

goCluster - User Manual

Gunnar Wrobel

May 18, 2005

Biozentrum & Swiss Institute of Bioinformatics Klingelbergstrasse 50-70, CH-4056 Basel, Switzerland,
<http://www.biozentrum.unibas.ch/personal/primig/gocluster/>

1 Introduction

The *goCluster* package provides a modular framework that allows to combine a variety of clustering approaches with different statistical approaches. It also allows to correct for multiple testing problems. *goCluster* uses S4-methods to provide an object oriented structure which can be easily extended by new clustering algorithms, statistical procedures as well as visualization methods.

This vignette provides a short overview on *goCluster* and explains how the package can be used to combine clustering with gene ontology (GO) term analysis.

For a deeper understanding of the statistical aspect of such an analysis the interested reader is referred to the “GOSTats” vignette, which explains the concept (as well as the statistical problems) of such a GO term analysis.

2 Getting started

```
> library("goCluster", verbose = FALSE)
```

```
Loading required package: Biobase
```

```
Welcome to Bioconductor
```

```
  Vignettes contain introductory material.  To view,
```

```
  simply type: openVignette()
```

```
  For details on reading vignettes, see
```

```
  the openVignette help page.
```

```
Loading required package: YEAST
```

```
> data(benomylsetup)
```

```
> benomyldata <- benomylsetup$data$dataset[1:400, ]
```

This first step loads the package together with a prepared analysis of the “benomyl dataset” (**benomylsetup**). This dataset characterizes the stress response of yeast to the micro-tubule destabilizing drug benomyl. The original expression values are included in the configuration object. A reduced dataset (400 genes) is being extracted and stored in the variable **benomyldata**. This dataset is an **exprSet** object as defined in the core *Biobase*-package and it will be the sample dataset to demonstrate the *goCluster*-setup and analysis process. The selection of the first four hundred gene is intended to reduce the execution time for this vignette.

At this point it would be possible to run the complete analysis using the following two statements:

```
> benomyl <- new("goCluster")
> execute(benomyl) <- benomylsetup
```

This is possible because the `benomylsetup` variable is a list that holds all necessary configuration information for the *goCluster*-analysis. But since this vignette is meant to explain the usage of *goCluster* it is not very instructive to use a preconfigured setup. Instead the `config`-method provided by the *goCluster*-package will be demonstrated in the next section.

goCluster provides functions for an interactive configuration. Since interactivity is something the vignette cannot provide the configuration process in the following sections will just contain a static transcript of the steps to take.

If you do not like the commandline configuration procedure you can also test the very simple GUI provided with *goCluster*. It is not yet complete but you can use the following call in order to perform a complete *goCluster* analysis:

```
> result <- goClusterTk()
```

3 Configuring *goCluster*

The following six sections have to be configured for a *goCluster*-analysis:

1. Data
2. Annotation
3. Clustering
4. Significance analysis
5. Statistical analysis
6. Visualization

All steps will be performed using a single *goCluster*-object that will be created with the following statement:

```
> benomyl <- new("goCluster")
```

Using the `config`-method with this object will start the configuration process. The result of the `config` call is a complete *goCluster* setup object that is stored in the *benomylConfigured* variable.

```
> benomylConfigured <- config(benomyl)
```

3.1 Data

The following sections provide a transcript of the dialog after calling the `config`-method. It has been splitted to the six sections mentioned above for better readability.

Please select a name or title for the dataset.

```
# Benomyl
```

Please specify the variable name of the dataset to be analyzed.

```
# benomyldata
```

Please specify the variable name of the unique ID for the rows of the dataset. Alternatively you can specify "rownames" and the current rownames of the dataset will be used as unique ID.

```
# rownames
```

Here you can specify a title for the analysis. The dataset to be analyzed is specified by the name of the variable that holds the expression values. The benomyl data has been stored in the `benomyldata`-variable (see above).

Finally the unique ID has to be specified so that *goCluster* will be able to link each gene to its corresponding annotation data. Often the rows of a dataset will be labeled with the unique ID. In that case you can specify *rownames* here and *goCluster* will retrieve the information from the dataset. But you may also specify the name of a character vector that holds the unique IDs.

In case of the benomyl dataset the rownames have already been replaced with the corresponding yeast symbolic names which provides a convenient link to the *YEAST*-annotation-package available on the bioconductor website.

goCluster will check for the correct class of the dataset variable and it will also verify the length of the unique ID vector by comparing it to the size of the dataset.

3.2 Annotation

Please select one of the following classes for the annotation data:

- 1) Chromosome-abp
- 2) GO-abp

```
# 2
```

Please specify the name of the Bioconductor meta package that applies to your dataset. This has to be an *AnnBuilder* package that holds all information to your array.

If you are using Affymetrix arrays there is a high chance you will find a prepackaged dataset at the Bioconductor website and for spotted array you will have to create your own *AnnBuilder* package.

```
# YEAST
```

If the unique id you specified for the dataset is not the true linker between genes and annotation, please specify the linking annotation here. This is the case for packages like *hgu95av2* where the affymetrix probe ids are the ids that link all annotations together, but the locus link ids are the actual ids that have been

used to establish the package in the first place. The name you specify here needs to yield the name of an environment together with the package you specified the step before. If you don't specify anything this option is disabled.

#

Please choose the type of ontology to use.
You can use the following abbreviations:
Molecular function (MF),
Biological Process (BP),
Cellular component (CC),
all three (GO).
You can select several ontologies
by separating them with a comma.

GO

Please choose the type of GO annotations
that you are willing to accept:
IC: inferred by curator,
IDA: inferred from direct assay,
IEA: inferred from electronic annotation,
IEP: inferred from expression pattern,
IGI: inferred from genetic interaction,
IMP: inferred from mutant phenotype,
IPI: inferred from physical interaction,
ISS: inferred from sequence or structural similarity,
NAS: non-traceable author statement,
ND: no biological data available,
TAS: traceable author statement,
NR: not recorded,
ALL: accept all types given above.
You can select several evidence types
by separating them with a comma. If
you need more information on the
different types, please read this page:
<http://www.geneontology.org/GO.evidence.html>

ALL

The initial class selection lets you specify the type of annotation you want to use for the analysis. The “abp”-suffix signifies that the annotation type is dependant on an *AnnBuilder*-package corresponding to the array type the data was generated with. In the second step you will need to specify the name of this *AnnBuilder*-package.

As mentioned in the previous section the correct package for the benomyl dataset is the *YEAST*-meta-package. The gene ontology has been selected as the annotation type and the corresponding *goCluster*-class allows to exclude certain parts of the gene ontology. Here we chose to include all available information.

goCluster will automatically validate that the meta package you specified provides the necessary environments which hold all requested type of annotation data.

3.3 Clustering

Please select one of the following classes for the gene selection or clustering algorithm:

- 1) Clara
- 2) Hclust
- 3) Kmeans
- 4) Pam

2

Please select the agglomeration method that should be used. For details see the documentation to the "hclust" function (?hclust).

- 1) complete
- 2) ward
- 3) single
- 4) average
- 5) mcquitty
- 6) median
- 7) centroid

1

Please select the distance measure to be used. For details see the documentation to the "hclust" function (?hclust).

- 1) euclidean
- 2) maximum
- 3) manhattan
- 4) canberra
- 5) binary
- 6) minkowski

1

This section allows to specify the algorithm employed in order to partition the genes into groups. While *goCluster* currently only provides four different clustering algorithms it can easily be extended to utilize any kind of gene selection algorithm that will return one or several gene lists. These can then be analysed within the *goCluster*-framework.

Here we select the hierarchical clustering algorithm. The distance measure is set to euclidian and agglomeration is according to the complete linkage algorithm.

3.4 Significance Analysis

Please select one of the following classes for the

significance analysis:

1) Base
2) Bonferroni
3) FDR

3

Please select how often randomization
should be performed. [0, 10000]

4

Please select the false discovery rate
for the selection of GO terms. [0, 1]

0.05

Here we choose the false discovery rate in order to correct for the massive multiple testing performed when testing the gene ontology terms for each of the clusters identified.

The procedure will replace each of the clusters identified with the same number of randomly selected genes. This will be repeated as often as given in the second parameter. We selected only four repetitions here in order to increase the speed of calculation but in a standard analysis the randomization should be performed around a hundred times.

The significance analysis will use the statistical function configured in the next section in order to establish a histogram over the p-values obtained. This distribution can then be combined with the FDR threshold specified by the user in order to select significant annotation terms.

Please note that this type of statistical approach is not entirely valid for an annotation type like the gene ontology (directed acyclic graph). The results should be considered to be an indication rather than mathematically precise values (the interested reader is referred to a discussion of the problem written by R. Gentleman [<http://bioconductor.org/Docs/Papers/2003/Compendium/GOstats.pdf>]). This problem does not occur in case you use an annotation type that provides terms which are independent of each other.

3.5 Significance Analysis

Please select one of the
following classes for the
statistical analysis algorithm:

1) Hyper

1

Currently you can only choose the hyper geometric distribution in order to identify the enrichment of functionally related genes within the clusters.

3.6 Visualization

Please select one of the
following classes for the
visualization:

```
1 ) Hclust
2 ) Heatmap
3 ) None
```

```
# 1
```

Since hierarchical clustering has been chosen as clustering method the corresponding visualization method is being selected.

3.7 Retrieving the configuration

The `setup`-method allows you to retrieve the configuration stored in a *goCluster*-object. The value returned will be a list that contains all relevant parameters. The following command demonstrates how you can retrieve the options set for the clustering algorithm:

```
> setup(benomy1Configured)[["algo"]]

$method
[1] "complete"

$distance
[1] "euclidean"
```

To get an overview of a *goCluster* object you can simply print the variable.

```
> print(benomy1Configured)
```

```
goCluster-Object:
```

```
-----
Dataset:
```

```
-----
```

```
Type: Bioconductor ExprSet object
```

```
Details:
```

```
-> Expression Set (exprSet) with
->      400 genes
->      16 samples
->      phenoData object with 0 variables and 0 cases
->      varLabels
```

```
Annotation:
```

```
-----
```

```
Type: Gene Ontology (GO)
```

```
Details:
```

-> GO information retrieved from meta package YEAST
-> GO ontologies selected: MF, BP, CC
-> Accepted type of evidence: IC, IDA, IEA, IEP, IGI, IMP, IPI, ISS, NAS, ND, TAS, NR

Algorithm:

Type: Hclust

Details:

-> Distance measure: euclidean
-> Clustering method: complete
-> The object holds no result yet.

Significance:

Type: False Discovery Rate

Details:

-> The object holds no result yet.

Statistic:

Type: Hypergeometric distribution

Details:

-> The object holds no result yet.

Visualization:

Type: Hclust

Details:

-> The object holds no result yet.

4 Executing *goCluster*

The configured object can be analysed by calling its `execute`-method.

```
> benomylExecuted <- execute(benomylConfigured)
```

```
Loading required package: GO
```

```
Building GO annotation...
```

```
Done!
```

```
Clustering the dataset.
```

```
Randomizing groups.
```

```
25 %..50 %..75 %..100 %..
```

```
Analyzing random data.
```

```
Going to analyze 1596 groups.
```

```
This might take a while...
```

```
33 %..67 %..100 %..
```

```
Analyzing original data.
```

```
Going to analyze 399 groups.
```

```
This might take a while...
```

```
Selecting significant annotations.
```

The selected (significant) gene ontology terms can be found in the `selection`-slot of the significance analysis object. This object is stored in the `sign`-slot of the main `goCluster`-object.

The object can be printed again in order to get an overview of the result.

```
> print(benomylExecuted)
```

```
goCluster-Object:
```

```
-----
```

```
Dataset:
```

```
-----
```

```
Type: Bioconductor ExprSet object
```

```
Details:
```

```
-> Expression Set (exprSet) with
```

```
->         400 genes
```

```
->         16 samples
```

```
->         phenoData object with 0 variables and 0 cases
```

```
->         varLabels
```

```
Annotation:
```

```
-----
```

```
Type: Gene Ontology (GO)
```

Details:

- > GO information retrieved from meta package YEAST
- > GO ontologies selected: MF, BP, CC
- > Accepted type of evidence: IC, IDA, IEA, IEP, IGI, IMP, IPI, ISS, NAS, ND, TAS, NR

Algorithm:

Type: Hclust

Details:

- > Distance measure: euclidean
- > Clustering method: complete
- > The dataset has been clustered.

Significance:

Type: False Discovery Rate

Details:

- > The clustered data has been analyzed.
- > A total of 1285 annotation terms have been selected

Statistic:

Type: Hypergeometric distribution

Details:

- > A total of 1197 clusters (this might include randomized datasets) have been analyzed.

Visualization:

Type: Hclust

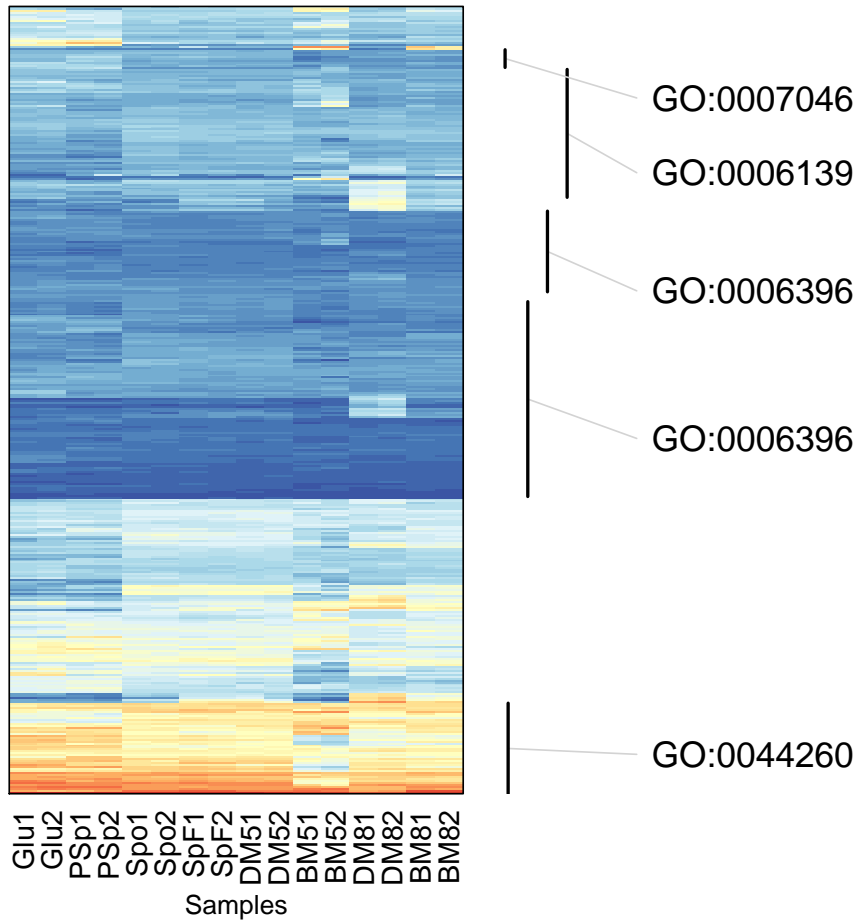
Details:

- > The clustered data has been prepared for visualization.
- > A total of 3 annotation types can be visualized.

5 Visualizing the result

The result can be visualized using the *display* function. The resulting plot depends on the visualization function chosen earlier and will be a hierarchical clustering in this case. The display function can take different arguments depending on the type of function selected. In case of the hierarchical clustering display it is necessary to choose the type of GO annotation that should be displayed alongside the plot.

```
> display(benomylExecuted, selection = 2)
```



In this type of plot *goCluster* highlights regions in which an annotation term was found to be significantly overrepresented. In case such regions overlap, only the term with the lowest p-value will be displayed.

In this case *goCluster* only identifies a small amount of terms since we initially reduced the dataset to only 400 genes.

In order to display one of the identified GO terms the visualization method can be switched to the heatmap. *benomylExecuted* is copied to *benomylExecuted2* and the visualization method is replaced in this new object by swapping the *visu* slot to a new object of the type *clusterVisualHeatmap*.

```
> benomylExecuted2 <- benomylExecuted
> benomylExecuted2@visu <- new("clusterVisualHeatmap")
> benomylExecuted2@visu <- execute(benomylExecuted2@visu, benomylExecuted2)
```

After executing just this subobject again, the same dataset can be visualized in a different manner. This time the selection variable isolates a specific GO term. The genes which were responsible for the identification of the GO term as being overrepresented in one of the clusters will be displayed in a heatmap.

```
> display(benomylExecuted2, selection = "GO:0009058")
```

