Practical session about normalization and differential analysis of RNAseq data using DiCoExpress

Marie-Laure Martin-Magniette, Christine Dillmann, Elodie Marchadier

November 2024

This file illustrates the steps of normalization and differential analysis of RNAseq data on a real dataset to compare different genotypes.

The analysis will be performed by using DiCoExpress. DiCoExpress is an R workspace to allow users without advanced statistical knowledge and programming skills to perform a full RNAseq analysis from quality controls to co-expression analysis through differential analysis based on contrasts inside generalized linear models. Hence, with DiCoExpress, the user can focus on the statistical modeling of gene expression according to the experimental design and on the interpretation of the results of such analysis in biological terms. Today, we focus on the differential analysis.

The biological experiment was performed on Podospora anserina in Ronald de Vries lab in Utrecht University (Netherland). P. anserina is a filamentous ascomycete fungus that colonized herbivore dung. Thanks to its enzymatic machinery, P. anserina is able to degrade biomass very efficiently. In the experiment you will study, transcriptome response of P. anserina has been characterized in the presence of two different feedstocks : soybean hulls (SH) and corn stover (CS). After 4, 24, and 48 h of incubation with the feedstock, the mycelium was harvested and experiments were performed in triplicate.

The data are composed of files included in the directory `Data`:

- `Podospora_anserina_COUNTS.csv` contains the raw counts of the experiment (19 columns: the first one contains the gene names, the others correspond to 18 different samples);
- `Podospora_anserina_TARGET.csv` contains information about the sample and the experimental design.

In the rest of this file, we:

1/ Present the structure of DicoExpress directories

2/ Install the R packages from BioClite

3/ Import and describe the data

4/ Perform a standard normalization (TMM) and explore its effects

5/ Perform a differential analysis using a glm model

6/ Compare lists of differentially expressed genes

7/ Explore the functions of differentially expressed genes

1 Data organization

The project directory is named `dicoexpress_NO` and must contain the following :

- `TP_dicoexpress.html` : this file.

- `Documentation` is a directory that contains the official dicoexpress tutorial, the documentation file, and a file describing R architecture.
- `Data` is a directory that contains the data files
- `Template_scripts` is a directory that contains the R scripts for the project. You may change lines of codes in those scripts.
- `Results` is a directory that will contain the output files and the plots.
- `Sources` is a directory that contains the source scripts. Indeed, `DiCoExpress` is not a R package, but rather a collection of R functions

2 Installing R packages

One of the Rscript that can be found in the `Sources` directory is the script named `Install.Packages.R` that contains the following instructions to install a serie of R packages :

```
## install CRAN packages
packages <- c("ggplot2","FactoMineR","gplots","reshape2","ggpubr","plyr","dplyr","RColorBrewer","data.ta
if (length(setdiff(packages, rownames(installed.packages()))) > 0) {
  install.packages(setdiff(packages, rownames(installed.packages())))
}
print(sapply(c(packages), require, character.only=T))
```

```
##      ggplot2   FactoMineR       gplots     reshape2       ggpubr         plyr
##         TRUE         TRUE         TRUE         TRUE         TRUE         TRUE
##        dplyr RColorBrewer   data.table        coseq
##         TRUE         TRUE         TRUE         TRUE
```

```
## install Bioconductor packages
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
#BiocManager::install(version = "3.14")
BiocManager::install(c("edgeR"))
```

In addition, the package `edgeR` must be installed from BioClite.

We have also to specify the working directory and where are the data and where the results must be saved. To do so, use the File menu in RStudio to choose the `Template_scripts` directory. Then, click o the *More* button and choose *Set as working directory.* The file `Load_Functions.R` contains the instructions to load all the libraries that are required by `DiCoExpress`.

```
## Call the edgeR library
library("edgeR")

source("../Sources/Load_Functions.R")
Load_Functions()

Working_Directory <- ".."
Data_Directory <- paste0(Working_Directory,"/Data")
Results_Directory <- paste0(Working_Directory,"/Results")
```

Information about the system (including package and **R** versions) are provided at the end of this file, in Section “Session information”.

3 Data importation

The data used in this practical session correspond to 18 samples of RNAseq obtained from a strain *Podospora anserina* grown on 2 conditions : soybean hulls (SH) and corn stover (CS). After 4, 24, and 48 h of incubation with the feedstock, RNAseq was extracted from mycelium and experiments were performed in triplicate (batches) leading to 18 samples (2 cond * 3 time points * 3 replicates)

Data can be loaded by using the function `Load_Data_Files` of DiCoExpress. We need to specify the name of the project, a filter to only keep the control condition and also the separator in the files.

```
Project_Name <- "Podospora_anserina"
Filter=NULL
Sep=";"

Data_Files <- Load_Data_Files(Data_Directory, Project_Name, Filter, Sep)
```

```
## Samples are not organized or named in the same manner in the expression file and the target file. The
```

```
Project_Name <- Data_Files$Project_Name
Target <- Data_Files$Target
Raw_Counts <- Data_Files$Raw_Counts
Annotation <- Data_Files$Annotation
Reference_Enrichment<-Data_Files$Reference_Enrichment
```

A quick overview of the count table is obtained by

```
head(Raw_Counts)
```

```
##            SH_04h_R1a SH_04h_R1g SH_04h_R1l SH_24h_R1b SH_24h_R1i SH_24h_R1f
## Pa_0_10            92         55         65         53         76         95
## Pa_0_100          394        630        872        562        832        718
## Pa_0_1000          91        310        383        175        298        354
## Pa_0_1010          63        201        209        141        189        238
## Pa_0_1020          63         73         81         52        101        107
## Pa_0_1030         752       1835       1603        333        465        839
##            SH_48h_R1q SH_48h_R1r SH_48h_R1d CS_04h_R1m CS_04h_R1c CS_04h_R1o
## Pa_0_10            37         58         12       1743        668       1244
## Pa_0_100          589        508        288       6343       4541       4384
## Pa_0_1000         182        302         88        225        293        159
## Pa_0_1010         235        267        113        259        305        221
## Pa_0_1020         112         91         87        110         99         92
## Pa_0_1030         654       1000        373       1169       1021        901
##            CS_24h_R1j CS_24h_R1k CS_24h_R1p CS_48h_R1e CS_48h_R1h CS_48h_R1n
## Pa_0_10           350        403        264         93        116        213
## Pa_0_100          536        981       1030        501        603        412
## Pa_0_1000         246        524        455        109        144        156
## Pa_0_1010         182        379        363        182        257        237
## Pa_0_1020          83        114        123         31        100         90
## Pa_0_1030         449        885        692        258        511        440
```

We observe that the row names are gene names.
The dimensions of the raw count matrix are given with `dim`. Notice that the columns of the COUNT file have been arranged so that the replicates from the same condition are grouped together. This ensures a prettier output for the quality control steps.

The experimental design is described in the object `Target`:

```
print(Target)
```

```
##              Medium Treatment Replicate
## SH_04h_R1a     SH       04h       R1a
## SH_04h_R1g     SH       04h       R1g
## SH_04h_R1l     SH       04h       R1l
## SH_24h_R1b     SH       24h       R1b
## SH_24h_R1i     SH       24h       R1i
## SH_24h_R1f     SH       24h       R1f
## SH_48h_R1q     SH       48h       R1q
## SH_48h_R1r     SH       48h       R1r
## SH_48h_R1d     SH       48h       R1d
## CS_04h_R1m     CS       04h       R1m
## CS_04h_R1c     CS       04h       R1c
## CS_04h_R1o     CS       04h       R1o
## CS_24h_R1j     CS       24h       R1j
## CS_24h_R1k     CS       24h       R1k
## CS_24h_R1p     CS       24h       R1p
## CS_48h_R1e     CS       48h       R1e
## CS_48h_R1h     CS       48h       R1h
## CS_48h_R1n     CS       48h       R1n
```

Questions: *Give the number of genes and describe the experimental design by giving the name of the factors and the number of modalities.*

4 Quality control and normalization (Session 2.4)

A first exploratory analysis is performed to evaluate the data quality and whether normalization assumptions are verified. We expect to have similar library sizes. DiCoExpress has a function to perform a control quality. Run the function and open the pdf report in the repository`Results/TP_control/Quality_Control`.

```
Filter_Strategy="NbReplicates"

Color_Group=NULL

CPM_Cutoff=1

Normalization_Method="TMM"



Quality_Control(Data_Directory, Results_Directory, Project_Name, Target,
                Raw_Counts, Filter_Strategy, Color_Group, CPM_Cutoff,
                Normalization_Method)
```

```
## ################################################
## Filtering
## ################################################
##
## #### Description of Raw counts table ####
## Number of samples: 18
## Number of genes: 10803
##
## Number of genes discarded by the filtering: 1007
## Number of genes analyzed after filtering: 9796
##
## ################################################
##  Statistics on the normalization factors
## ################################################
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.6689  0.9072  1.0015  1.0129  1.1567  1.2227
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## pdf
##   2
```

Filtering methods

- "NbConditions" : number of librairies with more than CPM_Cutoff >= number of biological conditions
- "NbReplicates" : number of librairies with more than CPM_Cutoff >= min(number of replicates)
- "filterByExpr" : filterByExpr function of edgeR package

Normalization methods - "TMM" is the weighted trimmed mean of M-values (to the reference) proposed by Robinson and Oshlack (2010), where the weights are from the delta method on Binomial data. If refColumn is unspecified, the library whose upper quartile is closest to the mean upper quartile is used.

- "RLE" is the scaling factor method proposed by Anders and Huber (2010). We call it "relative log expression", as median library is calculated from the geometric mean of all columns and the median ratio of each sample to the median library is taken as the scale factor.

- "upperquartile" is the upper-quartile normalization method of Bullard et al (2010), in which the scale factors are calculated from the 75% quantile of the counts for each library, after removing genes which are zero in all libraries. This idea is generalized here to allow scaling by any quantile of the distributions.

- "none", then the normalization factors are set to 1.

Question: *Comment the quality control file (type of plots and what do you see about the library sizes, the sample repartition, the normalization impact)*

Question: *PCA reports are generated, what were used as variables and individuals to perform these PCA ?*

Question: *The replicate numbers are provided, are they useful in this experiment ?*

5 Differential analysis (Sessions 2.1, 2.2, 2.3 and 2.5)

The differential analysis is performed using a generalized linear model to decompose the gene expression. The first step is to specify the statistical model and the questions that can be adressed with this model. An advantage of DiCoExpress is to do this for you.

```r
Replicate=FALSE
Interaction=TRUE

Model <- GLM_Contrasts(Results_Directory, Project_Name,
                                Target, Replicate, Interaction)
```

```
GLM_Model <- Model$GLM_Model
Contrasts <- Model$Contrasts
```

Question: *Write the equation relating the log of the mean expression and the experimental design.*

Question: *Using the command `colnames(GLM_Model)`, explain the parameter vector*

A TARGET file allowing to take into account the interactions between factors can also be used. In this case, the conditions (CS and SH) and the time (4, 24 and 48h will be indicated in two different columns.

5.1 Changes through time

Now it is possible to evaluate some contrasts. We will focus here in the contrasts between the three times in the CS condition.

```
### Here, choose the contrasts you're interested in among
Contrasts$Contrasts
```

```
##  [1] "[CS-SH]"                          "[04h-24h]"
##  [3] "[04h-48h]"                        "[24h-48h]"
##  [5] "[04h_CS-04h_SH]"                  "[24h_CS-24h_SH]"
##  [7] "[48h_CS-48h_SH]"                  "[CS_04h-CS_24h]"
##  [9] "[CS_04h-CS_48h]"                  "[CS_24h-CS_48h]"
## [11] "[SH_04h-SH_24h]"                  "[SH_04h-SH_48h]"
## [13] "[SH_24h-SH_48h]"                  "[CS_04h-CS_24h]-[SH_04h-SH_24h]"
## [15] "[CS_04h-CS_48h]-[SH_04h-SH_48h]" "[CS_24h-CS_48h]-[SH_24h-SH_48h]"
```

```
### For example, if you want to compare changes through time in the CS condition
###, there are three contrasts : [CS_04h-CS_48h],[CS_24h-CS_48h],[CS_04h-CS_24h]
Index_Contrast= c(8,9,10)
### To get all possible contrasts :
#Index_Contrast=1:nrow(Contrasts)

Alpha_DiffAnalysis=0.05
NbGenes_Profiles=20
NbGenes_Clustering=50

res <- DiffAnalysis.edgeR(Data_Directory, Results_Directory, Project_Name,
                   Target, Raw_Counts, GLM_Model, Contrasts, Index_Contrast,
                   Filter_Strategy, Alpha_DiffAnalysis, NbGenes_Profiles,
                   NbGenes_Clustering,
                   CPM_Cutoff, Normalization_Method)
```

```
## ################################################
##  Results of the differential analysis
## ################################################
##
## Number of DEGs for each contrast:
##          Contrast Nb_DEG
## 1 [CS_04h-CS_24h]   4629
## 2 [CS_04h-CS_48h]   5422
## 3 [CS_24h-CS_48h]   1499
##
##          Contrast Expression Nb_DEG
## 1 [CS_04h-CS_24h]         Up   2220
## 2 [CS_04h-CS_24h]       Down   2409
## 3 [CS_04h-CS_48h]         Up   2627
## 4 [CS_04h-CS_48h]       Down   2795
## 5 [CS_24h-CS_48h]         Up    759
## 6 [CS_24h-CS_48h]       Down    740
## ################################################
```

All the results of the differential analysis is in the repository `Results/Podospora_anserina/DiffAnalysis`.

Question: *Describe the organization of the repository*

Before extracting the genes differentially expressed, it is important to look at the raw pvalue histogram. Its form has the expected form so we can identify the genes differentially expressed.

Question: *What is the file name where you have all the pvalue histograms?*

Question: *Are you satisfied by their forms?*

Question: *What does the contrast [CS:24h-CS:48h] mean? Do you understand the writing of the associated contrast (file TP_control_Contrasts_Interest_Matrix.txt)?*

Question: *What does the contrast [CS:48h-CS:4h] mean? Do you understand the writing of the associated contrast (file TP_control_Contrasts_Interest_Matrix.txt)?*

5.2 To get functional annoation of DEG

```
for (cont in Contrasts$Contrasts[Index_Contrast])
{
  directory <- paste(Results_Directory,Project_Name,"DiffAnalysis",cont,sep="/")
  file <-paste(Project_Name,cont,"DEG.BH.txt",sep="_")
  dirfile <- paste(directory,file,sep="/")
  SignifDEG <- read.table(dirfile,header=TRUE,sep="\t")

  GeneAnnot <- read.table(paste(Data_Directory,"podo-go-terms-slim-2023.txt",sep="/"),sep="\t",header=
  colnames(GeneAnnot)[1]<-"genes"
  SignifDEGNames<-merge(SignifDEG,GeneAnnot,all.x=T, all.y=F)

  write.table(SignifDEGNames,paste(directory,paste(Project_Name,cont,"DEG.BH_GeneAnnot.txt",sep="_"),
}
```

5.3 Output : log(average gene counts) for differentially expressed genes

We may wish to analyze the expression profiles for the subset of DEG. To do this, there are two files of interest :

*`Podospora_anserina_Compare_table.txt` gives the result of mean comparison for each gene and each contrast.

```
tabcompdirectory <- paste(Results_Directory,Project_Name,"DiffAnalysis",sep="/")
tabcompfile <- paste(Project_Name,"_Compare_table.txt",sep="")
tabcompdirfile <- paste(tabcompdirectory,tabcompfile,sep="/")
tabcomp <- read.table(tabcompdirfile,header=TRUE)
head(tabcomp)
```

```
##     Gene_ID X.CS_04h.CS_24h. X.CS_04h.CS_48h. X.CS_24h.CS_48h.
## 1   Pa_0_10                1                1                0
## 2  Pa_0_100                1                1                0
## 3 Pa_0_1000                1                0                0
## 4 Pa_0_1010                0                1                1
## 5 Pa_0_1020                0                0                0
## 6 Pa_0_1030                1                0                0
```

*`Podospora_anserina_NormCounts_log2_Mean_SD.txt` gives the average count for each gene in each condition, as well as the corresponding standard deviations.

```
tabmeandirectory <- paste(Results_Directory,Project_Name,"DiffAnalysis",sep="/")
tabmeanfile <- paste(Project_Name,"_NormCounts_log2_Mean_SD.txt",sep="")
tabmeandirfile <- paste(tabmeandirectory,tabmeanfile,sep="/")
tabmean <- read.table(tabmeandirfile,header=TRUE,sep="\t")
rownames(tabmean)<-tabmean[,1]
tabmean<-tabmean[,-1]
head(tabmean)
```

```
##             SH_04h_Mean SH_24h_Mean SH_48h_Mean CS_04h_Mean CS_24h_Mean
## Pa_0_10        6.393224    6.498428    5.347625    9.640087    7.957517
## Pa_0_100       9.496483    9.742986    9.214395   11.787199    9.241246
## Pa_0_1000      8.058727    8.350873    7.833449    7.277563    8.174869
## Pa_0_1010      7.385451    7.838933    8.016445    7.521253    7.767189
## Pa_0_1020      6.450655    6.681501    7.022565    6.159285    6.303378
## Pa_0_1030     10.613152    9.287110    9.714109    9.496933    8.915880
##             CS_48h_Mean SH_04h_SD SH_24h_SD SH_48h_SD  CS_04h_SD  CS_24h_SD
## Pa_0_10        7.591365 0.9274493 0.1849305 0.6451734 0.65830330 0.58513557
## Pa_0_100       9.505220 0.2025302 0.1719907 0.1385841 0.18672601 0.14995837
## Pa_0_1000      7.621387 0.5443065 0.2868724 0.4370606 0.39585421 0.07530470
## Pa_0_1010      8.344646 0.4053764 0.1474949 0.1448351 0.21230741 0.12589513
## Pa_0_1020      6.585459 0.4059773 0.3401718 0.4421092 0.01092487 0.25752403
## Pa_0_1030      9.138629 0.1753616 0.4637320 0.2966651 0.07003551 0.07291659
##             CS_48h_SD
## Pa_0_10     0.6161731
## Pa_0_100    0.3624398
## Pa_0_1000   0.2435124
## Pa_0_1010   0.1463858
## Pa_0_1020   0.5883635
## Pa_0_1030   0.1751615
```

Question: *Could you create the Zsignif table containing the log2 mean of all DEG ?*

To help you, the list of DEG can be obtained from `tabcomp` using the apply function :

```
# list of DEG

 if(length(Index_Contrast)==1)
  {
    deglist <- tabcomp[,1][tabcomp[,-1]>0]
  }
  if(length(Index_Contrast)>1)
  {
    deglist <- tabcomp[,1][apply(tabcomp[,-1],1,sum)>0]
  }


# subset of tabmean
Zsignif <- tabmean[rownames(tabmean)%in%deglist,]
Zfile <- paste(tabmeandirectory,"Zsignif.csv",sep="/")
write.table(Zsignif,Zfile,sep=";",row.names=FALSE)
```

6 Functional enrichment of DEG (Session 2.6)

Among the lists of DEG, some functions may be over-represented and on the contrary, some may be under-represented. GO terms can be used to represent the functional annotation of genes and identify such biases among the lists of DEG.

The GO identifiers are provided in `Podospora_anserina_Enrichment.csv` file of the `Data` folder.

```
Title=NULL
Alpha_Enrichment=0.01
Enrichment(Results_Directory, Project_Name, Title, Reference_Enrichment,Alpha_Enrichment)
```

```
# to add the name of the GO terms in the generated
for (cont in Contrasts$Contrasts[Index_Contrast])
{
    directory <- paste(Results_Directory,Project_Name,"DiffAnalysis",cont,sep="/")
    file <-paste(Project_Name,cont,"Significant_Enrichments.txt",sep="_")
    dirfile <- paste(directory,file,sep="/")
    SignifEnrich <- read.table(dirfile,header=TRUE,sep="\t")

    GOnames <- read.table(paste(Data_Directory,"GO_terms.csv",sep="/"),sep=" ",header=T)
    colnames(GOnames)[2]<-"Term"
    SignifEnrichNames<-merge(SignifEnrich,GOnames,all.x=T, all.y=F)

    write.table(SignifEnrichNames,paste(directory,paste(Project_Name,cont,"Significant_EnrichmentsGOName
}
```

7 Compare lists of DEG using Venn diagrams

A Venn diagram is a graphical representation of the number of DEG shared by different contrasts.

Question: *Which contrasts are you looking at?*

In order to compare the DEG lists, you can use the function `Venn_IntersectUnion`

```
Title <- "CS - over time"
Groups=c("[CS_24h-CS_48h]","[CS_04h-CS_24h]","[CS_04h-CS_48h]")
### or alternatively :
#Groups = Contrasts$Contrasts[c(1,4:9)]

Operation="Union"

Venn_IntersectUnion(Data_Directory, Results_Directory, Project_Name, Title,
                    Groups, Operation)
```

```
## pdf
##   2
```

Question: *Where do you find the result files?*

Question: *Comment the Venn diagram*

8 Session information

It is important to save the parameters of the analysis. It is done with these lines

```r
Output = file(paste0(Results_Directory,"/",Project_Name,"/Parameter_Information.txt"), open="wt")
sink(Output)
sink(Output, type="message")
cat("Project_Name:",Project_Name,"\n")
if(!is.null(Filter))
    {
      cat("Filter", "\n")
      print(Filter)
    }

cat("Filter_Strategy:",Filter_Strategy,"\n")
cat("CPM_Cutoff:",CPM_Cutoff,"\n")
cat("Normalization_Method:",Normalization_Method,"\n")
cat("Replicate:",Replicate,"\n")
cat("Interaction:",Interaction,"\n")
sink(type="message")
sink()

writeLines(capture.output(sessionInfo()),
           paste0(Results_Directory,"/",Project_Name,"/SessionInfo.txt"))
```