Micro Array data Processing

Adrien Bessy

* Here is the pipeline to go from raw data that are measured intensities and locations for an array that has been hybridized (Cel files) to differentially expressed genes (final processed data). Once you have your processed data, you can use a tool to apply multiple filers on several variables.
* Raw data comes from two papers (Winter, 2011 and William, 2004) and from our own data (F. Parcy). Winter, 2011: The data are gene expression of 9-day-old 35S::LFY-GR dexamethasone-treated seedlings compared to 9-day-old WT dexamethasone-treated seedlings, 4 replicates for each condition. William, 2004: Gene expression in 9-day-old LFY-GR, 35S::LFY or Landsberg erecta seedlings treated with the steroid dexamethasone and/or the protein synthesis inhibitor cycloheximide, 1 replicate for each condition. Our data: SP1 vs Col and SP3 vs Col. In addition, gene expression from Wild type and mutant Arabidopsis plants grown in short days (9L:15D) for 30 days at 21°C, then shifted to long days (16L:8D). These last data are already normalized, so we have to skip the Normalization step.
* Data coming from Affymetrix Arabidopsis ATH1 Genome Array, so adapted methods were used for these kind of arrays (affy package). All the packages used for normalization and identification of differentially expressed genes come from [https://bioconductor.org](https://bioconductor.org/) that stores rather reliable packages.

1. Go to <https://www.ncbi.nlm.nih.gov/geo/> (a public functional genomics data repository)
2. Type GSE28062, then GSE911, then GSE576 in the Search bar.
3. Download CEL files corresponding to each one at the bottom of the pages (tick "Select All" and download). For GSE576, download the txt format, then select the relevant data inside.
4. Unpack them to obtain CEL files.
5. Put CEL files of GSE28062 in one directory, and CEL files of GSE911 in another one.
6. Concerning GSE911, three different comparisons will be done, so you have to put 35LFY and LER in one sub-directory, Dex and Mock in another one, and DexCyclo and Cyclo in another one. Concerning GSE576, I select the id probe, GSM8829, GSM8830, GSM8837, GSM8838 columns to paste in one new file, then id probe, GSM8831, GSM8832, GSM8839, GSM8840 columns to paste in one new file, and finally, id probe, GSM8833, GSM8834, GSM8841, GSM8842 columns to paste them in one new file, then id probe, GSM8827,GSM8828, GSM8835, GSM8836 columns to paste them in a new file. Don't forget to remove the last line that is "!series\_matrix\_table\_end".
7. Be in the directory where there are CEL files to analyze, then type within the R environment the following commands:
   1. Dependencies

## try http:// if https:// URLs are not supported  
source("https://bioconductor.org/biocLite.R")  
## browseVignettes("package\_name") for package information  
biocLite("affy") ## contains methods for Affymetrix arrays  
biocLite("gcrma") ## gcrma adjusts for background intensities in Affymetrix array data which include optical noise and non-specific binding.  
biocLite("RankProd") ## contains the Rank Product method that is a non-parametric method for identifying differentially expressed (up- or down- regulated) genes based on the estimated percentage of false predictions (pfp).  
library(affy)   
library(gcrma)  
library(RankProd)

* 1. If you use windows

The best is to use RStudio software <https://www.rstudio.com/>. Think to set your working directory where you put your CEL files (Session->Set Working Directory).

* 1. Normalization

## ReadAffy() reads all your cel files  
bm=ReadAffy()  
bm ## the body map displays the size of the array (feature number), number of samples and number of genes  
eset<-gcrma(bm)  
write.exprs(eset,file='gcRMA\_expression\_estimates.txt') ## In this file, you can see the order of your samples, which is useful for your data.cl variable

* 1. Identification of differentially expressed genes

For GSE96799:

data0<-read.table('GSM2544192.csv', header = TRUE, sep = "\t", quote = '"')

data1<-read.csv('tab\_sep\_LFY\_regulated\_genes.csv', header = TRUE, sep = "\t", quote = '"')

data00<-data0[grep("AT",data0$Locus.Identifier),]

data000<-data00[-grep("gb\_",data00$Locus.Identifier),]

new\_column <- gsub("\\..\*","",data000$Locus.Identifier)

data000$Locus.Identifier <- new\_column

data000 <- data000[order(data000$Locus.Identifier, data000$PValueLogRatio\_Rep\_1\_2h

, increasing=TRUE),]

data0000 <- data000[!duplicated(data000$Locus.Identifier),]

tables<-merge(data1, data0000, by=c('Locus.Identifier'),all=T)

write.table(tables,file='tab\_sep\_LFY\_regulated\_genes\_with\_GSE96799.csv',sep='\t',row.names=F)

For GSE576:

data0<-read.table('col\_3DAF\_vs\_lfy\_3DAF.txt', header = TRUE, sep = "\t", quote = '"')  
data <- data0[,-1]  
rownames(data) <- data0[,1]  
data.cl<-c(0,0,1,1)

proper from each analysis, not for GSE976:

## exprs() function extracts expression data  
data<-exprs(eset)  
## For data.cl variable, look at the 'gcRMA\_expression\_estimates.txt' file to observe the order of each sample. The control has to be zero.  
data.cl<-c(0,0,0,0,1,1,1,1) ## for GSE28062  
data.cl<-c(0,1,0,1) ## for GSE911  
data.cl<-c(1,1,1,0,0,0) ## for sp1/Col and sp3/Col

data.gnames<-rownames(data)  
RP.out<-RP(data,data.cl,gene.names=data.gnames,rand=123) ## we chose to set rand to 123 to make this pipeline reproducible.  
result\_0.1<-topGene(RP.out,num.gene=22810,gene.names=data.gnames) ## create a table with FC, pfp and P.value for all the genes (22810 genes here)  
data1<-result\_0.1$Table1  
data1<-data1[,-2] ## suppress the RP/Rsum column  
data1=cbind(rownames(data1),data1)

Proper from each analysis:

colnames(data1)[c(1,4,5)]=c('ath1\_probe','pfp\_LFYGR\_WT\_down','P.value\_LFYGR\_WT\_down')  
data2<-result\_0.1$Table2  
data2<-data2[,c(1,4,5)]  
data2=cbind(rownames(data2),data2)  
colnames(data2)[c(1,3,4)]=c('ath1\_probe','pfp\_LFYGR\_WT\_up','P.value\_LFYGR\_WT\_up')  
tables<-merge(data1, data2, by=c('ath1\_probe','gene.index'),all=T)  
colnames(tables)[c(2,3)]=c('probe\_target\_number','FC\_LFYGR\_WT\_GSE28062')  
write.table(tables,file='LFYgr\_vs\_WT\_GSE28062.csv',sep=';',row.names=F)

Repeat all these steps for each comparison (here, four comparisons), so quit R (with q() command) and go to the relevant directory.

1. When you did all your comparisons, quit R, put all your results in a directory.
2. Go into this directory
3. Then, write the following commands to merge your csv files.
   1. Merging of your csv files

multmerge = function(){  
filenames=list.files(full.names=TRUE)  
datalist = lapply(filenames, function(x){read.csv(file=x,, header = TRUE, sep = ";", quote = '"')})  
Reduce(function(x,y) {merge(x,y, by=c('ath1\_probe','probe\_target\_number'),all=T)}, datalist)  
}  
all<-multmerge()  
write.table(all,file="all.csv",sep=";",row.names=F)

Then, you get one csv file that contains ath1 probe, probe target number, fold change (FC), pfp and Pvalue for all your samples.

* 1. Short description and Locus.Identifier

1. Copy the ah1\_probe column and paste it in the entry text in <https://www.arabidopsis.org/tools/bulk/microarray/index.jsp>.
2. Select all, then save it in a csv format.
3. Replace the 'Array Element' header by 'ath1\_probe' header.
4. In the terminal, be in the same directory that your both files then type the following commands:

mydata1 = read.csv("all.csv", header = TRUE, sep = ";", quote = '"')  
mydata2 = read.csv("short\_description.csv", header = TRUE, sep = "\t", quote = '"')  
mydata2<-mydata2[,c(1,2,3)]  
myfulldata<-merge(mydata1, mydata2, by=c('ath1\_probe'),all=T)  
write.table(myfulldata,file="all\_and\_SD.csv",sep=";",row.names=F)

Before using csv operation tool, check, by using excel, if you have comma or scientific notation (E+) that is not compatible with csv operation tool. So you can replace scientific notation by standard notations, then comma by dot.

Here is a way to do multiple filters on several variables:

* 1. Use of LFY-regulated\_genes\_base:

1. Download the 'LFY-regulated\_genes\_base' here: <https://github.com/adrbessy/LFY-regulated_genes_base>.
2. Put your csv file in the 'data' directory of this base.
3. Type the following commands on your terminal or on RStudio:

install.packages('shiny')  
install.packages('DT')  
library(shiny)  
runApp('C:Users/.../csv\_operation\_tool')

1. Launch the app on your web browser
2. You can do multiple filtering, then saving your resulting table by using the easy copy/paste function or PDF function.
   1. Session information

It is very useful to record the session information when performing analyses.

sessionInfo()

## R version 3.3.0 (2016-05-03)  
## Platform: x86\_64-w64-mingw32/x64 (64-bit)  
## Running under: Windows 8.1 x64 (build 9600)  
##   
## locale:  
## [1] LC\_COLLATE=French\_France.1252 LC\_CTYPE=French\_France.1252   
## [3] LC\_MONETARY=French\_France.1252 LC\_NUMERIC=C   
## [5] LC\_TIME=French\_France.1252   
##   
## attached base packages:  
## [1] stats graphics grDevices utils datasets methods base   
##   
## loaded via a namespace (and not attached):  
## [1] magrittr\_1.5 assertthat\_0.1 formatR\_1.4 tools\_3.3.0   
## [5] htmltools\_0.3.5 yaml\_2.1.13 tibble\_1.2 Rcpp\_0.12.7   
## [9] stringi\_1.1.2 rmarkdown\_1.1 knitr\_1.14 stringr\_1.1.0   
## [13] digest\_0.6.9 evaluate\_0.10

devtools::session\_info()

## Session info --------------------------------------------------------------

## setting value   
## version R version 3.3.0 (2016-05-03)  
## system x86\_64, mingw32   
## ui RTerm   
## language (EN)   
## collate French\_France.1252   
## tz Europe/Berlin   
## date 2016-10-22

## Packages ------------------------------------------------------------------

## package \* version date source   
## assertthat 0.1 2013-12-06 CRAN (R 3.3.0)  
## devtools 1.11.1 2016-04-21 CRAN (R 3.3.0)  
## digest 0.6.9 2016-01-08 CRAN (R 3.3.0)  
## evaluate 0.10 2016-10-11 CRAN (R 3.3.1)  
## formatR 1.4 2016-05-09 CRAN (R 3.3.1)  
## htmltools 0.3.5 2016-03-21 CRAN (R 3.3.1)  
## knitr 1.14 2016-08-13 CRAN (R 3.3.0)  
## magrittr 1.5 2014-11-22 CRAN (R 3.3.1)  
## memoise 1.0.0 2016-01-29 CRAN (R 3.3.0)  
## Rcpp 0.12.7 2016-09-05 CRAN (R 3.3.1)  
## rmarkdown 1.1 2016-10-16 CRAN (R 3.3.1)  
## stringi 1.1.2 2016-10-01 CRAN (R 3.3.1)  
## stringr 1.1.0 2016-08-19 CRAN (R 3.3.1)  
## tibble 1.2 2016-08-26 CRAN (R 3.3.1)  
## withr 1.0.1 2016-02-04 CRAN (R 3.3.0)  
## yaml 2.1.13 2014-06-12 CRAN (R 3.3.1)