Gene set enrichment analysis with **topGO**

Adrian Alexa, Jörg Rahnenführer

April 21, 2009
`http://www.mpi-sb.mpg.de/∼alexa`

Contents

1 Introduction **2**

1.1 Step 1: Preparing the data 2

1.2 Step 2: Running the desired tests 3

1.3 Step 3: Analysis of results 4

2 Session Information **6**

References **6**

1 Introduction

This manuscript provides a quick tour into topGO. There is an accompanying document which provides details on the functions used in this document as well as showing more advance functionality implemented in the topGO package.

The topGO package is designed to work with different test statistics and with multiple GO graph algorithms, see [Alexa, A., *et al.*, 2006].

This section describes a simple working-session using topGO. There are only a handful of commands necessary to perform a gene set enrichment analysis and we will be briefly presented them bellow.

A typical session can be divided into three steps:

1. Collection and preprocessing of the data. The list of genes and the correspondent gene' score, the criteria for selecting genes based on their scores and the gene-to-GO annotations are all collected to form a suitable R object.
2. Using the object created in the first step the user can perform enrichment analysis using various statistical tests and methods that deal with the GO topology.
3. Analysis of the results.

But before going through each of these steps we need to decide which biological question we want to answer. This will most likely dictate which test statistic we need to use. The choice of the test is also restricted by the data available at hand. The knowledge of the test will dictate the way the gene expression data needs to be process. Here we will test for enrichment of GO terms with differentially expressed genes using the Kolmogorov-Smirnov test and Fisher's exact test as examples.

1.1 Step 1: Preparing the data

In this step a convenient R object of class topGOdata is created containing all the information required for the remaining two steps. The user needs to provide a list of genes, GO annotations and a criteria for selecting interesting genes.

In most cases we want to test enrichment of GO terms with differentially expressed genes. Thus, the starting point is a list of genes and the respective p-values for differential expression. A toy example of a list of gene identifiers and the respective p-values is provided by the geneList object.

```
> library(topGO)
> library(ALL)
> data(ALL)
> data(geneList)
```

The geneList data is based on an differential expression analysis of the ALL dataset. It contains just a small number, 323, of genes with the corespondent p-values. For these genes we need GO annotations to be able to form the gene groups. The information on where to find the GO annotations is stored in the ALL object and its easily accessible.

```
> affyLib <- paste(annotation(ALL), "db", sep = ".")
> library(package = affyLib, character.only = TRUE)
```

The chip used in the experiment is the hgu95av2 from Affymetrix, as we can see from the affyLib object. When we loaded the geneList object a function which will select the differentially expressed genes from the list was also loaded under the name of topDiffGenes. For example using this function we can see that there are 50 genes with a row p-value less than 0.01 out of a total of 323 genes.

```
> sum(topDiffGenes(geneList))
```

```
[1] 50
```

We now have all the data necessary to build on object of type `topGOdata`. This object will contain all the gene identifiers and their score, the GO annotations, the GO hierarchical structure and all the other information needed to perform the enrichment analysis.

```
> sampleGOdata <- new("topGOdata", description = "Simple session", ontology = "BP",
+     allGenes = geneList, geneSel = topDiffGenes, nodeSize = 10, annot = annFUN.db,
+     affyLib = affyLib)
```

A summary of the `sampleGOdata` object can be seen by typing the object name at the R prompt. Having all the data stored into this object facilitates the access to identifiers, annotations and to obtain basic data statics.

```
> sampleGOdata
```

1.2 Step 2: Running the desired tests

Once we have an object of class `topGOdata` we can start with the enrichment analysis. Since for each gene we have a score and there is also a procedure to select interesting genes based on the scores we will use two types of test statistics: Fisher's exact test which is based on gene counts, and a Kolmogorov-Smirnov like test which needs gene scores.

The function `runTest` is used to apply the specified enrichment test and method to the data. It has three basic arguments. The first argument needs to be on object of class `topGOdata`. The second argument is of type character and specifies which method for dealing with the GO graph structure will be used (this argument is optional). And the third argument specifies the test statistic and is of type character.

First we perform a classical enrichment analysis by testing the over-representation of GO terms with differentially expressed genes. In the classical case each GO category is tested independently.

```
> resultFisher <- runTest(sampleGOdata, algorithm = "classic", statistic = "fisher")
```

`runTest` returns an object of class `topGOresult`. A short summary of this object is shown bellow.

```
> resultFisher

Description: Simple session
Ontology: BP
'classic' algorithm with the 'fisher' test
408 GO terms scored: 33 terms with p < 0.01
Annotation data:
    Annotated genes: 318
    Significant genes: 50
    Min. no. of genes annotated to a GO: 10
    Nontrivial nodes: 365
```

Next we will test the enrichment using the Kolmogorov-Smirnov test. We will use the both the classic and the elim methods.

```
> resultKS <- runTest(sampleGOdata, algorithm = "classic", statistic = "ks")
> resultKS.elim <- runTest(sampleGOdata, algorithm = "elim", statistic = "ks")
```

Please note that some statistical tests will not work with any method. The compatibility matrix between the methods and statistical tests is shown in Table **??**.

The p-values computed by the `runTest` function are unadjusted for multiple testing. We do note advocate against adjusting the p-values of the tested groups, but in many cases the an adjusted p-value can be misleading.

	GO.ID	Term	Annotated	Significant	Expected	Rank in classicFisher	classicFisher	classicKS	elimKS
1	GO:0007067	mitosis	183	17	28.77	362	0.99993	2.9e-09	2.9e-09
2	GO:0051301	cell division	126	15	19.81	332	0.95458	9.6e-06	9.6e-06
3	GO:0000087	M phase of mitotic cell cycle	188	22	29.56	355	0.99391	7.9e-10	0.0019
4	GO:0022403	cell cycle phase	192	26	30.19	322	0.92925	2.6e-10	0.0041
5	GO:0000226	microtubule cytoskeleton organization	30	4	4.72	252	0.72881	0.0054	0.0054
6	GO:0019219	regulation of nucleobase, nucleoside, nu...	47	12	7.39	72	0.04216	0.0067	0.0067
7	GO:0050789	regulation of biological process	181	35	28.46	50	0.02882	0.0024	0.0075
8	GO:0032946	positive regulation of mononuclear cell ...	30	12	4.72	1	0.00062	0.0120	0.0120
9	GO:0050671	positive regulation of lymphocyte prolif...	30	12	4.72	2	0.00062	0.0120	0.0120
10	GO:0050794	regulation of cellular process	180	34	28.30	76	0.05189	0.0042	0.0131

Table 1: *Significance of GO terms according to* classic *and* elim *methods.*

1.3 Step 3: Analysis of results

After the enrichment tests are performed the researcher needs tools for analysing and interpreting the results. `GenTable` is an easy to use function for analysing the most significant GO terms and the corresponding p-values. For example, we want to see which are the top 10 significant GO terms identified by the elim method. We also want to compare the ranks and the p-values of these GO terms in the case where the classic method was employe used.

```
> allRes <- GenTable(sampleGOdata, classicFisher = resultFisher, classicKS = resultKS,
+     elimKS = resultKS.elim, orderBy = "elimKS", ranksOf = "classicFisher",
+     topNodes = 10)
```

The `GenTable` function returns a data.frame containing the top `topNodes` GO terms identified by the elim algorithm, see `orderBy` argument. The data.frame includes some statistics on the GO terms and the p-values obtained from the methods passes as arguments. Table 1 shows the results.

For accessing the GO term's p-values from a `topGOresult` object the user should use the `score` functions. As a simple example, we look at the differences between the results of the classic and the elim methods in the case of the Kolmogorov-Smirnov test. Due to the fact that there are few significant GO terms, 17 terms with a p-value less than 0.01 in the classic method, one would expect that only few GO terms will have different p-values between these two methods. This, of course, depends on the value of the cutoff parameter used by the elim method.

```
> pValue.classic <- score(resultKS)
> pValue.elim <- score(resultKS.elim)[names(pValue.classic)]
> plot(pValue.classic, pValue.elim, xlab = "p-value classic", ylab = "p-value elim",
+     cex = 0.5)
```

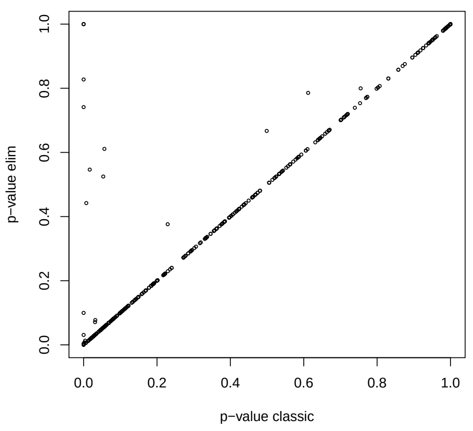

Figure 1: *p-values scatter plot for the* classic *and* elim *methods.*

We can see in Figure 1 that there are indeed few differences between the two methods. Some GO terms witch are found significant by the classic method are less significant in the elim, as expected.

Another insightful way of looking at the results of the analysis is to investigate how the significant GO terms are distributed over the GO graph. For the `elim` algorithm the subgraph induced by the 5 most significant GO terms is plotted. In the plot, the significant nodes are represented as boxes. The plotted graph is the upper induced graph generated by these significant nodes.

```
> showSigOfNodes(sampleGOdata, score(resultKS.elim), firstTerms = 5, useInfo = "all")
```

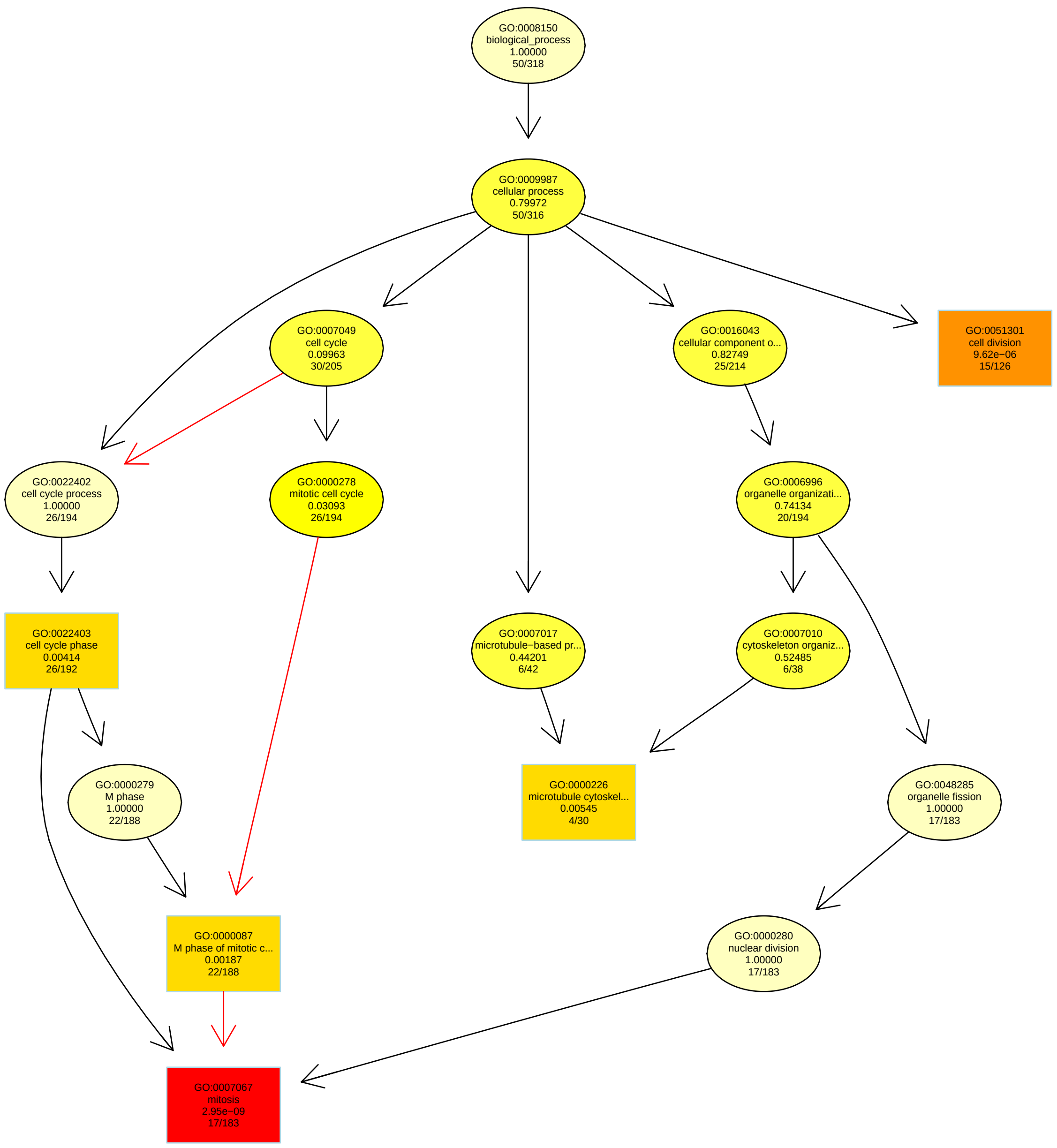
GO:0008150 biological_process 1.00000 50/318
GO:0009987 cellular process 0.79972 50/316
GO:0007049 cell cycle 0.09963 30/205
GO:0016043 cellular component o... 0.82749 25/214
GO:0051301 cell division 9.62e−06 15/126
GO:0022402 cell cycle process 1.00000 26/194
GO:0000278 mitotic cell cycle 0.03093 26/194
GO:0006996 organelle organizati... 0.74134 20/194
GO:0022403 cell cycle phase 0.00414 26/192
GO:0007017 microtubule−based pr... 0.44201 6/42
GO:0007010 cytoskeleton organiz... 0.52485 6/38
GO:0000279 M phase 1.00000 22/188
GO:0000226 microtubule cytoskel... 0.00545 4/30
GO:0048285 organelle fission 1.00000 17/183
GO:0000087 M phase of mitotic c... 0.00187 22/188
GO:0000280 nuclear division 1.00000 17/183
GO:0007067 mitosis 2.95e−09 17/183

Figure 2: *The subgraph induced by the top* 5 *GO terms identified by the* elim *algorithm for scoring GO terms for enrichment. Rectangles indicate the* 5 *most significant terms. Rectangle color represents the relative significance, ranging from dark red (most significant) to light yellow (least significant). Black arrows indicate* is-a *relationships and red arrows* $part$-of *relationships. For each node, some basic information is displayed. The first two lines show the GO identifier and a trimmed GO name. In the third line the raw p-value is shown. The forth line is showing the number of significant genes and the total number of genes annotated to the respective GO term.*

2 Session Information

The version number of R and packages loaded for generating the vignette were:

- R version 2.9.0 (2009-04-17), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US;LC_NUMERIC=C;LC_TIME=en_US;LC_COLLATE=en_US;LC_MONETARY=C;LC_MESSAGES=en_U`
- Base packages: base, datasets, graphics, grDevices, grid, methods, stats, tools, utils
- Other packages: ALL 1.4.4, AnnotationDbi 1.6.0, Biobase 2.4.0, DBI 0.2-4, GO.db 2.2.11, graph 1.22.0, hgu95av2.db 2.2.11, Rgraphviz 1.22.0, RSQLite 0.7-1, SparseM 0.79, topGO 1.12.0, xtable 1.5-5
- Loaded via a namespace (and not attached): cluster 1.11.13, lattice 0.17-22

References

[Alexa, A., *et al.*, 2006] Alexa, A., *et al.* (2006). Improvined scoring of functional groups from gene expression data be decorrelating go graph structure. *Bioinformatics*, 22(13):1600–1607.