Scripts documentation

Contents

With the following R scripts, it's possible to perform a statistical analysis on RNA-seq data. Bioinformatics and biostatistics major steps and methods for the anaysis of this type of data is detailed in *"A survey of best practices for RNA-seq data analysis"*, Conesa 2016.

These three scripts can be run either directly on R/Rstudio or using a command line on a cluster for example but **R version 3.2 minimum** is required. Packages importation is integrated in each script but you need to change the environment variable `R_LIBS` if you don't have the right to write in the folder which are save R packages:

```
export R\_LIBS=‘‘./your_dir_for_R_packages/’’
```

1 Normalization

Normalization is a process designed to identify and remove systematic technical differences between samples. This process aims at ensuring that technical bias has minimal impact on the results of statistical analyses such as differential expression analysis. The most common symptom of the need for normalization is differences in the total number of aligned reads (sequencing depth).

1.1 Methods

All the normalization methods considered here are global procedures, in the sense that only a single factor is used to scale the counts of every sample, C_j. For the three following methods, we have:

$$C_j = \frac{\exp(\frac{1}{N}\sum_{l=1}^{N}\log\tilde{C}_l)}{\tilde{C}_j}$$

with N the number of samples.

Further details can be found in *"A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis"*, Dillies 2012. None of the methods is the best for all datasets but on all methods, only the RLE and TMM normalization methods seem to be robust to the presence of different library sizes and widely different library compositions.

Notations:

- G: number of genes,
- N: number of samples,
- K_{gj}: counts (number of reads) of gene g in sample j,
- D_j: total number of reads for sample j.

Upper quartile

Gene counts are divided by a quartile (usually 75% quartile) of count distribution which are different from 0 in the sample. This normalization factor is itself divided by the average upper quartile across all samples to ensure that the corrected counts have a scale similar to the original ones:

$$\tilde{C}_j = \frac{10^6}{D_j Q_j^{(p)}}$$

with $Q_j^{(p)}$ the p-th quantile

RLE

This normalization method is based on the hypothesis that most genes are not DE. A scaling factor for a given sample is computed as the median of the ratio of a gene read count over its geometric mean across all samples:

1. compute a pseudo-reference sample: $R_g = \left(\prod_{j=1}^{N} K_{gj}\right)^{\frac{1}{N}}$,

2. center samples compared to the reference: $\tilde{K}_{gj} = \frac{K_{gj}}{R_g}$,

3. calculate normalization factor: $\tilde{C}_j = \text{median}_g\{K_{gj}\}$.

The underlying idea is that non-DE genes should have similar read counts across samples, leading to a ratio of 1. Assuming most genes are not DE, the median of this ratio provides an estimate of the correction factor that should be applied to all read counts of the corresponding sample to fulfill the hypothesis.

TMM

This normalization method is also based on the hypothesis that most genes are not DE. The TMM factor is computed with one sample being considered as a reference sample and the others as test samples (the method is not sensitive to which sample is chosen as a reference). For each test sample, TMM is computed as the weighted mean of log ratios between this test and the reference, after exclusion of the most expressed genes and the genes with the largest log ratios:

1. remove extreme data for fold-changed (M) and average intensity (A) (figure 1):

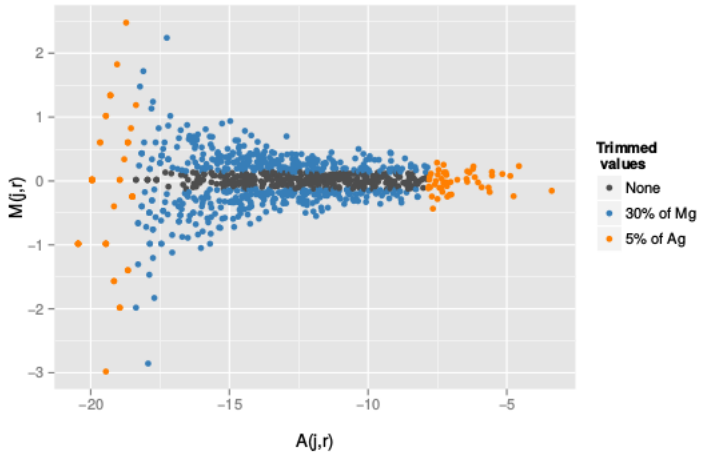

Figure 1: Extreme data removed

2. on remaining data, calculate the weighted mean of M-values:

$$TMM(j,r) = \frac{\sum_{g:not\ trimmed} w_g(j,r) M_g(j,r)}{\sum_{g:not\ trimmed} w_g(j,r)}$$

with $w_g(j,r)$ an appropriate weight that approximates the inverse of the variance for M,

3. calculate normalization factor: $\tilde{C}_j = 2^{TMM(j,r)}$.

According to the hypothesis of low DE, this TMM should be close to 1. If it is not, its value provides an estimate of the correction factor that must be applied to the library sizes (and not the raw counts) in order to fulfill the hypothesis.

1.2 Plots

The diagnostic plots can be used to choose the best normalization method. Normalization methods search for homogeneity, especially inside a given experimental condition (*e.g.*, control/treated).

Boxplots

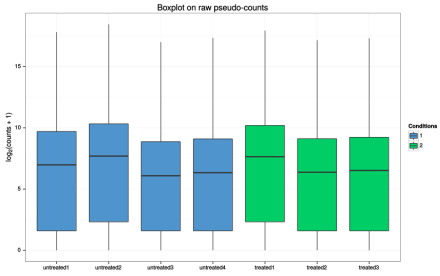

Figure 2: Bad

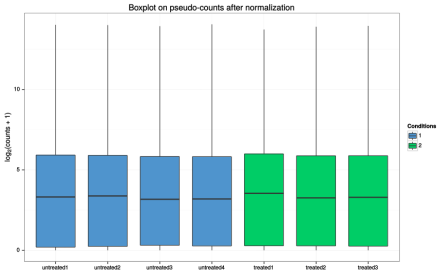

Figure 3: Good

Density plots

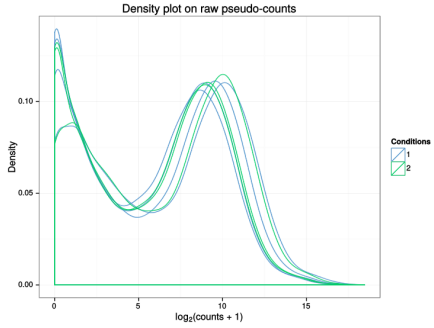

Figure 4: Bad

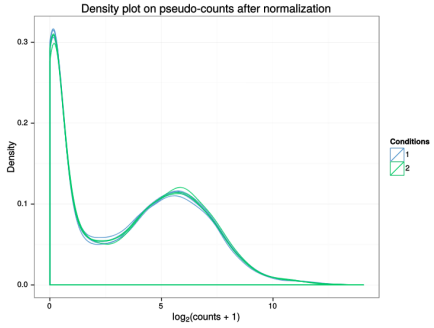

Figure 5: Good

MAplots

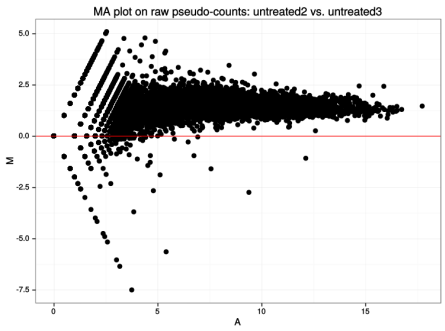

Figure 6: Bad

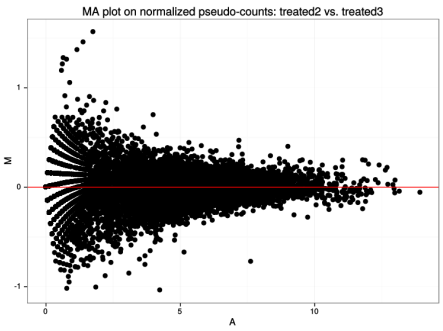

Figure 7: Good

1.3 Script Normalization.R

This script uses the R package `edgeR` to import data, perform the three normalization methods described above and produce diagnostic plots.

To run it, two options are mandatory:

- with a command line (Linux):

```
Rscript --vanilla <script_path>/Normalization.R
-f <raw_data_path>/<file_name>.csv
-o <results_path>
```

- in R or Rstudio : before execution, you need to comment part "Parameter initialization when the script is launched from command line", uncomment part "Parameter initialization when the script is launched from R or Rstudio" and change paths in opt$file and opt$out. The all script can then be run.

Parameters:

- file: path and name of the raw counts file. Columns in this file need to be **separated by tabulation** and have **samples names as columns names** (e.g. gene_id lib1 lib2),
- out: folder path where results are stored (if the folder does not exist it will be created).

Outputs:

- plots: boxplot, density plot and MDS on raw counts and normalized counts (three methods) are produced (<method>_<type>.svg) and saved in file defined in the out parameter. Tables to reproduce these plots are also saved (<method>_<type>.csv).

- pseudo-counts: three files with normalized pseudo-counts for each method (<method>_pseudocounts.csv : tabulate separation between columns (e.g. gene_id lib1 lib2)),
- normalization factors: the files with samples names and sizes and normalization factors (<method>_info.txt : tabulate separation between columns (e.g. sample.name lib.size norm.factors)).

2 Differential expression analysis

A differential expression analysis selects genes with significantly different level of expression between experimental conditions. At least TWO replicates by condition are needed to carry out a reliable statistical analysis. Nevertheless, an analysis is performed with Fisher test if only one replicate is available in a condition but the results of this test are not trustful.

2.1 Methods

Three steps are performed on normalized counts to obtain a list of differentially expressed (DE) genes:

1. Estimate dispersion (not with less than 2 replicates)
2. Filter low expressed genes (not with less than 2 replicates)
3. Perform a test between conditions

2.1.1 Estimation of dispersion

In edgeR, pseudo-counts are modeled by a negative binomial distribution, so an estimation of dispersion parameters is needed. The edgeR approach implements the empirical Bayes strategy proposed by Robinson and Smyth 2007[1]; Robinson and Smyth 2008[2] for estimating the tagwise negative binomial dispersions. First, it uses the count data for estimating a common dispersion parameter for all the genes by conditional maximum likelihood (figure 8). The profile likelihood is then adjusted to the prior distribution for tagwise dispersions (figure 9). Finally, the adjusted profile is used as a prior value for a second estimation round, which results in the final estimates of dispersion (figure 10).

2.1.2 Independent filtering

Weakly expressed genes in RNA-Seq data have low chance to be declared differentially expressed. The reason is that the variability of these counts is often too large for the differences to be relevant.

The idea of independent filtering is to filter out those genes from the procedure that have no, or little chance of showing significant evidence, without even looking at their test statistic. At first sight, there may seem to be little benefit in filtering out these genes. However, these genes have an influence on

[1] Robinson, M. D., & Smyth, G. K. (2007). Moderated statistical tests for assessing differences in tag abundance. Bioinformatics, 23(21), 2881-2887.

[2] Robinson, M. D., & Smyth, G. K. (2008). Small-sample estimation of negative binomial dispersion, with applications to SAGE data. Biostatistics, 9(2), 321-332.

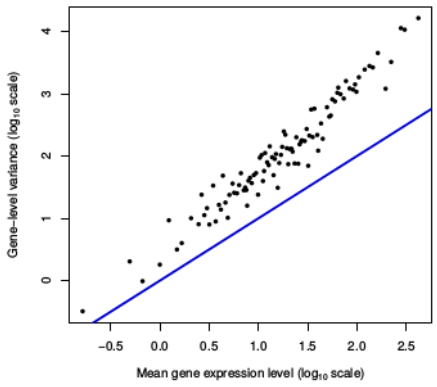

Figure 8: Step 1

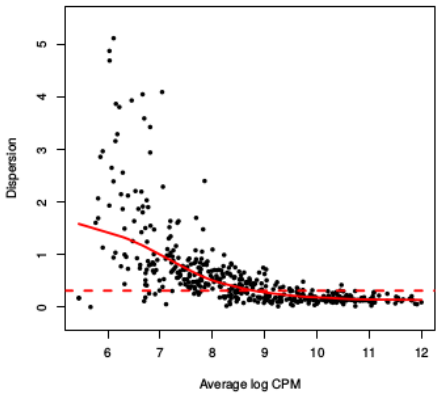

Figure 9: Step 2

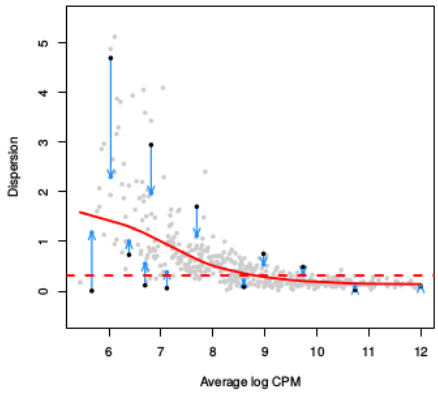

Figure 10: Step 3

the multiple testing adjustment, which is less severe if such genes are removed. By removing the weakly-expressed genes from the input to the FDR procedure, the power of our test is improved.

HTSFilter (package HTSFilter) is a function which implements a data-based filtering procedure based on the calculation of a similarity index among biological replicates for read counts arising from replicated transcriptome sequencing (RNA-seq) data.

2.1.3 Test

When there is more than one replicate by condition, a test based on the negative binomial distribution is performed. Otherwise, Fisher's exact test is used. Fisher's exact test is **NOT** recommended because it does not estimate the variability in gene expression.

Negative binomial model and test Count data are usually modeled with the Poisson distribution but due to heterogeneity between individuals, the vari-

ance is often much larger than the mean so this distribution isn't realistic for this type of data. With a negative binomial model, the variability is properly capture. We can suppose that the count for gene g K_g follow a negative binomial distribution with parameters ϕ_g and p_g, $K_g \sim NB(\phi_g^{-1}, p_g)$, parametrised in ters of its mean λ_g and variance σ_g^2, via

$$p_g = \frac{\lambda_g}{\sigma_g^2} \quad \text{and} \quad \phi_g = \frac{\sigma_g^2 - \lambda_g}{\lambda_g^2}$$

Then, we wish to test if the mean of counts under experimental condition A is the same as that in condition B. Thus, the null hypothesis is :

$$\mathrm{H}_0 : \lambda_{gA} = \lambda_{gB}, \quad \text{for each gene } g$$

Fisher exact test Fisher exact test fixes the marginal totals of the 2 x 2 contingency table (table **??**) and tests differential expression using the null hypotheses that treatment do not affect gene expression levels (i.e., that the two groups are independent).

	Group A	Group B	Total
Gene g	n_{gA}	n_{gB}	n_g
Remaining genes	$N_A - n_{gA}$	$N_B - n_{gB}$	$N - n_g$
Total	N_A	N_B	N

Table 1: 2x2 contingency table for gene g

Multiple testing correction Because multiple tests are carried out simultaneously, a correction for multiple testing **IS NEEDED** to keep an acceptable false-positive rate. For example, if you don't control the global risk, with 20 tests, we have a 64% chance of observing at least one significant result, even if all of the tests are actually not significant. There is two types of correction based on:

- the control of the probability that at least one test is declared positive whereas it is not (i.e., it controls the probability to have at least one type I error). This quantity is called familywise error rate (FWER). The Bonferroni correction is a correction designed to control FWER,
- and the control of the expected proportion of type I error among the rejected hypothesis (FDR: false discovery rate). The Benjamini & Hochberg (BH) correction is a correction designed to control FDR.

FWER correction is appropriate when you want to guard against any false positives. However, in many cases, a certain number of false positive is not a problem and the user will prefer not to filter out possible positive genes. In this case, the more relevant quantity to control is the false discovery rate (FDR). In summary, BH correction is less severe than Bonferroni correction and will yield to a larger number of DEG.

2.2 Plots

Heatmap

To explore the similarity between genes, it is often instructive to combine clustering methods with a graphical representation of the "primary" count table by means of the so-called clustering image map (CIM) or heatmap.

A CIM (or heatmap) is a two-dimensional, rectangular, colored grid, representing each data point (rectangle) with a color that quantitatively and qualitatively reflects the original experimental observations. The rows (and/or columns) of the matrix are rearranged (independently) according to some hierarchical clustering method, so that genes or groups of genes with similar expression patterns are adjacent. The computed dendrogram (tree) resulting from the clustering is added to a side of the image to indicate the relationships among genes.

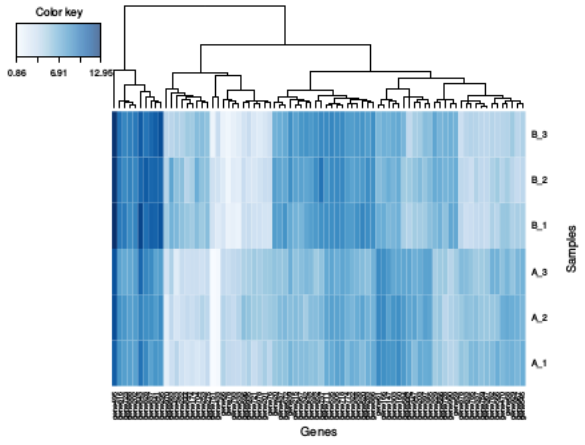

Figure 11: Example

MDS or PCA plots

This type of plot is useful for visualizing the overall effect of experimental covariates and batch effects.

The plot is produced from the output of Principal Component Analysis (PCA) on the transposed of the given count matrix. PCA is used to reduce the dimension of multidimensional datasets. The method builds linear combination of the original data that are designed so as to be the best summary of the original dataset (formally, they are the combinations with the largest variability so as to preserve the entropy of the original data at best).

The purpose of multidimensional scaling (MDS) is to provide a visual representation of the pattern of proximities (i.e., dissimilarities or distances) among a set of objects. MDS takes a set of dissimilarities and returns a set of points in 2D such that the distances between the points are the most similar to the original dissimilarities as possible. In the present case, the dissimilarity between each pair of samples (columns) is the root-mean-square deviation (Euclidean

distance) for the top genes. Distances on the plot can be interpreted as leading log2-fold-change, meaning the typical (root-mean-square) log2-fold-change between the samples for the genes that distinguish those samples.

PCA and MDS are equivalent when used with the dissimilarity used in MDS is the Euclidean distance. With an MDS analysis (or with PCA), other effects than the experimental conditions are sometimes visible on the 2D projection of the data.

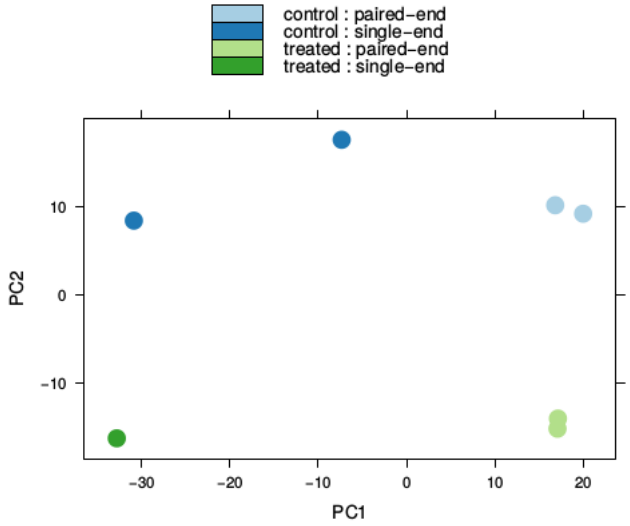

Figure 12: Here, an effect of extracting methods (single-end or paired-end) and condition (treated or control).

2.3 Script DEG.R

This script uses the R package `edgeR` to import data, do tests to find differentially expressed genes and produces diagnostic plots.

Two options are mandatory to run the test:

- with a command line (Linux):

```
Rscript --vanilla <script_path>/Normalization.R
-f <raw_data_path>/<file_name>.csv
-n <normalized_factor_path>/<file_name>.csv
-o <results_path> --pool1 lib1,lib2...
--pool2 lib3,lib4... --filter true or false
--correct "BY" or "BH" --MAplots true or false
--alpha 0.05
```

- in R or Rstudio: before execution, you need to comment part “Parameter initialization when the script is run from command line”, uncomment part “Parameter initialization when the script is run from R or Rstudio” and change parameters opt$XXX. The all script can then be run.

Parameters:

- file: path and name of the raw counts file. Columns in this file need to be **separated by tabulation** and **have samples names as columns names** (e.g. gene_id lib1 lib2),
- norm: file with normalized factors and library name (e.g. sample.name lib.size norm.factors). This file is obtained with script 'Normalization.R',
- out: folder path where results are stored (if the folder doesn't exist it will be created),
- pool1: library name in pool 1 separeted by ',',
- pool2: library name in pool 2 separeted by ',',
- filter: if TRUE low expressed genes are removed,
- alpha: significance level of the tests (i.e. acceptable rate of false-positive in the list of differentially expressed genes),
- correct: method used to adjust p-values for multiple testing ('BH' or 'bonferroni'),
- MAplots: if TRUE all MAplots are saved.

Outputs:

- diagnostic plots: a pdf file with all plots (boxplot, density plot, MA-plots and MDS on raw counts and normalized counts (methods chosen at the beginning), histograms on p-values, MA plot and heatmap and MDS on DE genes),
- results: list of DEG (resDEG.csv : tabulate separation between columns (e.g. gene_id log fold-change p-value adjusted p-value over-expressed pool)) and data to reproduce plots (<plot_name>.csv),
- some statistics: number of over-expressed genes (total, in pool 1, in pool 2), number of genes with no counts and number of gene filtered by HTSFilter (statistics.txt),
- json files: two json files are produced in case the results are display in an interface.

3 Gene set enrichment analysis

List of differentially expressed genes is not sufficient for biologists to find the underlying biological mechanism involved. More biological information about this set of genes is needed to enhance the interpretation of such a list. The Gene Ontology consortium[3] provides a database with a correspondence between gene and various functions. These functions are structured as a directed acyclic

[3] http://geneontology.org/

graph, and each term has defined relationships to one or more other terms in the same domain.

With this database, it is possible to find which functions are over-represented in a set of genes compared to the other genes used in the analysis.

3.1 Methods

First, we need to choose a test to find if a GO is over-represented in the set of genes of interest compared to all the genes in the analysis. Two tests can be used: Fisher's exact test based on gene counts and Kolmogorov-Smirnov test based on gene ranks.

Then, a algorithm is chosen to take into account the dependency between GO nodes. Three are proposed in the script : "classic", "elim" and "weight". The last two are explained in *"Improved scoring of functional groups from gene expression data by decorrelating GO graph structure"* (Alexa et al., 2006).

"Classic" algorithm

For all nodes, tests are realized without take into account dependency between nodes. Then a correction for multiple testing is performed as for differential expression analysis. This algorithm is simple but not recommended because of the dependency between nodes.

"Elim" algorithm

The "elim" method investigates the nodes in the GO graph bottom-up. At each level, tests and Bonferroni correction of p-value are performed for each node. If a node is found significant all genes mapped to it are removed in all its ancestors thus following tests are performed without these genes.

"Weight" algorithm (only with Fisher's exact test)

With the strategy implemented in the "elim" algorithm a node is considered to be significant if it p-value is below a given threshold. Because of the removal process, more significant nodes on higher levels can be missed. An alternative is implemented in the "weight" method. Nodes are not removed but genes are weighted.

At the beginning, all genes have a weight set to 1. Let u be the currently processed node in the bottom-up process. If node u has a lower p-value than its children, genes contained in the children are down-weighted (old weight are multiplied with new weight) and then p-values are updated. Else if at least one child has a lower p-value than u, all genes of these children are down-weighted in the node u and its ancestors and then p-value are recomputed on the new weights.

Weight between node u and v are calculated with a predefined function which can be, in the script, a simple ratio $\frac{u}{v}$ or a ratio of natural logs $\frac{\log v}{\log u}$.

3.2 Plots

To represent differences between GO in a set of genes (generally differentially expressed genes) and GO in all the genes, the script create a barplot with the

percent of gene of the sample for each GO at level 2 (see figure 13).

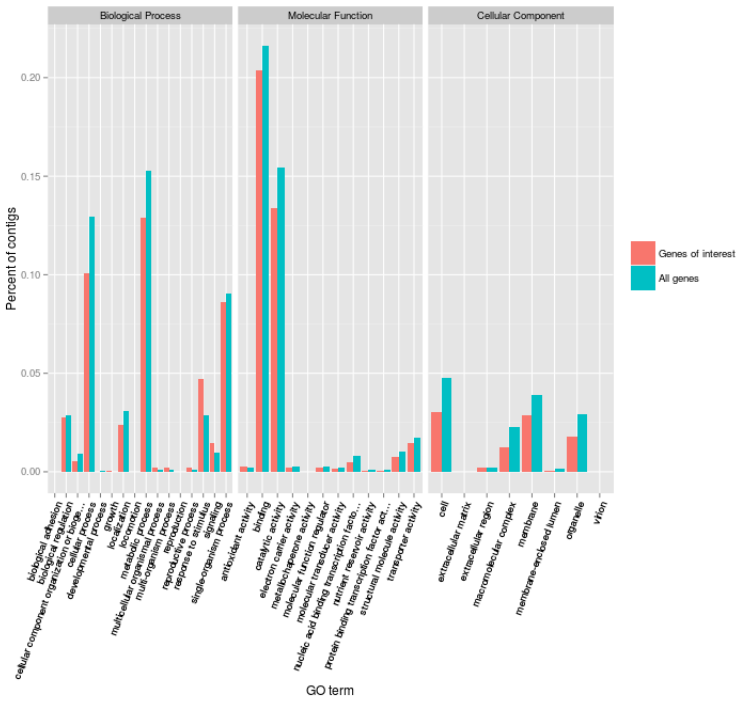

Figure 13: Barplot of level 2 GO

After the enrichment analysis, significant GO can be represented in a DAG (see figure 14). Boxes indicate the 10 most significant terms. Box color represents significance, ranging from dark red (most significant) to light yellow (least significant).

3.3 Script GOEnrichment.R

This script performs an enrichment analysis for Gene Ontology (GO) and produces

Two options are mandatory to run the test:

- with a command line (Linux):

```
Rscript --vanilla <script_path>/GOEnrichment.R
-f <raw_data_path>/<file_name>.csv
--fileFormat "topGO" or "twoColumns"
--interest <interest_data_path>/<interest_file_name>.csv
-o <results_path> --test "fisher" or "ks"
--algorithm "classic", "elim" or "weight"
--paramAlgo "bonferroni", "BH", 0.01, 0.05, "ratio" or "log"
--alpha 0.05
--target "contigs", "gene"...
```

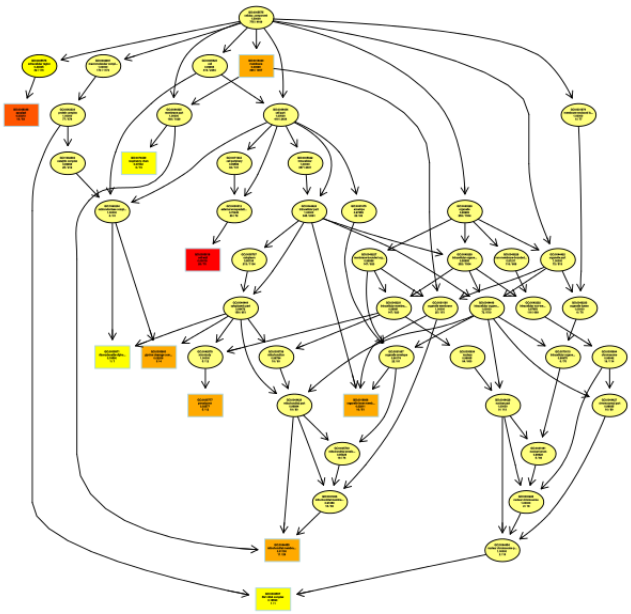

Figure 14: DAG of cellular component ontology

- in R or Rstudio: before execution, you need to comment part "Parameter initialization when the script is run from command line", uncomment part "Parameter initialization when the script is run from R or Rstudio" and change parameters opt$XXX. The all script can then be run.

Parameters:

- file: tabulated matrix with the correspondence gene to GO,
- fileFormat: format of previous file, either one gene by line ("topGO": gene_id1 GO_1,GO_2,...) or two columns ("twoColumns": gene_id GO_id),
- interest: matrix with id of genes of interest in the first column (all these genes must be in the first file),
- out: folder path where results are stored (if the folder doesn't exist it will be created),
- test: test used to find significant GO ('fisher' or 'ks' Kolmogorov-Smirnov),
- algorithm: algorithm used to correct p-value ('classic', 'elim' or 'weight'),
- paramAlgo:
  - for 'classic' algorithm: method used to correct p-values ('bonferroni' or 'BH'),
  - for 'elim' algorithm: value used to remove genes from significant GO,

  - for 'weight' algorithm: weighted function used ('ratio' or 'log'),

- alpha: significance level of the tests,
- target: type of data (e.g. "contigs" or "genes").

Outputs:

- plots: barplot and three DAG,
- results: list of significant GO (GOSig.csv : tabulate separation between columns (e.g. ontology GO GODef pval)), list of genes associated with significant GO (GOgeneSig.csv : tabulate separation between columns (e.g. ontology GO GODef contigs pval) and data to reproduce barplot (barplot.csv),