Package 'geneRecommender'

November 12, 2026

Version 1.84.0

Date 2005-04-5

Title A gene recommender algorithm to identify genes coexpressed with a query set of genes

Author Gregory J. Hather <ghather@gmail.com>, with contributions from Art B. Owen <art@stat.stanford.edu> and Terence P. Speed <terry@stat.berkeley.edu>

Maintainer Greg Hather <ghather@gmail.com>

Depends R (>= 1.8.0), Biobase (>= 1.4.22), methods

Imports Biobase, methods, stats

Description This package contains a targeted clustering algorithm for the analysis of microarray data. The algorithm can aid in the discovery of new genes with similar functions to a given list of genes already known to have closely related functions.

License GPL (>= 2)

biocViews Microarray, Clustering

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

git_branch RELEASE_3_23

git_last_commit aac4477

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

gr.cv 2
gr.main 3
gr.normalize 5

Index **7**

gr.cv *A function for cross validation*

Description

gr.cv performs leave-one-out cross validation with gr.main for each element of the query.

Usage

```
gr.cv(normalized.dataset, query, fun = median)
```

Arguments

normalized.dataset	A matrix or ExpressionSet containing the normalized gene expression data. The rows correspond to genes, the columns correspond to experiments, and the entries correspond to the gene expression levels. The rows must be labeled. The values contained in normalized.dataset must be either finite or NA.
query	A vector containing the query set of genes. These should correspond to the row names of normalized.dataset. The query must contain at least 2 elements
fun	A function used in choosing the number of experiments to include in the calculation. See the help file for gr.main for details.

Details

In addition to measuring performance, the results of the cross validation can be used to determine if some element(s) in the query might not belong. If one of the elements in the output vector was very large, one would suspect that the associated gene was regulated differently than the other genes in the query.

Value

A vector containing the rank of each element in the query produced by applying gr.main to the query with that element removed.

Author(s)

Gregory J. Hather <ghather@stat.berkeley.edu>
with contributions from from Art B. Owen <art@stat.stanford.edu>
and Terence P. Speed <terry@stat.berkeley.edu>.

References

Art B. Owen, Josh Stuart, Kathy Mach, Anne M. Villeneuve, and Stuart Kim. A Gene Recommender Algorithm to Identify Coexpressed Genes in C. elegans. Genome Research 13:1828-1837, 2003.

See Also

gr.main, gr.normalize

Examples

```
#This example uses the geneData dataset from the Biobase package
data(geneData)
my.query <- c("31730_at", "31331_at", "31712_at", "31441_at")
normalized.data <- gr.normalize(geneData)
gr.cv(normalized.data, my.query)
```

---

gr.main *A gene recommender algorithm to identify genes coexpressed with a query set of genes*

Description

The function gr.main implements the Gene Recommender algorithm described in Owen et al (2003). Note that in order for gr.main to provide meaningful results, the normalization procedure gr.normalize must first be applied to the gene expression matrix.

Usage

```
gr.main(normalized.dataset, query, fun = median, ngenes = NULL, extra = FALSE)
```

Arguments

normalized.dataset	A matrix or ExpressionSet containing the normalized gene expression data. The rows correspond to genes, the columns correspond to experiments, and the entries correspond to the gene expression levels. The rows must be labeled. The values contained in normalized.dataset must be either finite or NA.
query	A vector containing the query set of genes. These should correspond to the row names of normalized.dataset. The query must contain at least 2 elements
fun	A function used in choosing the number of experiments to include in the calculation. See below for details.
ngenes	The number of genes to return in the result. It's default value is the number of genes found at 50 percent recall.
extra	A logical value. When false, the output list will contain only one item, result. When true, several other quantities (listed below) will be calculated and added to the output list.

Details

Given data from a large number of microarray experiments and a query set of genes, which genes have expression profiles that are similar to the query? The Gene Recommender algorithm (Owen et al, 2003) answers this question by first identifying the set of experiments over which the query genes behave similarly. Next, the algorithm ranks all the genes based on the strength of the correlation with the query across the chosen set of experiments.

The algorithm must choose how generous to be in including experiments. How many experiments should be included? The algorithm tries every number of experiments and chooses the number which minimizes a score. In the paper, the score was defined as the median of the ranks of the query genes. In `gr.main`, the score can be computed with the user-defined function, `fun`.

Value

A list containing entries:

`result` An array of dimensions (`ngenes`, 2). Column 1 contains the resulting genes, with the highest scoring genes listed first. Column 2 contains character strings, indicating whether the corresponding gene is from the query list or not.

`fifty.percent.recall` Number of genes found at 50 percent recall.

`experiments.included` Experiments included in the analysis.

`experiments.excluded` Experiments excluded from the analysis.

`s.g.i` An array used as a measure of biological significance for each gene. The output is ranked by this quantity.

`z.g.i` An array used as a measure of statistical significance for each gene.

`contribution` An array indicating the contribution of each experiment to each gene result. For a given gene and a given experiment, the `contribution` indicates how strongly the experiment suggests that the gene should be high ranking. Using notation from the article, `contribution` is defined as $\bar{Y}_{Q,j} \times Y_{ij}$.

Note

The results of `gr.main` will differ from the results generated from the C code released by Owen et al (2003). This is due to differences in the implementation. See the vignette for details.

Author(s)

Gregory J. Hather <ghather@stat.berkeley.edu>
with contributions from from Art B. Owen <art@stat.stanford.edu>
and Terence P. Speed <terry@stat.berkeley.edu>.

References

Art B. Owen, Josh Stuart, Kathy Mach, Anne M. Villeneuve, and Stuart Kim. A Gene Recommender Algorithm to Identify Coexpressed Genes in C. elegans. Genome Research 13:1828-1837, 2003.

See Also

gr.normalize, gr.cv

Examples

```
#This example uses the geneData dataset from the Biobase package
data(geneData)
my.query <- c("31730_at", "31331_at", "31712_at", "31441_at")
normalized.data <- gr.normalize(geneData)
gr.main(normalized.data, my.query, ngenes = 10)
```

---

gr.normalize *A function for normalization*

Description

gr.normalize normalizes a matrix of gene expression data as part of the implementation of the Gene Recommender algorithm described in Owen et al (2003). gr.normalize must be applied to the data before running gr.main.

Usage

```
gr.normalize(unnormalized.dataset)
```

Arguments

unnormalized.dataset
: A matrix or ExpressionSet containing the normalized gene expression data. The rows correspond to genes, the columns correspond to experiments, and the entries correspond to the gene expression levels. The rows must be labeled.

Details

gr.normalize normalizes the data so that for each gene, the gene expression measurements are distributed uniformly between -1 and 1.

Value

The normalized gene expression data, in the same format as the input.

Author(s)

Gregory J. Hather <ghather@stat.berkeley.edu>
with contributions from from Art B. Owen <art@stat.stanford.edu>
and Terence P. Speed <terry@stat.berkeley.edu>.

References

Art B. Owen, Josh Stuart, Kathy Mach, Anne M. Villeneuve, and Stuart Kim. A Gene Recommender Algorithm to Identify Coexpressed Genes in C. elegans. Genome Research 13:1828-1837, 2003.

See Also

gr.main, gr.cv

Examples

```
#This example uses the geneData dataset from the Biobase package
data(geneData)
my.query <- c("31730_at", "31331_at", "31712_at", "31441_at")
normalized.data <- gr.normalize(geneData)
```

Index

∗ **manip**
gr.cv, 2
gr.main, 3
gr.normalize, 5

gr.cv, 2
gr.main, 3
gr.normalize, 5