tRanslatome: an R/Bioconductor package to portray translational control

Toma Tebaldi <t.tebaldi@unitn.it>,
Erik Dassi <erik.dassi@unitn.it>,
Galena Kostoska <kostoska.galena@unitn.it>,
Gabriella Viero <viero@fbk.eu>,
Alessandro Quattrone <alessandro.quattrone@unitn.it>,

October 17, 2016

Contents

1 Introduction

One way to achieve a comprehensive estimation of the influence of different layers of control on gene expression is to analyze the changes in abundances of molecular intermediates at different levels. For example, comparing changes between abundances of mRNAs in active translation with respect to the corresponding changes in abundances of total mRNAs (by mean of parallel high-throughput profiling) we can estimate the influence of translational controls on each transcript. The tRanslatome package represents a complete platform for comparing data coming from two parallel high-throughput assays, profiling two different levels of gene expression. The package focusses on the comparison between the translatome and the transcriptome, but it can be used to compare any variation monitored at two "-omics" levels (e.g. the transcriptome and the proteome). The package provides a broad variety of statistical methods covering each step of the standard data analysis workflow: detection and comparison of differentially expressed genes (DEGs), detection and comparison of enriched biological themes through Gene Ontology (GO) annotation. The package provides tools to visually compare/contrast the results. An additional feature lies in the possibility to detect enrichment of targets of translational regulators using the experimental annotation contained in the AURA database <http://aura.science.unitn.it/>.

2 tRanslatome in practice

The following code illustrates an analysis pipeline with tRanslatome. For demonstrating tRanslatome in practice we use a dataset coming from a study of Parent et al [**?**]. The dataset is named "tRanslatomeSampleData". In this study, the authors profiled the total and the polysome-bound transcripts in differentiated and undifferentiated human HepaRG cells. Therefore this example presents two levels of gene expression analysis (transcriptome, labelled as "tot" and translatome, labeled as "pol") on cells in two different conditions (undifferentiated, labeled as "undiff" vs. differentiated, labeled as "diff"). Experiments were done by the authors in biological triplicate, labeled as "a","b" and "c". All the steps contained in the code will be explained in more detail in the following sections.

```
> ##loading the tRanslatome package
> library(tRanslatome)
> ##loading the training data set
> data(tRanslatomeSampleData)
> translatome.analysis <- newTranslatomeDataset(expressionMatrix,
+                         c("tot.undiff.a", "tot.undiff.b", "tot.undiff.c"),
+                         c("tot.diff.a", "tot.diff.b", "tot.diff.c"),
+                         c("pol.undiff.a", "pol.undiff.b", "pol.undiff.c"),
+                         c("pol.diff.a", "pol.diff.b", "pol.diff.c"),
+                         label.level= c("transcriptome", "translatome"),
+                         label.condition=c("undifferentiated", "differentiated"))
> ##identification of DEGs with the use of the limma statistical method
> limma.DEGs <- computeDEGs(translatome.analysis,
+
> ##enrichment analysis of the selected DEGs
> CCEnrichment <- GOEnrichment(limma.DEGs,ontology="CC", classOfDEGs="up",
+
> ##performing a comparison of the biological themes enriched
> ##in the two levels of gene expression
> CCComparison <- GOComparison(CCEnrichment)
```

3 DEGs detection

The initial core of the package consists of the class holding input data and results, called `TranslatomeDataset`. Objects of this class can be created through the `newTranslatome-Dataset` function. This function takes as input a normalized data matrix coming from the high throughput experiment with entities (genes, transcripts, exons) in rows and samples (normalized signals coming from microarray, next generation sequencing, mass spectrometry) in columns. Since tRanslatome doesn't provide any normalization, signals contained in the data matrix should be normalized before, unless the DEGs selection method doesn't

provide also a normalization step, as in the case of edgeR and DEseq. In our worked example microarray data were previously quantile normalized.

The function has the following input parameters:

- expr.matrix, a matrix that contains the normalized signal intensity data, each row representing a gene and each column representing a sample;
- cond.a, a vector of column names belonging to expression matrix. These columns contain the signal intensity data coming from the samples of the first expression level of the control condition (in our example: total RNA, undifferentiated cells);
- cond.b, a vector of column names belonging to expression matrix. These columns contain the signal intensity data coming from the samples of the first expression level of the treatment condition (in our example: total RNA, differentiated cells);
- cond.c, a vector of column names belonging to expression matrix. These columns contain the signal intensity data coming from the samples of the second expression level of the control condition (in our example: polysomal RNA, undifferentiated cells);
- cond.d, a vector of column names belonging to expression matrix. These columns contain the signal intensity data coming from the samples of the second expression level of the treatment condition (in our example: polysomal RNA, differentiated cells);
- label.level, character vector specifying the names given to the two levels. By default levels are named "1st level" and "2nd level", but the user can specify others: in our example the two levels are named "transcriptome" and "translatome";
- label.condition, character vector specifying the names given to the two conditions. By default, these values are "control" and "treated", but user can specify others: in our example the two levels are named "undifferentiated" and "differentiated";
- data.type, character vector specifying the type of data represented by `expr.matrix`. By default it is set to `array`, the other accepted value is `ngs`;

Once the object is initialized one can call the function for the identification of DEGs, called `computeDEGs()`. `computeDEGs()` takes as input a label specifying the method that we want to employ in order to detect DEGs and returns a table containing for each gene the associated fold changes, statistical significances and classification according to their expression behaviour in the two levels. The function has the following input parameters:

- object, an object of class `TranslatomeDataset` containing the data needed for DEGs identification;
- method, a label that specifies the statistical method for DEGs detection. It can have one the following values: `limma` [?], `t-test` [?], `RP` [?], `TE` [?], `SAM` [?], `ANOTA` [?], `DESeq` [?], `edgeR` [?] and `none`;

- significance.threshold, a threshold on the statistical significance below which the genes are considered as differentially expressed, the default is set to 0.05;
- FC.threshold, additional threshold on the absolute log2 fold change, above which the genes are considered as differentially expressed, the default is set to 0;
- log.transformed, a boolean variable specifying whether the data have been previously log2 transformed. By default it is set to FALSE;
- mult.cor, a boolean variable specifying whether the significance threshold is applied on the multiple test corrected or on the original p-values obtained from the DEGs detection method. By default it is set to TRUE.

The function `computeDEGs()` generates an object of class `DEGs`, containing the result of the differential expression analysis at the two expression levels. This object is returned and also stored in the DEGs slot of the `TranslatomeDataset` object. DEGs can then be later retrieved with the accessor `getDEGs` function.

One way to visualize the results obtained by the `computeDEGs()` function is to use the `Scatterplot()` method on the object of class `DEGs` generated by the `computeDEGs()` function, generating a plot in logarithmic scale where each gene is a dot whose position is uniquely determined by its fold change (FC) at the first expression level, represented on the x-axis, and the FC at the second expression level, represented on the y-axis.

Optional input parameters of the `Scatterplot()` method are:

- outputformat, a character string specifying if the plot is saved in jpeg (`jpeg`), postscript (`postscript`), pdf (`pdf`) format, or it is simply displayed on the screen(`on screen`). By default this value is `on screen`.
- track, a character vector of gene names that will be explicitly highlighted in the scatterplot, if they match any gene contained in the object of class `DEGs`. By default this vector is empty.

Figure **??** shows the graphical output of this method applied to the example object of class `DEGs`, where the two expression levels are names "transcriptome" and "translatome". We adopt a color code to label different classes of DEGs: blue for genes differentially expressed only at the first level, yellow for genes differentially expressed only at the second level, green for genes changing homodirectionally at both levels, red for the genes changing antidirectionally at the two levels. The Spearman's Correlation Coefficient between the fold changes of all the genes and between the fold changes of all the DEGs is also displayed.

Another way to visualize the results obtained from the `computeDEGs()` function is to use the `Histogram()` method, showing in a plot the number of the genes differentially expressed (up-regulated or down-regulated) at each expression level.

Optional input parameters of `Histogram()` are:

- outputformat, a character string specifying if the plot is saved in jpeg (`jpeg`), postscript (`postscript`), pdf (`pdf`) format, or it is simply displayed on the screen(`on screen`). By default this value is `on screen`.

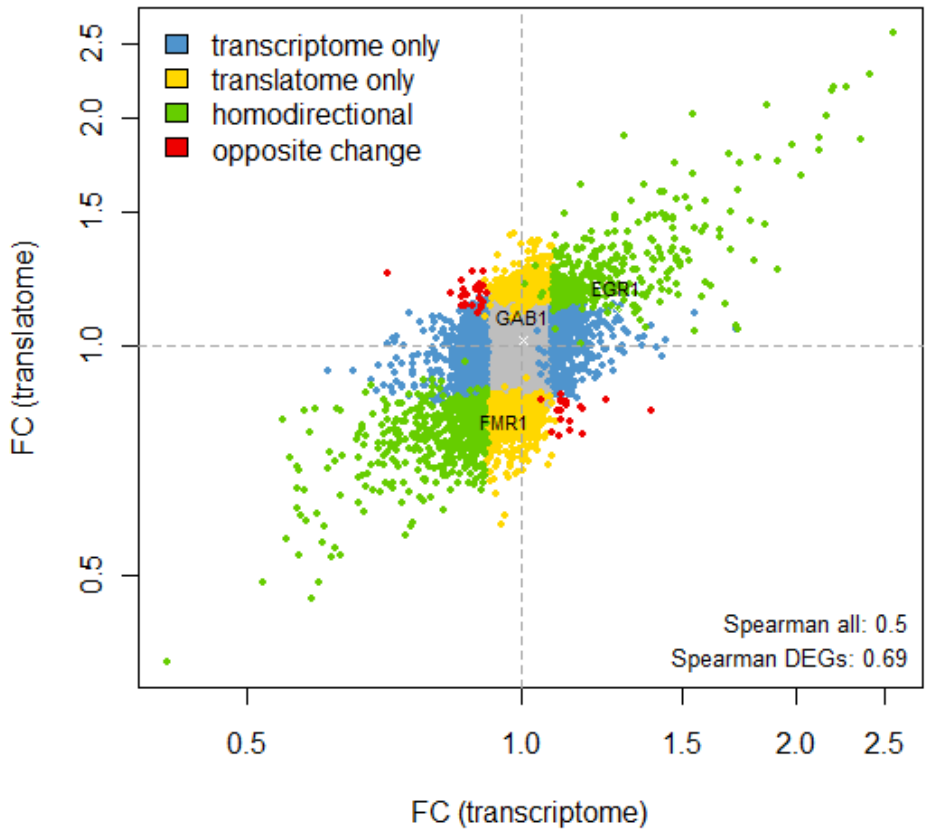

Figure 1: tRanslatome provides the method `Scatterplot()`, drawing a scatterplot in which each gene is mapped according to its fold change at the first level (represented on the x-axis) and the fold change at the second level (y-axis). The track parameter was set to `c("GAB1","FMR1","EGR1")`. We adopt a color code to label different classes of DEGs: blue for genes differentially expressed only at the first level; yellow for genes differentially expressed only at the second level, green for genes changing homodirectionally at both levels, red for the genes changing antidirectionally at the two levels.

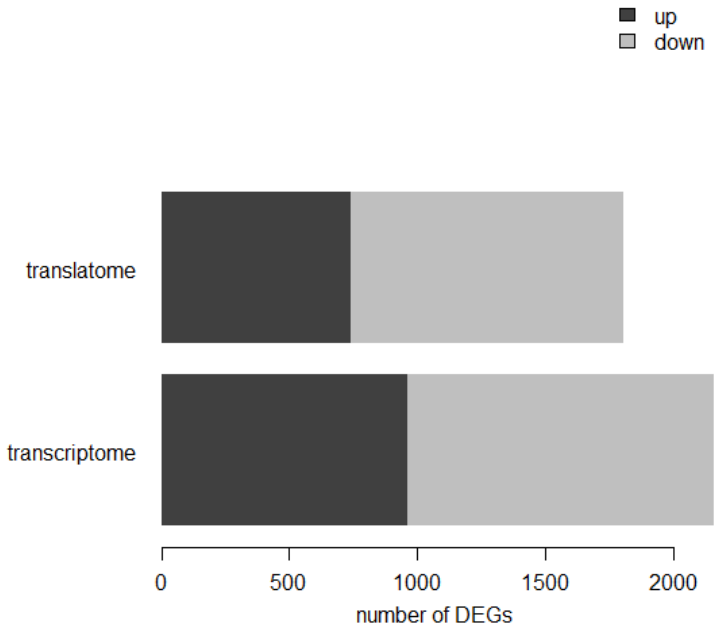

Figure 2: Summary histrogram of the identified differentially expressed genes (DEGs). It shows the distribution of up-regulated (in black) and down-regulated (in gray) genes at the two expression levels, transcriptome and translatome in our example.

- plottype, a label specifying whether the histogram should be just a brief "summary" of the data analysis, showing the number of genes up and down regulated in the first and second level (Figure **??**), or it should be a "detailed" histogram presenting the distribution of all the possible classes of DEGs, according to their behaviour in the two expression levels (Figure **??**).

4 Quality control

The MA-plots show in logarithmic scale the relationship between the average log2 signal intensity (A) and the log2 fold change (M) for each gene. The general assumption of genome-wide profiles is that most of the genes don't show any change in their expression. Therefore the majority of the genes should be located around 0 on the y-axis of the MA-plot. If this is not the case and there isn't a biological justification, an alternative normalization method should be applied [**?**]. tRanslatome enables the generation of MA-plots within the `MAplot()` method, which can be applied to object of class `DEGs` (Figure **??**).

Optional input parameters of `MAplot()` are:

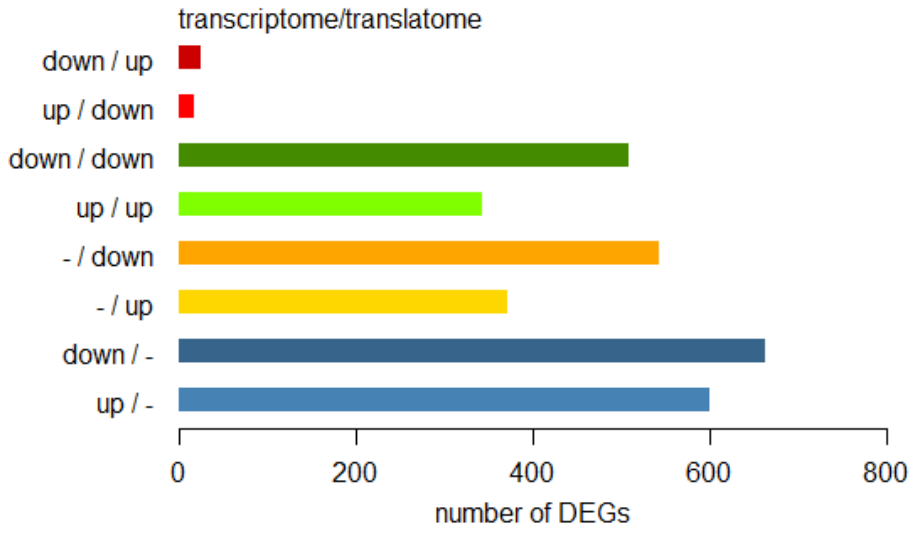

Figure 3: Detailed histogram of the identified differentially expressed genes (DEGs). It shows the number of genes up or down regulated only at the first expression level ("up/-" or "down/-", in blue tones), only at the second expression level ("-/up" or "-/down", yellow tones), at both expression levels with the same trend ("up/up" or "down/down", in green tones), at both expression levels with opposite trends ("up/down" or "down/up", in red tones).

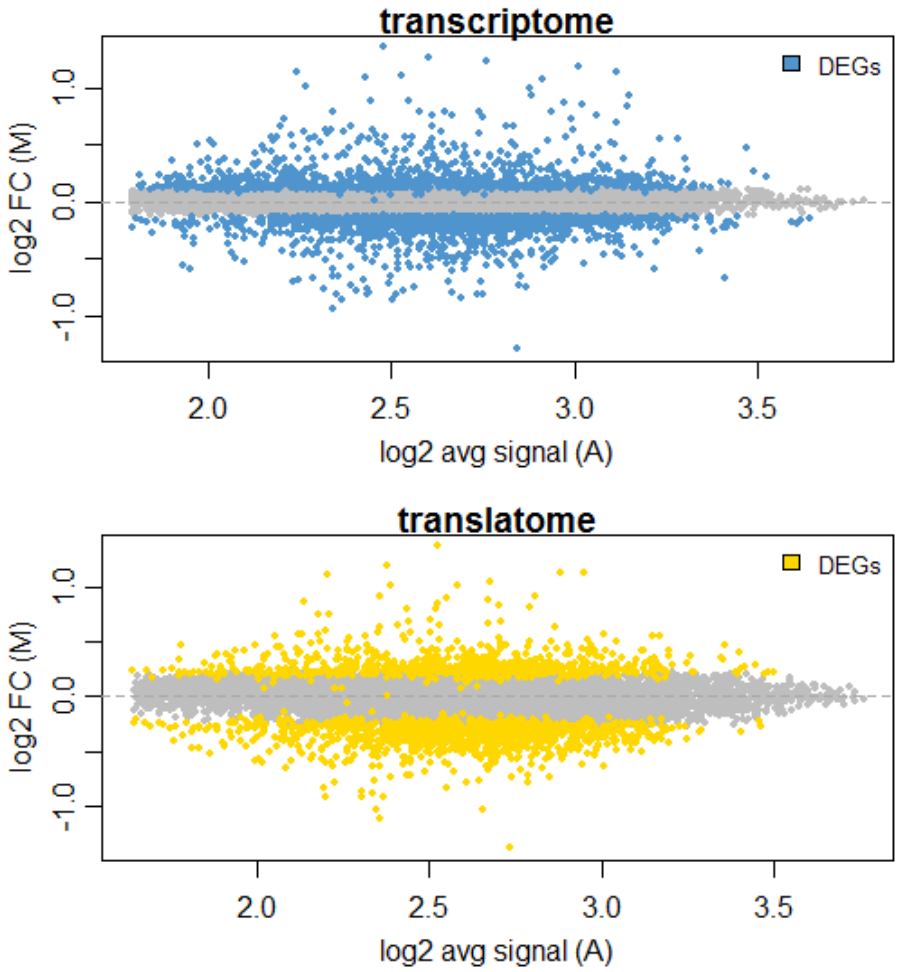

Figure 4: MA plots for the first expression level (upper panel) and the second expression level (lower panel).

- outputformat, a character string specifying if the plot is saved in jpeg (`jpeg`), postscript (`postscript`), pdf (`pdf`) format, or it is simply displayed on the screen(`on screen`). By default this value is `on screen`.

- track, a character vector of gene names that will be explicitly highlighted in the scatterplot, if they match any gene contained in the object of class `DEGs`. By default this vector is empty.

The upper panel in Figure **??** refers to the first expression level (transcriptome in our example), whereas the lower panel refers to the second expression level (translatome in our example). DEGs at each level are color labeled.

In tRanslatome there is also a method to visualize the relationship between the FC of the selected DEGs with respect to the standard deviation of their signals. The `SDplot()`

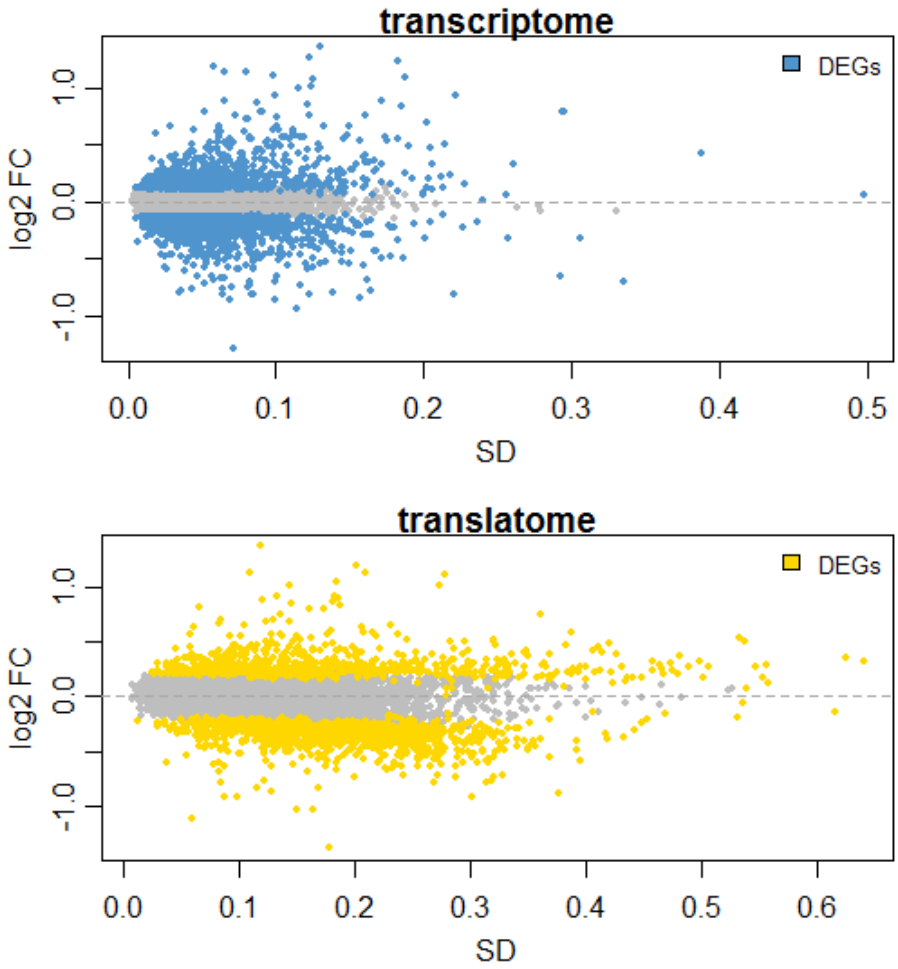

Figure 5: Standard deviation against log2 fold change of the genes. DEGs at the first and at the second expression level are labeled respectively in blue and yellow.

method plots for each of the two expression levels the standard deviation of the genes against their logarithmic fold change.

The set of input parameters is exactly like in `MAplot()`. The upper panel in Figure **??** refers to the first expression level (transcriptome in our example), whereas the lower panel refers to the second expression level (translatome in our example).

Alternatively, the relationship between the log2 fold change and the coefficient of variation (CV) of each gene can be visualized with the `CVplot()` method (See Figure **??**).

5 GO Enrichment

The Gene Ontology (GO) project provides a standardized controlled vocabulary to describe gene product attributes in all organisms [**?**]. GO consists of three hierarchically structured

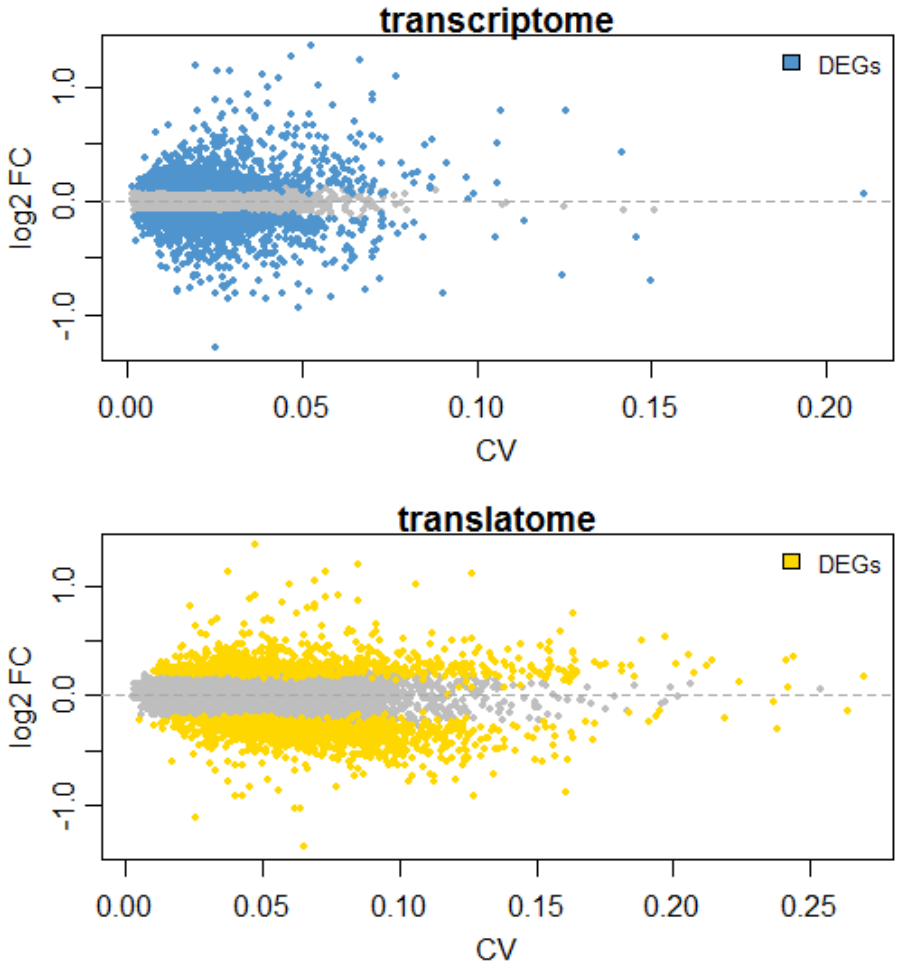

Figure 6: Coefficient of variation against log2 fold change of the genes. DEGs at the first and at the second expression level are labeled respectively in blue and yellow.

vocabularies (ontologies) that describe gene products in terms of their associated biological processes (BP), cellular components (CC) and molecular functions (MF). One of the most frequent applications of the GO to the results of high-throughput experiments is the enrichment analysis - the identification of GO terms that are significantly over-represented in a given set of differentially expressed genes [?] [?]. Enrichment analysis describes the functional characteristics of the given set of DEGs, suggesting the occurrence of possible mechanisms of regulation associated to the condition under examination [?]. tRanslatome offers different methods to perform enrichment analysis at both expression levels by detecting the overrepresented GO terms in the lists of DEGs returned from the `getDEGs()` function.

`GOEnrichment()` is a function in tRanslatome which identifies GO terms significantly enriched in the two lists of DEGs (corresponding to the two expression levels under analysis). The enrichment analysis can be performed on the whole set of GO ontologies, or restricted to one single ontology (either molecular function, cellular component or biological process). Moreover, the enrichment analysis can be performed on different classes of DEGs: only up-regulated genes, only down-regulated genes or both independently from the direction of their changes.

The function `GOEnrichment()` has the following input parameters:

- objectan object of class `DEGs`
- ontologya character string specifying the GO ontology of interest: `CC` for Cellular Component, `BP` for Biological Process, `MF` for Molecular Function or `all` for all the three ontologies. The default is set to `all`.
- classOfDEGsa character string specifying the class of genes for which we want to detect enriched GO terms: `up` for considering only up-regulated genes, `down` for considering only down-regulated genes, `both` for considering all DEGs, independently from the direction of their changes. The default is set to `both`.
- test.methoda character string specifying the statistical method to calculate the enrichment. By default it is set to `classic` (enrichment is measured with the classic Fisher exact test), but it can also be set to `elim`, `weight`, `weight01` or `parentchild`. All these methods are implemented in the `topGO` Bioconductor package
- test.thresholda numeric value specifying the significance threshold upon which the GO terms are considered significantly over-represented. By default it is se to `0.05`.
- mult.cora boolean variable specifying whether the significance threshold is applied to the multiple test corrected or to the original p-values obtained from the selected enrichment method. By default it is set to `TRUE`.

The output of the function `GOEnrichment()` is an object of class `GOsets`, containing the results of the enrichment analysis.

The method `Radar()` can be applied to the object of class `GOsets` in order to display the top enriched GO terms for the first and second expression level in a radar plot display (Figure ??).

The `Radar()` method has the following mandatory input parameters:

- object, an object of class `GOsets`.
- ontology, a label selecting the ontology of interest (either `CC`, `BP` or `MF`).

  Optional input parameters are:

- outputformat, a character string specifying if the plot is saved in jpeg (`jpeg`), postscript (`postscript`), pdf (`pdf`) format, or it is simply displayed on the screen(`on screen`). By default this value is `on screen`.
- n.nodes.1stlevel, a numeric value specifying the number of top enriched GO terms, from the first level, that will be represented on the plot. By default the value is set to `5`.
- n.nodes.2ndlevel, a numeric value specifying the number of top enriched GO terms, from the second level, that will be represented on the plot. By default the value is set to `5`.
- mult.cor, a boolean variable specifying whether the displayed significance values are multiple test corrected or the original p-values obtained from the selected enrichment method. By default it is set to `TRUE`.

The method `Heatmap()`, applied to an object of class `GOsets`, displays the top enriched GO terms for the first and second expression level in form of a heatmap (Figure **??**).

The `Heatmap()` method has the same input parameter of the `Radar()` method.

6 GO Comparison

The function `GOComparison()` takes as input an object of class `GOsets`, containing the results of a GO enrichment analysis applied to both expression levels, and returns as output an object of class `GOsims`, containing a variety of comparisons among the enriched GO terms. These comparisons include the calculation of semantic similarity scores between terms differentially enriched at the two levels, using the Wang method [**?**].

This function has only one input parameter:

- object, an object of class `GOsets`.

The output is a an object of class `GOsims`, containing an identity comparison and a semantic similarity comparison between the GO terms enriched at the two expression levels under analysis.

The method `IdentityPlot()`, applied to an object of class `GOsims`, displays in a barplot, for each GO ontology, the number of GO terms showing enrichment in both expression levels or only in one expression level. (Figure **??**).

The method `SimilarityPlot()`, applied to an object of class `GOsims`, displays in a barplot, for each GO ontology, the average semantic similarity value between GO terms showing enrichment in the first or in the second expression level. (Figure **??**).

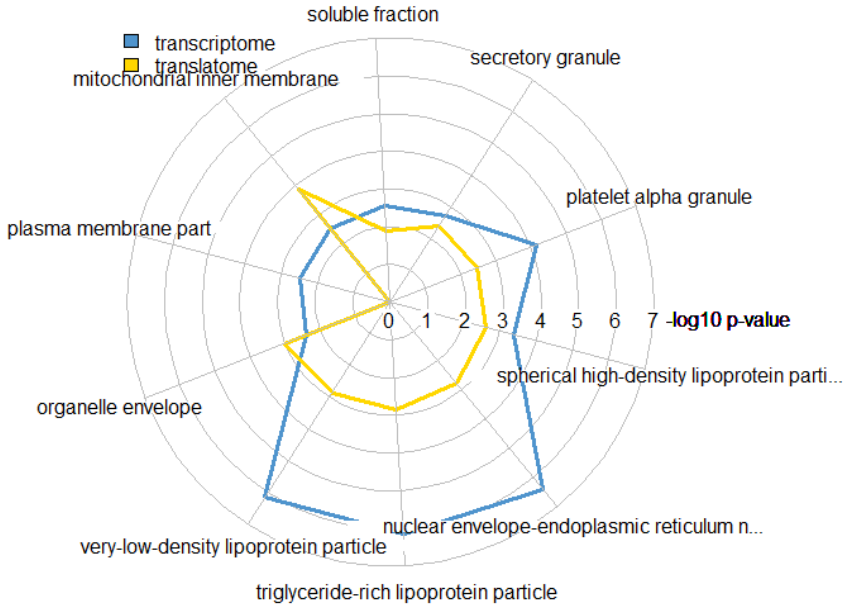

Figure 7: Top enriched GO terms are displayed in form of a radar plot. Enrichment p-values values for the first expression level are labeled in blue, whereas enrichment p-values for the second expression level are labeled in yellow.

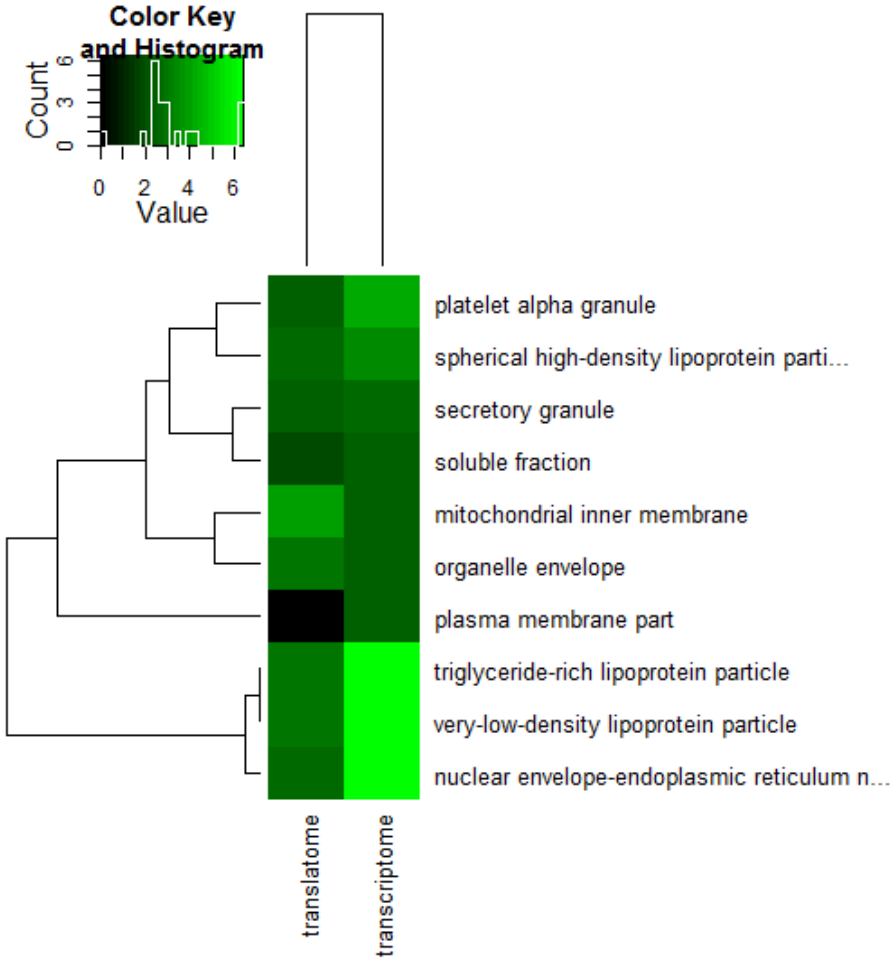

Figure 8: Heatmap of the top enriched GO terms for each expression level, transcriptome and translatome in our example. The color scale is based on the -log10 of the enrichment p-value.

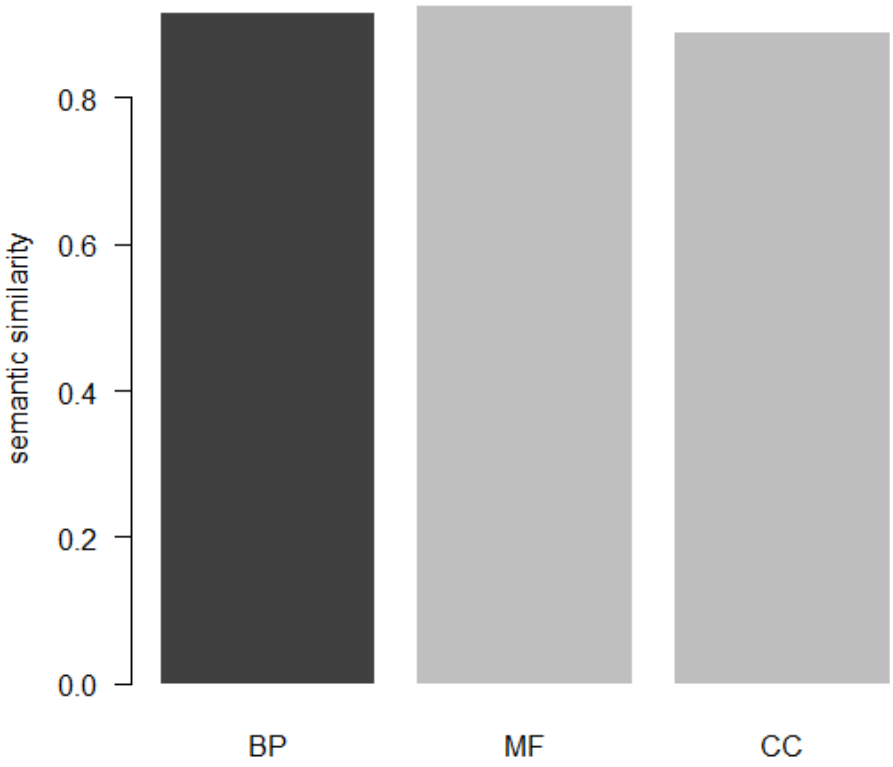

Figure 9: Barplot of the number of GO terms showing enrichment in both expression levels, transcriptome and translatome in our example, or only in one expression level.

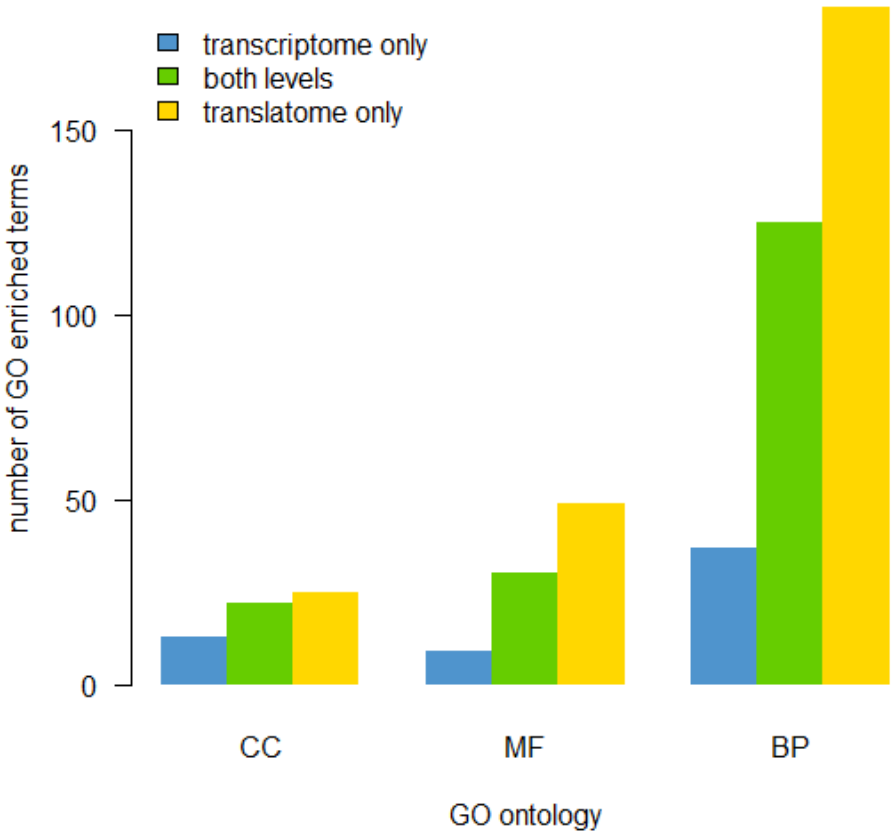

Figure 10: Barplot of the average semantic similarity value between GO terms showing enrichment in the first or in the second expression level, transcriptome and translatome in our example. Semantic similarity values are calculated with the Wang method [**?**]

.

7 Regulatory Enrichment

RegulatoryEnrichment is a function which, given as input an object of class `DEGs`, identifies overrepresented post-transcriptional regulators (RNA-binding proteins, microRNA, etc) possibly controlling differentially expressed genes. The analysis is applied to a dataset of experimentally determined post-transcriptional interactions extracted from the AURA database(http://aura.science.unitn.it). However, the user can specify a custom dataset onto which the analysis can be performed (see arguments for details). Moreover, the function can identify enriched regulators for separate classes of genes of interest: only up-regulated genes, only down-regulated genes or both of them together. Moreover, the function can identify enriched regulators for separate classes of genes of interest: only up-regulated genes, only down-regulated genes or both of them together. The method works by exploiting two lists: one containing all genes regulated by each of the post-transcriptional regulators, and the other containing the number of regulated and non-regulated genes for each of these post-transcriptional regulators in the backgroung gene set (usually the whole genome). By means of these two lists it is possible to compute a Fisher enrichment p-value indicating wether a significant group of genes in the DEGs list is likely to be regulated by one or more of these post-transcriptional regulators. The output of the function is an object of class `EnrichedSets`, containing the results of the enrichment analysis.

The method `Radar()` and the method `Heatmap()` can be applied also to objects of class `EnrichedSets` in order to display the top enriched regulatory elements for the first and second expression level in a radar plot or in a heatmap display.

References

[1] Parent R, Kolippakkam D, Booth G, Beretta L. Mammalian target of rapamycin activation impairs hepatocytic differentiation and targets genes moderating lipid homeostasis and hepatocellular growth. *Cancer Res.*, 2007, 67(9):4337-45.

[2] Smyth GK. Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Stat Appl Genet Mol Biol.*, 2004, 3:Article3.

[3] Tian L, Greenberg SA, Kong SW, Altschuler J, Kohane IS, Park PJ. Discovering statistically significant pathways in expression profiing studies. *Proc Natl Acad Sci USA.*, 2005, 102(38):13544-9.

[4] Breitling R, Armengaud P, Amtmann A, Herzyk P. Rank products: a simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments *FEBS Lett.*, 2004, 573(1-3):83-92.

[5] Courtes FC et al. (2013) Translatome analysis of CHO cells identify key growth genes. *Journal of Biotechnology*, 167, 215-24.

[6] Tusher VG, Tibshirani R, Chu G. Significance analysis of microarrays applied to the ionizing radiation response *Proc Natl Acad Sci USA.*, 2001, 98(9):5116-21.

[7] Larsson O, Sonenberg N, Nadon R. anota: Analysis of differential translation in genome-wide studies. *Bioinformatics*, 2011, 27(10):1440-1.

[8] Anders S, Huber W. Differential expression analysis for sequence count data. *Genome Biology*, 2010, 11(10):R106.

[9] Robinson MD, McCarthy DJ, Smyth GK. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 2010, 26(1):139-40.

[10] Dudoit S, Yang YH, Callow MJ, Speed TP. Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. *Stat. Sin.*, 2002, 12:1 111-139.

[11] Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, Harris MA, Hill DP, Issel-Tarver L, Kasarskis A, Lewis S, Matese JC, Richardson JE, Ringwald M, Rubin GM, Sherlock G. Gene ontology: tool for the unification of biology. *Nat. Genet.*, 2000, 25(1):25-9.

[12] Khatri P, Draghici S. Ontological analysis of gene expression data: current tools, limitations, and open problems. *Bioinformatics*, 2005, 21(18):3587-95.

[13] Alexa A, Rahnenfuhrer J, Lengauer T. Improved scoring of functional groups from gene expression data by decorrelating go graph structure. *Bioinformatics*, 2006, 22(13):1600-7.

[14] Yu G, Li F, Qin Y, Bo X, Wu Y, Wang S. GOSemSim: an R package for measuring semantic similarity among GO terms and gene products. *Bioinformatics*, 2010, 26(7):976-8