Genexpressie van L. plantarum WCFS1 en NC8 bij glucose en ribose

Teun van Dorp, Luc Hengeveld, Michelle Memelink

16/01/2021

Algemeen doel

Het doel is om te onderzoeken of er verschil is in genexpressie bij de bacterie *Lactobacillus plantarum* WCFS1 en NC8, als deze op verschillende voedingsbodems groeien. De twee gebruikte soorten voedingsbodem zijn glucose en ribose. Er is op beide bodems twee keer een experiment uitgevoerd per stam. De resultaten die hieruit zijn gekomen worden gebruikt in dit onderzoek. De resultaten die door middel van deze code verkregen worden hebben dus betrekking op die resultaten. Uit deze resultaten kan uiteindelijk afgeleid worden welke genen up of down gereguleerd zijn of gelijk zijn gebleven in de expressie.

Ophalen bibliotheken

De benodigde bibliotheken worden aangeroepen.
Output: welke package is ingeladen.

```
library(edgeR)
library(xlsx)
library(rstudioapi)
library(rafalib)
library(pathview)
library(ggplot2)
library(factoextra)
```

Bestand openen en inlezen

Er kan via een pop-up een map geselecteerd worden waarin het bestand staat met de naam “RNA-Seq-counts.txt”. Vervolgens wordt er dus de naam van het bestand dat ingelezen moet worden aangegeven, deze wordt tevens ingeladen. Dan wordt het bestand ingelezen en worden alle ID’s van de genen eruit gehaald. Ook wordt in een andere vector de namen opgeslagen van de verschillende experimenten, dit zijn de kolomnamen.

```
fDir <- selectDirectory(caption = "Select Directory")
fName <- "/RNA-Seq-counts.txt"
cnts <- read.delim(paste0(fDir,fName), comment.char="#")
row.names(cnts) <- cnts[,"ID"]
exp <- c("WCFS1.glc", "WCFS1.glc", "WCFS1.rib", "WCFS1.rib", "NC8.glc",
"NC8.glc", "NC8.rib", "NC8.rib")
```

DGElist aanmaken

Er wordt een DGElist gemaakt, door gebruik te maken van EdgeR. Het is een lijst welke de genen bevat met de counts die geanalyseerd moeten worden. De aangemaakte vector met de kolomnamen wordt meegegeven voor het maken ven de DGElist. Output: als eerst worden de groepen aangegeven en de vier niveaus. Vervolgens wordt er van elk experiment apart een samenvatting gegeven, deze bevat onder andere het gemiddelde, de mediaan en de eerste en derde kwartiel. Vervolgens wordt er de gemaakte DGElist getoond. Er wordt daarna ook per experiment verkregen resultaten gegeven. De tweede bevat een kolom met de naam van het experiment. De tweede kolom bevat de algemene namen van de condities (welke voedingsbodem). Daarna een kolom die de totaal aantal counts voor elk conditie bevat. De laatste kolom bevat normalisatie factor waarden.

```
group <- factor(exp)
print(group)

## [1] WCFS1.glc WCFS1.glc WCFS1.rib WCFS1.rib NC8.glc   NC8.glc   NC8.rib
## [8] NC8.rib
## Levels: NC8.glc NC8.rib WCFS1.glc WCFS1.rib

print(summary(cnts[,2:9]))

##   WCFS1.glc.1        WCFS1.glc.2      WCFS1.rib.1        WCFS1.rib.2
##  Min.   :     0.0   Min.   :     0   Min.   :     0.0   Min.   :     0
##  1st Qu.:   271.5   1st Qu.:   272   1st Qu.:   275.5   1st Qu.:   308
##  Median :   960.0   Median :  1016   Median :   993.0   Median :  1086
##  Mean   :  4575.8   Mean   :  4784   Mean   :  4037.4   Mean   :  4209
##  3rd Qu.:  2954.5   3rd Qu.:  3180   3rd Qu.:  2853.5   3rd Qu.:  3136
##  Max.   :304949.0   Max.   :348490   Max.   :299861.0   Max.   :309971
##    NC8.glc.1          NC8.glc.2          NC8.rib.1        NC8.rib.2
##  Min.   :     0.0   Min.   :     0.0   Min.   :     0   Min.   :     0
##  1st Qu.:   235.5   1st Qu.:   271.5   1st Qu.:   295   1st Qu.:   315
##  Median :   950.0   Median :  1045.0   Median :  1070   Median :  1133
##  Mean   :  4426.2   Mean   :  4624.6   Mean   :  4485   Mean   :  4487
##  3rd Qu.:  2873.5   3rd Qu.:  3240.0   3rd Qu.:  3226   3rd Qu.:  3386
##  Max.   :306306.0   Max.   :307691.0   Max.   :385049   Max.   :347366

y <- DGEList(counts=cnts[,2:9],group=group)
print(y)

## An object of class "DGEList"
## $counts
##         WCFS1.glc.1 WCFS1.glc.2 WCFS1.rib.1 WCFS1.rib.2 NC8.glc.1
NC8.glc.2
## lp_0001        8100        9599        8144        7000      7117
8278
## lp_0002       12679       15856       11539       11049     10815
14348
## lp_0004        1795        1946        1470        1607      1489
1407
```

```
## lp_0005        8538        8740        5699        7402       6497
8565
## lp_0009       56040       42130       31941       23500      61965
37353
##         NC8.rib.1 NC8.rib.2
## lp_0001      7457      6980
## lp_0002     10552     10735
## lp_0004      1587      1699
## lp_0005      6581      8342
## lp_0009     20498     18188
## 2214 more rows ...
##
## $samples
##                 group lib.size norm.factors
## WCFS1.glc.1 WCFS1.glc 10153710            1
## WCFS1.glc.2 WCFS1.glc 10615392            1
## WCFS1.rib.1 WCFS1.rib  8959060            1
## WCFS1.rib.2 WCFS1.rib  9340139            1
## NC8.glc.1     NC8.glc  9821662            1
## NC8.glc.2     NC8.glc 10261922            1
## NC8.rib.1     NC8.rib  9951954            1
## NC8.rib.2     NC8.rib  9957519            1
```

Normaliseren

De waardes worden genormaliseerd. Dit wordt gedaan door het bijsnijden van de gemiddelde van M-waarden: verwijderen van de laagste en hoogste waarden.

```
y <- calcNormFactors(y, method="TMM")
lijst_norm_all <- y
```

Filteren op de low read counts

Van de genormaliseerde data worden de genen uit de data set verwijderd als ze onder het gemiddelde eerste kwartiel vallen. Het gemiddelde is berekend over alle acht de experimenten. Als de waarde er bovenligt dan is die TRUE, als de waarde er onder valt dan is die FALSE. Alle genen die een FALSE hebben gekregen worden niet bewaard in de data set. Output: een samenvatting welke onder andere de kwartielen gemiddelde, hoogste en laagste waardes bevatten per experiment. Vervolgens wordt er aangegeven hoeveel genen FALSE en TRUE hebben gekregen. Daarna wordt de DGElist weer weergeven op dezelfde manier als eerder hierboven benoemt onder “DGElist aanmaken”.

```
print(summary(y$counts))
```

```
##   WCFS1.glc.1       WCFS1.glc.2      WCFS1.rib.1       WCFS1.rib.2
##  Min.   :     0.0   Min.   :     0   Min.   :     0.0   Min.   :     0
##  1st Qu.:   271.5   1st Qu.:   272   1st Qu.:   275.5   1st Qu.:   308
##  Median :   960.0   Median :  1016   Median :   993.0   Median :  1086
##  Mean   :  4575.8   Mean   :  4784   Mean   :  4037.4   Mean   :  4209
##  3rd Qu.:  2954.5   3rd Qu.:  3180   3rd Qu.:  2853.5   3rd Qu.:  3136
```

```
##  Max.   :304949.0   Max.   :348490   Max.   :299861.0   Max.   :309971
##    NC8.glc.1          NC8.glc.2          NC8.rib.1        NC8.rib.2
##  Min.   :     0.0   Min.   :     0.0   Min.   :     0   Min.   :     0
##  1st Qu.:   235.5   1st Qu.:   271.5   1st Qu.:   295   1st Qu.:   315
##  Median :   950.0   Median :  1045.0   Median :  1070   Median :  1133
##  Mean   :  4426.2   Mean   :  4624.6   Mean   :  4485   Mean   :  4487
##  3rd Qu.:  2873.5   3rd Qu.:  3240.0   3rd Qu.:  3226   3rd Qu.:  3386
##  Max.   :306306.0   Max.   :307691.0   Max.   :385049   Max.   :347366
```

```
firstQuartile <- c()
for(i in 2:9) {
  firstQuartile[i-1] <- unname(quantile(y$counts, .25))
}
highEnough <- y$counts > mean(firstQuartile)
keep <- rowSums(highEnough) >= 4
summary(keep)
```

```
##    Mode   FALSE    TRUE
## logical     495    1724
```

```
y <- y[keep, keep.lib.sizes=FALSE]
print(y)
```

```
## An object of class "DGEList"
## $counts
##         WCFS1.glc.1 WCFS1.glc.2 WCFS1.rib.1 WCFS1.rib.2 NC8.glc.1
NC8.glc.2
## lp_0001        8100        9599        8144        7000      7117
8278
## lp_0002       12679       15856       11539       11049     10815
14348
## lp_0004        1795        1946        1470        1607      1489
1407
## lp_0005        8538        8740        5699        7402      6497
8565
## lp_0009       56040       42130       31941       23500     61965
37353
##         NC8.rib.1 NC8.rib.2
## lp_0001      7457      6980
## lp_0002     10552     10735
## lp_0004      1587      1699
## lp_0005      6581      8342
## lp_0009     20498     18188
## 1719 more rows ...
##
## $samples
##                 group lib.size norm.factors
## WCFS1.glc.1 WCFS1.glc 10099303    0.9850714
## WCFS1.glc.2 WCFS1.glc 10561170    0.9830048
## WCFS1.rib.1 WCFS1.rib  8908525    0.9986438
## WCFS1.rib.2 WCFS1.rib  9282792    1.0375329
```

```
## NC8.glc.1     NC8.glc  9771733    0.9390775
## NC8.glc.2     NC8.glc 10204798    1.0155475
## NC8.rib.1     NC8.rib  9897428    0.9899344
## NC8.rib.2     NC8.rib  9895710    1.0557378
```

Controleren van data

Er wordt gekeken of de data juist is. Er wordt gecontroleerd of het inlezen en normaliseren op de juiste wijze is verlopen.
Output: Er wordt per experiment resultaten getoond. De tweede bevat een kolom met de naam van de experiment. De twee kolom bevat de algemene namen van de condities (welke voedingsbodem). Daarna een kolom wat die de totaal aantal counts voor elk conditie bevat. De laatste kolom bevat normalisatie factor waardes.

```
print(y$samples)
```

```
##                 group lib.size norm.factors
## WCFS1.glc.1 WCFS1.glc 10099303    0.9850714
## WCFS1.glc.2 WCFS1.glc 10561170    0.9830048
## WCFS1.rib.1 WCFS1.rib  8908525    0.9986438
## WCFS1.rib.2 WCFS1.rib  9282792    1.0375329
## NC8.glc.1     NC8.glc  9771733    0.9390775
## NC8.glc.2     NC8.glc 10204798    1.0155475
## NC8.rib.1     NC8.rib  9897428    0.9899344
## NC8.rib.2     NC8.rib  9895710    1.0557378
```

Matrix aanmaken

Er wordt een matrix gemaakt. Er wordt aangegeven welke kolom welk experiment bevat. Vervolgens wordt er een relatie gelegd tussen conditie (welke voedingsbodem), type stam en de experimenten. Er wordt aangegeven welke experimenten gelijk aan elkaar zijn gekeken naar de conditie, dus een groep glucose en ribose. Output: een matrix Met als eerste kolom de namen van de experimenten. De tweede kolom bevat de conditie glucose voor NC8. De derde kolom bevat de conditie ribose voor NC8. de Vierde kolom bevat de conditie glucose voor WCFS1 en de laatste kolom bevat de conditie ribose voor WCFS1. Met een één wordt aangegeven dat, dat experiment bij die conditie en stam hoort. Vervolgens wordt er aangegeven hoeveel kolommen data bevatten. De kolommen met de condities hebben minstens één, één erin staan, dus niet alle waardes in de kolom zijn nul in de matrix. Als laatste wordt er voor gezorgd dat elk niveau met de basislijnniveau wordt gecontrasteerd.

```
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"
```

Spreidingsbreedte Bepalen

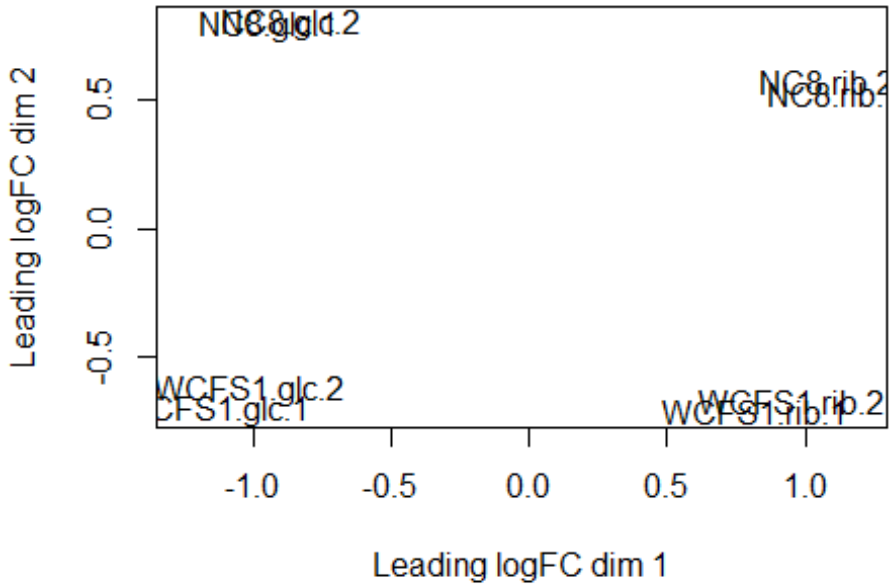
Er wordt op drie verschillende manieren de spreidingsbreedte bepaald. De eerste is de common, dit is de spreiding bepaald over alle waardes en daar het gemiddelde van genomen. De tweede manier is om de spreiding te bepalen van het gemiddelde bij iedere waarde op de x-as en hier vervolgens een lijn van de maken. De derde manier is dat er per experiment resultaat een spreiding wordt bepaald, dus per gen.

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

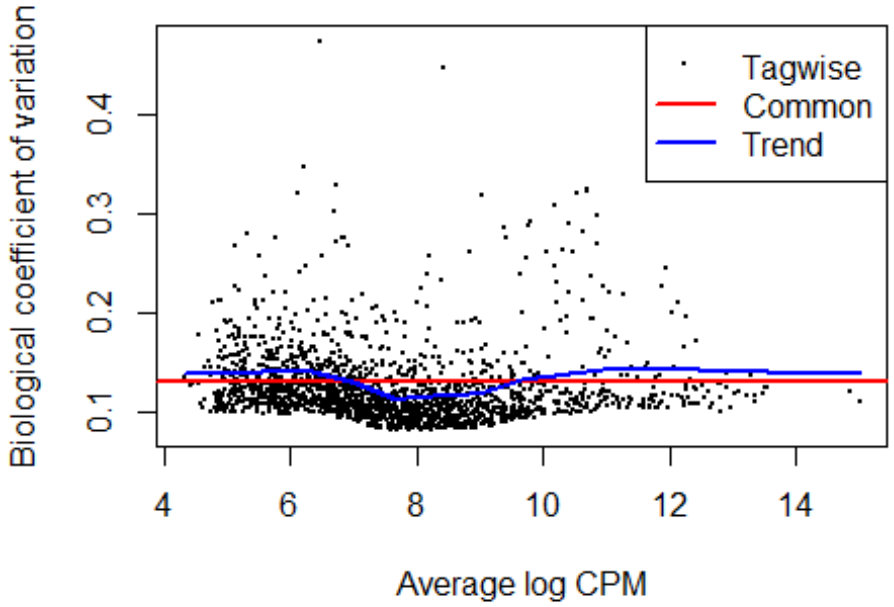
Grafieken Maken (PCA en BCV)

De twee grafieken worden aangemaakt. De eerste grafiek is een multidimensionaal schaaldiagram van afstanden tussen genexpressieprofielen. De experimenten zijn op een tweedimensionale scatterplot geplot, zodat de afstanden op de plot de typische log2-voudige veranderingen tussen de experimenten benaderen. De tweede grafiek is een plot welke biologische variatiecoëfficiënt toont. De genewise biologische variatiecoëfficiënt (BCV) zijn uitgezet tegen de overvloed aan genen (in log2 tellingen per miljoen). Output: in de eerste grafiek als er gekeken wordt naar de eerste dimensie, is te zien dat zowel glucose als ribose experimenten bijna dezelfde waarde hebben. Echter is er een groot verschil te zien in de waarde gekeken naar glucose ten opzichten van ribose. Dit betekend dat er sprake is van verschillende genexpressie. Als er gekeken wordt naar de tweede dimensie, is te zien dat bij zowel de ribose experimenten als de glucose experimenten bijna dezelfde waarden hebben. Echter is er wel verschil zichtbaar tussen glucose en ribose, maar dit bevat een minder groot afstand dan het verschil gekeken naar de eerste dimensie tussen glucose en ribose, Dus kan er met zekerheid gesteld worden dat er spraken is van verschil in genexpressie. Ook is te zien dan de experimenten van de stammen rond dezelfde waarde liggen gekeken naar de tweede dimensie. In de tweede grafiek zijn de verschillende spreidingswaardes (variatiecoëfficiënt) getoond. De x-as geeft aan hoe vaak bepaald gen is geteld. De y-as geeft de mate van spreiding aan.

```
plotMDS(spreiding)
```




```
plotBCV(spreiding)
```



Fit data

Op basis van de design matrix en de samples, wordt er een model lijn (lineaire) gemaakt hoe de waardes zich tot elkaar verhouden.

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

Bepalen van de fold change

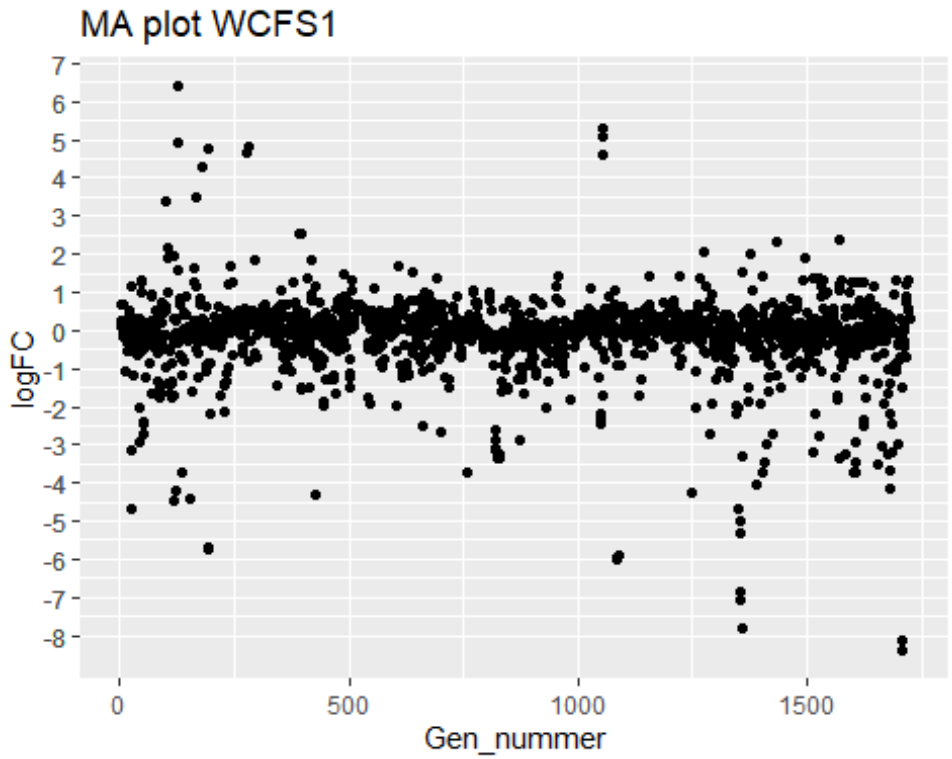
De input is het gemaakte model. Er wordt aangegeven wat er gemeten moet worden, dit is het verschil in genexpressie tussen glucose en ribose bij WCFS1 en bij NC8. Dit verschil is de fold change. Vervolgens wordt het model passende gemaakt en geef je het vorm.

```
mcWCFS1 <- makeContrasts(exp.r=WCFS1.glc-WCFS1.rib, levels=design)
fitWCFS1 <- glmLRT(fit, contrast=mcWCFS1)
mcNC8 <- makeContrasts(exp.r=NC8.glc-NC8.rib, levels=design)
fitNC8 <- glmLRT(fit, contrast=mcNC8)
```

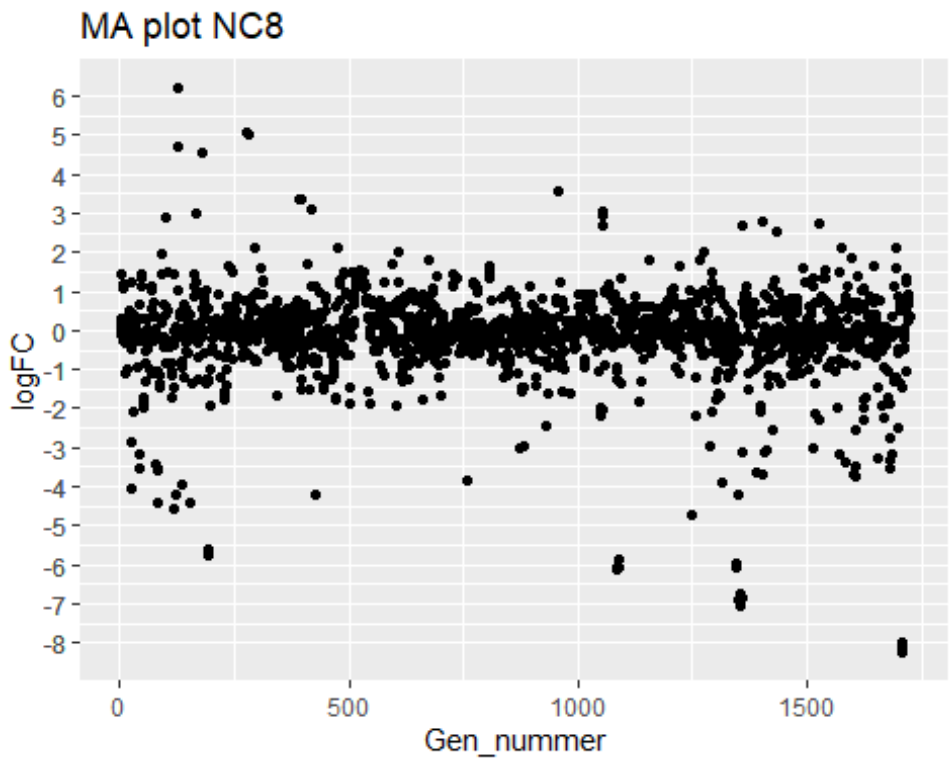
MA Plot

Er worden MA grafieken gemaakt waarin afgelezen kan worden welke fold change gekozen kan worden om nog genoeg genen in de data set te houden en welke een goede FDR waarde bevatten, maar ook dat je genen pakt met de hoge en lage fold change waardes. Alle genen die tussen de twee waardes liggen kunnen er uit gefilterd worden. Output: MA plot, met op de y-as de fold change en op de x-as het aantal genen. Eerste gen in de tabel is nummer 1 en de tweede nummer 2 enzovoort. De eerste grafiek behoord bij WCFS1 en de tweede grafiek met dezelfde opbouw behoort bij NC8.

```
Gen_nummer = c()
for (i in 0:length( fitWCFS1$table$logFC)){
  Gen_nummer[i] <- i
}
data <- data.frame(fitWCFS1)
ggplot <- ggplot(data, aes(x = Gen_nummer, y = logFC))  + geom_point() +
scale_y_continuous(breaks = round(seq(min(data), max(data), by = 0.5),0))
print(ggplot + ggtitle("MA plot WCFS1"))
```




```
Gen_nummer = c()
for (i in 0:length( fitNC8$table$logFC)){
  Gen_nummer[i] <- i
}
data <- data.frame(fitNC8)
ggplot <- ggplot(data, aes(x = Gen_nummer, y = logFC))  + geom_point() +
scale_y_continuous(breaks = round(seq(min(data), max(data), by = 0.5),0))
print(ggplot + ggtitle("MA plot NC8"))
```



Beste resultaten weergeven

Als laatste wordt het resultaat weergegeven, wat voortkomt uit het model. Er komt als resultaat een tabel met alle genen en de bijbehorende waardes uit. Er wordt per stam een tabel aangemaakt namelijk WCFS1.rib met WCFS1.glc en NC8.rib met NC8.glc. Output: de eerste kolom bevat de gen ID's. De tweede kolom bevat de log fold change waardes, dus of een gen up gereguleerd of down gereguleerd is of gelijk gebleven. De derde kolom bevat de verschillen in counts, dus hoe vaak gen geteld is. De vijfde kolom bevat de p-waardes. Dit zijn de exact p-waarde voor differentiële expressie. De laatste kolom bevat de p-waarde aangepast voor meervoudige tests.

```
print(topTags(fitWCFS1))
```

```
## Coefficient:  1*WCFS1.glc -1*WCFS1.rib
##             logFC    logCPM        LR        PValue           FDR
## lp_3658 -8.134276 12.124232 1242.9357 2.846504e-272 4.907373e-269
## lp_2154 -5.890498 10.214967  911.9207 2.514313e-200 2.167338e-197
## lp_0371 -5.667997  8.980998  888.5357 3.048201e-195 1.751699e-192
## lp_2151 -5.972273 11.165349  873.8553 4.736558e-192 2.041456e-189
## lp_3660 -8.398716 11.394780  832.7239 4.144601e-183 1.429058e-180
## lp_3659 -8.141327 11.032494  805.6414 3.202694e-177 9.202408e-175
## lp_2152 -6.032237 11.041941  775.7457 1.012723e-170 2.494191e-168
## lp_2153 -5.913993 10.449397  757.2681 1.054535e-166 2.272523e-164
## lp_2757 -4.068558  7.596042  695.3986 2.994422e-153 5.735983e-151
## lp_0575  4.687822  9.934456  689.7375 5.097641e-152 8.788334e-150
```

```
print(topTags(fitNC8))
```

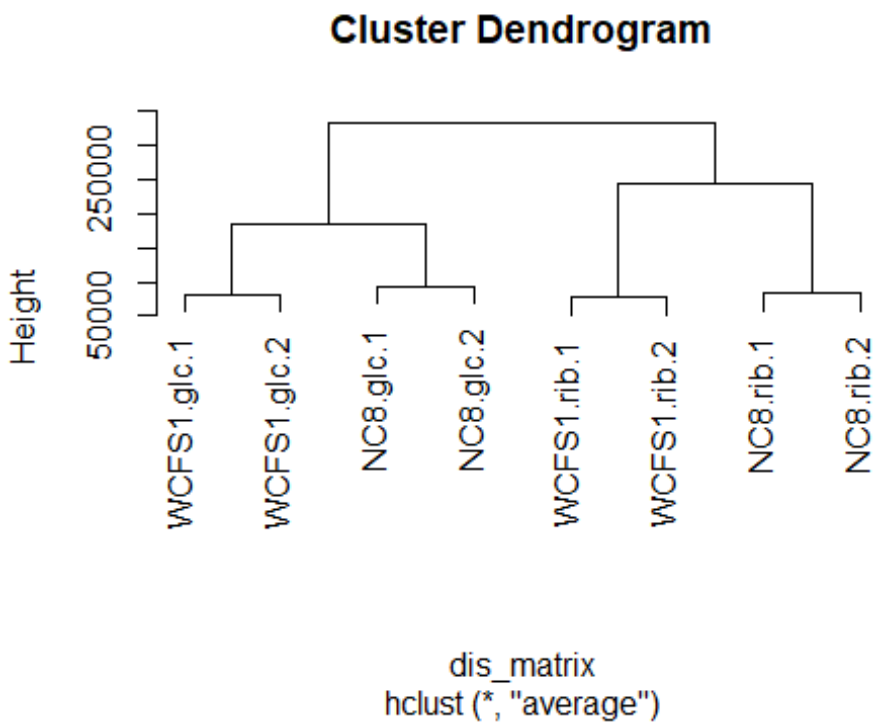
```
## Coefficient:  1*NC8.glc -1*NC8.rib
##             logFC    logCPM        LR        PValue           FDR
## lp_3658 -8.246398 12.124232 1267.9814 1.026572e-277 1.769810e-274
## lp_0371 -5.776817  8.980998  944.1419 2.489712e-207 2.146132e-204
## lp_2154 -5.839579 10.214967  906.7970 3.267755e-199 1.877870e-196
## lp_2151 -6.056332 11.165349  891.5993 6.577352e-196 2.834839e-193
## lp_3660 -8.098869 11.394780  799.3202 7.583200e-176 2.614687e-173
## lp_2152 -6.107747 11.041941  791.7473 3.360178e-174 9.654912e-172
## lp_2153 -6.072328 10.449397  789.2907 1.149410e-173 2.830833e-171
## lp_3659 -7.964803 11.032494  785.2242 8.803030e-173 1.897053e-170
## lp_0575  5.070715  9.934456  783.5501 2.035231e-172 3.898599e-170
## lp_3314 -3.746701  8.715154  684.0707 8.703262e-151 1.500442e-148

resWCFS1<-topTags(fitWCFS1, n = length(fitWCFS1$fitted.values))
resNC8<-topTags(fitNC8, n = length(fitNC8$fitted.values))
ongefilterdWCFS1 <- resWCFS1
ongefilterdNC8 <- resNC8
```

Genen Clusteren

Aan de hand van de matrix welke de afstanden bevat is er op twee manieren geclusterd, namelijk hiërarchisch geclusterd en via de methode “K means”. Er is vervolgens ook nog met de methode “K means” een clustering gemaakt met alle genen. Output: de eerste grafiek geeft het resultaat van het hiërarchisch clusteren weer, waarin gezien kan worden welke experimenten het meeste op elkaar lijken enzovoort. Dit geldt hetzelfde voor de tweede grafiek, maar dan met de “K means” methode. De laatste grafiek geeft weer welke genen bij elkaar horen tot een cluster, ook met de “K means” methode is dit weergegeven.

```
dis_matrix <- dist(t(lijst_norm_all$counts[,1:8]), method = "euclidean")
hclust_object <- hclust(dis_matrix, method = "average")
plot(hclust_object)
```




```
km.res <- kmeans(dis_matrix, 4, nstart = 1)
fviz_cluster(km.res, data = dis_matrix, geom = "point")
```


```
km.res$cluster
```

```
## WCFS1.glc.1 WCFS1.glc.2 WCFS1.rib.1 WCFS1.rib.2   NC8.glc.1   NC8.glc.2
##           2           2           1           1           3           3
##   NC8.rib.1   NC8.rib.2
##           4           4

km.res <- kmeans(y, 4, nstart = 1)
fviz_cluster(km.res, data = y, geom = "point")
```

Filteren fold change en FDR-waarde

Aan de hand van de MA plot is ervoor gekozen op alles tussen de 1 en de -1 aan fold change weg te filteren en als de FDR-waarde groter is dan 0,001 wordt het gen er ook uitgefilterd, dus meer richting de nul. de overgebleven genen worden weer in een data frame gezet. Er is uiteindelijk een data frame voor WCFS1 en NC8.

```
resWCFS1 <- data.frame(resWCFS1)
attach(resWCFS1$table)
resWCFS1 <- subset(resWCFS1, (logFC > 1 | logFC < -1) & FDR < 0.001)
resNC8 <- data.frame(resNC8)
attach(resNC8$table)
resNC8 <- subset(resNC8, (logFC > 1 | logFC < -1) & FDR < 0.001)
```

KEGG mapper

Met de overgebleven genen zijn KEGG pathways bekeken aan de hand van de fold change waardes. Er is voor zowel WCFS1 en NC8 vier pathways bekeken, namelijk Glycolyse / Gluconeogenese, citroenzuurcyclus en pentosefosfaat. Output: lokaal worden de acht

pathways opgeslagen. met groen wordt aangegeven dat er een hogere genexpressie is bij ribose en met rood wordt aangegeven dat de genexpressie bij glucose hoger is en met geen kleur wordt aangegeven dat het gelijk is.

```
logFC <- resWCFS1$logFC
names(logFC) <- row.names(resWCFS1)
pathview(gene.data = logFC, species = "lpl", pathway = "lpl00010",
gene.idtype = "KEGG", out.suffix = "WCFS1.Glycolysis_Gluconeogenesis")
pathview(gene.data = logFC, species = "lpl", pathway = "lpl00020",
gene.idtype = "KEGG", out.suffix = "WCFS1.CytrateCycle")
pathview(gene.data = logFC, species = "lpl", pathway = "lpl00030",
gene.idtype = "KEGG", out.suffix = "WCFS1.PentosePhosphate")
logFC <- resNC8$logFC
names(logFC) <- row.names(resNC8)
pathview(gene.data = logFC, species = "lpl", pathway = "lpl00010",
gene.idtype = "KEGG", out.suffix = "NC8.Glycolysis_Gluconeogenesis")
pathview(gene.data = logFC, species = "lpl", pathway = "lpl00020",
gene.idtype = "KEGG", out.suffix = "NC8.CytrateCycle")
pathview(gene.data = logFC, species = "lpl", pathway = "lpl00030",
gene.idtype = "KEGG", out.suffix = "NC8.PentosePhosphate")
```

Annotatie toevoegen aan genen

De twee data frames, welke de fold change bevatten, krijgen extra informatie toegevoegd vanuit het bestand genaamd: “WCFS1_anno.txt”. Dit bestand moet in dezelfde map staan welk aan het begin van het programma is geselecteerd. Het bestand bevat informatie over de genen. Deze informatie wordt toegevoegd aan de bestaande data frames.

```
annotation = read.delim("WCFS1_anno.txt", header=TRUE, row.names = 1)
drops <- c("X","X.1","X.2","X.3", "X.4")
annotation <- annotation[ , !(names(annotation) %in% drops)]
resWCFS1 <- cbind(resWCFS1, annotation[rownames(resWCFS1),])
resNC8 <- cbind(resNC8, annotation[rownames(resNC8),])
```

Resultaten opslaan in een Excel bestand

De data van de tabellen, welke de fold change waardes bevatten, worden lokaal opgeslagen in een Excel bestand. Dit bestand bestaat uit 2 verschillende sheets: een sheet met alle data en een ander sheet met de beste waarden.

```
write.xlsx(ongefilterdWCFS1, paste0(fDir, "results.xlsx"), sheetName = "WCFS1
ongefilterd",
           col.names = T, row.names = T, append = F)
write.xlsx(ongefilterdNC8, paste0(fDir, "results.xlsx"), sheetName = "NC8
ongefilterd",
           col.names = T, row.names = T, append = T)
write.xlsx(resWCFS1, paste0(fDir, "results.xlsx"), sheetName = "WCFS1
gefilterd",
           col.names = T, row.names = T, append = T)
write.xlsx(resNC8, paste0(fDir, "results.xlsx"), sheetName = "NC8 gefilterd",
```

```
            col.names = T, row.names = T, append = T)
write.xlsx(km.res$cluster, paste0(fDir, "results.xlsx"), sheetName = "K-means
Clustering",
            col.names = F, row.names = T, append = T)
```