**Transcriptie analyse (BRNA)**

**RNA-seq analyse in R**

Door: Dirk van der Torre en Robin Zanoni  
Klas: Bin3BM   
Docent: Koen Bossers  
Datum: 5 April 2018  
1e mogelijkheid

Inhoudsopgave

Introductie en data import 3

Filteren op lage expressie 4

Quality control op count data 6

Hierarchical clustering met een Heatmap 9

Normalisatie (voor compositie bias) 12

DGE testing 15

GSEA/Pathway level analyse 17

# Introductie en data import

RNA-seq is een techniek om transcriptoom in kaart te brengen. Dit wordt gedaan door het meten van de expressie van genen. Dit geeft echter veel data wat geanalyseerd moet worden. Door gebruik van statistische analyse in R kan dit. Er word begonnen met het inladen van packages die nodig zijn voor analyse.

library(limma)

library(edgeR)

library(gplots)

library(RColorBrewer)

library(org.Hs.eg.db)

library(GO.db)

De data wat gebruikt wordt zijn van 4 celtypes in zowel behandelde “treated” als onbehandelde “control”. Hiermee is een RNA-seq experiment uitgevoerd hiervan is de kwaliteit van de ruwe reads al bepaald. Deze reads zijn vervolgens ge-aligned tegen een human referentie genoom wat resulteert in de count tabel. De onderzoeksvraag is wat is het effect van een behandeling op chronische inflammatie door middel van transscriptoom analyse. Deze data is te vinden op ELO van Hogeschool Leiden.

annotatie <- read.csv(file="Data/annotables\_unique\_grch37(1).csv", header = TRUE)

metadata <- read.csv(file="Data/chronische\_inflammatie\_metadata.csv", header = TRUE)

rawcounts <- read.csv(file="Data/chronische\_inflammatie\_rawcounts(1).csv", header = TRUE)

Doordat de counts per million (cpm) alleen numerieke kolommen accepteert, wordt de eerste kolom met de namen van de genen weg gehaald, deze namen kunnen altijd terug gevonden worden in de rawcount data.

countdata <- rawcounts[,-1]

rownames(countdata) <- rawcounts[,1]

Hierna worden achter de nummer code van de countdata neergezet of het sample treated (t) of control ( c) is zodat dit later makkelijker uit elkaar te halen is.

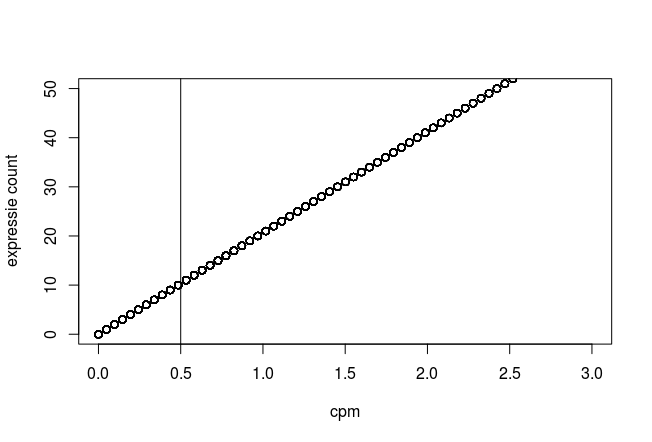
colnames(countdata) <- c("N61311.c", "N61311.t", "N052611.c", "N052611.t", "N080611.c", "N080611.t", "N061011.c", "N061011.t")

# Filteren op lage expressie

Genen met een lage expressie zeggen weinig over de differentiële expressie en kunnen interfereren met de statistische berekeningen later. Ook kunnen deze lage expressies een vals positieve geven om deze reden worden eerst alle genen gefilterd op lage expressie. Er is te zien dat de libraries verschillen in de grootte er verschillen 15 miljoen sequenties tussen de grootste en de kleinste library om dit tegen te gaan werken we met log2 counts. Je normaliseert door het verschil tussen het total aantal reads per sample.

CountsPM <- cpm(countdata)

Om te bepalen welke counts gebruikt worden voor verdere analyse is er een threshold bepaald.

Daarvoor is eerst de kolom/sample van de cpm geplot. Daarin is een vertical lijn gezet op 0,5 cpm, dit is een veel gebruikte threshold voor minimale expressie. Te zien in Figuur 1 is dat er bij een cpm van 0.5 er een expressie counts van 10 zijn. Dit is het minimum aantal counts per gen, hierdoor is het een betrouwbare threshold.

par(mfrow=c(1,1))

plot(CountsPM[,1], countdata[,1],ylim=c(0,50),xlim=c(0,3), xlab="cpm", ylab="expressie count")

abline(v=0.5)

***Figuur 1: counts per miljoen analyse:*** *Bij een cpm van 0,5 is er een expressie counts van 10 dit is een minimale aantal counts per gen.*

Vervolgens word bepaald welke genen een lager cpm hebben van 0,5 deze samples worden getagd als FALSE. Daarna worden de sampels er uit gefilterd die minder dan 2 samples hebben waarvan de cpm hoger en/of gelijk is aan 0,5 dit omdat deze genen niet significant genoeg tot uiting komen.

thresh <- CountsPM > 0.5

table(rowSums(thresh))

keep = rowSums(thresh) >= 2

counts.keep <- countdata[keep,]

summary(keep)

Na dat alle genen met een te laag cpm er uitgefilterd zijn worden de overgebleven cpm’s omgezet in een DGE list object (differential gene expression) Hier wordt bekeken of naast de genen meerdere expressies bevatten . Dit formaat wordt gebruikt voor het normaliseren van de library grote.

countsDGE <- DGEList(counts.keep)

countsDGE$counts

countsDGE$samples

# Quality control op count data

Nu de lage expressie genen verwijderd zijn en de counts staan opgeslagen in de DGE list object, kan er gekeken worden naar verschillende plots om de kwaliteit van de data na te gaan en te bekijken wat de meeste variatie veroorzaakt. Dit wordt op twee manieren gedaan de eerste is door middel van een barplot met aansluitende boxplot en de ander is een clustering van de samples gebaseerd op celtype en behandeling. Als eerst wordt er een barplot gemaakt dit is om te kijken

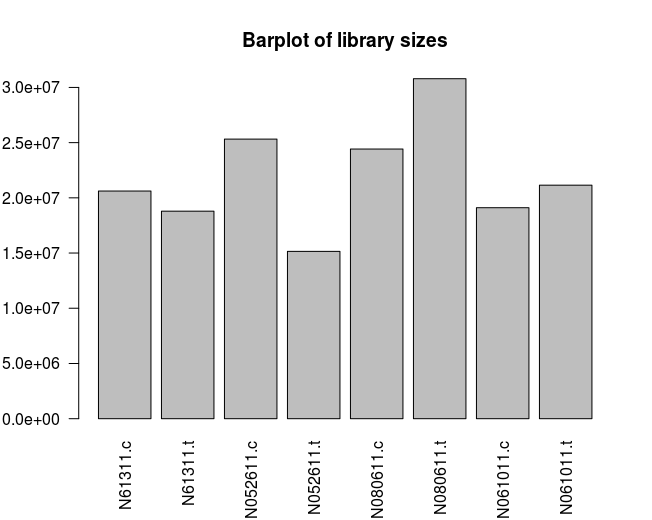
naar de verdeling van de samples. Aangezien de count data niet normaal verdeeld is worden deze counts omgezet in logcounts. Zoals te zien is in figuur 2 is er een verschil tussen de library grote van de controle en threated van elk celtype.

countsDGE$samples$lib.size

barplot(countsDGE$samples$lib.size,names=colnames(countsDGE),las=2)

title("Barplot of library sizes")

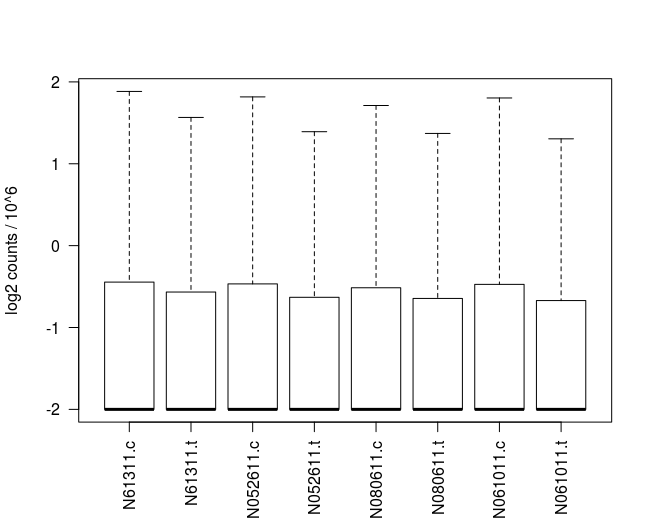
logcounts <- cpm(CountsPM,log=TRUE)



***Figuur 2:*** *Barplot library grootte*

Met behulp van de logcounts wordt een boxplot gemaakt zoals zichtbaar is in figuur 3. Hierin is te zien dat de eerste en tweede kwart van de data een verschil en variatie heeft in expressie niveaus. De derde en vierde kwartalen heeft zeer weinig variatie.

boxplot(logcounts, xlab="", ylab="log2 counts / 10^6",las=2, outline=FALSE)



***Figuur 3:*** *boxplots van de logcounts per sample.*

Nu worden er twee multiple dimension scaling (MDS) plots gemaakt om te kijken naar de clustering van de samples op basis van cel type en controle versus behandeld. Eerst wordt een MDS gemaakt van de controle versus behandeld. Er wordt gekeken naar de verdeling van de data tussen de controle en behandeld zoals te zien is in figuur 4. Hierin is te zien dat de controle versus treated steeds het zelfde verschil geeft tussen de samples. Visueel is te zien dat de paarse (controle) en oranje (treated) erg op elkaar lijken en bij elkaar clusteren. Er is dus niet veel variatie als je alle controle samples met de behandelde samples vergelijkt.

par(mfrow=c(1,1))

plotMDS(countsDGE)

levels(metadata$dex)

metadata

countsDGE$samples

col.behandeling <- c("purple", "orange")[metadata$dex]

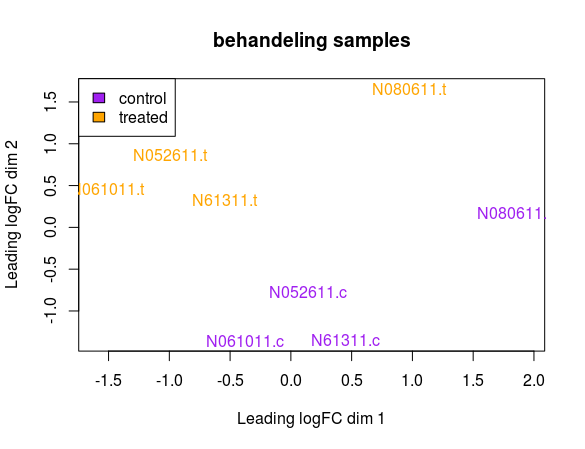
data.frame(metadata$dex,col.behandeling)

par(mfrow=c(1,1))

plotMDS(countsDGE, col=col.behandeling)

legend("topleft",fill=c("purple", "orange"),legend=levels(metadata$dex))

title("behandeling samples")

***Figuur 4:*** *clustering van samples op basis van behandeling*

Na de MDS plot van de controle versus de treated wordt er een MDS plot gemaakt van de verschillende cel type om te kijken of dit meer variatie geeft, zie figuur 5.

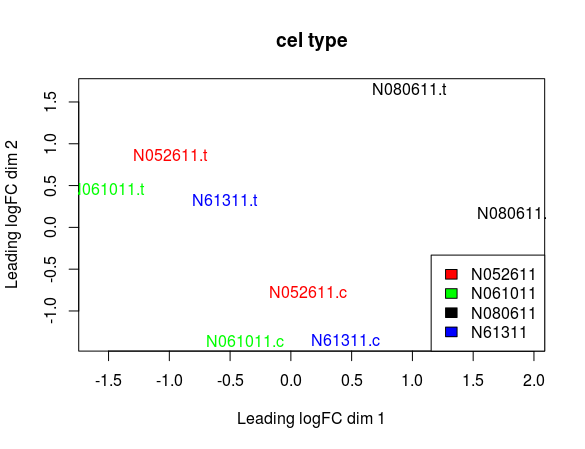
col.celtype <- c("red", "green", "black", "blue")[metadata$celltype]

plotMDS(countsDGE, col=col.celtype)

legend("bottomright",fill=c("red", "green", "black", "blue"),legend=levels(metadata$celltype))

title("cel type")

par(mfrow=c(1,1))

Te zien in figuur 5 is visueel te zien dat de kleuren van de cel type niet bij elkaar clusteren. Dit geeft aan dat er meer variatie is tussen de verschillende cel type, dan tussen de behandeling. De cel type zijn hierdoor de eigenschap waar naar gekeken moet worden om verschil in expressie te vinden.

***Figuur 5:*** *clustering van samples op basis van cel type.*

# Hierarchical clustering met een Heatmap

Een alternatieve voor de MDS plot is een hierarchical clustering met een heatmap om de relaties te bekijken tussen monsters. In dit geval wordt er gekeken naar de 500 meest variabele genen. Dit is geen vast gegeven en kan zelf gekozen worden. Eerst wordt er een variantie bepaald per gen.

var\_genes <- apply(CountsPM, 1, var)

Na de bepaling van deze variantie worden de namen van de eerste 500 genen bekend gemaakt.

select\_var <- names(sort(var\_genes, decreasing = TRUE))[1:500]

Na de naamgeving worden de logcounts verkregen van de meest variabele genen.

highly\_variable\_lcpm <- logcounts[select\_var,]

Van deze logcounts word een heatmap geplot na dat de kleuren voor behandeld (oranje) en de controle (paars) zijn aangewezen.

#Kleuren aanwijzen

mypalette <- brewer.pal(7, "RdYlBu")

morecols <- colorRampPalette(mypalette)

col.cell <- c("purple", "orange")[metadata$dex]

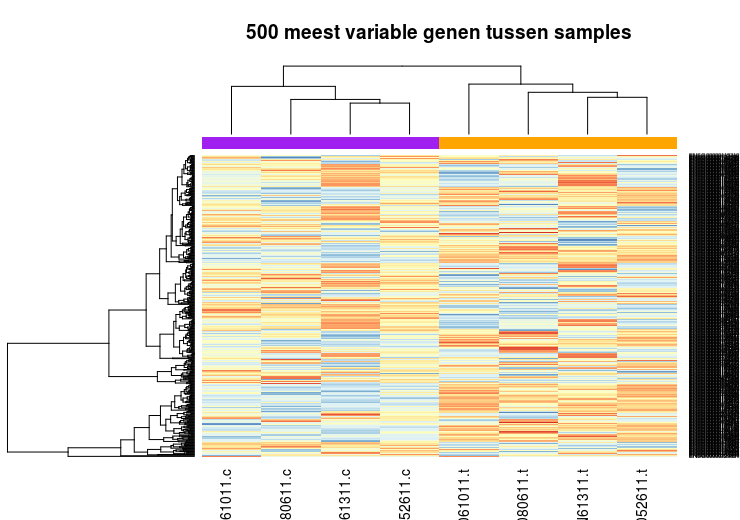
#Heatmap plotten

heatmap.2(highly\_variable\_lcpm,col=rev(morecols(50)),

trace="none", main="500 meest variable genen tussen samples",

ColSideColors=col.cell, scale='row')

Er is mogelijk een Error wat plaats vind in verband met dat de margins te groot zijn maar des al niet te min wordt de heatmap als nog betrouwbaar gemaakt. In figuur 6 is de heatmap zichtbaar van de 500 meest variabele genen tussen de samples van de behandelde versus de controle. Hier is te zien dat de controle en behandelde samples apart clusteren. De relatie tussen de controle samples is het zelfde als de relatie tussen de treated samples.



***Figuur 6:*** *heatmap van alle samples controle (paars) versus behandeld (oranje).*

Er wordt hierna nog een heatmap gemaakt waar de variabele genen tussen de cel type gezien kunnen worden. Dit wordt gedaan door de logcounts per million gemiddelde over de samples te berekenen van de logcounts per million van de controle samples te delen door de behandelde sampels voor elke cel type.

N61311 <- highly\_variable\_lcpm[,1]/highly\_variable\_lcpm[,2]

N052611 <- highly\_variable\_lcpm[,3]/highly\_variable\_lcpm[,4]

N080611 <- highly\_variable\_lcpm[,5]/highly\_variable\_lcpm[,6]

N061011 <- highly\_variable\_lcpm[,7]/highly\_variable\_lcpm[,8]

test = data.frame(N61311, N052611, N080611, N061011)

Van deze logcounts word een heatmap geplot na dat de kleuren voor de verschillende cel type aanwezig zijn.

#kleuren aanwijzen  
mypalette <- brewer.pal(3, "RdYlBu")

morecols <- colorRampPalette(mypalette)

col.cell <- c("purple", "orange", "green", "blue")[metadata$celltype]

col.cell <- c("blue", "purple", "green", "orange")

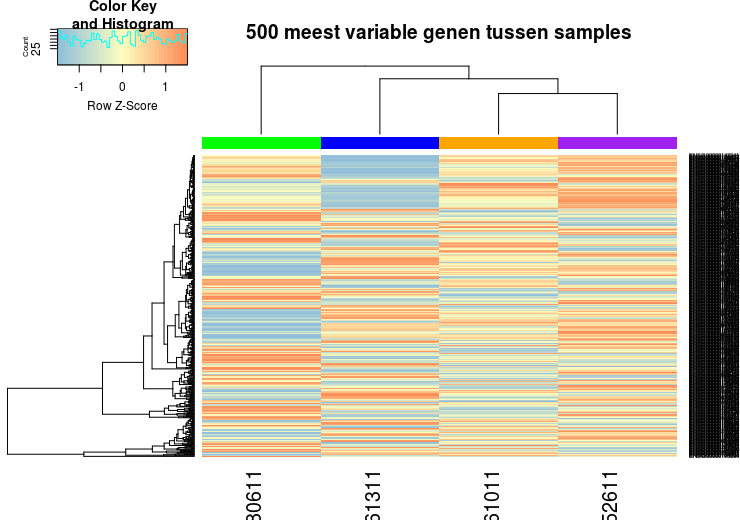
#Heatmap plotten

heatmap.2(as.matrix(test),col=rev(morecols(50)),

trace="none", main="500 meest variable genen tussen samples",

ColSideColors=col.cell, scale='row')

Ook hier kan een error plaats vinden over te grote margins, deze error is negeerbaar aangezien de heatmap nog betrouwbaar is. Zoals zichtbaar is in figuur 7 dat de cel type 61011 en 52611 het meest aan elkaar verwant zijn ten opzichte van de andere 2 cel type.



***Figuur 7:*** *heatmap van alle cel type*

# Normalisatie (voor compositie bias)

De normalisatie voor de compositie bias wordt uitgevoerd tussen libraries om de compositie bias te elimineren. Door de onderliggende uitkomsten van de behandelings- effect te combineren in een overal effect dit resulteerd in de compositie bias, dit genereert een set van normalisatie factors. Deze normalisatie factoren bepalen de effectieve library grote. Er is eerst genormaliseerd met calcNormFactors deze manier wordt gebruikt voor de meeste scalings normalisatie manieren

countsDGEnorm <- calcNormFactors(countsDGE)

countsDGEnorm$samples

countsDGE$samples

In onderstaande tabel 1 is zichtbaar wat de library size is en wat de normalisatie factoren zijn van dezelfde samples

***Tabel 1:*** *Overzicht tabel counts DGE*

|  |  |  |  |
| --- | --- | --- | --- |
| Sample name | group | Lib. size | Norm. factors |
| N61311.c | 1 | 20615750 | 1.0558949 |
| N61311.t | 1 | 18789410 | 1.0203974 |
| N052611.c | 1 | 25320612 | 0.9915741 |
| N052611.t | 1 | 15149527 | 0.9489665 |
| N080611.c | 1 | 24420766 | 1.0313658 |
| N080611.t | 1 | 30786651 | 0.9767417 |
| N061011.c | 1 | 19104845 | 1.0281090 |
| N061011.t | 1 | 21144737 | 0.9523625 |

Hierna wordt de grootste en kleinste normalisatie factor er uit gehaald.

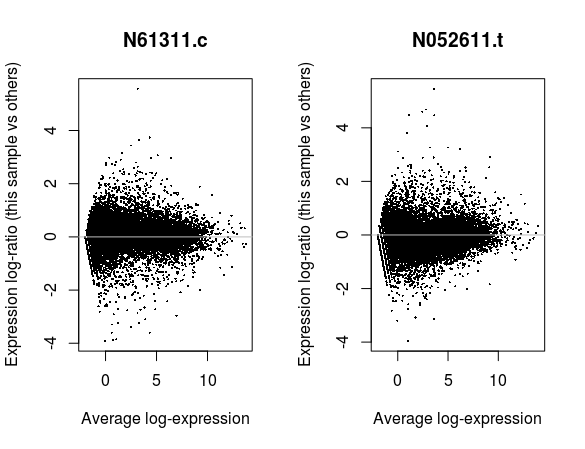
max(countsDGEnorm$samples[3]) # N61311.c

min(countsDGEnorm$samples[3]) # N052611.t

maxNum = which(colnames(logcounts)=="N61311.c")

minNum = which(colnames(logcounts)=="N052611.t")

Eerst wordt er een scatterplot gemaakt met de logcounts die nog niet genormaliseerd zijn voor de compositie bias maar wel voor de library grote zie figuur 8.



***figuur 8:*** *scatter plot van de kleinste en grootste libraries***.**

Na normalisatie van compositie bias zouden de punten van de scatter plot dichter bij de nul lijn moeten liggen zie figuur 9. De punten zijn niet veel veranderd vergeleken met de scatter plot zonder normalisatie, dit komt omdat de normalisatie factoren zeer klein zijn.

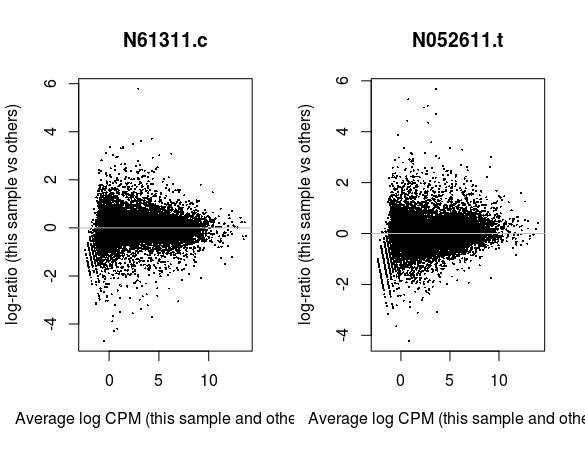
par(mfrow=c(1,2))

plotMD(countsDGEnorm, column=maxNum)

abline(h=0,col="grey")

plotMD(countsDGEnorm, column=minNum)

abline(h=0,col="grey")

***Figuur 9:*** *Scatter plot van grootste en kleinste libraries na normalisatie voor compositie bias.*

# DGE testing

Eerst maken we een design matrix om te bepalen welke samples we met elkaar willen vergelijken.

Daarna veranderen we de namen van de kolommen, zodat we weten over welke samples we het hebben.

subject <- factor(metadata$celltype)

treat <- factor(metadata$dex)

design <- model.matrix(~0+subject+treat)

colnames(design) <- c("N052611", "N061011", "N080611", "N61311", "treated")

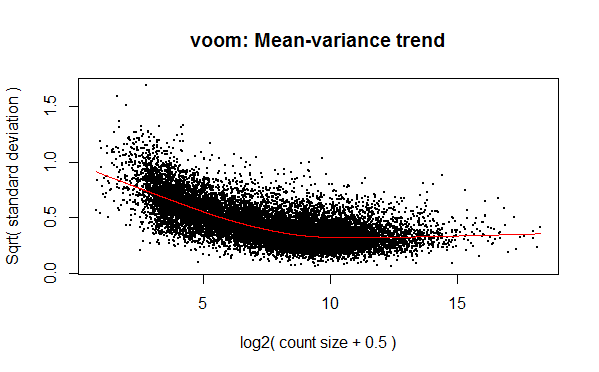
Er wordt een plot gemaakt m.b.v. Limma voom. Limma voom past de library grootes aan de normalisatie factoren die al zijn berekend in countsDGEnorm$samples. Voom transformatie geeft een Elist object met hulp van de design matrix. De gemiddelde variantie trend kan worden gevisualiseerd worden met een rode lijn. Die geeft aan of er genen zijn die variabel eruit zien in de data en of lage counts er correct uitgefilterd zijn.

par(mfrow=c(1,1))

v <- voom(countsDGEnorm, design, plot=TRUE)

v$design

In figuur 10 is de Limma voom plot te zien. En kan er uit geconcludeerd worden dat de lage counts er correct uitgefilterd zijn. Alks dat niet het geval was, dan was de trend door de punten meer gebogen.

***Figuur 10:*** *Limma voom plot*

Na de Limma voom is de differentiatie van de gen expressie getest. Door gebruik van de empirical Bayes shrinkage op de variabelen en verwachte moderate t- statistiek en de bij horende p-waarde.

fit <- lmFit(v)

names(fit)

fit <- eBayes(fit)

Hierna wordt er een samenvatting gemaakt van de DE genen voor het contrast.

results <- decideTests(fit)

summary(results)

Hierbij wordt de annotatie toegevoegd. Eerst worden alle genen er uit verwijderd die niet nodig zijn en gekeken of deze waarde TRUE zijn en naar de enzymen die significant aanwezig zijn.

head(annotatie)

useme = annotatie[annotatie$ensgene %in% rownames(fit),]

all(useme$ensgene==rownames(fit))

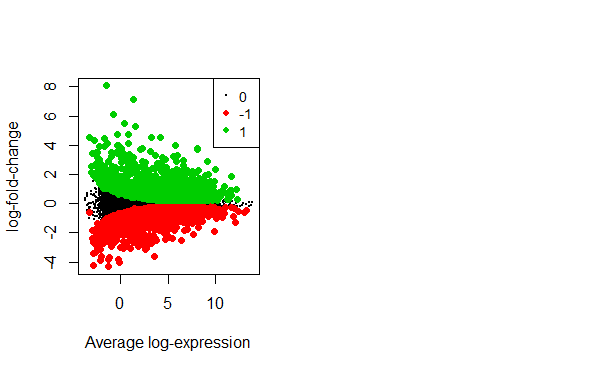
fit$genes <- useme

topTable(fit, coef="treated",sort.by="p")

Na de DGE testing wordt er een MA plot gemaakt om te bekijken welke genen up of down gereguleerd zijn. De top 100 significante genen worden hier in gehighlight zoals te zien is in figuur 11.

par(mfrow=c(1,2))

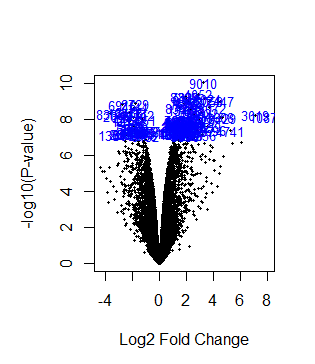
plotMD(fit,coef="treated",status=results[,"treated"], values = c(-1, 1))



***Figuur 11:*** *MA plot 100 significante genen in het groen de up gereguleerde genen en in het rood de down gereguleerde genen.*

Als aansluiting voor de MA plot wordt er een volkano gemaakt om naar significantie van genen te kijken. Zie figuur 12 een hogere y coordinaat geeft een hogere significantie aan. De blauwe labels geven de 100 significantste gene aan.

volcanoplot(fit,coef="treated",highlight=100,names=fit$genes$SYMBOL)



***Figuur 12:*** *Fold change plot*

We willen eigenlijk alleen maar genen overhouden die een verdubbeling van expressie hebben. Daarvoor kunnen we de treat en decideTest functie gebruiken. Een lfc (log fold change) van 1 wil zeggen dat we genen over willen houden die 2x zo veel tot uiting kunnen komen, of juist 2 keer zo weinig.

t.treat <- treat(fit,lfc=1)

res.treat <- decideTests(fit.treat)

summary(res.treat)

# Top genes from linear model fit

topTreat(fit.treat,coef=3)

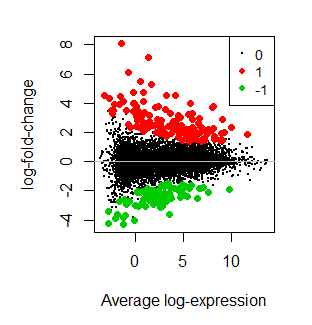
Zoals te zien is in figuur 13 is er veel ruis verwijderd in de plot vergeleken met figuur 11.

par(mfrow=c(1,2))

plotMD(fit,coef="treated",status=results[,"treated"], values = c(-1, 1))

plotMD(fit.treat,coef="treated",status=res.treat[,"treated"])

abline(h=0,col="grey")



***Figuur 13:*** *MA plot na ruis verwijdering.*

# GSEA/Pathway level analyse

GSEA is gen/eiwit klassen detecteren die onrepresentatief zijn in de samples, door middel van de pathway analyse worden de onrepresentatieve genen/eiwitten gekoppeld aan pathways om te kunnen zien of deze anders zijn. De 10 meest significante up gereguleerde eiwit anthologieën die gerelateerd zijn aan een biologisch proces worden bekeken. Hierbij hebben we rekening gehouden met de gen lengten en alleen gekeken naar van biologische processen (BP), omdat we daarin zijn geïnteresseerd.

gen\_lengten = fit$genes$end - fit$genes$start + 1

go\_length <- goana(fit, coef="treated",species = "Hs", covariate=gen\_lengten, geneid="entrez")

topGO(go\_length, ontology="BP", n=20)

Dit gaf de volgende pathways als resultaat.

# Term Ont N Up Down P.Up P.Down

# GO:0042221 response to chemical BP 2706 628 475 8.184667e-16 2.918390e-02

# GO:0007167 enzyme linked receptor protein signaling pathway BP 729 219 137 9.331050e-16 7.043901e-02

# GO:0050896 response to stimulus BP 5551 1172 987 1.101079e-15 2.297995e-04

# GO:0007169 transmembrane receptor protein tyrosine kinase signaling pathway BP 520 167 94 8.242853e-15 2.366843e-01

# GO:0007165 signal transduction BP 3726 829 694 9.704034e-15 4.191474e-05

# GO:0023052 signaling BP 4051 891 759 1.437951e-14 5.662438e-06

# GO:0007154 cell communication BP 4064 893 760 1.458908e-14 6.798134e-06

# GO:0070887 cellular response to chemical stimulus BP 2065 491 354 4.063544e-14 1.430072e-01

# GO:0051716 cellular response to stimulus BP 4702 1003 842 1.398039e-13 6.222644e-04

# GO:0006629 lipid metabolic process BP 996 267 169 2.232218e-13 3.289951e-01

# GO:0009888 tissue development BP 1204 312 247 3.955324e-13 7.882289e-05

# GO:0048513 animal organ development BP 2150 507 429 6.049820e-13 2.268003e-06

# GO:0044255 cellular lipid metabolic process BP 771 215 127 9.425349e-13 4.996529e-01

# GO:0008150 biological\_process BP 11472 2161 1899 9.841516e-13 3.442495e-02

# GO:0009653 anatomical structure morphogenesis BP 1747 427 347 1.903690e-12 1.444220e-04

# GO:0010033 response to organic substance BP 2125 490 368 5.365737e-12 8.365117e-02

# GO:0051179 localization BP 4417 937 731 6.479997e-12 4.362609e-01

# GO:0048856 anatomical structure development BP 3712 807 718 1.134552e-11 2.373758e-08

# GO:0006810 transport BP 3492 757 546 1.440822e-11 9.286914e-01

# GO:0003012 muscle system process BP 267 94 55 1.531733e-11 5.296110e-02

De onderzoeksvraag was: Wat is het effect van een behandeling op chronische inflammatie door middel van transscriptoom analyse? Het verdere onderzoek ging om de chromische inflammatie in verschillende weefsels aan te tonen met betrekken welke pathways daarbij een belangrijke rol in spelen. Het afweersysteem of immuunsysteem bestaat uit een groep cellen en weefsels die het lichaam beschermt tegen vreemde indringers zoals virussen, bacteriën en/of schimmels. Het immuunsysteem voorkomt dat de indringers schade aanbrengen. Als het afweersysteem niet alleen beschermt tegen pathogenen, maar ook lichaamseigen cellen aanvalt, dan gaat er wat mis. Psoriasis is daar een goed voorbeeld van. Reacties die te maken hebben met het reactie van het lichaam/weefsel/cel zijn dus interessant om te weten. Genen die differentieel tot expressie komen in de volgende pathways kunnen hier een interessante rol in spelen:

* GO:0042221 (response to chemical)
* GO:0007167 (enzyme linked receptor protein signaling pathway)
* GO:0050896 (response to stimulus)