Package 'Salsa'

November 19, 2020

Type Package

Title Data mining facilities for Capsicum gene expression profiles

Version 0.4

Date 2020-11-19

Author Octavio Martinez and Christian Escoto-Sandoval

Maintainer `<octavio.martinez@cinvestav.mx>`

Description Salsa is an auto-contained R package with data and functions to explore the transcriptome of 12 chili pepper (Capsicum annum L.) accessions during fruit development. Data consist of curated gene expression profiles for more than 29000 genes sampled every ten days from 0 to 60 Days After Anthesis (DAA), incorporating also gene identification plus Gene Ontology (GO) annotations. Functions include extensive resources for expression pattern searching and plotting facilities, GO enrichment analysis as well as methods to evaluate expression pattern differences in sets of genes and accessions.

License GPL-3

LazyData TRUE

R topics documented:

`Salsa-package` *Data mining facilities for Capsicum gene expression profiles*

Description

Salsa is an auto-contained R package with data and functions to explore the transcriptome of 12 chili pepper (Capsicum annum L.) accessions during fruit development. Data consist of curated gene expression profiles for more than 29000 genes sampled every ten days from 0 to 60 Days After Anthesis (DAA), incorporating also gene identification plus Gene Ontology (GO) annotations. Functions include extensive resources for expression pattern searching and plotting facilities, GO enrichment analysis as well as methods to evaluate expression pattern differences in sets of genes and accessions.

Details

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

Index: This package was not yet installed at build time.

Author(s)

Octavio Martinez and Christian Escoto-Sandoval

Maintainer: <octavio.martinez@cinvestav.mx>

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

Examples

```
## Assume that the interest is on genes
# of Transcription Factors (TF) in two accessions
# "CM" (Domesticated) and "QU" (Wild)
# Isolate SEPs with the genes of interest
temp.TF.CM <- get.SEP(acc.key="CM", isTF=TRUE)
temp.TF.QU <- get.SEP(acc.key="QU", isTF=TRUE)
temp.TF.CM.QU <- get.SEP(acc.key=c("CM", "QU"), isTF=TRUE)

# Obtain a summary of the content of the
# three SEPs
SEP.summary(temp.TF.CM.QU, temp.TF.CM,
temp.TF.QU, conf.level=0.99)

# Plot the corresponding SEPs to have an idea
# of mean expression per time (DAA)
SEPs.plot(list(temp.TF.CM.QU, temp.TF.CM,
temp.TF.QU), colors=c("grey", "red", "blue"),
conf.level=0.99)
title(main="Transcription Factors in QU and CM")
legend("topright", legend=c("CM and QU", "CM", "QU"),
col=c("grey", "red", "blue"), pch=1, lwd=2)

# Using the previously Isolated SEPs of all
# TFs, Isolate only the ones that have "GATA"
# in their description
temp.GATA.CM <- get.SEP(descr="GATA", previous.sep=temp.TF.CM)
temp.GATA.QU <- get.SEP(descr="GATA", previous.sep=temp.TF.QU)

# Perform a test of Euclidean distances between
# SEPs of GATA TFs in accessions CM and QU
analyze.2.SEPs(temp.GATA.CM, temp.GATA.QU, conf.level=0.99)

# Clean the temporal objects created
rm(list=c(ls(patt="temp.TF"), ls(patt="temp.GATA")))

# Obtain the gene ids of all
# GATA TFs which are expressed
# in all 12 accessions
ids.GATA <- get.ids(descr="GATA", ExistInAll=TRUE, isTF=TRUE)
length(ids.GATA)

# Look for GO terms of aspect BP
# which contain "cell cycle",
# "negative regulation", and "mitotic"
# within their description.
get.GO.terms.by.desc("cell cycle",
"negative regulation",
"mitotic", only.aspect="BP")

# Now perform a GO enrichment
# analysis having as target the
# genes in "ids.GATA"
analyze.GO(ids.GATA, aspect="BP", aspect.id=1814)

# See two of the descriptions of the
# GATA TFs in the gene data.frame.
head(gene[is.element(gene$id, ids.GATA), ], 2)

# Remove the temp object
rm("ids.GATA")

# Finally browse in NCBI the data about
# the first one of these two genes.
browse.gene(ProtId="XP_016577838.1")

# Also browse in GO the meaning of the
# GO term "GO:0045930"
# (previously analyzed)
browse.GO(GO="GO:0045930")
```

`acc` *Accessions*

Description

Accessions (genotypes) available in the data

Usage

```
data("acc")
```

Format

A data frame with 12 observations on the following 3 variables.

`acc.key` a character vector of two letters coding the 12 accessions.

`acc.type` a character vector; D = Domesticated, W = Wild, C = An F1 cross between D and W parents.

`acc.name` a character vector; Common name of the accession.

Details

All 12 accessions are of the specie Capsicum annuum, the 4 W accessions belong to the sub-specie var. glabriusculum (chle piquin or chiltepin).

Source

Cinvestav Irapuato, Mexico.

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

Examples

```
acc[acc$acc.type=="W", ] # Data for Wild accessions
```

`all.GO` *Gene Ontology (GO) annotations available*

Description

Aspects, keys and descriptions of GO annotations for the chili pepper genes.

Usage

```
data("all.GO")
```

Format

A data frame with 4518 observations on the following 4 variables.

`aspect` a character vector with values "BP" = Biological processes, "CC" = Cell Component or "MF" = Molecular Function.

`aspect.id` a numeric vector with identifiers for each GO annotation.

`GO` a character vector with the GO identifiers.

`GO.desc` a character vector with the description for each GO term.

Details

This data.frame is used in GO enrichment analyses.

Source

http://geneontology.org/

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

Examples

```
# A random row of the data.frame
all.GO[sample(c(1:nrow(all.GO)), size=1), ]
```

`analyze.2.SEPs` *Test two SEPs trough Euclidean distances*

Description

Mean Euclidean distances within and between the two SEPs are tested via the t-test to decide if there is a global difference in the expression profiles contained within the two sets of SEPs. Null hypothesis is that the mean distances within and between SEPs are equal, while the one-tail alternative hypothesis is that the mean distance between SEPs is larger than the mean distance within SEPs. This last possibility implies that the two SEPs are globally different.

Usage

```
analyze.2.SEPs(sep1, sep2, conf.level = 0.95, do.print = TRUE)
```

Arguments

`sep1`	A SEP `data.frame` obtained with the function `get.SEP`
`sep2`	A SEP `data.frame` obtained with the function `get.SEP`
`conf.level`	Confidence level to perform the `t-test`
`do.print`	`logical`. If `TRUE` the function prints the results, otherwise invisible returns a `data.frame` with main results

Details

Let’s `n1` and `n2` be the number of rows of `sep1` and `sep2`, respectively. Then the number of distances within SEPs is given by `(n1*(n1-1/2)) + (n2*(n2-1/2))`, while the number of distances between SEPs is given by `n1*n2`.

Value

The function prints results of the t-test if `do.print = TRUE` and invisible returns a numeric vactor with named components

`n.sep1`	Number of SEPs in sep1
`n.sep2`	Number of SEPs in sep2
`t.value`	Value of the t statistic
`df`	Degrees of freedom for the test
`p-value`	P value reached by the test
`conf.level`	Confidence level
`L.CL`	Lower Confidence Limit of the Confidence Interval
`mean.dis.bet`	Mean distance between SEPs
`mean.dis.wit`	Mean distance within SEPs

Note

This univariate test solves the problem of performing multiple tests (one for each one of the seven time points sampled), but is sensitive to outliers.

Author(s)

O. Martinez

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

See Also

`get.SEP`, `SEP.summary`.

Examples

```
# SEPs of gene with id=3 in "D" versus
# SEPs of gene with id=3 in "W" or "C"
analyze.2.SEPs(sep1=get.SEP(id=3, acc.type="D"),
sep2=get.SEP(id=3, acc.type=c("W", "C")))

# Test of SEPs for two different genes
analyze.2.SEPs(sep1=get.SEP(id=3),
sep2=get.SEP(id=19))
```

`analyze.all.GO` *General GO enrichment analysis*

Description

Performs a general GO enrichment analysis for a set of genes identified by their ids and for all the terms in a given GO aspect. The analysis is performed using the Fisher's exact test. Optionally filters results per a False Discovery Rate (FDR).

Usage

```
analyze.all.GO(ids, aspect = "BP", only.FDR.le = NULL)
```

Arguments

`ids`	(`numeric`) Set of target gene identifiers to form the 2 x 2 contingency table to be tested.
`aspect`	Must be one of the valid aspects of GO: "BP" (Biological Process) or "CC" (Cell Component) or "MF" (Molecular function).
`only.FDR.le`	`NULL` (if no filtering of the results is desired) or a number between 0 and 1 with the threshold for the maximum False Discovery Rate (FDR) to be reported.

Details

Using the `ids` and EACH ONE of the terms of the GO aspect, a contingency table is formed by crossing the criteria Annotated and Target (in each case FALSE or TRUE) and the resulting 2 x 2 contingency table is analyzed using Fisher's exact test under the null hypothesis of criteria independence. Results are returned in a `data.frame` with rows corresponding to each GO aspect (if not filtered by FDR, i.e., if `only.FDR.le = NULL` (the default) or only with rows that fulfill the FDR asked. If there are no rows that fulfill the FDR criterion, a `data.frame` with 0 rows is returned without warning.

Value

A data.frame with the following columns

`aspect`	GO aspect analyzed.
`aspect.id`	Numerical identifier of the GO aspect.
`desc`	Description of the GO aspect analyzed.
`odds`	Odds ratio for the contingency table.
`P`	P value for the test of independence.
`AnnTarg`	Annotated with aspect and in Target set
`NotAnnTarg`	Not Annotated with aspect and in Target set
`AnnNotTarg`	Annotated with aspect and Not in Target set
`NotAnnNotTarg`	Not Annotated with aspect and Not in Target set

Note

Running can take some time, be patient. The total of genes taken into account corresponds to the total of genes with valid expression in the data and which are annotated in the GO aspect.

Author(s)

O. Martinez

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

See Also

`analyze.GO`, `get.ids`, `get.GO.terms.by.desc`, `browse.GO`.

Examples

```
# A dummy set of ids that will give
# "significant" results.
my.temp.ids <- c(unique(get.genes.by.GO.desc(
"mitochondrial outer membrane translocase complex"
, only.aspect="CC")$id), gene$id[1:26])

# Calling the function with that set of ids
# and filtering by a minimum FDR of 1%
# (can take some time)
my.temp.res <- analyze.all.GO(ids=my.temp.ids,
aspect = "CC", only.FDR.le = 0.01)

# See some of the results
head(my.temp.res)
tail(my.temp.res)

# Clean temporary objects
rm(my.temp.ids, my.temp.res)
```

`analyze.GO` *GO enrichment analysis*

Description

Performs a GO enrichment analysis for a set of genes identified by their ids. The analysis is performed using the Fisher's exact test.

Usage

```
analyze.GO(ids, aspect, aspect.id, print.all = TRUE)
```

Arguments

`ids`	(`numeric`) Set of target gene identifiers to form the 2 x 2 contingency table to be tested.
`aspect`	Must be one of the valid aspects of GO: "BP" (Biological Process) or "CC" (Cell Component) or "MF" (Molecular function).
`aspect.id`	Numeric identifier of the aspect.
`print.all`	`logic`; `TRUE` will print the results, `FALSE` will only invisibly return the results.

Details

A contingency table is formed by crossing the criteria Annotated and Target (in each case FALSE or TRUE) and the resulting 2 x 2 contingency table is analyzed using Fisher's exact test under the null hypothesis of criteria independence. By default results are printed by the function and a data.frame with the results is invisibly returned.

Value

A data.frame with the following columns

`aspect`	GO aspect analyzed.
`aspect.id`	Numerical identifier of the GO aspect.
`desc`	Description of the GO aspect analyzed.
`odds`	Odds ratio for the contingency table.
`P`	P value for the test of independence.
`AnnTarg`	Annotated with aspect and in Target set
`NotAnnTarg`	Not Annotated with aspect and in Target set
`AnnNotTarg`	Annotated with aspect and Not in Target set
`NotAnnNotTarg`	Not Annotated with aspect and Not in Target set

Note

The total of genes taken into account corresponds to the total of genes with valid expression in the data and which are annotated in the GO aspect.

Author(s)

O. Martinez

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

See Also

`analyze.all.GO`, `get.ids`, `get.GO.terms.by.desc`, `browse.GO`.

Examples

```
# An example with the first 1000 gene identifiers
# (a fully arbitraty set)
analyze.GO(ids=gene$id[1:1000])

# Performs a GO enrichment analysis in
# a set of 100 genes taken at random.
# Also GO aspect and term are random
temp.asp <- sample(c("BP", "CC", "MF"), size=1)
analyze.GO(ids=sample(gene$id, size=100), aspect=temp.asp, aspect.id=sample(all.GO$aspect.id[all.GO$aspect=
rm(temp.asp)

# Same random experiment but this time
# with 1000 genes
```

```
temp.asp <- sample(c("BP", "CC", "MF"), size=1)
analyze.GO(ids=sample(gene$id, size=1000), aspect=temp.asp, aspect.id=sample(all.GO$aspect.id[all.GO$aspect
rm(temp.asp)
```

`browse.gene` *Opens a window in your browser*

Description

The input of the function is a protein identifier for a gene in the data (`ProtId` in the `data.frame` gene). It will cause to open a window containing the information of the NCBI that exist about that protein.

Usage

```
browse.gene(ProtId = "XP_016567627.1")
```

Arguments

`ProtId`	One of the protein identifiers existent in the `data.frame` gene

Value

The function is used by its side effect (open a window in your web browser)

Author(s)

O. MArtinez

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

See Also

`gene`, `browse.gene`, `get.genes.by.desc`.

Examples

```
# If you have a gene of interest, say
gene[gene$id==35802,]
# You can obtain information of
# that protein in the NCBI:
browse.gene("XP_016555511.1")
```

browse.GO *Opens an URL in the page corresponding to the GO identifier*

Description

Opens in your web browser the URL in THE GENE ONTOLOGY RESOURCE page corresponding to the GO identifier in the input.

Usage

```
browse.GO(GO = "GO:0000002")
```

Arguments

`GO`	Valid `GO` are the ones in the column `GO` of the data.frame `all.GO`

Details

See http://www.informatics.jax.org/vocab/gene_ontology/

Value

The function is used by its side effect (to open a web page).

Note

This function is a convenient wrapper for the function `browseURL`.

Author(s)

O. Martinez

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

See Also

`get.GO.terms.by.desc`

Examples

```
# Opens the page corresponding to GO:0000003 (Reproduction)
# (in your web browser)
browse.GO("GO:0000003")
```

DAA *Days After Anthesis*

Description

Time points where all accessions were sampled in all 12 accessions.

Usage

```
data("DAA")
```

Format

The format is: num [1:7] 0 10 20 30 40 50 60

Details

DAA is used for plotting and analysis of data.

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

Examples

```
DAA # Just the time points of sampling.
```

expected *Expected value of a contingency table*

Description

Returns the expected value for a contingency table under the hypothesis of independence.

Usage

```
expected(x)
```

Arguments

x	A matrix (of observed values)

Details

Mainly for internal use.

Value

A matrix of the same size of x with the expected values under independence.

Author(s)

O. Martinez

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

See Also

`get.mat.from.ids`, `analyze.GO`

Examples

```
# An arbitrary matrix
matrix(c(1:6), nrow=2, ncol=3)
# And its expected value under independence
expected(matrix(c(1:6), nrow=2, ncol=3))
# Used internally as
expected(get.mat.from.ids())
```

`FPKM.expr` *Expression data per accession and time in FPKM units*

Description

Contains mean expression data for each gene (`id`) and each accession (`acc.key`) at each one of the seven times sampled (`mT00` to `mT60`) in FPKM (fragments per kilobase of exon model per million reads mapped).

Usage

```
data("FPKM.expr")
```

Format

A data frame with 313919 observations on the following 14 variables.

`id` a numeric vector; gene numeric identifier.

`acc.key` a character vector; two letters code for accessions.

`acc.type` a character vector; D = Domesticated, W = Wild.

`model` a character vector; six letters code for the expression model (see details).

`ExistInAll` a logic; If TRUE the gene was expressed in all 12 accessions.

`mT00` a numeric vector; FPKM expression at 0 DAA.

`mT10` a numeric vector; FPKM expression at 10 DAA.

`mT20` a numeric vector; FPKM expression at 20 DAA.

`mT30` a numeric vector; FPKM expression at 30 DAA.

`mT40` a numeric vector; FPKM expression at 40 DAA.

`mT50` a numeric vector; FPKM expression at 50 DAA.

`mT60` a numeric vector; FPKM expression at 60 DAA.

`mean.over.time` a numeric vector; mean over times of gene expression.

`coded.expr.level` a numeric vector; code for expression levels, see details.

Details

FPKM (fragments per kilobase of exon model per million reads mapped) is a normalised estimation of gene expression based on RNA-seq data. Column `model` is formed by all the 729 permutations of the letters D, S, I, meaning that the expression in the corresponding interval was Decreasing, Steady or Increasing, respectively. Column `coded.expr.level` codes mean expression level with the following meaning: 1 expression less or equal to 1 FPKM, 2 more than 1 but less than 10 FPKM, 3 more than 10 but less than 100 FPKM, 4 more than 100 but less than 1000 FPKM and 5 more than 1000 FPKM.

Source

Cinvestav Irapuato, Mexico.

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

Examples

```
# A random row of the data.frame
FPKM.expr[sample(c(1:nrow(FPKM.expr)), size=1), ]
# A gene with high expression level (an average expression larger than 1000 FPKM)
head(FPKM.expr[FPKM.expr$coded.expr.level==5,], 1)
# What is codded by this gene?
gene[gene$id==102, ]
# Plot of the expression of gene with
# id=102 in accession "AS"
plot(DAA, FPKM.expr[(FPKM.expr$id==102)&(FPKM.expr$acc.key=="AS"), 6:12], type="b", ylab="Expression in FPKM
# A line marking the mean expression.
abline(h=2413.108, col="grey")
```

`gene` *Identifiers for genes*

Description

For each one of the expressed genes this data frame contains protein identifiers and descriptions.

Usage

```
data("gene")
```

Format

A data frame with 29946 observations on the following 4 variables.

`id` a numeric vector; numeric gene identifier.

`ProtId` a character vector; protein identifier.

`Prot.Desc` a character vector; protein description

`isTF` a logical vector; Is the coded protein a Transcription Factor?

Details

Descriptors are based in the reference genome CM334 v1.6.

Source

http://passport.pepper.snu.ac.kr/?t=PGENOME

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

Examples

```
# A random row of the data.frame
gene[sample(c(1:nrow(gene)), size=1), ]

# Show data for some Transcription Factors.
head(gene[gene$isTF==TRUE,])
```

`gene.summary` *Graphic and numeric summary of a gene*

Description

Gives a plot as well as a summary of mean standardized expression over time for a gene in the types of accessions where it was expressed (D, W and C).

Usage

```
gene.summary(id, leg.pos, T.col, D.col, W.col, C.col)
```

Arguments

`id`	Identifier of the gene.
`leg.pos`	Possition of the legend in the plot: one of `bottomright`, `bottom`, `bottomleft`, `left`, `topleft`, `top`, `topright`, `right` and `center`.
`T.col`	Color to be used for plotting the expression mean of all accessions.
`D.col`	Color to be used for plotting the expression mean of D accessions.
`W.col`	Color to be used for plotting the expression mean of W accessions.
`C.col`	Color to be used for plotting the expression mean of C accessions.

Details

This function is used mainly for its side effect: plotting of mean expression per group and summary of mean expression. But also it invisibly returns a list with the main results of the summary of mean expression.

Value

Invisibly returns a list with the following components

`IdDesc`	Identification of the gene.
`Acc.x.group`	Number of accessions where the gene was expressed (per group).
`Mean.r.x.group`	Mean expression correlation (r) of SEPs within each group.
`StatMaxExpx.group`	A matrix giving the mean and standard deviation (S) for the time at which the maximum expression was reached for each accession group.

Note

The plot and summary is obtained from the data in SEP.id which in turn summarizes the standardized expression profile for each one of the genes.

Author(s)

Octavio Martinez.

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

See Also

`SEP.id`

Examples

```
# Summary for the gene with highest mean expression in the dataset
# See: FPKM.expr[FPKM.expr$mean.over.time>17000,]
gene.summary(id = 7975, leg.pos = "topleft")

# Obtains a summary of one gene
# selecting it at random
gene.summary(id=sample(SEP.id$id, size=1))
```

`get.genes.by.desc` *Select genes that fulfill a description criteria*

Description

Shows the rows of the gene `data.frame` which include into `Prot.Desc` the character chains pass to the function.

Usage

```
get.genes.by.desc(...)
```

Arguments

`...`	One or more character strings that you want to be part of the `Prot.Desc` for the gene.

Details

All the character strings pass in the input must be present in the `Prot.Desc`, otherwise a `NULL` value is returned by the function.

Value

Either, a `NULL` value (when there are not `Prot.Desc` that fulfill the input) or a subset of the gene `data.frame` which rows in which the `Prot.Desc` column includes all terms in the input.

Note

For general descriptions the `data.frame` returned can be large.

Author(s)

O. Martinez

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

See Also

`gene`

Examples

```
# A case where the function cannot
# find a Prot.Desc that fulfills
# the search
get.genes.by.desc("my favorite gene")

# Obtain a subset of genes which includes
# "WRKY" in their description
head(get.genes.by.desc("WRKY"))
# How many of those genes are there?
nrow(get.genes.by.desc("WRKY"))

# Having also ""factor 53"
head(get.genes.by.desc("WRKY", "factor 53"))
# How many of those?
nrow(get.genes.by.desc("WRKY", "factor 53"))

# How many containing "DNA polymerase"
nrow(get.genes.by.desc("DNA polymerase"))

# A more concise search
get.genes.by.desc("DNA polymerase I")
```

`get.genes.by.GO.desc` *Select genes annotated with GO terms*

Description

Obtains a `data.frame` of all genes annotated with one or more Gene Ontology (GO) aspects and terms, returning an informative `data.frame`.

Usage

```
get.genes.by.GO.desc(..., only.aspect = NULL)
```

Arguments

`...`	One or more character strings that you want to be part of the description of the Gene Ontology term description.
`only.aspect`	If not `NULL` it must contain the code for a single GO aspect that you want to look for, say BP = Biological Process, CC = Cell Component or MF = Molecular function.

Details

All the character strings pass to the function must be part of one or more GO term descriptions, otherwise a `NULL` value is return.

Value

Either, a `NULL` value (when there are not GO terms that fulfill the input) or a`data.frame` with the following columns:

`id`	Gene numeric identifier
`aspect`	GO aspect
`GO`	GO term
`GO.desc`	Description of the GO term
`ProtId`	Identifier of the protein coded by the gene
`Prot.Desc`	Description of the protein coded by the gene
`isTF`	logical; Is the gene annotated as a Transcription Factor

Note

The output `data.frame` can be redundant, having the same genes various times, depending on how many GO aspects each gene is annotated.

Author(s)

O. MArtinez

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

See Also

`get.GO.terms.by.desc`, `all.GO`, `analyze.GO`, `browse.GO`, `GO.annot`.

Examples

```
# A case where the function cannot
# find any gene fulfilling the search
# and thus returns NULL
get.genes.by.GO.desc("my favorite GO term")

# Obtain a set of genes which include
# "mitotic cell cycle" in their GO annotations
# (see only the fist two rows)
head(get.genes.by.GO.desc("mitotic cell cycle"),2)

# How many cases are there?
nrow(get.genes.by.GO.desc("mitotic cell cycle"))
# Note: that one gene can be annotated in
# various of the GO terms:
# How many different genes ("id"s)
length(unique(get.genes.by.GO.desc("mitotic cell cycle")$id))
# How many different GO terms ("id"s)
length(unique(get.genes.by.GO.desc("mitotic cell cycle")$GO))

# Using the only.aspect parameter
nrow(get.genes.by.GO.desc("wall", only.aspect="BP"))
nrow(get.genes.by.GO.desc("wall", only.aspect="CC"))
nrow(get.genes.by.GO.desc("wall", only.aspect="MF"))
```

`get.GO.terms.by.desc` *Finds GO terms that fulfill the input*

Description

Finds all Gene Ontology (GO) terms that include in their descriptions all words pass to the function in the input.

Usage

```
get.GO.terms.by.desc(..., only.aspect = NULL)
```

Arguments

`...`	One or more words (character strings) that must be present in the GO description to be present in the output.
`only.aspect`	If not `NULL` it must contain the code for a single GO aspect that you want to look for, say BP = Biological Process, CC = Cell Component or MF = Molecular function.

Value

Either, NULL when the function does not find one or more of the terms pass in the input, of a data.frame with columns

aspect	GO aspect
aspect.id	Numeric identifier of the GO term within aspect
GO	GO identifier
GO.desc	GO descrioption

In this data.frame all cases of GO.desc include all terms pass to the function in the input.

Author(s)

O. Martinez

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

See Also

get.genes.by.GO.desc, all.GO, analyze.GO, browse.GO, GO.annot.

Examples

```
# When the functio does not find
# the term pass in the input:
get.genes.by.GO.desc("My favorite GO term")

# Obtains a set of GO terms which includes
# "mitotic cell cycle" in their description
# Show just the first two of them
head(get.GO.terms.by.desc("mitotic cell cycle"),2)
# How many of those?
nrow(get.GO.terms.by.desc("mitotic cell cycle"))

# A more concise search
# (asking for two terms)
get.GO.terms.by.desc("mitotic cell cycle",
"negative regulation")

# Using a specific aspect
# (only.aspect "BP", "MF" or "CC")
get.GO.terms.by.desc("wall", only.aspect="CC")
get.GO.terms.by.desc("wall", only.aspect="MF")
```

get.ids *Obtains a set of gene identifiers*

Description

Obtains a set of gene identifiers that fulfill one or more of the criteria pass in the input.

Usage

```
get.ids(descr = NULL, ExistInAll = NULL, isTF = NULL)
```

Arguments

descr	One or more words (character strings) that must be present in the Prot.Desc of the gene data.frame to be presented in the output of the function.
ExistInAll	A logical variable; if TRUE only genes that were expressed in all 12 accessions will be present in the output.
isTF	A logical variable; if TRUE only genes that are annotated as Transcription Factors will be present in the output.

Details

One or more of the criteria in the input must be not NULL, otherwise the function will produce an error.

Value

Either NULL, when no gene fulfills all criteria pass in the input, or a integer vector including all gene idendifiers of the genes that fulfill all criteria pass to the function in the input.

Author(s)

O. Martinez

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

See Also

gene, get.genes.by.desc, browse.gene.

Examples

```
# Returns NULL
# (because descr is not found in
# any gene description)
get.ids(descr="My favorite gene")

# Returns the ids of all genes expressed
# in the 12 accessions
# get.ids(ExistInAll=TRUE)
# See the first 6 of them
head(get.ids(ExistInAll=TRUE))
# How many?
length(get.ids(ExistInAll=TRUE))

# How many genes are annotated as
# Transcription Factors?
length(get.ids(isTF=TRUE))

# And how many of those include in
# the description the character
# strings "WRKY"
length(get.ids(descr="WRKY"))

# See some of the Prot.Desc of those genes
gene$Prot.Desc[is.element(gene$id, tail(get.ids(descr="WRKY")))]

# Example using multiple criteria
# of selection
get.ids(descr=c("zipper", "HAT"), ExistInAll=TRUE, isTF=TRUE)
```

get.mat.from.ids *2 by 2 Contingency Table*

Description

Returns a 2 by 2 Contingency Table to be used in GO enrichment analysis (mainly for internal use)

Usage

```
get.mat.from.ids(s.t, aspect, aspect.id)
```

Arguments

s.t	Vector of integers. The set of gene identifiers in the group which is the target of the analysis.
aspect	Character string. The GO aspect (BP, CC or MF)
aspect.id	Integer. Numeric identifier of the aspect in `all.GO`.

Details

Used by `analyze.GO`.

Value

A 2 by 2 matrix suitable for GO enrichment analysis.

Note

Used by `analyze.GO`.

Author(s)

Octavio Martinez.

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

See Also

`analyze.GO`, `expected`

Examples

```
get.mat.from.ids(s.t = SEP$id[1:1000], aspect = "BP", aspect.id = 1)
```

`get.SEP` *Obtains a SEP data frame*

Description

Obtains a Standardized Expression Profile (SEP) data frame with rows that fulfill various criteria determined by the input.

Usage

```
get.SEP(ids, descr, acc.key, acc.type, model, ExistInAll,
 TimeMaxExp, isTF, coded.expr.level, previous.sep)
```

Arguments

`ids`	`integer`. A vector with the gene identifiers (`id`) that must be present in the SEP.
`descr`	One or more words (character strings) that you want to be part of the `Prot.Desc` for the genes in the output.
`acc.key`	One or more `acc.key` (key for the accessions) that you want to be in the output SEP.
`acc.type`	One or more `acc.type` (key for the accession types) that you want to be in the output SEP.
`model`	One or more `model` (key for SEP models) that you want to be in the output SEP.
`ExistInAll`	`logical`. If `TRUE` only cases of genes expressed in all 12 accession will be present in the output SEP.
`TimeMaxExp`	`numeric`. If present must be a vector of times where the maximum expression of the gene was reached, i.e., a time existent in the vector `DAA`, 0, 10, 20, 30, 40, 50 and or 60.
`isTF`	`logical`. If `TRUE` only genes annotated as Transcription Factors will be present in the output SEP.
`coded.expr.level`	`numeric` One or more coded expression levels, i.e., numbers 1, 2, 3, 4 and 5. Only genes with those coded expression levels will be present in the output SEP.
`previous.sep`	If `NULL` the SEP used for the selection will be the original `SEP` present in the package, otherwise this parameter must contain a valid SEP, possibly selected with a previous call to the function. In that case such SEP will be used for the selection.

Details

This function can be used to select a subset of data for further analysis.

Value

A SEP data frame containing rows that fulfill all criteria pass to the function in the input.

Note

This function can be used in the initial data mining steps to select sets of data to be employed in further analysis.

Author(s)

O. Martinez

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

See Also

SEP, SEP.summary, SEPs.plot, analyze.2.SEPs, FPKM.expr.

Examples

```
# Return NULL because there is no gene
# with the description pass to the function
get.SEP(descr="MyFavorite gene")

# Obtain a SEP with all the Transcription Factors that are
# expressed in all 12 accessions.
temp.TF.all <- get.SEP(ExistInAll = TRUE, isTF = TRUE)
# How many rows has that SEP?
nrow(temp.TF.all)

# Some characteristics of that SEP
length(unique(temp.TF.all$id)) # How many different genes?
table(temp.TF.all$acc.key) # Number of cases per accession

# Now, assume that we want the Transcription Factors
# which include in the description the character "WRKY"
# and are present only in wild accessions.
# Using the previous SEP we can obtain:
temp.TF.WRKY.W <- get.SEP(descr="WRKY", acc.type="W",
previous.sep=temp.TF.all)
nrow(temp.TF.WRKY.W)

# Now assume that from that from the previous SEP we want only
# cases where the model is "DSSSSS", i.e., genes that are
# highly expressed in flower but then present an steady state
# along the remaining time intervals:
temp.TF.WRKY.W.DSSSSS <- get.SEP(model="DSSSSS",
previous.sep=temp.TF.WRKY.W)

# See the resulting SEP
# (note that all cases have the same standardized expression)
temp.TF.WRKY.W.DSSSSS

# An alternative way to obtain "temp.TF.WRKY.W.DSSSSS" with
# a single call of get.SEP is:
temp.TF.WRKY.W.DSSSSS.2 <- get.SEP(descr="WRKY", acc.type="W",
model="DSSSSS", ExistInAll=TRUE)
temp.TF.WRKY.W.DSSSSS.2

# Remove the objects that we created
rm(list=ls(patt='temp.TF'))
```

`GO.annot` *GO annotations*

Description

GO annotations for the genes.

Usage

```
data("GO.annot")
```

Format

A data frame with 784324 observations on the following 4 variables.

`aspect` a character vector where BP = Biological Process, CC = Cell Component and MF = Molecular Function.

`id` a numeric vector; gene numeric identifier.

`GO` a character vector; GO identifier.

`aspect.id` a numeric vector; the identifier of the aspect.

Source

http://geneontology.org/

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

Examples

```
# A random row of the data.frame
GO.annot[sample(c(1:nrow(GO.annot)), size=1), ]

head(GO.annot)
# How many annotations per aspect:
table(GO.annot$aspect)
```

SEP *Standardized Expression Profile (SEP)*

Description

For each gene expressed in each one of the 12 accessions this data frame gives the Standardized Expression Profiles (SEP) as well as auxiliary variables.

Usage

```
data("SEP")
```

Format

A data frame with 313919 observations on the following 13 variables.

`id` a numeric vector; gene identifier.

`acc.key` a character vector; accession identifier.

`acc.type` a character vector; accession type identifier.

`model` a character vector; each value consist of six concatenated letters which for each one of the 6 consecutive intervals are D if the expression decremented, S if the expression was steady, I if the expression incremented.

`ExistInAll` a logic vector; TRUE if the gene (id) was expressed in all 12 accessions.

`seT0` a numeric vector; mean expression at time 0.

`seT10` a numeric vector; mean expression at time 10.

`seT20` a numeric vector; mean expression at time 20.

`seT30` a numeric vector; mean expression at time 30.

`seT40` a numeric vector; mean expression at time 40.

`seT50` a numeric vector; mean expression at time 50.

`seT60` a numeric vector; mean expression at time 60.

`TimeMaxExp` a numeric vector; time at which the maximum expression over time was reached.

Details

SEPs are standardized over time, thus the mean value of each vector is equal to zero while its standard deviation (S) is equal to one.

Source

Cinvestav Irapuato, Mexico.

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

Examples

```
# A random row of the data.frame
SEP[sample(c(1:nrow(SEP)), size=1), ]

# Structure of the dataset
str(SEP)

# Number of SEPs that exist in only some or all accessions.
table(SEP$ExistInAll)

# Summary of numeric variables in SEP
summary(SEP[,6:13])
```

`SEP.id` *Summaries of SEP for each gene expressed*

Description

For each gene expressed this data.frame presents the number of SEPs per group of accessions where the gene was expressed, general mean SEPs as well as mean SEPs per group of accessions (D = Domesticated, W = Wild and C = Crosses). Includes mean correlation coefficients for SEPs in the whole set of SEPs for each gene, and in the SEPs of each group. Finally, it includes the mean and standard deviation (S) for the time at which the maximum expression of the gene was reached for the total, D, W and C groups. Function gene.summary gives a summary of the data in this data.frame per gene id.

Usage

```
data("SEP.id")
```

Format

A data frame with 29946 observations on the following 45 variables.

`id` a numeric vector of gene identifiers.

`n.t` a numeric vector with the total number of accessions where the gene was expressed.

`n.D` a numeric vector with the number of D (domesticated) accessions where the gene was expressed.

`n.W` a numeric vector with the number of W (wild) accessions where the gene was expressed.

`n.C` a numeric vector with the number of C (crosses) where the gene was expressed.

`t.mT0` a numeric vector; mean at time 0 in the whole set of SEPs.

`t.mT10` a numeric vector; mean at time 10 in the whole set of SEPs.

`t.mT20` a numeric vector; mean at time 20 in the whole set of SEPs.

`t.mT30` a numeric vector; mean at time 30 in the whole set of SEPs.

`t.mT40` a numeric vector; mean at time 40 in the whole set of SEPs.

`t.mT50` a numeric vector; mean at time 50 in the whole set of SEPs.

`t.mT60` a numeric vector; mean at time 60 in the whole set of SEPs.

`D.mT0` a numeric vector; mean at time 0 in the D set of SEPs.

D.mT10 a numeric vector; mean at time 10 in the D set of SEPs.

D.mT20 a numeric vector; mean at time 20 in the D set of SEPs.

D.mT30 a numeric vector; mean at time 30 in the D set of SEPs.

D.mT40 a numeric vector; mean at time 40 in the D set of SEPs.

D.mT50 a numeric vector; mean at time 50 in the D set of SEPs.

D.mT60 a numeric vector; mean at time 60 in the D set of SEPs.

W.mT0 a numeric vector; mean at time 0 in the W set of SEPs.

W.mT10 a numeric vector; mean at time 10 in the W set of SEPs.

W.mT20 a numeric vector; mean at time 20 in the W set of SEPs.

W.mT30 a numeric vector; mean at time 30 in the W set of SEPs.

W.mT40 a numeric vector; mean at time 40 in the W set of SEPs.

W.mT50 a numeric vector; mean at time 50 in the W set of SEPs.

W.mT60 a numeric vector; mean at time 60 in the W set of SEPs.

C.mT0 a numeric vector; mean at time 0 in the C set of SEPs.

C.mT10 a numeric vector; mean at time 10 in the C set of SEPs.

C.mT20 a numeric vector; mean at time 20 in the C set of SEPs.

C.mT30 a numeric vector; mean at time 30 in the C set of SEPs.

C.mT40 a numeric vector; mean at time 40 in the C set of SEPs.

C.mT50 a numeric vector; mean at time 50 in the C set of SEPs.

C.mT60 a numeric vector; mean at time 60 in the C set of SEPs.

m.r.t a numeric vector; mean correlation between SEPs.

m.r.D a numeric vector; mean correlation between SEPs in D.

m.r.W a numeric vector; mean correlation between SEPs in W.

m.r.C a numeric vector; mean correlation between SEPs in C.

m.TME.t a numeric vector; mean of maximum expression time in the whole set.

S.TME.t a numeric vector; Stantard deviation of maximum expression time in the whole set.

m.TME.D a numeric vector; mean of maximum expression time in the D set.

S.TME.D a numeric vector; Stantard deviation of maximum expression time in the D set.

m.TME.W a numeric vector; mean of maximum expression time in the D set.

S.TME.W a numeric vector; Stantard deviation of maximum expression time in the W set.

m.TME.C a numeric vector; mean of maximum expression time in the C set.

S.TME.C a numeric vector; Stantard deviation of maximum expression time in the C set.

Details

NA values are present where the corresponding value could not be estimated.

Source

Cinvestav Irapuato, Mexico

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

Examples

```
# A random row of the data.frame
SEP.id[sample(c(1:nrow(SEP.id)), size=1), ]

# Table with the number of SEPs in the whole (total) set.
table(SEP.id$n.t)

# Plot of mean SEPs per group.
plot(DAA, apply(SEP.id[SEP.id$n.t==12, 6:12], 2, mean), ylim=c(-0.43, 0.25), type="b", ylab="Mean Standaridz
points(DAA, apply(SEP.id[SEP.id$n.D==6, 13:19], 2, mean), type="b", col="red")
points(DAA, apply(SEP.id[SEP.id$n.W==4, 20:26], 2, mean), type="b", col="blue")
points(DAA, apply(SEP.id[SEP.id$n.C==2, 27:33], 2, mean), type="b", col="violet")
legend("topright", legend=c("Total", "D", "W", "C"), pch=1, lty=1, col=c("black", "red",  "blue", "violet"))
```

`SEP.summary` *Summary of one or many SEPs*

Description

Gives a comprehensive summary for the content of one or more Standardized Expression Profile (SEPs) `data.frames`. The output is a list with four components containing general statistics, means per expression time, lower limit of the confidence interval for the means and upper limit of the confidence interval for the means, respectively. For each one of the SEPs in the input, one row is produced in each one of the rows of the output `data.frame`.

Usage

```
SEP.summary(..., conf.level = 0.95)
```

Arguments

`...`	One or more SEPs.
`conf.level`	Confidence Level for the calculation of Confidence Intervals for the mean expression.

Details

The order of the rows in the results is the order at which each SEP was set in the input. The `t.test` function is used to obtain the Confidence Intervals for the mean expression at each one of the 7 times sampled. If the expression at one or more time points are uniform, the Confidence Interval limits for the mean expression at those points are set equal to the mean at those points, avoiding errors. For SEPs with only one row, CIs (`LL.time.means` and `UL.time.means`) are set equal to the value of the corresponding mean and a warning is shown in the output.

Value

A `list` with four components: `general` `time.means` `LL.time.means` and `UL.time.means`. For each SEP in the input, one row is produced in each one of the four components. The order of the rows in the four components of the output corresponds to the order of the SEPs in the input.

`general`	`data.frame` with columns `n.rows` Number of rows in the SEP, `n.ids` Number of different gene identifiers in the SEP, `n.acc` Number of different accession keys in the SEP, `n.type` Number of different accession types in the SEP, `n.mod` Number of different models in the SEP, `mean.TimeMaxExp` Mean time of maximum expression for SEPs, `LL.TimeMaxExp` Lower Limit for the Mean time of maximum expression for SEPs, `UL.TimeMaxExp` Upper Limit for the Mean time of maximum expression for SEPs.
`time.means`	`data.frame` with seven columns corresponding to the mean expression at each one of the times sampled, from `m.T0` up to `m.T60`.
`LL.time.means`	`data.frame` with seven columns corresponding to the lower limits for the mean expression at each one of the times sampled, from `LL.m.T0` up to `LL.m.T60`.
`UL.time.means`	`data.frame` with seven columns corresponding to the upper limits for the mean expression at each one of the times sampled, from `UL.m.T0` up to `UL.m.T60`.

Each one of the rows in each one of the components corresponds to one of the SEPs in the input, in the same order; i.e., the first row corresponds to to first SEP in the input, the second row corresponds to the second SEP in the input, etc.

Note

The output gives a summary of the content of each SEP and is used to plot the mean expression of SEPs with their confidence levels at each time, allowing the judgment of significance of differences per time.

Author(s)

O. Martinez

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

See Also

`SEP`, `get.SEP`, `SEP.id`, `SEPs.plot`, `analyze.2.SEPs`.

Examples

```
# Summary of a SEP with a single gene (id)
SEP.summary(get.SEP(ids=3))

# Summary of a SEP with many ids
# (all genes that are Transcription Factors)
SEP.summary(get.SEP(isTF=TRUE))

# Summary of three different SEPs
# (each one containing all Transcription Factors
# in each one of the three acc.type)
SEP.summary(get.SEP(acc.type="D", isTF=TRUE),
get.SEP(acc.type="W", isTF=TRUE),
get.SEP(acc.type="C", isTF=TRUE))

# Summary in a case where all expression values
# at each time are the same (and thus confidence
# intervals are set to be equal to the means)
SEP.summary(get.SEP(model="DSSSSS"))
```

`SEPs.plot` *Plot mean expression times in SEPs*

Description

For one or more SEPs produces a plot of the means of standardized expression at each one of the times sampled, including their confidence intervals.

Usage

```
SEPs.plot(seps, colors, lwd = 2, conf.level = 0.95, CIs.sep = 0.5)
```

Arguments

`seps`	Either, a single SEP or a `list` of two or more SEPs.
`colors`	A single color in the case when `seps` is a single SEP, or a vector of colors with the same number of colors than the number of `seps` passed on that `list`. Any valid code for `col` (color) is accepted.
`lwd`	Line width.
`conf.level`	Confidence Level for the estimation of the Confidence Intervals for the means at each one of the times.
`CIs.sep`	A value to avoid the overlapping of Confidence Intervals in the X axis. The default of 0.5 days works well in most cases.

Details

It is very important that if two or more SEPs are pass to this function, the groups of SEPs must be defined as a `list`. Otherwise an unexpected error will be produced. If the expression at one or more time points are uniform, the Confidence Interval limits for the mean expression at those points are set equal to the mean at those points, avoiding errors. In cases of SEPs with only one row CIs are not shown (it is not posible to estimate such CIs).

Value

`NULL` (the function is used by its side effect, i.e., to produce a plot of SEPs).

Note

Confidence Intervals for the means are shown as thin vertical lines. It is not wise to produce plots with more than a few SEPs, say a maximum of five. Plots with more SEPs will be difficult to interpret.

Author(s)

O. Martinez

References

O. Martinez et al. (In preparation). An R package for data mining chili pepper fruit transcriptomes.

See Also

SEP.summary, get.SEP, SEP.id, SEPs.plot, analyze.2.SEPs.

Examples

```
# Plot of a single SEP
temp.sep.3 <- get.SEP(ids=3)
SEPs.plot(seps=temp.sep.3, colors="grey")

# Obtaining SEPs per acc.type
temp.sep.3.D <- get.SEP(ids=3, acc.type="D")
temp.sep.3.W <- get.SEP(ids=3, acc.type="W")
temp.sep.3.C <- get.SEP(ids=3, acc.type="C")

# and plotting them together
# Note: in that case argument seps
# MUST be a list!
SEPs.plot(seps=list(temp.sep.3, temp.sep.3.D,
temp.sep.3.W, temp.sep.3.C),
colors=c("grey", "red", "blue", "violet"))

# Clean temporary objects
rm(list=ls(patt="temp.sep.3"))

# Plot SEPs for two different genes
SEPs.plot(seps=list(get.SEP(ids=3), get.SEP(ids=19)), colors=c("red", "blue"))

# Plot of SEPs with all transcription
# factors group by acc.type
SEPs.plot(seps=list(get.SEP(acc.type="D", isTF=TRUE),
get.SEP(acc.type="W", isTF=TRUE),
get.SEP(acc.type="C", isTF=TRUE)),
colors=c("red", "blue", "violet"))
```

Index