Package 'goseq'

June 2, 2023

Version 1.53.0

Date 2019/04/26

Title Gene Ontology analyser for RNA-seq and other length biased data

Author Matthew Young

Maintainer
Matthew Young <my4@sanger.ac.uk>, Nadia Davidson <nadia.davidson@mcri.edu.au>

Depends R (>= 2.11.0), BiasedUrn, geneLenDataBase (>= 1.9.2)

Imports mgcv, graphics, stats, utils, AnnotationDbi,
GO.db,BiocGenerics

Suggests edgeR, org.Hs.eg.db, rtracklayer

LazyLoad yes

Description Detects Gene Ontology and/or other user defined categories
which are over/under represented in RNA-seq data

License LGPL (>= 2)

biocViews ImmunoOncology, Sequencing, GO, GeneExpression,
Transcription, RNASeq

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

git_branch devel

git_last_commit 6519846

git_last_commit_date 2023-04-25

Date/Publication 2023-06-02

R topics documented:

---

genes *Androgen stimulation of prostate cancer Cell lines.*

Description

This data set gives the RNA-seq data from an experiment measuring the effects of androgen stimulation on prostate cancer. Information is given about all (ENSEMBL) genes for which there was at least one mapping read in either the treated or untreated RNA-seq experiment. The edgeR package was used to determine which genes were differentially expressed. The details of the analysis can be found in the goseq vignette.

Usage

```
data(genes)
```

Format

A named vector of ENSEMBL genes, with 1 representing differential expression.

Source

Determination of tag density required for digital transcriptome analysis: application to an androgen-sensitive prostate cancer model, 2008, Li et. al.

References

Li, H., Lovci, M. T., Kwon, Y. S., Rosenfeld, M. G., Fu, X. D., Yeo, G. W. (2008) *Determination of tag density required for digital transcriptome analysis: application to an androgen-sensitive prostate cancer model* Proceedings of the National Academy of Sciences of the United States of America Date: Dec 23 Vol: 105 Issue: 51 Pages: 20179-84

Examples

```
data(genes)
head(genes)
```

getgo *Fetch GO categories*

Description

Obtains all gene ontology (GO) categories associated with a set of genes using the relevant organism package.

Usage

```
getgo(genes, genome, id,fetch.cats=c("GO:CC","GO:BP","GO:MF"))
```

Arguments

genes	A vector or list of genes to get the associated GO categories.
genome	A string identifying the genome that genes refer to. For a list of supported organisms run supportedGenomes.
id	A string identifying the gene identifier used by genes. For a list of supported gene IDs run supportedGeneIDs.
fetch.cats	A vector specifying which categories to fetch the mapping between category names and genes for. See details for vaild options.

Details

This function attempts to make use of the organism packages (org.<Genome>.<GeneID>.db) to obtain the mapping between gene ID and GO categories. As with getlength it is preferable that the same gene identifier system is used for both summarization and retrieving GO categories.

Valid options for the fetch.cats arguement are any combination of "GO:CC", "GO:BP", "GO:MF" & "KEGG". The three GO terms refer to the Cellular Component, Biological Process and Molecular Function respectively. "KEGG" refers to KEGG pathways.

Note that getgo is a convenience function, designed to make extracting mappings between GO categories and Gene ID easy. For less common organisms and/or gene ID getgo may fail to return a mapping even when a legitimate mapping exists in the revelant organism package. If getgo fails, you should always try to build the mapping yourself from the organism package (if one exists) before deciding that the information is unavailable. Further information and examples of this can be found in the package Vignette.

Value

A list where each entry is named by a gene and contains a vector of all the associated GO categories. This can be used directly with the gene2cat option in goseq.

Author(s)

Matthew D. Young <myoung@wehi.edu.au>

See Also

supportedGenomes, supportedGeneIDs, goseq

Examples

```
genes <- c("ENSG00000124208", "ENSG00000182463", "ENSG00000124201", "ENSG00000124205", "ENSG00000124207")
getgo(genes,'hg19','ensGene')
```

---

getlength *Retrieves Gene length data*

Description

Gets the length of each gene in a vector.

Usage

```
getlength(genes, genome, id)
```

Arguments

genes	A vector or list of the genes for which length information is required.
genome	A string identifying the genome that genes refer to. For a list of supported organisms run supportedGenomes.
id	A string identifying the gene identifier used by genes. For a list of supported gene IDs run supportedGeneIDs.

Details

Length data is obtained from data obtained from the UCSC genome browser for each combination of genome and id. As fetching this data at runtime is time consuming, a local copy of the length information for common genomes and gene ID are included in the **geneLenDataBase** package. This function uses this package to fetch the required data.

The length of a gene is taken to be the median length of all its mature, mRNA, transcripts. It is always preferable to obtain length information directly for the gene ID used to summarize your count data, rather than converting IDs and then using the supplied databases. Even when two genes have a one-to-one mapping between different identifier conventions (which is often not the case), they frequently refer to slightly different regions of the genome with different lengths. It is therefore recommended that the user perform the full analysis in terms of only one gene ID, or manually obtain their own length data for the identifier used to bin reads by gene.

Value

Returns a vector of the gene lengths, in the same order as genes. If length data is unavailable for a particular gene NA is returned in that position. The returned vector is intended for use with the bias.data option of the nullp function.

Author(s)

Matthew D. Young <myoung@wehi.edu.au>

See Also

supportedGenomes, supportedGeneIDs, nullp, **geneLenDataBase**

Examples

```
genes <- c("ENSG00000124208", "ENSG00000182463", "ENSG00000124201", "ENSG00000124205", "ENSG00000124207")
getlength(genes,'hg19','ensGene')
```

---

goseq *goseq Gene Ontology analyser*

Description

Does selection-unbiased testing for category enrichment amongst differentially expressed (DE) genes for RNA-seq data. By default, tests gene ontology (GO) categories, but any categories may be tested.

Usage

```
goseq(pwf, genome, id, gene2cat = NULL,
test.cats=c("GO:CC", "GO:BP", "GO:MF"),
method = "Wallenius", repcnt = 2000, use_genes_without_cat=FALSE)
```

Arguments

pwf	An object containing gene names, DE calls, the probability weighting function. Usually generated by nullp.
genome	A string identifying the genome that genes refer to. For a list of supported organisms run supportedGenomes.
id	A string identifying the gene identifier used by genes. For a list of supported gene IDs run supportedGeneIDs.
gene2cat	A data frame with two columns containing the mapping between genes and the categories of interest. Alternatively, a list where the names are genes and each entry is a vector containing GO categories associated with that gene (this is the output produced by getgo). If set to NULL goseq attempts to fetch GO categories automatically using getgo.
test.cats	A vector specifying which categories to test for over representation amongst DE genes. See details for allowed options.
method	The method to use to calculate the unbiased category enrichment scores. Valid options are "Wallenius", "Sampling" & "Hypergeometric". "Hypergeometric" and "Sampling" should almost never be used (see details).

repcnt
: Number of random samples to be calculated when random sampling is used. Ignored unless method="Sampling".

use_genes_without_cat
: A boolean to indicate whether genes without a categorie should still be used. For example, a large number of gene may have no GO term annotated. If this option is set to FALSE, those genes will be ignored in the calculation of p-values (default behaviour). If this option is set to TRUE, then these genes will count towards the total number of genes outside the category being tested (default behaviour prior to version 1.15.2).

Details

The pwf argument is almost always the output of the function nullp. This is a data frame with 3 columns, named "DEgenes", "bias.data" and "pwf" with the rownames set to the gene names. Each row corresponds to a gene with the DEgenes column specifying if the gene is DE (1 for DE, 0 for not DE), the bias.data column giving the numeric value of the DE bias being accounted for (usually the gene length or number of counts) and the pwf column giving the genes value on the probability weighting function.

goseq obtains length data from UCSC and GO mappings from the organim packages (see link{getgo} and getlength for details). If your data is in an unsupported format you will need to obtain the GO category mapping and supply them to the goseq function using the gene2cat arguement.

To use your own gene to category mapping with goseq, use the gene2cat arguement. This arguement takes a data.frame, with one column containing gene IDs and the other containing the associated categories. As the mapping from gene <-> category is in general many to many there will be multiple rows containing the same gene identifier. Alternatively, gene2cat can take a list, where the names are the genes and the entries are the GO categories associated with the genes. This is the format produced by the getgo function and is more space efficient than the data.frame representation.

If gene2cat is left as NULL, goseq attempts to use getgo to fetch GO catgeory to gene identifier mappings.

The PWF is usually calculated using the nullp function to correct for length bias. However, goseq will work with any vector of weights. Any bias can be accounted for so long as a weight for each gene is supplied using this arguement. NAs are allowed in the "pwf" and "bias.data" columns of the PWF data frame (these usually occur as a result of missing length data for some genes). Any entry which is NA is set to the weighting of the median gene.

Valid options for the test.cats arguement are any combination of "GO:CC", "GO:BP", "GO:MF" & "KEGG". The three GO terms refer to the Cellular Component, Biological Process and Molecular Function respectively. "KEGG" refers to KEGG pathways.

The three methods, "Wallenius", "Sampling" & "Hypergeometric", calculate the p-values as follows.

"Wallenius" approximates the true distribution of numbers of members of a category amongst DE genes by the Wallenius non-central hypergeometric distribution. This distribution assumes that within a category all genes have the same probability of being chosen. Therefore, this approximation works best when the range in probabilities obtained by the probability weighting function is small. "Wallenius" is the recommended method for calculating p-values.

"Sampling" uses random sampling to approximate the true distribution and uses it to calculate the p-values for over (and under) representation of categories. In practice, its use quickly becomes computationally prohibitive because `repcnt` would need to be set very high for most applications.

CAUTION: "Hypergeometric" should NEVER be used for producing results for biological interpretation. If there is genuinly no bias in power to detect DE in your experiment, the PWF will reflect this and the other methods will produce accuracte results.

"Hypergeometric" assumes there is no bias in power to detect differential expression at all and calculates the p-values using a standard hypergeometric distribution. Useful if you wish to test the effect of selection bias on your results.

Value

goseq returns a data frame with several columns. The first column gives the name of the category, the second gives the p-value for the associated category being over represented amongst DE genes. The third column gives the p-value for the associated category being under represented amongst DE genes. The p-values have not been corrected for multiple hypothesis testing. The fourth and fifth columns give the number of differentially expressed genes in the category and total genes in the category respectively. If any of the categories was a GO term, there will be two additional columns for the GO term and its ontology.

Author(s)

Matthew D. Young <myoung@wehi.edu.au>

References

Young, M. D., Wakefield, M. J., Smyth, G. K., Oshlack, A. (2010) *Gene ontology analysis for RNA-seq: accounting for selection bias* Genome Biology Date: Feb 2010 Vol: 11 Issue: 2 Pages: R14

See Also

`nullp`, `getgo`, `getlength`

Examples

```
data(genes)
pwf <- nullp(genes,'hg19','ensGene')
pvals <- goseq(pwf,'hg19','ensGene')
head(pvals)
```

makespline *Monotonic Spline*

Description

Fits a monotonic cubic spline to the data provided, using the penalized constrained least squares method from the mgcv package.

Usage

```
makespline(x, y, newX=NULL, nKnots = 6, lower_bound = 10^-3)
```

Arguments

x	The predictor variable.
y	The response variable. Must be the same length as x.
newX	The points at which to return the value on the fitted spline. If not specified x is used.
nKnots	The number of knots to use in fitting the spline.
lower_bound	The spline cannot drop below this value.

Details

This uses the pcls function from the **mgcv** package to produce the fit. The montonicity constraint is enforced using mono.con from the same package. The lower_bound argument is only used on the rare occasions when the fitting function becomes negative or arbitrarily close to zero. If this does occur lower_bound is added everywhere to ensure that no one length is given essentially infinite weighting.

Value

Returns a vector of values containing the value of the fit at each point newX.

Author(s)

Matthew D. Young <myoung@wehi.edu.au>.

References

Package **mgcv**. In particular this function is a modification of an example given in the man page for pcls.

Examples

```
y <- c( rbinom(50,p=0.4,size=1), rbinom(50,p=0.6,size=1) )
x <- 1:100
plot(x,y)
p <- makespline(x,y)
lines(x,p)
```

`nullp` *Probability Weighting Function*

Description

Calculates a Probability Weighting Function for a set of genes based on a given set of biased data (usually gene length) and each genes status as differentially expressed or not.

Usage

```
nullp(DEgenes, genome, id, bias.data=NULL,plot.fit=TRUE)
```

Arguments

`DEgenes`	A named binary vector where 1 represents DE, 0 not DE and the names are gene IDs.
`genome`	A string identifying the genome that `genes` refer to. For a list of supported organisms run `supportedGenomes`.
`id`	A string identifying the gene identifier used by `genes`. For a list of supported gene IDs run `supportedGeneIDs`.
`bias.data`	A numeric vector containing the data on which the DE may depend. Usually this is the median transcript length of each gene in bp. If set to `NULL` `nullp` will attempt to fetch length using `getlength`.
`plot.fit`	Plot the PWF or not? Calls `plotPWF` with default values if `TRUE`.

Details

It is essential that the entire analysis pipeline, from summarizing raw reads through to using `goseq` be done in just one gene identifier format. If your data is in a different format you will need to obtain the gene lengths and supply them to the `nullp` function using the `bias.data` arguement. Converting to a supported format from another format should be avoided whenever possible as this will almost always result in data loss.

`NA`s are allowed in the bias.data vector if you do not have information about a certain gene. Setting a gene to `NA` is preferable to removing it from the analysis.

If `bias.data` is left as `NULL`, `nullp` attempts to use `getlength` to fetch GO catgeory to gene identifier mappings.

It is recommended you review the fit produced by the `nullp` function before proceeding by leaving `plot.fit` as `TRUE`.

Value

A data frame with 3 columns, named "DEgenes", "bias.data" and "pwf" with the rownames set to the gene names. Each row corresponds to a gene with the DEgenes column specifying if the gene is DE (1 for DE, 0 for not DE), the bias.data column giving the numeric value of the DE bias being accounted for (usually the gene length or number of counts) and the pwf column giving the genes value on the probability weighting function. This object is usually passed to `goseq` to calculate enriched categories or `plotPWF` for further plotting.

Author(s)

Matthew D. Young <myoung@wehi.edu.au>

References

Young, M. D., Wakefield, M. J., Smyth, G. K., Oshlack, A. (2010) *Gene ontology analysis for RNA-seq: accounting for selection bias* Genome Biology Date: Feb 2010 Vol: 11 Issue: 2 Pages: R14

See Also

`supportedGenomes`, `supportedGeneIDs`, `goseq`, `getlength`

Examples

```
data(genes)
pwf <- nullp(genes, 'hg19', 'ensGene')
```

`plotPWF` *Plot the Probability Weighting Function*

Description

Plots the Probability Weighting Function created by `nullp` by binning together genes.

Usage

```
plotPWF(pwf,binsize="auto",pwf_col=3,pwf_lwd=2,xlab="Biased Data in <binsize> gene bins.",ylab="Propo
```

Arguments

`pwf`	A data frame with 3 columns named DEgenes, bias.data & pwf and row names giving the gene names. Usually generated by `nullp`.
`binsize`	Calculate and plot the fraction of genes that are DE in bins of this size. If set to "auto" the best binsize for visualization is attempted to be found automatically.
`pwf_col`	The colour of the probability weighting function
`pwf_lwd`	The width of the probability weighting function

xlab	The x-axis label. <binsize> is replaced by the binsize used.
ylab	The y-axis label.
...	Extra arguments that are passed to plot.

Details

This function is almost always called using the output from the nullp function. However, it can be used to visualize the length (or any other type of quantifyable) bias in ability to detect DE in a data set. The pwf arguement needs to be a data frame with 3 columns each containing numeric entries (although NAs are permitted in the bias.data and pwf columns), which must be named "DEgenes", "bias.data" and "pwf", although they can appear in any order. The row names are taken to be the gene names. The DEgenes column should be 0s or 1s where 1 represents a DE gene, 0 a gene which is not DE. The bias.data column is a quantification of the quantity for which there is a bias in detecting DE for the associated gene, this is usually gene length or the number of counts associated with a gene. Finally, the pwf column gives the probability weighting to be applied for a given gene.

Value

Nothing is returned.

Author(s)

Matthew D. Young <myoung@wehi.edu.au>

References

Young, M. D., Wakefield, M. J., Smyth, G. K., Oshlack, A. (2010) *Gene ontology analysis for RNA-seq: accounting for selection bias* Genome Biology Date: Feb 2010 Vol: 11 Issue: 2 Pages: R14

See Also

nullp

Examples

```
data(genes)
pwf <- nullp(genes, 'hg19', 'ensGene',plot.fit=FALSE)
plotPWF(pwf,binsize=200)
```

supportedOrganisms *Supported Organisms*

Description

Lists which genomes and gene ids are supported for gene lengths and for gene ontology

Usage

```
supportedOrganisms()
```

Details

Goseq allows a user to provide their own bias data (usually gene lengths) and/or gene categories (usually gene ontologies), but goseq also provides this data automatically for many commonly used species. This function lists which genome and gene ids are automatically supported by goseq. The first to third columns list the genomes, gene ids, and gene id descriptions respectively. The fourth column indicates whether this combination of genome and id are available in the geneLengthDataBase. If a particular combination is absent, goseq may still automatically fetch the gene lengths from either a TxDB annotation package (must be installed) or download the data from UCSC. For example gene lengths for `hg38` are not supported in `geneLengthDataBase` but may be fetched by these other means. However, this is not always the case. The final column indicates if GO terms will be automatically fetched for the geneome and id combination. Goseq relies on an `org` annotation package (e.g. `org.Hs.eg.db`) existing for the organism. In general, if either the lengths or GO terms are not supported, the user must enter this information manually.

Value

A data.frame containing supported genomes and gene ids

Author(s)

Nadia Davidson <nadia.davidson@mcri.edu.au>

Examples

```
supportedOrganisms()
```

Index