Gene expression of L. plantarum WCFS1

Ivar van den Akker, Herke Schuffel, Yuri Wit en Peter Cserei

11/24/2020

Contents

Samenvatting . . . 1
Inleiding . . . 1
Materiaal en Methode . . . 2
Library importeren . . . 2
Bestand inladen . . . 2
DGE list maken . . . 3
Data normaliseren . . . 3
Filteren op low counts . . . 3
Design matrix maken . . . 3
Despersie berekenen . . . 4
Fit data . . . 4
Determine fold changes . . . 4
Print top tags . . . 4
Filteren . . . 5
Resultaten . . . 5
Cluster genes using hierarchal clustering . . . 5
PCA plot . . . 6
Dispersie plot . . . 7
Conclusie . . . 8

Samenvatting

Inleiding

Doel: When L. plantarum is grown on a ribose-rich medium genes are upregulated that are required for metabolizing ribose.

Materiaal en Methode

Dit R-script analyseert de expressie van genen van L. plantarum varianten: WCFS1 en NC8. Deze twee bacterien zijn gegroeid op verschillende bodems namelijk glucose (glc) en ribose (rib).

Library importeren

Importeren van library's die nodig zijn voor het functioneren van onderstaande script.

```
library(limma)
library(edgeR)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.6.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(xlsx)
```

```
## Warning: package 'xlsx' was built under R version 3.6.2
```

Bestand inladen

Het eerste variabel slaat de locatie van het bestand op. Vanzelfsprekend wordt ook de naam van het nodige bestand in een variabel opgeslagen.

De read.delim functie wordt over het algemeen gebruikt om tekstbestanden in te lezen waarbij data georganiseerd is in een data matrix layout. Binnen deze functie zijn de bovenste twee variabelen aan elkaar te plakken met paste0.

Daarbij wordt het kolom dat zorgt voor het oplopende nummering van rijen omgezet naar de ID kolom. Dit is gedaan om te voorkomen dat er verwarring ontstaat. De nummering van R zou namelijk niet overeen komen met de ID's van de ingeladen data. Dit is van belang voor eventuele latere annotatie.

```
fDir <- "./"
fName <- "RNA-Seq-counts.txt"
cnts <- read.delim(paste0(fDir,fName), comment.char="#")
row.names(cnts) <- cnts[,"ID"]

f2Name <- "WCFS1_anno.txt"
annotation <- read.delim(paste0(fDir,f2Name), comment.char = "#")
row.names(annotation) <- annotation[,row_number(0)]
```

DGE list maken

EdgeR slaat data op in een list-based data object genaamd een DGEList. Dit type object is gemakkelijk te gebruiken binnen R aangezien een DGEList gemanipuleerd kan worden zoals ieder ander lijst in R. Daarbij maakt een DGElist het mogelijk om berekeningen en statistiek toe te passen op de data. In het script worden de counts in de kolommen 2 t/m 9 gegroepeerd met de acht aangemaakte labels.

```
label <- c("WCFS1.glc",  "WCFS1.glc",   "WCFS1.rib",    "WCFS1.rib",    "NC8.glc",  "NC8.glc",  "NC8.ri
group <- factor(label)
y <- DGEList(counts=cnts[,2:9],group=group)
```

Data normaliseren

Normalisatie wordt toegepast om biologische variatie uit de data te halen, zodat vervolgens de verschillende datasets met elkaar vergeleken kunnen worden. In R wordt dit gedaan door norm factors te berekenen. De TMM (trimmed mean of M-values) methode verwijdert eerst de laagste en hoogste waarden en bepaalt daarna het gemiddelde van de rest van de genen.

Hieronder zijn de norm factors weergegeven, zonder calcNormFactors toe te passen zou deze voor alle rijen 1 zijn. De data is genormaliseerd nadat het met de norm factors vermenigvuldigd is. Dit proces gaat automatisch in R na het veranderen van de norm factors.

```
y <- calcNormFactors(y, method="TMM" )
print(y$samples)
```

```
##                 group lib.size norm.factors
## WCFS1.glc.1 WCFS1.glc 10153710    0.9850714
## WCFS1.glc.2 WCFS1.glc 10615392    0.9830048
## WCFS1.rib.1 WCFS1.rib  8959060    0.9986438
## WCFS1.rib.2 WCFS1.rib  9340139    1.0375329
## NC8.glc.1     NC8.glc  9821662    0.9390775
## NC8.glc.2     NC8.glc 10261922    1.0155475
## NC8.rib.1     NC8.rib  9951954    0.9899344
## NC8.rib.2     NC8.rib  9957519    1.0557378
```

Filteren op low counts

Deze functie zorgt ervoor dat genen geselecteert worden die niet significant genoeg zijn voor de dataset. Aangezien de dataset waarmee gewerkt wordt al een keer is gefilterd, resulteert de functie “filter by expression” maar in een hele kleine hoeveelheid genen, namelijk 20 stuks.

```
keep <- filterByExpr(y)
y <- y[keep,]
```

Design matrix maken

Een design matrix geeft aan hoe de samples gegroepeerd zijn in de experimenten. In dit geval is te zien dat glucose 1 en 2 gekoppeld zijn aan glucose van het desbetreffende bacterie. Hetzelfde geldt voor ribose.

```
design <- model.matrix(~0+group, data=y$samples)
colnames(design) <- levels(y$samples$group)
print(design)
```

```
##             NC8.glc NC8.rib WCFS1.glc WCFS1.rib
## WCFS1.glc.1       0       0         1         0
## WCFS1.glc.2       0       0         1         0
## WCFS1.rib.1       0       0         0         1
## WCFS1.rib.2       0       0         0         1
## NC8.glc.1         1       0         0         0
## NC8.glc.2         1       0         0         0
## NC8.rib.1         0       1         0         0
## NC8.rib.2         0       1         0         0
## attr(,"assign")
## [1] 1 1 1 1
## attr(,"contrasts")
## attr(,"contrasts")$group
## [1] "contr.treatment"
```

Despersie berekenen

Er zijn twee verschillende manieren getest om de dispersie te berekenen.

```
y <- estimateDisp(y, design)

f <- estimateGLMCommonDisp(y,design)
f <- estimateGLMTrendedDisp(y,design, method="power")
f <- estimateGLMTagwiseDisp(y,design)
```

Fit data

De glmFit functie maakt een DGEGLM object aan waarin allerlei data wordt verwerkt en berekent daarbij onder andere de p-values en fold-changes.

```
fit <- glmFit(y,design)
```

Determine fold changes

```
WCFSglcrib <- makeContrasts(exp.r=WCFS1.glc-WCFS1.rib, levels=design)
WCFSglcribfit <- glmLRT(fit, contrast=WCFSglcrib)
NC8glcrib <- makeContrasts(exp.r=NC8.glc-NC8.rib, levels=design)
NC8glcribfit <- glmLRT(fit, contrast=NC8glcrib)
```

Print top tags

WCFS glucose vs WCFS ribose

```
WCFSglcribres<-topTags(WCFSglcribfit, n=nrow(WCFSglcribfit))
```

NC8 glucose vs NC8 ribose

```
NC8glcribres<-topTags(NC8glcribfit, n=nrow(NC8glcribfit))
```

Filteren

Om genen te selecteren die voor ons onderzoek van toepassing zijn hebben we cut of values ingesteld voor de p-value en de fold changes. Voor de p-value is er gekozen om deze onder de 0.05 te houden en de fold changes moeten hoger zijn dan 1 of lager zijn dan -1. WCFS glucose vs WCFS ribose

```
WCFSfinal <- filter(WCFSglcribres$table,  WCFSglcribres$table$FDR<0.005 &(WCFSglcribres$table$logFC>1|W
```

NC8 glucose vs NC8 ribose

```
NC8final <- filter(NC8glcribres$table,  NC8glcribres$table$FDR<0.005 &(NC8glcribres$table$logFC>1|NC8gl
```

Annotatie toevoegen aan de genen die zijn gefiltered en van hoog naar laag sorteren

```
WCFS1_sig <- cbind(WCFSfinal, annotation[rownames(WCFSfinal),])
NC8_sig <- cbind(NC8final, annotation[rownames(NC8final),])
WCFS1_sorted <- WCFS1_sig[order(WCFS1_sig[, "logFC"], decreasing = TRUE, na.last = FALSE), ,drop=FALSE]
NC8_sorted <- NC8_sig[order(NC8_sig[, "logFC"], decreasing = TRUE, na.last = FALSE), ,drop=FALSE]
```

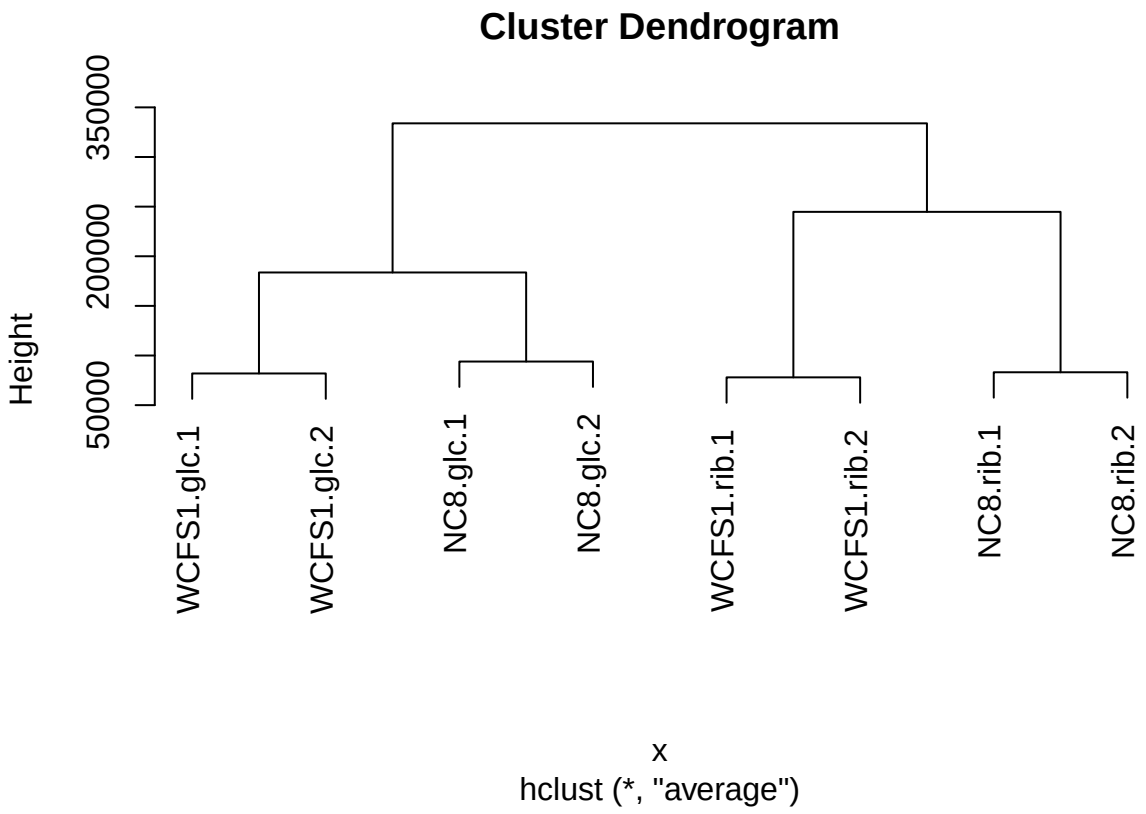
Excel file maken met de data en annotaties

```
#write.xlsx(WCFS1_sorted, file = "Merged_Genes.xlsx", sheetName = "WCFS1_data",
#  col.names = TRUE, row.names = TRUE, append = FALSE)
#write.xlsx(NC8_sorted, file = "Merged_Genes.xlsx", sheetName = "NC8_data",
#  col.names = TRUE, row.names = TRUE, append = TRUE)
```

Resultaten

Cluster genes using hierarchal clustering

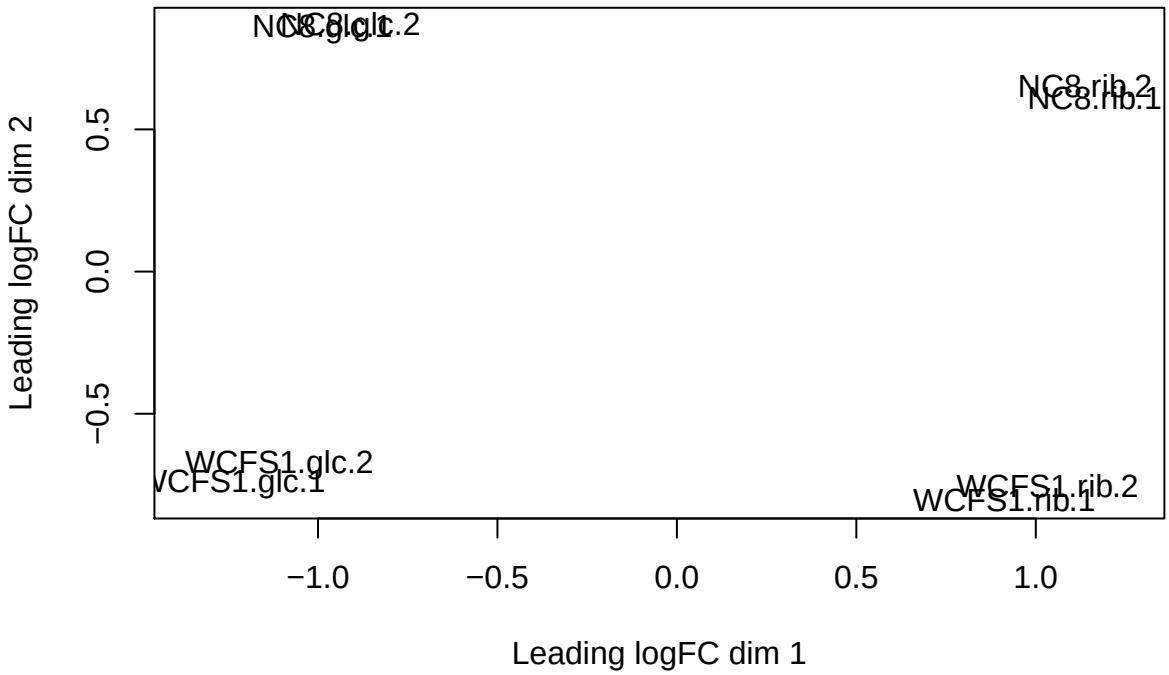
```
x <- t(y$counts)
x <- dist(x, method = "euclidean")
x <- hclust(x, method = "average")
plot(x)
```



PCA plot

De PCA plot (principal component analysis) comprimeert de datasets door de data zo veel mogelijk om te zetten naar de 1ste dimensies. Het verminderen van dimensies van de variatie in de samples maakt het mogelijk om de ze gemakkelijk te visualiseren en met elkaar te vergelijken. De X en Y as geven de fold-change in de log2 schaal weer. De X-as verklaart de hoogste variatie, in dit geval laat het zien dat het verschil in fold-change tussen WCFS1.glucose en WCFS1.ribose zeer groot is. Daarbij valt ook op dat de groepen die in de design matrix gemaakt zijn weinig variatie met elkaar hebben.

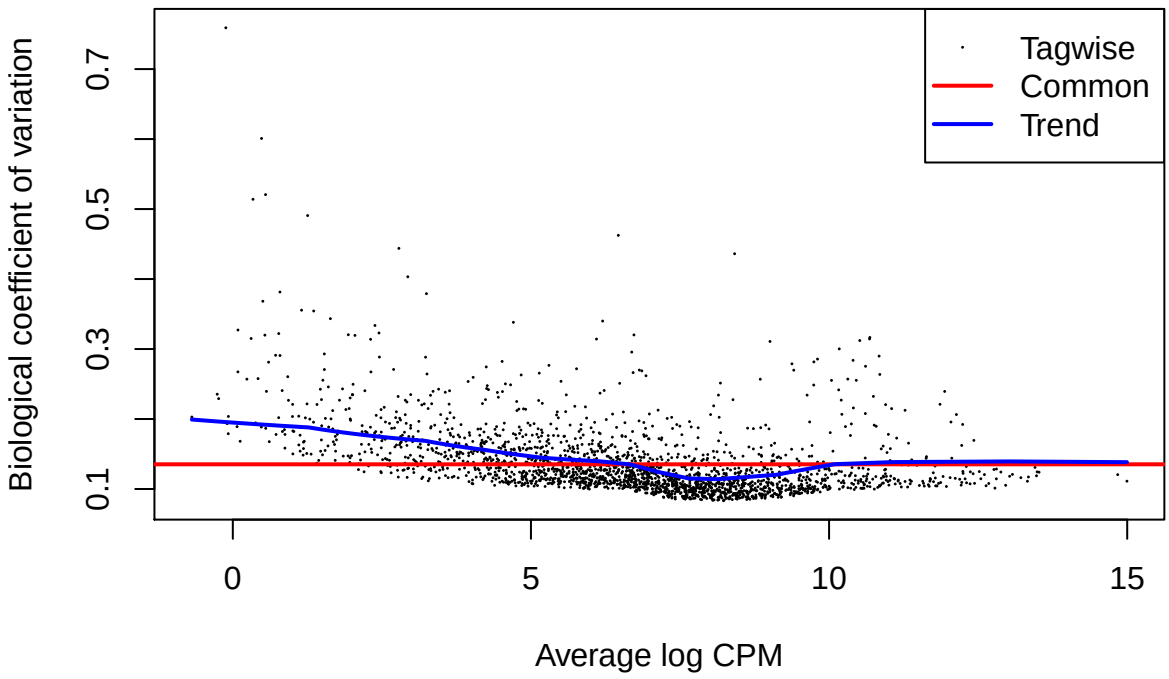
```
plotMDS(y)
```



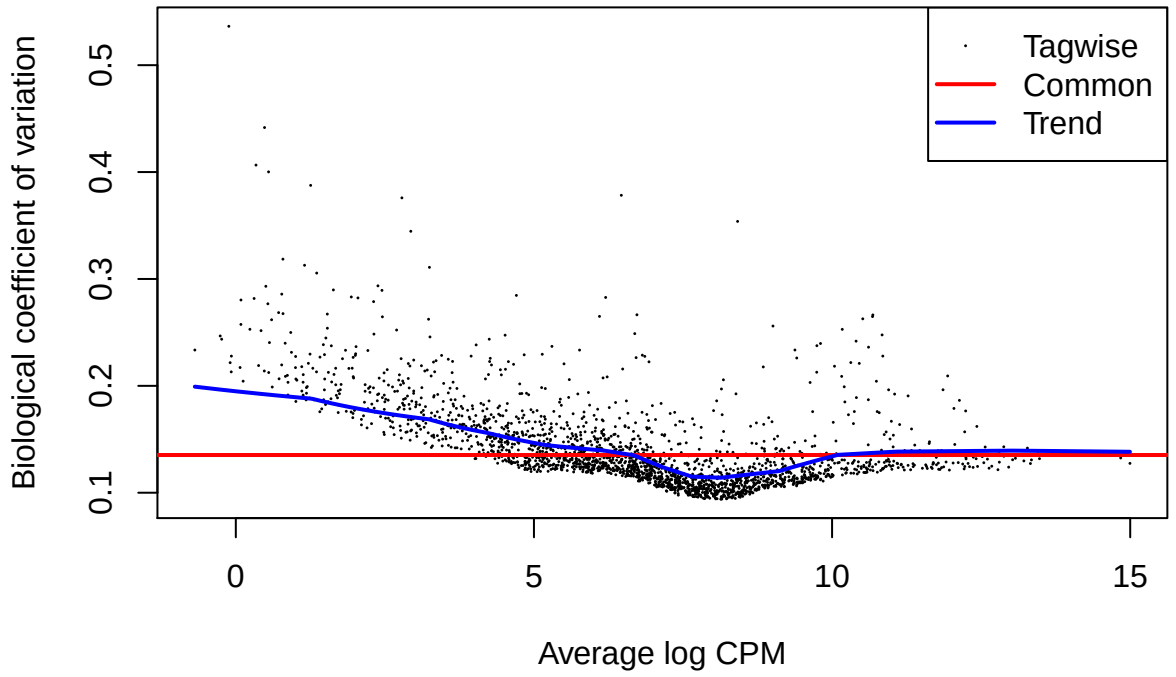
Dispersie plot

De dispersie plot laat de spreiding van de data zien. Dit zegt iets over hoe goed de verschillende samples met elkaar te vergelijken zijn. Op de X-as staat de CPM (counts per million) op de log schaal en Y-as de dispersie. De rode lijn is de common, het ideale scenario. De blauwe lijn laat de afwijking zien van variatie van de verschillende samples. Hier is te zien dat de trendlijn redelijk overeen komt met de common lijn.

```
plotBCV(y)
```



```
plotBCV(f)
```



Conclusie