT protein’)

min.entities.per.attr
  a number : the minimum number of entities per attribute accepted

entity.space.name
  a string that will be presented on the plot representing the meaning of the entities

attribute.space.name
  a string that will be presented on the plot representing the meaning of the attributes

plot.saveRDS.file
  if not NULL must be a string represented a file location where the plot will be saved as an RDS object. The plot can be then retrieved at any time using readRDS function.

Details

The attributes that appear only once or just a in very few entities do not bring additional information. In general there are many such ‘non-informative’ attributes. Thus it’s good to know the proportion of attributes that will be still kept if we impose a minimum number of entities per attribute.
mi.GO.Yeast

Value

A number: which is either the input value of the min.entities.per.attr or, in case min.entities.per.attr is null, a proposed min.entities.per.attr threshold. The assumption is that attributes characterizing just one entity are the most frequent. The proposed threshold is the minimum number of entities per attribute whose frequency matches 1/3 of the above maximum frequency.

Author(s)

Adrian Pasculescu

References


Examples

data(Yeast.GO.assocs)
min.entities.per.attr <- entities_attribute_stats(entity.attribute= Yeast.GO.assocs,
min.entities.per.attr=NULL,
entity.space.name='Yeast genes',
attribute.space.name='Gene Ontology')

mi.GO.Yeast

d precalculated mutual information between Gene Ontology attributes
of Yeast genes

Description

precalculated mutual information between Gene Ontology attributes of Yeast genes

Usage

data("mi.GO.Yeast")

Format

The format is: num [1:2266, 1:2266] NA NA NA NA NA NA NA NA NA NA ... - attr(*, "dimnames")=List of 2 ..$ : chr [1:2266] "GO:0000001" "GO:0000002" "GO:0000009" "GO:0000011" ...
..$ : chr [1:2266] "GO:0000001" "GO:0000002" "GO:0000009" "GO:0000011" ...

Details

For convenience this data set was pre-generated using attribute_mut_inf from this package
validate_association

Value

This data loads a symetric matrix of mutual information values calculated between pairs of Gene Ontology attributes of Yeast genes

References


See Also

attribute_mut_inf

Examples

data(mi.GO.Yeast)
str(mi.GO.Yeast)

validate_association(validate the associations between entities and attributes)

Description

verifies if the input structure has two columns, if there are no NAs or NULLs and if there are no duplicated associations

Usage

validate_association(entity.attribute, message = TRUE)

Arguments

data frame or matrix with 2 columns The assumption is that first column represents some 'entities' like gene names or gene ids. And the second column represents 'attributes' of entities (for example Gene Ontology ID 'GO:0007260' which is 'tyrosine phosphorylation of STAT protein')

message a logical. If set to TRUE will print a message in case of successful validation.

Value

Returns TRUE and print "Validation OK!" message in case of success. Otherwise stops with an explanatory message.

Author(s)

Adrian Pasculescu
References


Examples

data(Yeast.GO.assocs)

validate_association(Yeast.GO.assocs)

Yeast.GO.assocs  Gene Ontology attributes associated to Yeast Gene entities

Description

Gene Ontology attributes associated to Yeast Gene entities

Usage

data("Yeast.GO.assocs")

Format

A data frame with 70487 observations on the following 2 variables.

- SGDID  a character vector
- GOID  a character vector

Value

This data loads a data frame with two columns: Yeast gene ids and their Gene Ontology ids

References


Examples

data(Yeast.GO.assocs)
str(Yeast.GO.assocs)
Index

* Yeast.GO.assocs
  attribute_mut_inf, 3
  clusterJudge_z_score, 6
* attribute_mut_inf
  clusterJudge, 4
  consolidate_entity_attribute, 8
* cluster.Judge
  clusterJudge_z_score, 6
* consolidate_entity_attribute
  clusterJudge, 4
* datasets
  mi.GO.Yeast, 14
  Yeast.GO.assocs, 16
* mi.GO.Yeast
  attribute_mut_inf, 3
* package
  ClusterJudge-package, 2
* validate_association
  attribute_mut_inf, 3

attribute_mut_inf, 3, 15

ClusterJudge (ClusterJudge-package, 2
clusterJudge, 4
ClusterJudge-package, 2
clusterJudge_z_score, 6
consolidate_entity_attribute, 4, 5, 8
convert_Yeast_SGDId_2_systematic, 9
download_Yeast_GO_mapping, 10, 12
download_Yeast_Go_terms, 12

tentities_attribute_stats, 13

help, 5, 11

mi.GO.Yeast, 4, 14
validate_association, 15

Yeast.GO.assocs, 11, 16