

Project_logboek_Genexpressie

Mark van de Streek

2023-04-04

Contents

Gen expressie Logboek	1
Het onderzoek	2
Verdeling van de groepen	2
3. Verkenning van de data	2
3.1 Inladen van de data	3
3.2 Eerste kijk op data	4
3.3 Verdeling van de data	4
3.4 Normalisatie van de data en de afstand tussen de samples	7
4 Differentieel tot expressie gebrachte genen (DEG's)	10
4.1 Voorbewerking van de data	10
4.2 Fold Change Value (FC)	12
4.3 DEG analyse met behulp van R-paketten	13
5. Analyse en Visualisaties van de DEG's	16
5.1 Volcano plot	16
5.2 Venn Diagram	17
DAVID	18

Gen expressie Logboek

Het doel van dit logboek is om het onderzoek wat benoemd wordt te reproduceren. Verder wordt er gekeken of er nog meer biologische vragen beantwoordt kunnen worden, die de onderzoekers niet hebben behandeld.

Het onderzoek gaat over het roken van sigaretten en de elektronische sigaret, ook wel de vape genoemd. In het onderzoek is gebruikt gemaakt van RNA-seq data. Er is dus gekeken naar expressie van bepaalde genen. Om precies te zijn is er gekeken naar genen die aanwezig zijn in leukocyten (witte bloedcellen).

Het onderzoek

Er is gekeken naar de activiteit van genen van de volgende groepen mensen:

- Vapers (mensen die een elektronische sigaret roken)
- Mensen die de traditionele sigaret roken
- Mensen die geen tabaksproducten gebruiken

Om mensen te verkrijgen voor het onderzoek werd er om te beginnen een online vragenlijst gemaakt. Vervolgens werd er voor deze vragenlijst reclame gemaakt online, op bijvoorbeeld Reddit en Twitter. Aan de hand van de deze vragenlijst werden de mensen die geschikt leken voor het onderzoek uitgenodigd voor een gesprek. In dit gesprek werden nog een aantal vragen en checklists gemaakt om er zeker van te zijn dat de persoon geschikt is voor het onderzoek.

Uit de resultaten bleek dat de mensen die op dit moment Vapen, maar vroeger niet rookten, een significant andere genregulatie hebben ten opzichte van mensen uit de controle groep. Ook mitochondriale genen en immuunresponsgenen waren significant ontregeld.

Verdeling van de groepen

In alle verschillende groepen waren een aantal mensen die voor het onderzoek werden gebruikt. Onderstaand de aantallen in een kleine tabel.

Table 1: Aantal mensen per groep

Vapers	Traditionele rokers	Niet-rokers
37	22	23

Er werd bij de deelnemers ook gekeken naar geslacht, leeftijd, etniciteit, rassen en ook naar mensen die kunnen lezen en schrijven in het Engels. Het onderzoek vond plaats in Los Angeles.

De opzet van het onderzoek was om de verschillen tussen alle drie de groepen duidelijk weer te geven. In vele eerdere onderzoeken werden vaak deelnemers gekozen die vapen, maar vaak met een geschiedenis in andere rookgewoonten. Er namen dus mensen deel aan het onderzoek die eerder traditionele sigaretten rookten en op latere tijd zijn overgestapt naar elektronische sigaretten.

De groep die werd geclassificeerd als vapers in dit onderzoek heeft in de 6 maanden voor het onderzoek geen traditionele sigaret gebruikt. De onderzoekers hebben bewust voor een periode van ‘maar’ 6 maanden gekozen, omdat de elektronische sigaret nog niet zo heel lang bestaat. door het kiezen van een periode van 6 maanden, zijn er meer mensen die deel konden nemen aan het onderzoek. Voor de traditionele rokers groep werd een periode van 1 jaar gebruikt voor deelname.

Met behulp van bioinformatische bevindingen zijn in dit onderzoek onderscheid gemaakt tussen gezonde volwassen vapers, met en zonder geschiedenis van roken, en ‘exclusieve’ sigaretten rokers. Ook is specifiek de expressie van genen in rokers (zowel e-sigaret als traditionele rokers) vergeleken met niet-rokers.

3. Verkenning van de data

In de eerste sectie worden de verkennende data analyses uitgevoerd. Er wordt dus bijvoorbeeld gekeken naar de verdeling van de data, maar ook naar bijvoorbeeld de kwaliteit van de data. In het laatste gedeelte van deze sectie wordt een normalisatie uitgevoerd.

3.1 Inladen van de data

Onderstaand wordt de data van het bestand ingeladen. Helaas is er in het bestand nog niet duidelijk welke samples precies bij welke groep horen. Vandaar dat er nog een aantal stappen moeten worden uitgevoerd om de juiste namen bij de juiste samples te krijgen. De data betreft ongefilterde tellingen van de genexpressie.

```
data <- read.table(  
  "data/GSE169757_Partek_Hum_BC_RNA_Jan2018_UnfilteredCounts_82.txt",  
  header = T,  
  sep = "\t")  
  
# In de eerste kolom van het bestand staat alleen een index,  
# We willen hier de gennamen hebben  
row.names(data) <- data$gene_name  
  
Sys.setenv(VROOM_CONNECTION_SIZE=50007200)  
  
# Opzoeken van het ID en defineren in een object  
gse <- getGEO("GSE169757")  
  
## Found 1 file(s)  
  
## GSE169757_series_matrix.txt.gz  
  
## Rows: 0 Columns: 83-- Column specification -----  
## Delimiter: "\t"  
## chr (83): ID_REF, GSM5213644, GSM5213645, GSM5213646, GSM5213647, GSM5213648...  
## i Use 'spec()' to retrieve the full column specification for this data.  
## i Specify the column types or set 'show_col_types = FALSE' to quiet this message.File stored at:  
## /tmp/Rtmp1JVLvx/GPL18573.soft  
  
# Defineren van de specifieke kolom met alle namen  
gse_groups_column <- gse[[1]]@phenoData@data$characteristics_ch1.1  
  
# Lege vector om de uiteindelijke namen in op te slaan  
all_modified_group_names <- c()  
  
# Het door lopen van alle kolonnamen en het opzoeken van de namen.  
# Allereerst worden de indexen verkregen van alle namen  
# Met die indexen worden de juiste namen eruit gehaald.  
for (name in names(data[,9:ncol(data)])) {  
  name <- gsub("\\.", "-", name)  
  all_group_names <- gse_groups_column[which(gse[[1]]@phenoData@data$title %in% name)]  
  group_names <- strsplit(all_group_names, ": ")[[1]][2]  
  all_modified_group_names <- c(all_modified_group_names, group_names)  
}  
  
# Uniek maken door een getal erachter te zetten.  
unique_group_names <- make.unique(all_modified_group_names, sep = "_")  
  
# Het veranderen van de kolom namen in de data  
names(data)[9:ncol(data)] <- unique_group_names
```

3.2 Eerste kijk op data

Nu de data goed is ingeladen met alle juiste namen kunnen we kijken hoe het precies er uit ziet. Ook kunnen we kijken wat het precieze aantal rijen en kolommen zijn, ook wel de dimenties.

```
pander(head(data[1:4, c(1, 2, 8, 9, 10, 11)]))
```

Table 2: Table continues below

	Chromosome	Start	havana_gene	Control	Smoker
5S_rRNA	4	67439992	—	2	2.602
7SK	6	5.3e+07	—	466	73.82
A1BG	19	58345178	OTTHUMG00000183507.2	324.4	96.5
A1BG-AS1	19	58347751	OTTHUMG00000183508.1	286.5	81.92

	Control_1
5S_rRNA	5
7SK	357.3
A1BG	258.9
A1BG-AS1	181.3

In de data zit de volgende informatie: chromosome, start- en stop-positie, streng, gen symbool/gen naam, havana gen en natuurlijk de samples. Van elke sample is dus meer informatie op te zoeken dan simpelweg alleen de telwaarde. De havana gen kolom is een kolom die unieke waardes bevat die verwijzen naar een specifiek gen, dat op een specifieke locatie op het menselijk genoom ligt. De waardes zijn geannoteerd door het HAVANA-project, om genen, exonen, etc. op het menselijk genoom te identificeren.

```
dims <- dim(data)
sprintf("Aantal rijen: %.f en aantal kolommen: %.f", dims[1], dims[2])
```

```
## [1] "Aantal rijen: 38921 en aantal kolommen: 90"
```

We weten nu hoe de data eruit ziet en hoe groot het precies is. Vanaf kolom nummer negen beginnen de ‘samples’. De samples staan dus niet onder elkaar, maar naast elkaar.

Classificatie van variabelen Omdat we niet de hele tijd willen opzoeken welke kolomnummers bij welke groep hoeren, gaan we dit defineren. Op deze manier kunnen we makkelijk een variabel invullen om vervolgens snel alle data van de bijhorende groep te krijgen.

```
smoker <- grep("Smoker", names(data))
vaper <- grep("Vaper", names(data))
control <- grep("Control", names(data))
```

Grep() geeft de nummers van kolommen terug waar het woord in voorkomt.

3.3 Verdeling van de data

De groepen zijn nu geklassificeerd en we kunnen dus kijken hoe de data is verdeeld. We kunnen het duidelijk weergeven op twee manieren. Onderstaand worden er twee figuren gemaakt om te kijken hoe de data verdeeld is; een boxplot en een dichtheidsplot.

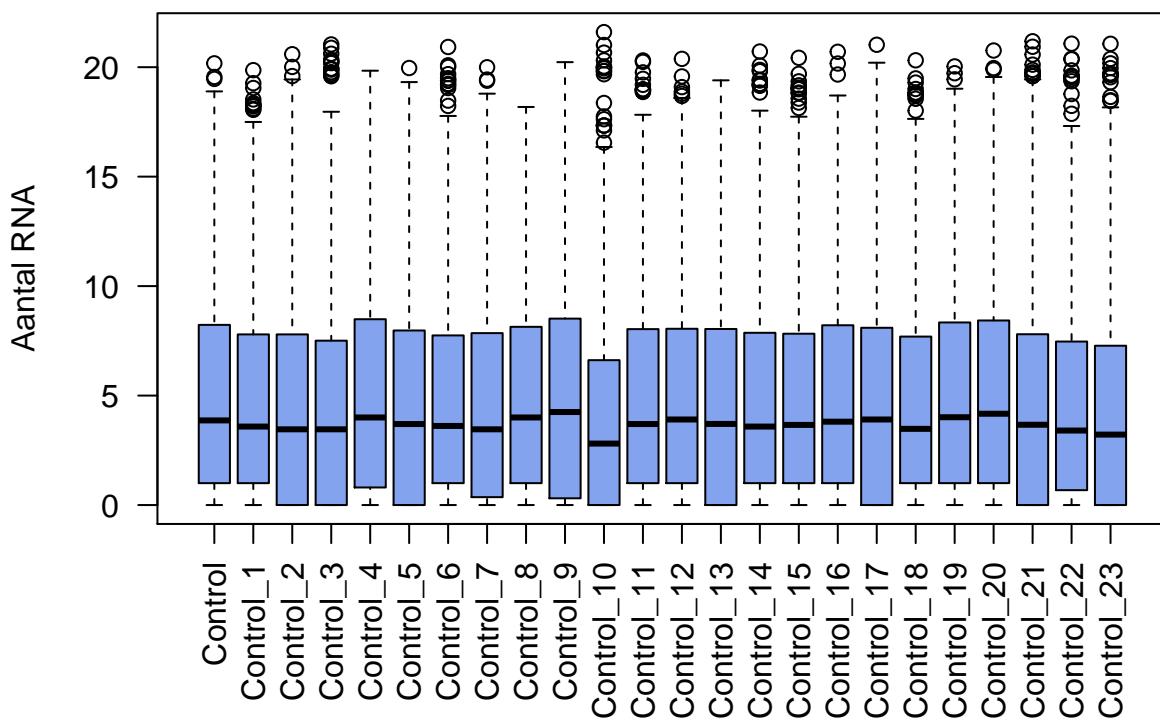
3.3.1 Boxplot Allereerst een boxplot. Als de boxen van de figuren allemaal rond een lijn zitten, is de data goed dicht bij elkaar. Een probleem van de ‘ruwe’ dat is de hoeveelheid nullen. Deze nullen ‘trekken’ de andere waarden behoorlijk naar beneden. De boxplot heeft geen details meer en zegt dan niks meer.

Om dit probleem op te lossen, moeten we alle waarden met 0 ‘wegfilteren’. Dit is makkelijk te doen met een log2 transformatie. Het bereik tussen de samples wordt op deze manier kleiner. De waarden met 0 domineren minder in de data.

Onderstaand de boxplot van de controle groep, met getransformeerde data.

```
boxplot(log2(data[control] + 1), las = 2, col = "#83A3EE",
        main = "log2 transformatie van de verdeling van de control groep",
        ylab = "Aantal RNA")
```

log2 transformatie van de verdeling van de control groep

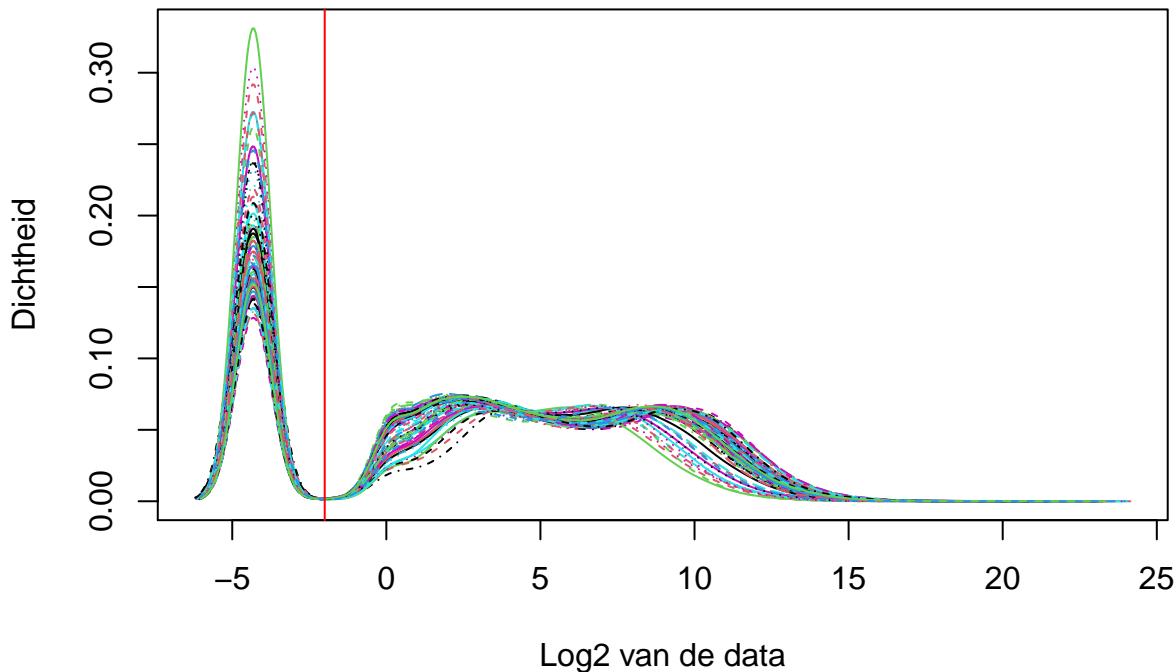


Zoals je kunt zien op het figuur liggen alle boxen behoorlijk dicht bij elkaar. Je zou dus zeggen dat deze data goed is verdeeld. Echter is dit nog maar een simpele log2 transformatie en kunnen we dus nog verder kijken naar normalisatie. Uiteraard kun je ook niet een conclusie trekken van één figuur.

3.3.2 Dichtheidsplot Een andere manier om uitschieters in de data weer te geven is een dichtheids grafiek. Dit plot geeft ook de verdeling van de data weer. Als alle lijnen van de grafiek op elkaar liggen, is de data goed verdeeld. Eventuele uitschieters zijn goed zichtbaar. We willen dus in de grafiek zo min mogelijk lijnen zien die afwijken van de andere lijnen.

```
plotDensity(log2(data[9:ncol(data)] + 0.05),
            main = "Dichtheidsplot van alle data samples",
            ylab = "Dichtheid",
            xlab = "Log2 van de data")
abline(v = -2, col = "red", lwd = 1, lty = 1)
```

Dichtheidsplot van alle data samples

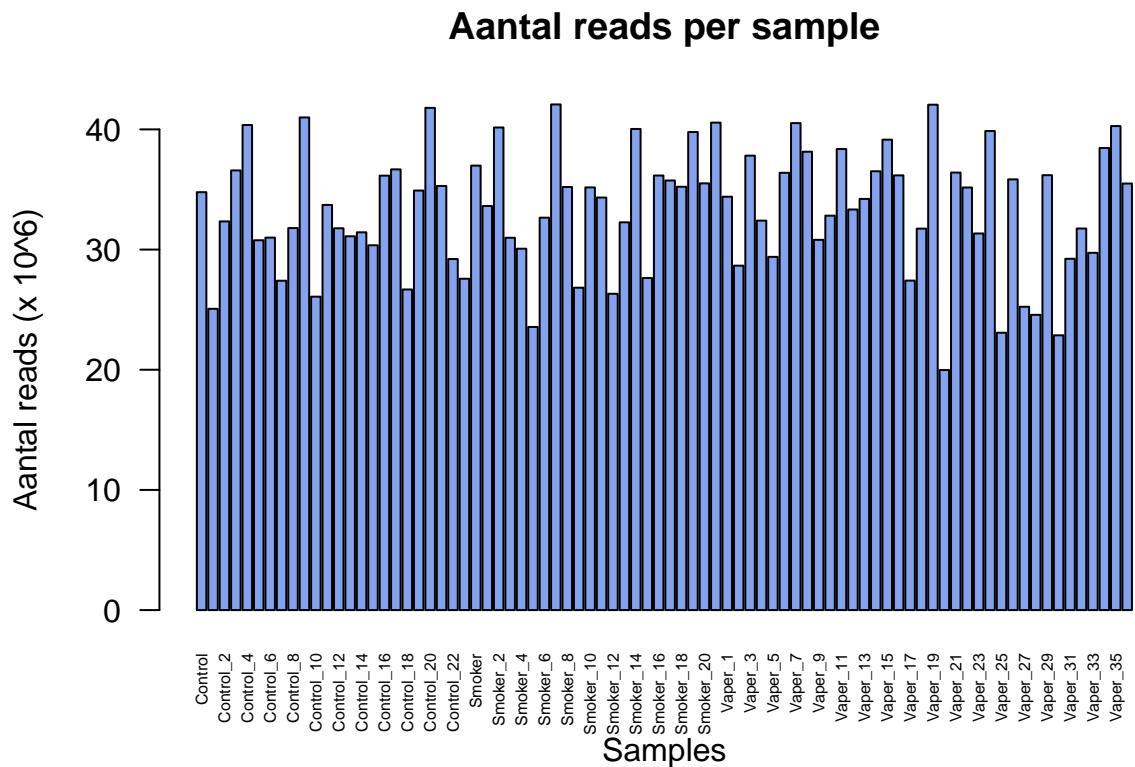


Zoals je kunt zien zit er een hele groot piek links van de rode lijn. Zoals eerder opgemerkt bevat de data heel veel nullen. De data is getransformeerd. De log2 van 0.05 is ~ -4.3 . Deze piek zijn dus alle waarden met nul en kunnen genegeerd worden (daarom is er om die reden ook een rode lijn geplaatst).

In de figuur is op twee plekken een duidelijk ander verloop te zien. Aan het begin en aan het einde van de stijging is andere verloop tussen de samples. Hier is de data dus niet helemaal gelijk verdeeld. Na het normaliseren van de data is het mogelijk om eventuele afwijkende samples weg te gooien, om deze uitschieters eruit te filteren.

3.3.3 Staafdiagram Een laatste methode die we gebruiken om te kijken is een staafdiagram van het aantal reads. We gebruiken de functie colSums om alle waarden op te tellen. Het doel van dit figuur is het prijsgeven van het aantal reads per sample. Het zegt dus iets over de dekking van de data.

```
barplot(colSums(data[c(control, smoker, vaper)]) / 1e6,
        main = "Aantal reads per sample",
        xlab = "Samples",
        ylab = "Aantal reads (x 10^6)",
        las = 2,
        cex.names = .5,
        col = "#83A3EE")
```



3.4 Normalisatie van de data en de afstand tussen de samples

Om te kijken of de data ver uit elkaar ligt, maken we twee figuren. We berekenen de afstanden tussen de datapunten en plotten dit.

Om zoveel mogelijk ruis uit de data te halen, wordt de data genormaliseerd. De gegevens worden aangepast om er voor te zorgen dat ze gelijk zijn. Onderstaand zal er een simpele normalisatie worden uitgevoerd met het DESeq2 pakket. De meest basale vorm van normalisatie wordt toegepast. Er wordt onder andere genormaliseerd op basis van gen-lengte.

```
# Maken van het DESEQ object
ddsMat <- DESeqDataSetFromMatrix(countData = floor(data[c(control, smoker, vaper)]),
  colData = data.frame(samples = names(data[c(control, smoker, vaper]))),
  design = ~ 1)
```

```
## converting counts to integer mode
```

```
# Uitvoeren van de normalisatie
rld.dds <- vst(ddsMat)
# Waarden ophalen
rld <- assay(rld.dds)
```

3.4.1 Afstanden weergeven met heatmap Nu de data genormaliseerd is, kunnen de afstanden berekend worden tussen de samples om aan te tonen hoe dicht de data bij elkaar ligt. Een mooie manier om de afstanden weer te geven is een heatmap. In een heatmap zijn de waarden die dicht bij elkaar liggen rood en de waarden die verder weg liggen blauw. De heatmap wordt gemaakt met de pheatmap package.

```

# Berekenen van de afstanden
sampledists <- dist(t(rld))
sampleDistMatrix <- as.matrix(sampledists)

# Het maken van een dataframe die gebruikt kan worden om te annoteren
group <- factor(c(rep(1, length(control)),
                     rep(2, length(smoker)),
                     rep(3, length(vaper))),
                     labels = c("Control", "Smoker", "Vaper"))

annotation <- data.frame(group = group)
rownames(annotation) <- names(data[c(control, smoker, vaper)])

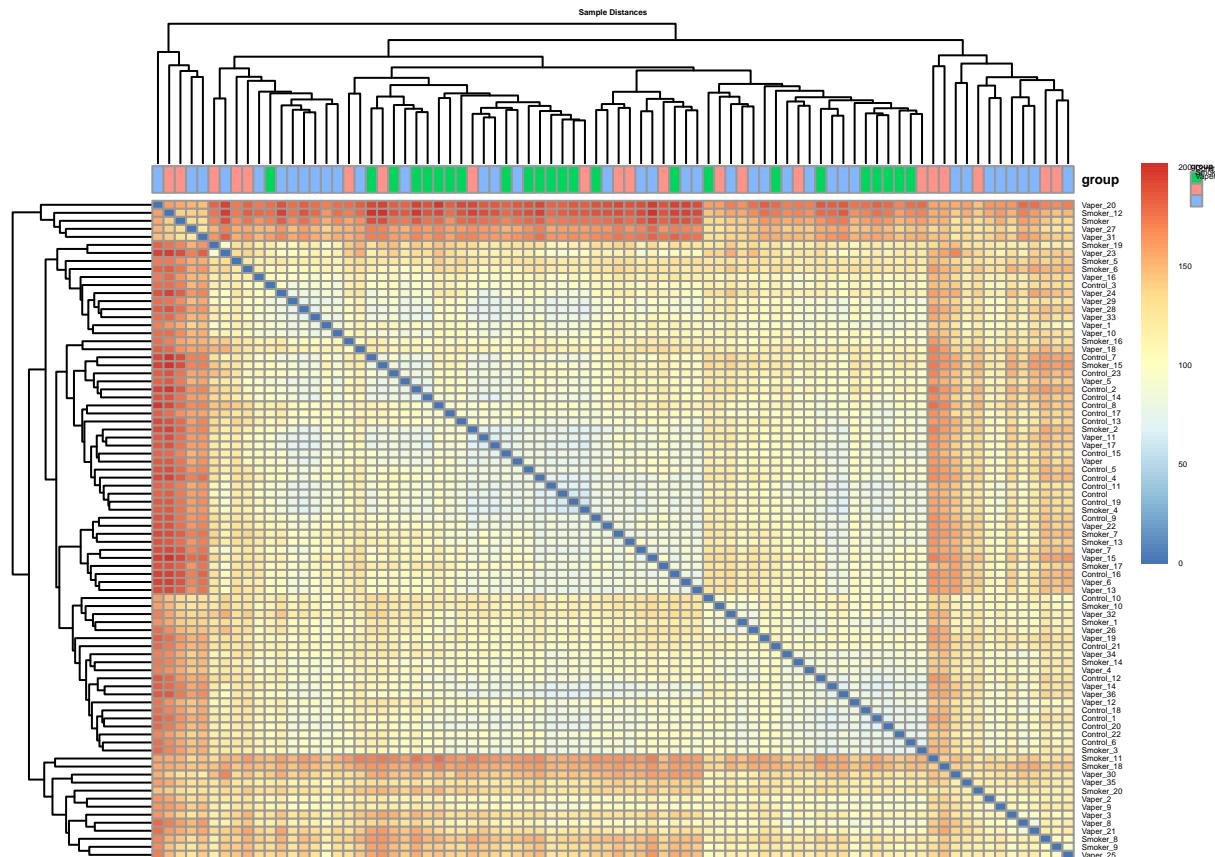
```



```

# Het maken van de heatmap
pheatmap(sampleDistMatrix, show_colnames = FALSE,
clustering_distance_rows = sampledists,
clustering_distance_cols = sampledists,
annotation_col = annotation,
cex = .5,
main = "Sample Distances")

```



Door de vele aanwezige samples is de heatmap zeer lastig af te lezen. Wel kun je mooi zien dat de eerste vier samples verder weg liggen dan de andere samples. Het is echter lastig om hier meer over te vertellen, omdat er zoveel samples aanwezig zijn.

3.4.2 Multi-dimensionale schaling (MDS) Op de data kunnen we ook multi-dimensionale schaling (MDS) toepassen. Dit is vergelijkbaar met de heatmap, maar hierbij worden de afstanden op een andere manier berekend. Bij MDS wordt Poisson afstandsberekening toegepast. Er wordt gekeken welke genen bij elkaar horen. De genen die bij elkaar horen, vormen een groepje in de grafiek.

Allereerst berekenen we de afstanden met de Poisson methode.

```
dds <- assay(ddsMat)
# Afstand berekenen
poisd <- PoissonDistance( t(dds), type = "deseq")

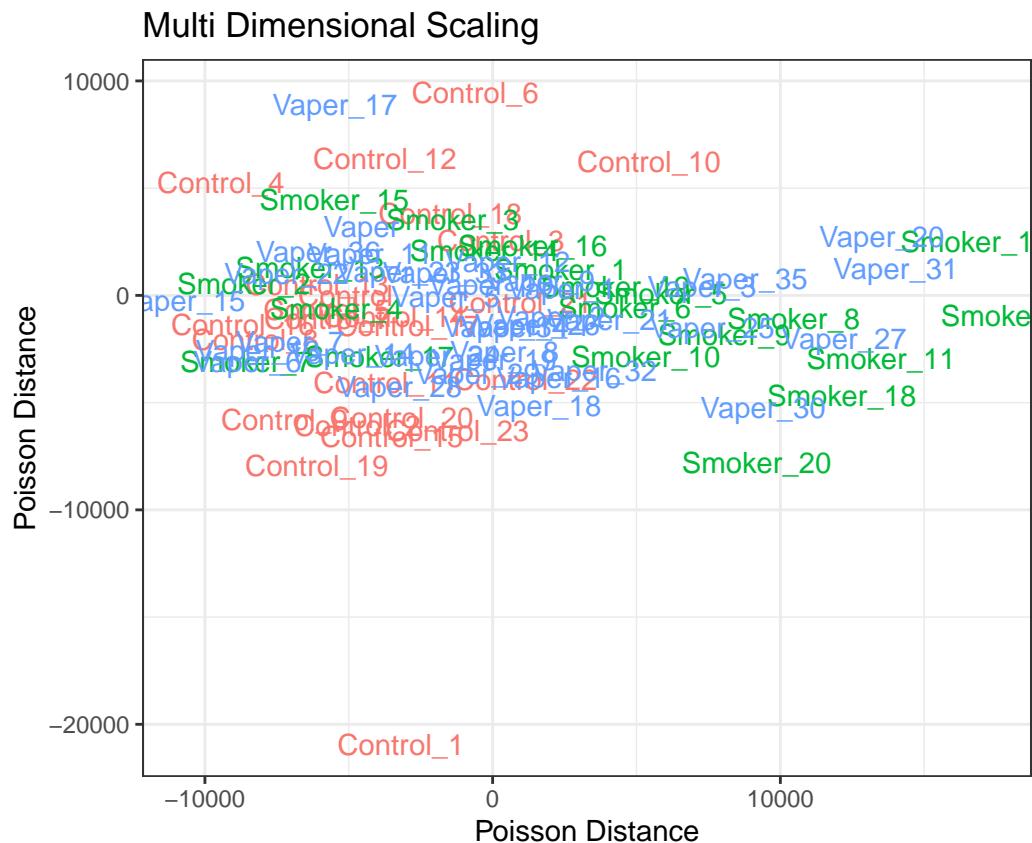
# De waarden ophalen uit de matrix
samplePoisDistMatrix <- as.matrix(poisd$dd)

# Het berekenen van de x- en y-coördinaten
mdsPoisData <- data.frame( cmdscale(samplePoisDistMatrix) )

# Namen aanpassen voor het beter aflezen
names(mdsPoisData) <- c('x_coord', 'y_coord')
```

En natuurlijk het maken van de grafiek.

```
coldata <- names(data[c(control, smoker, vaper)])
ggplot(mdsPoisData, aes(x_coord, y_coord, color = group, label = coldata)) +
  geom_text(size = 4) +
  ggtitle('Multi Dimensional Scaling') +
  labs(x = "Poisson Distance", y = "Poisson Distance") +
  theme_bw()
```



Zoals je kunt zien zijn er heel veel samples die verspreid zijn. Er zijn niet echt groepen. Toch kun je duidelijk zien dat de groepen vaper en control samen een groep vormen met blauw en groen. Er zijn van de rokers groep wel een aantal samples die ook in de groep vallen, maar dat zijn er niet veel. In de rokers groep zijn een aantal individuele punten die ver liggen van de andere waarden. Dit kan bruikbare data zijn om aan te tonen dat een gen/sample een specifieke afwijking heeft in genexpressie.

4 Differentieel tot expressie gebrachte genen (DEG's)

In deze sectie gaan we kijken naar DEG's. Een DEG is een gen die met expressie afwijkt van normale genen. Een DEG is afhankelijk van de Log fold change waarde (LFC) en de p-waarde. Voordat we alle gegevens meteen in het pakket gooien, voeren we eerst ‘handmatig’ een aantal stappen uit.

4.1 Voorbewerking van de data

Het uiteindelijke doel is om genen te vinden die afwijken in expressie van de ‘normale’ genen. Om dit aan te tonen, voeren we statistische testen uit. Echter zijn de testen heel erg gevoelig voor buitenliggende waarden. Om deze reden voeren we nog een normalisatie uit, maar dan op een andere manier.

4.1.1 Normalisatie op basis van reads We gaan de data normaliseren door middel van de counts per miljoen. Het aantal counts wordt berekend per miljoen. Deze data transformeren we vervolgens met de log2. Eerder keken we nog naar de lengte van de genen, maar nu kijken we puur alleen naar het aantal counts.

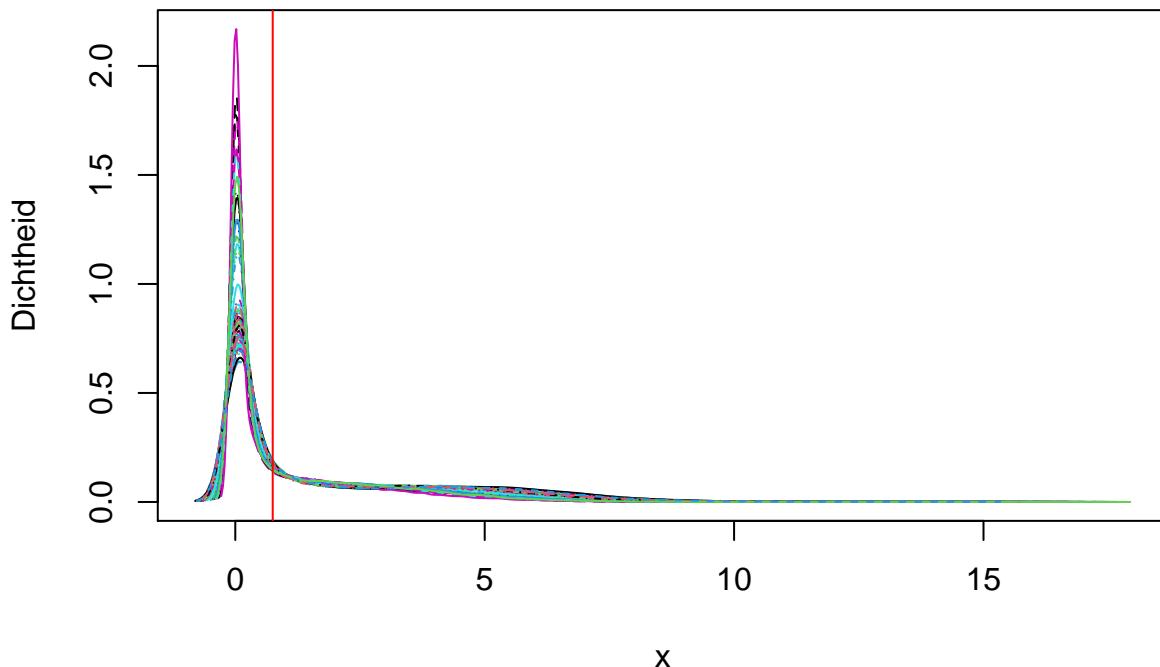
```
counts.fpm <- log2( (data[c(control, smoker, vaper)]) / (colSums(data[c(control, smoker, vaper)])) / 1e6)
plotDensity(counts.fpm,
main = "Dichtheidsplot van alle data samples",
```

```

ylab = "Dichtheid")
abline(v = .75, col = "red", lwd = 1, lty = 1)

```

Dichtheidsplot van alle data samples



de log2 functie kan geen ‘overige’ waarden transformeren. De eerste negen kolommen worden dus niet meegegeven, alleen de ‘echte’ data.

In vergelijking met het vorige dichtheidsplot, lijkt deze dat nu beter verdeeld te zijn. De meeste lijnen liggen bijna allemaal op één lijn. De normalisatiemethode is echter wel een beetje ‘kaal’, omdat het zoals eerder vermeld alleen maar rekening houdt met het aantal reads.

4.1.2 Wegfilteren van de nullen in de dataset Zoals eerder vermeld zijn er heel veel nullen aanwezig in de dataset. Normaal gesproken gebruik je een pakket in R die deze waarden wegfilterd. Sommige paketten hebben echter deze mogelijkheid niet. Voor nu gaan we onze eigen manier bedenken/onderzoeken.

In het onderstaande blok wordt de data gefilterd en zullen de verschillen aangetoond worden.

```

new_data <- data[rowSums(data[9:ncol(data)] > 5) <= ncol(data[9:ncol(data)]) * 0.25,]
new_data_dims <- dim(new_data)

cat(sprintf("Data is gefilterd,
            Oude dataset: %.f regels.
            Nieuwe data: %.f regels.
            Aantal regels weggefilterd: %.f",
            dims[[1]], new_data_dims[[1]], dims[[1]] - new_data_dims[[1]]))

## Data is gefilterd,
##          Oude dataset: 38921 regels.
##          Nieuwe data: 14576 regels.
##          Aantal regels weggefilterd: 24345

```

De genen moeten een expressiewaarde van hoger dan 5 hebben. Dit moet gelden voor 75% van de samples. Er zijn heel veel waarden onder de 5 aanwezig in de dataset. Daarom zijn er dus ook veel regels weggefilterd in de ‘nieuwe’ dataset.

4.2 Fold Change Value (FC)

De FC waarde is de log₂ waarde van de experiment/controle ratio. Deze waarde geeft simpelweg aan of een bepaald gen een hogere of lagere expressie heeft. Om te ver komen dat de data erg wordt beïnvloed door ruis gebruik je de log₂ waarde van de ratio. Dit is de Fold Change Value. Bij een FC-waarde van boven de 1, spreek je van een hogere expressie. Bij een FC-waarde tussen de 0 en 1, spreek je van een mindere expressie.

Hieronder wordt de Fold Change berekend van de genen in de dataset. Hiervoor wordt de FPM-data gebruikt (zie hierboven). Doordat de eerder genormaliseerde data een iets andere volgorde heeft, worden de kolommen voor de groepen opnieuw gedefineert.

```
# Het normaliseren van de gefilterde data
new_data.counts.fpm <- log2( (new_data[c(control, smoker, vaper)] / (colSums(new_data[c(control, smoker, vaper)]))) )

# Het berekenen van de ratios
vaper.means <- rowMeans(new_data.counts.fpm[grep("Vaper", names(new_data.counts.fpm))])
control.means <- rowMeans(new_data.counts.fpm[grep("Control", names(new_data.counts.fpm))])
FC <- vaper.means - control.means
```

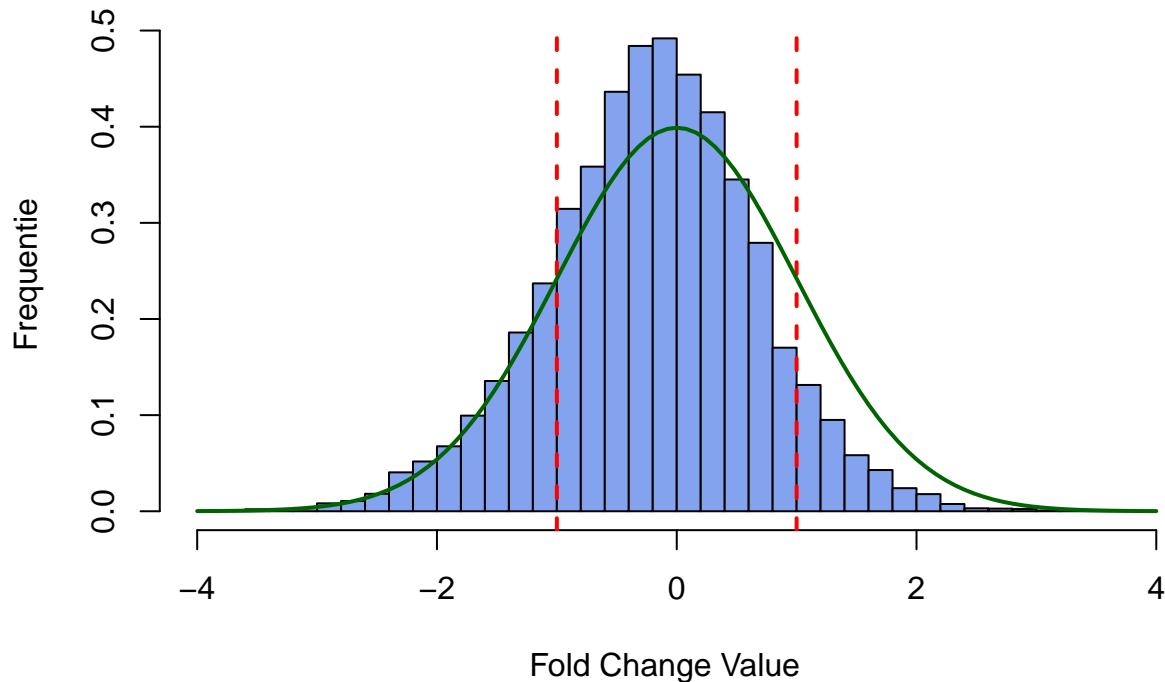
Het plotten van de LFC-waarde. Ook wordt er een normaalverdeling toegevoegd in het figuur. Je wil dat de LFC-waarde normaal verdeeld is, omdat de statistische testen zijn op aannames van normaliteit van de gegevens.

```
hist(FC,
     breaks = 40,
     main = "Histogram van de Fold Change Value (Vaper - Control)",
     xlab = "Fold Change Value",
     ylab = "Frequentie",
     col = "#83A3EE",
     freq = F)

# Lijn toevoegen bij -1 en 1
abline(v = -1, col = "red", lty = 2, lwd = 2)
abline(v = 1, col = "red", lty = 2, lwd = 2)

# Het plotten van de normaalverdeling.
# Allereerst datapunten maken tussen -4 en 4
x <- seq(-4, 4, by = 0.1)
curve(dnorm(x, 0, 1), from = -4, to = 4, col="darkgreen", add=T, lwd = 2)
```

Histogram van de Fold Change Value (Vaper – Control)



De groene lijn is de normaalverdeling. Zoals je kunt zien is de Fold Change redelijk normaal verdeeld op deze data.

4.3 DEG analyse met behulp van R-pakketten

Nu er duidelijk is welke stappen we naar op zoek zijn, kunnen we beginnen met het zoeken naar de DEG's. Het gebruiken hiervoor de DESeq module. Deze module hebben we ook al eerder gebruikt voor de heatmap en mds-plot. Het annotation dataframe wat we voor de heatmap hebben gebruikt kunnen we nu weer gebruiken. Het is wel belangrijk dat de controle groep als eerste voorkomt in het dataframe.

Om te beginnen maken we een DESeq object en voeren we hierop de testen uit. Omdat het experiment (maar) drie groepen bevat, voeren we ze allebei uit. Roker tegen controle en vaper tegen controle. Voor de visualisatie is het belangrijk om nog een lfcshrink toe te passen. Deze functie haalt de ruis weg.

```
# Maken van het object
dds <- DESeqDataSetFromMatrix(countData = floor(data[c(control, smoker, vaper)]),
                                colData = annotation,
                                design = ~ group)

## converting counts to integer mode

dds <- DESeq(dds)

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates
```

```

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

## -- replacing outliers and refitting for 503 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)

## estimating dispersions

## fitting model and testing

# Deze functie geeft terug welke resultaten er mogelijk zijn met de dataset
resultsNames(dds)

```

```

## [1] "Intercept"           "group_Smoker_vs_Control"
## [3] "group_Vaper_vs_Control"

```

Het object is aangemaakt en er is bekend welke testen er uitgevoerd kunnen worden. De twee resultaten kunnen dus gekoppeld worden aan objecten.

```

res.smoker.control <- results(dds, name = "group_Smoker_vs_Control")
res.vaper.control <- results(dds, name = "group_Vaper_vs_Control")

```

Voor het maken van de grafieken is het belangrijk om de ruis die komt van de log2 waarden van lfc-waarden, wordt weggefilterd. Dit wordt gedaan met de lfcShrink functie.

```
res.smoker.control <- lfcShrink(dds, coef = "group_Smoker_vs_Control", type = "apeglm")
```

```

## using 'apeglm' for LFC shrinkage. If used in published research, please cite:
##   Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
##   sequence count data: removing the noise and preserving large differences.
##   Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895

```

```
res.vaper.control <- lfcShrink(dds, coef = "group_Vaper_vs_Control", type = "apeglm")
```

```

## using 'apeglm' for LFC shrinkage. If used in published research, please cite:
##   Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
##   sequence count data: removing the noise and preserving large differences.
##   Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895

```

Met de summary functie wordt weergegeven hoeveel DEG's er zijn gevonden. Ook worden hier de percentages bijgegeven.

```
summary(res.smoker.control)
```

```

## 
## out of 38909 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)      : 4511, 12%
## LFC < 0 (down)    : 1676, 4.3%
## outliers [1]       : 0, 0%
## low counts [2]     : 9064, 23%
## (mean count < 1)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results

```

```
summary(res.vaper.control)
```

```

## 
## out of 38909 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)      : 1604, 4.1%
## LFC < 0 (down)    : 342, 0.88%
## outliers [1]       : 0, 0%
## low counts [2]     : 15099, 39%
## (mean count < 4)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results

```

Het is nu mogelijk om een MA plot te maken van de objecten.

```

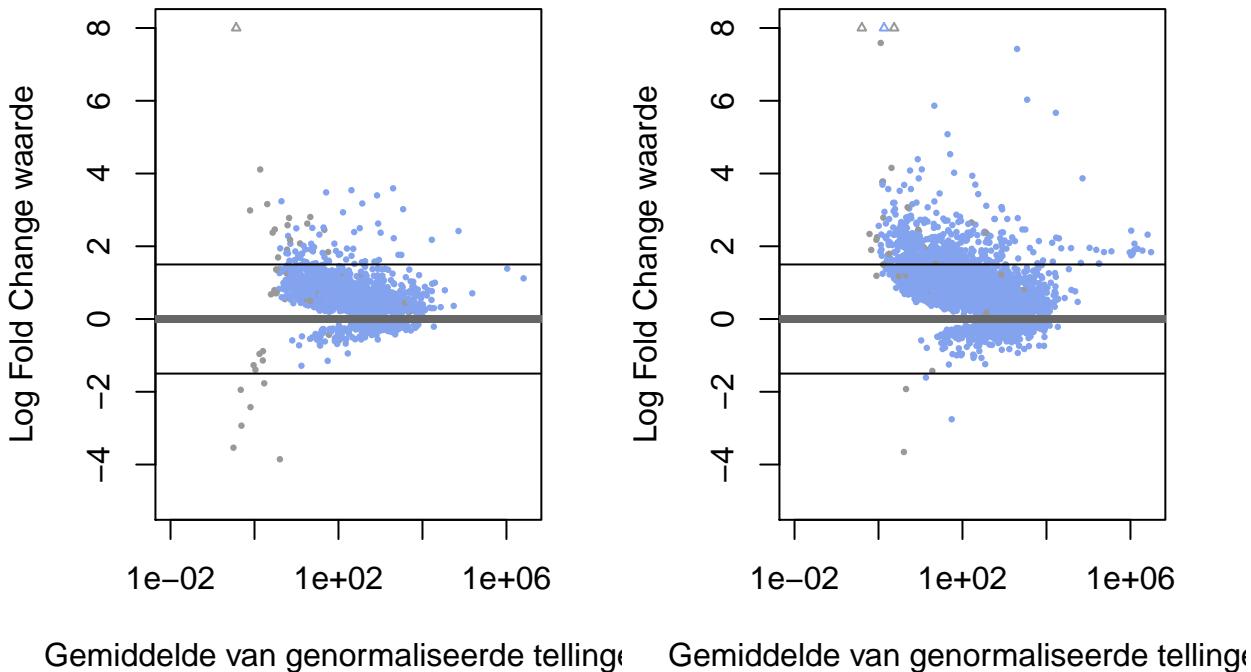
# Plots naast elkaar zetten met par()
par(mfrow = c(1, 2))

# Ma plots maken inclusief de LFC-lijnen bij -1.5 en 1.5
plotMA(res.vaper.control,
       xlab = "Gemiddelde van genormaliseerde tellingen",
       ylab = "Log Fold Change waarde",
       ylim = c(-5, 8),
       colSig = "#83A3EE",
       alpha = 0.1)
abline(h = c(-1.5, 1.5))

plotMA(res.smoker.control,
       xlab = "Gemiddelde van genormaliseerde tellingen",
       ylab = "Log Fold Change waarde",
       ylim = c(-5, 8),
       colSig = "#83A3EE",
       alpha = 0.1)
abline(h = c(-1.5, 1.5))
mtext("DEG's van de groep vaper vs controle (links) en de groep smoker vs controle (rechts)",
      side = 3,
      line = -1,
      outer = T)

```

DEG's van de groep vaper vs controle (links) en de groep smoker vs controle (rechts)



Wel goed om op te merken is dat er in het originele onderzoek gebruik is gemaakt van een LFC-threshold van 1.5. Deze kan niet worden toegevoegd aan de test. Om de grafiek wat ‘duidelijker’ te maken zijn er twee lijnen geplot. Een lijn bij -1.5 en 1.5. Alle blauwe stippen die onder/boven deze lijnen liggen kunnen worden beschouwd als DEG. Zoals je kunt zien zijn er heel veel genen met een significantie die net onder de grens van 1.5 LFC liggen. Deze genen worden door het MA-plot nog geclasseerd als DEG, maar dit is voor het onderzoek niet het geval.

In de volgende sectie zullen we een volcano plot behandelen. In dit plot kunnen we wel een grens opgeven voor de LFC. Op deze manier wordt er duidelijker welke genen nou echt DEG's zijn (volgens het originele onderzoek). Zonder deze grens zijn er 5000 DEG's bij de Rokers groep. Dit op een totaal aantal van ~ 38000 genen. Er worden standaard dus erg veel genen gekenmerkt als DEG en het is dus belangrijk om hier nog een extra ‘grens’ voor toe te voegen. Met deze grens kunnen de genen worden getoond die uitschieten.

5. Analyse en Visualisaties van de DEG's

Nu er een set met DEG's is, kunnen we visualiseren, d.w.z. figuren maken/namaken en we kunnen ook biologische vragen gaan beantwoorden/conclusies trekken.

5.1 Volcano plot

Allereerst gaan we een volcano plot maken. Zoals hierboven beschreven kan je in een volcano plot zowel de p-waarde als de LFC-waarde weergeven. Het doel is om vergelijkbare aantal DEG's te krijgen die in het originele onderzoek ook aanwezig waren.

```
# Definieren van volcano plot functie
deseq.volcano <- function(res, datasetName) {
  return(EnhancedVolcano(res, x = 'log2FoldChange', y = 'padj',
```

```

    lab=rownames(res),
    title = paste(datasetName, ""),
    subtitle = bquote(italic('FDR <= 0.1 and absolute FC >= 1.5')),
    labSize = 3, pointSize = 1.5, axisLabSize=10, titleLabSize=12,
    subtitleLabSize=8, captionLabSize=10,
    legendLabels=c('Niet sig.',
                  'Log (base 2) FC',
                  'p-waarde',
                  'p-waarde & Log (base 2) FC'),
    legendPosition = 'right',
    legendLabSize = 6,
    legendIconSize = 2,
    pCutoff = 0.1, FCcutoff = 1.5))}
```

```

# Het maken van de Volcano grafieken
#deseq.volcano(res = res.vaper.control, datasetName = "De genen van de groep vaper in vergelijking met
#deseq.volcano(res = res.smoker.control, datasetName = "De genen van de groep smoker in vergelijking me
```

Nog in te vullen.....

5.2 Venn Diagram

Uitleg geven over venn diagram....

```

# P-waarde (FDR) van 0.1
pval_threshold <- 0.1
# In het originele onderzoek is er geen log fold change gebruikt,
# maar een fold change. Dus om terug te gaan, log2(1.5)
lfc_threshold <- log2(1.5)

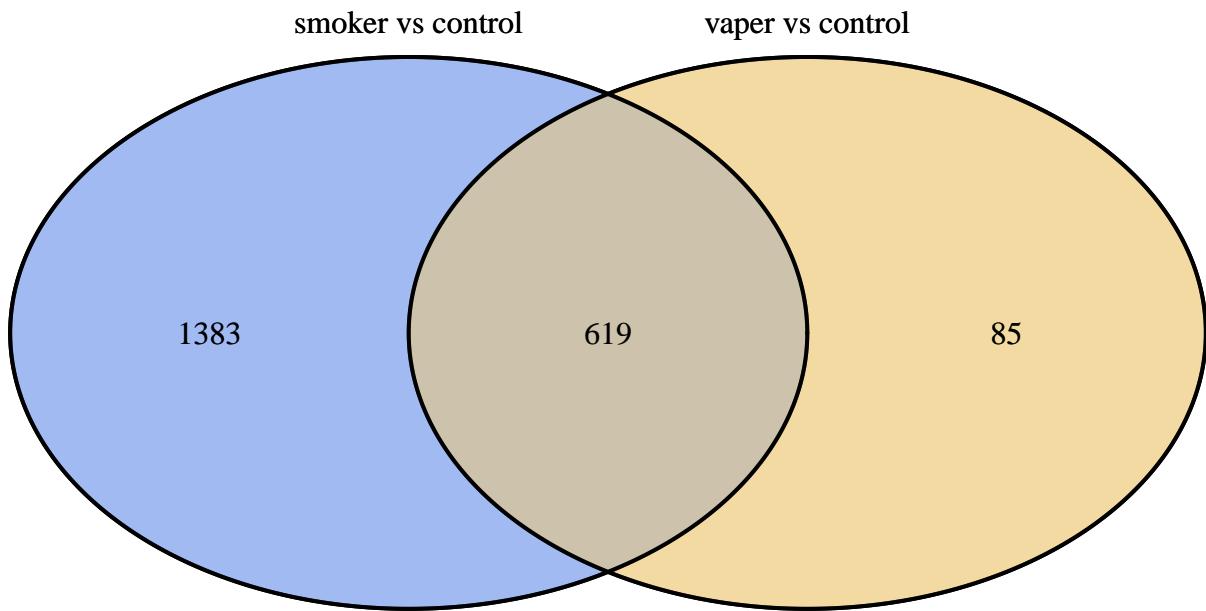
vaper.degs <- row.names(res.vaper.control)[which(res.vaper.control$padj <= pval_threshold & abs(res.vape
smoker.degs <- row.names(res.smoker.control)[which(res.smoker.control$padj <= pval_threshold & abs(res.sm
```

Invullen....

```

venn.plot <- draw.pairwise.venn(length(vaper.degs),
                                length(smoker.degs),
                                length( intersect(vaper.degs, smoker.degs) ),
                                category = c("vaper vs control", "smoker vs control"),
                                scaled = F,
                                fill = c("#eece83", "#83A3EE"),
                                alpha = rep(0.5, 2),
                                cat.pos = c(0, 0))

# Plotten van de diadram
grid.draw(venn.plot)
```



```
# data$Gene.Symbol[data$gene_name %in% vaper.degs]

vaper.degs.write <- sapply(vaper.degs, tools::file_path_sans_ext)
smoker.degs.write <- sapply(smoker.degs, tools::file_path_sans_ext)

write.table(vaper.degs.write, file = "vaper.degs.txt", row.names = F, quote = F, col.names = F)
write.table(smoker.degs.write, file = "smoker.degs.txt", row.names = F, quote = F, col.names = F)
```

DAVID

Nog invullen...

```
DAVID_genes <- read.csv("data/chart_31BB3D290B901680533944143.txt", sep = "\t", header = T)

# genes <- c("MT-TG", "MT-TR", "MT-TY", "MT-TI", "MT-TV", "MT-TH",
#           "MT-TN", "LARS", "CARS2", "MT-TK", "MT-TL1", "MT-TM",
#           "MT-TW", "MT-TA", "NARS2", "MT-TQ", "MT-TL2", "MT-TC",
#           "MT-TS2")

# vaper.degs.lfc <- c()
#
# for (gene in genes) {
#   lfc <- res.vaper.control$log2FoldChange[res.vaper.control@rownames == gene]
#   vaper.degs.lfc <- c(vaper.degs.lfc, lfc)
# }

# vaper.degs.lfc <- data.frame(LogFoldChange = vaper.degs.lfc)

# pathwayview(gene.data=vaper.degs.lfc,
#             pathway.id="00970",
#             species="hsa")
```

```

DAVID.vaper.terms <- DAVID_genes$Term[DAVID_genes$Category %in% "KEGG_PATHWAY"]
DAVID.vaper.terms.split <- c()

for (term in DAVID.vaper.terms) {
  splitted <- strsplit(term, ":" )[1][2]
  DAVID.vaper.terms.split <- c(DAVID.vaper.terms.split, splitted)
}

x <- DAVID_genes$PValue[DAVID_genes$Category %in% "KEGG_PATHWAY"]

barplot(-log10(x) ~ DAVID.vaper.terms.split, las = 2, cex.names = 0.5)

```

