

goseq

October 5, 2010

`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

<code>genes</code>	A vector or list of genes to get the associated GO categories.
<code>genome</code>	A string identifying the genome that <code>genes</code> refer to. For a list of supported organisms run supportedGenomes .
<code>id</code>	A string identifying the gene identifier used by <code>genes</code> . For a list of supported gene IDs run supportedGeneIDs .
<code>fetch.cats</code>	A vector specifying which categories to fetch the mapping between category names and genes for. See details for valid 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` argument 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. For less common organisms and/or gene ID `getgo` may fail to return a mapping even when a legitimate mapping exists in the relevant 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", "E
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

<code>genes</code>	A vector or list of the genes for which length information is required.
<code>genome</code>	A string identifying the genome that <code>genes</code> refer to. For a list of supported organisms run supportedGenomes .
<code>id</code>	A string identifying the gene identifier used by <code>genes</code> . 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", "E
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(DEgenes, pwf, genome, id, gene2cat = NULL,
test.cats=c("GO:CC", "GO:BP", "GO:MF"),
method = "Wallenius", repcnt = 2000)
```

Arguments

<code>DEgenes</code>	A named binary vector where 1 represents DE, 0 not DE and the names are gene IDs.
<code>pwf</code>	A vector of the same length as <code>DEgenes</code> containing the weighting to be applied to each gene in <code>DEgenes</code> . Usually produced by <code>nullp</code> .
<code>genome</code>	A string identifying the genome that <code>genes</code> refer to. For a list of supported organisms run supportedGenomes .
<code>id</code>	A string identifying the gene identifier used by <code>genes</code> . For a list of supported gene IDs run supportedGeneIDs .
<code>gene2cat</code>	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 <code>getgo</code>). If set to <code>NULL</code> <code>goseq</code> attempts to fetch GO categories automatically using <code>getgo</code> .
<code>test.cats</code>	A vector specifying which categories to test for over representation amongst DE genes. See details for valid options.

method	The method to use to calculate the unbiased category enrichment scores. Valid options are "Wallenius", "Sampling" & "Hypergeometric". "Hypergeometric" should almost never be used (see details).
repcnt	Number of random samples to be calculated when random sampling is used. Ignored unless method="Sampling".

Details

`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` argument.

To use your own gene to category mapping with `goseq`, use the `gene2cat` argument. This argument takes a `data.frame`, with one column containing gene IDs and the other containing the associated categories. As the mapping from gene \leftrightarrow 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 category to gene identifier mappings.

The `genome` and `id` arguments are only required for fetching go category mappings. If you are providing your own, these arguments are not required and will be ignored if specified.

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 argument. `NA`s are allowed in `pwf` (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` argument 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.

"Sampling" uses random sampling to approximate the true distribution and uses it to calculate the p-values for over (and under) representation of categories. Although this is the most accurate method given a high enough value of `repcnt`, its use quickly becomes computationally prohibitive.

CAUTION: "Hypergeometric" should NEVER be used for producing results for biological interpretation. If there is genuinely no bias in power to detect DE in your experiment, the PWF will reflect this and the other methods will produce accurate 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 3 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 and final 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.

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

Arguments

<code>x</code>	The predictor variable.
<code>y</code>	The response variable. Must be the same length as <code>x</code> .
<code>newX</code>	The points at which to return the value on the fitted spline. If not specified <code>x</code> is used.
<code>nKnots</code>	The number of knots to use in fitting the spline.

Details

This uses the `pcls` function from the `mgcv` package to produce the fit. The monotonicity constraint is enforced using `mono.con` from the same package.

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

<code>DEgenes</code>	A named binary vector where 1 represents DE, 0 not DE and the names are gene IDs.
<code>genome</code>	A string identifying the genome that genes refer to. For a list of supported organisms run supportedGenomes .
<code>id</code>	A string identifying the gene identifier used by genes. For a list of supported gene IDs run supportedGeneIDs .
<code>bias.data</code>	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 <code>nullp</code> will attempt to fetch length using getlength .
<code>plot.fit</code>	Plot the PWF or not?

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` argument. Converting to a supported format from another format should be avoided whenever possible as this will almost always result in data loss.

NAs 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 category 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 numeric vector containing the value on the probability weighting function for each gene. This is usually passed to the function `goseq` via the `pwf` argument.

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

prostate

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(prostate)
```

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(prostate)
head(genes)
```

Index

*Topic **datasets**

prostate, [7](#)

getgo, [1](#), [3-5](#)

getlength, [1](#), [2](#), [4-7](#)

goseq, [2](#), [3](#), [7](#)

makespline, [5](#)

nullp, [3-5](#), [6](#)

prostate, [7](#)

supportedGeneIDs, [1-3](#), [6](#), [7](#)

supportedGenomes, [1-3](#), [6](#), [7](#)