

Nasonia

Kimberly Olney & Heini Natri

06/11/2020

Loading libraries

```
# Install libraries for the first time using BiocManager:  
# if (!requireNamespace("BiocManager", quietly = TRUE))  
#   install.packages("BiocManager")  
# BiocManager::install()  
# source("https://bioconductor.org/biocLite.R")  
# BiocInstaller::biocLite("DESeq")  
  
# load libraries  
library(RColorBrewer)  
library(matrixStats)  
library(ggplot2)  
library(edgeR)  
library(limma)  
library(doParallel)  
library(variancePartition)  
library(clusterProfiler)  
library(GOSemSim)  
library(biomaRt)  
library(VennDiagram)  
library(ggrepel)  
library(dplyr)  
library(stringr)  
library(forcats)  
library("knitr")
```

Setting the environment parameters.

```
# package knitr is required to set the working directory  
require("knitr")  
# set the working directory  
opts_knit$set(root.dir = "~/Dropbox (ASU)/NASONIA_take3/DE_limmaVoom")  
  
# Defining colors  
viralPalette <- brewer.pal(8, "Set2")  
VV_Color <- viralPalette[1] # green  
GG_Color <- viralPalette[2] # orange  
VG_Color <- viralPalette[3] # blue  
GV_Color <- viralPalette[4] # pink
```

```

# Defining shapes
VV_Shape <- c(15) # square
GG_Shape <- c(16) # circle
VG_Shape <- c(17) # triangle
GV_Shape <- c(18) # diamond

```

Importing count and pheno type data to create DGEList object

```

# reading in expression data
# gene count information for each sample
# each column is a sample
# each row is the raw count (expression) for that gene
counts_pseudoNgir <- read.table("wilson_counts_VgirRef.txt",
                                 header = TRUE,
                                 sep = "\t")
counts_Nvit <-
  read.table("wilson_counts.txt", header = TRUE, sep = "\t")
DF <- data.frame(counts_Nvit, counts_pseudoNgir)
colnames = c(
  "X014444",
  "X014445",
  "X014446",
  "X014447",
  "X014448",
  "X014449",
  "X014450",
  "X014451",
  "X014452",
  "X014453",
  "X014454",
  "X014455"
)
counts <-
  sapply(colnames, function(x)
    rowMeans(DF[, grep(x, names(DF))]))

```

`head(counts) # inspect the file`

```

##      X014444 X014445 X014446 X014447 X014448 X014449 X014450 X014451 X014452
## [1,]     12    9.0    6.0    9.0    7.0   12.0    44    8.0   11.0
## [2,]     10    9.0    9.0    0.0    1.5    1.5     0    1.0    6.0
## [3,]      3    8.0    4.0    0.0    0.0    0.0     0    1.0    1.0
## [4,]    462   468.0   414.0   646.5   520.5   712.0    596   370.5   561.5
## [5,]     55   31.0   30.0   12.0   19.5   16.5    43   22.0   25.0
## [6,]   1362  1484.5  959.5  2172.0  1013.5  1419.0   1273  1198.5  1396.5
##      X014453 X014454 X014455
## [1,]    9.0   15.0   19.0
## [2,]    0.0    5.0    1.0
## [3,]    1.0    1.0    1.0
## [4,]   399.0   651.0   640.5
## [5,]   10.0    29.5   22.0
## [6,]  1242.5  1763.5  1483.5

```

```

genes <- read.table("genes.txt", header = TRUE, sep = "\t")
# the gene file contains information about the genes
# Geneid, Chr, Start, End, Length
head(genes)

##           Geneid      Chr Start End Strand Length
## 1 gene-LOC100679504 NC_045757.1 6142 8476 - 2335
## 2 gene-LOC116415970 NC_045757.1 9540 13384 - 3845
## 3 gene-LOC103315360 NC_045757.1 16491 18228 + 1738
## 4 gene-LOC100117425 NC_045757.1 54820 61373 - 6554
## 5 gene-LOC100678203 NC_045757.1 63374 65444 + 2071
## 6 gene-LOC100117470 NC_045757.1 65587 69067 + 3481

pheno <- read.table("wilson_pheno.csv", header = TRUE, sep = ",")
# the pheno file contains information about the samples
# sampleID,Strain
head(pheno)

```

```

##   sampleID Strain
## 1 14444    VV
## 2 14445    VV
## 3 14446    VV
## 4 14447    GG
## 5 14448    GG
## 6 14449    GG

```

```

# define colors and shapes for the species
viralPalette <- brewer.pal(8, "Set2")
VV_Color <- viralPalette[1]
GG_Color <- viralPalette[2]
VG_Color <- viralPalette[3]
GV_Color <- viralPalette[4]

```

```

VV_Shape <- c(15)
GG_Shape <- c(16)
VG_Shape <- c(17)
GV_Shape <- c(18)

```

```

# Create the DGEList object using the counts and genes
dge <- DGEList(counts = counts, genes = genes)
dge$samples$strain <- pheno$Strain
table(dge$samples$strain) # Inspecting the N of samples in each group

```

```

##
## GG GV VG VV
## 3 3 3 3

```

Filtering the expression data

Normalization of counts using RPKM and FPKM reads/fragments per kilobase of exon per million reads/fragments mapped gene count normalization using gene length accounting for gene length is necessary for comparing expression between different genes within the same sample.

```

# Filtering expression data
fpkm <- rpkm(dge, gene.length = dge$genes$Length)

```

```

dim(dge$genes) # N of genes before filtering

## [1] 15259      6

# mean fpkm for each strain
VV_mean_fpkm <- apply(as.data.frame(fpkm)
                       [(dge$samples$strain == "VV")],
                       1, mean, na.rm = TRUE)
GG_mean_fpkm <- apply(as.data.frame(fpkm)
                       [(dge$samples$strain == "GG")],
                       1, mean, na.rm = TRUE)
VG_mean_fpkm <- apply(as.data.frame(fpkm)
                       [(dge$samples$strain == "VG")],
                       1, mean, na.rm = TRUE)
GV_mean_fpkm <- apply(as.data.frame(fpkm)
                       [(dge$samples$strain == "GV")],
                       1, mean, na.rm = TRUE)

# Filtering expression data
# the mean fpkm in each strain must be
# greater than 0.5 to be considered expressed
# and kept for downstream analysis
keep <- (VV_mean_fpkm > 0.5 | GG_mean_fpkm > 0.5 |
          VG_mean_fpkm > 0.5 | GV_mean_fpkm > 0.5)

dge <- dge[keep, , keep.lib.sizes = FALSE]
dge <- calcNormFactors(dge, method = "TMM")
keep <- rowSums(dge$counts > 6) >= 2
dge <- dge[keep, , keep.lib.size = FALSE]
dge <- calcNormFactors(dge, method = "TMM")

# N of genes retained after filtering
dim(dge$genes)

```

```
## [1] 10737      6
```

voom transformation

```

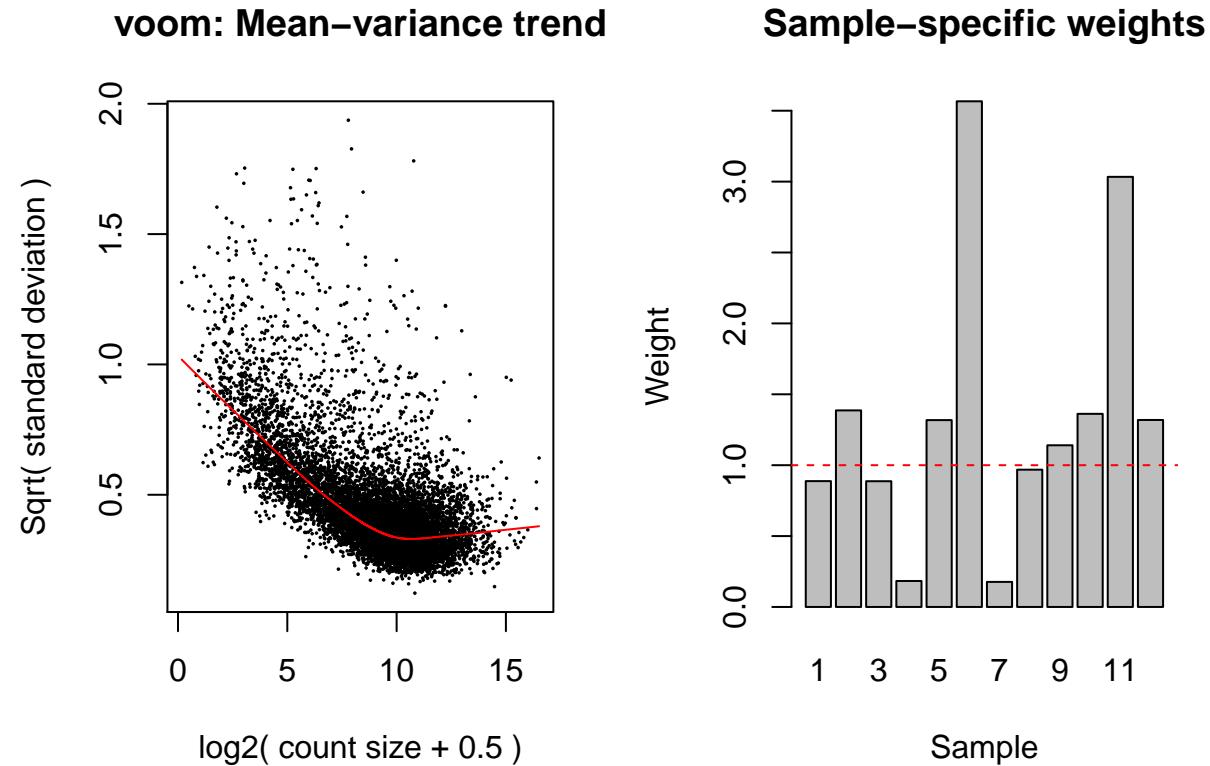
# create a design model matrix with the variable of interest
design <- model.matrix(~ 0 + dge$samples$strain)
colnames(design) <-
  gsub("dge\\\$samples\\\$strain", "", colnames(design))
head(design)

##   GG  GV  VG  VV
## 1  0  0  0  1
## 2  0  0  0  1
## 3  0  0  0  1
## 4  1  0  0  0
## 5  1  0  0  0
## 6  1  0  0  0

# run voom with quality weights.
# normalize expression intensities so that the log-ratios

```

```
# have similar distributions across a set of samples.
v <- voomWithQualityWeights(dge, design, plot = TRUE)
```



```
# to quantile normalize, add normalize.method="quantile"
```

Multi-dimensional scaling plot

To save figures as png, uncomment all png(filename) and dev.off()

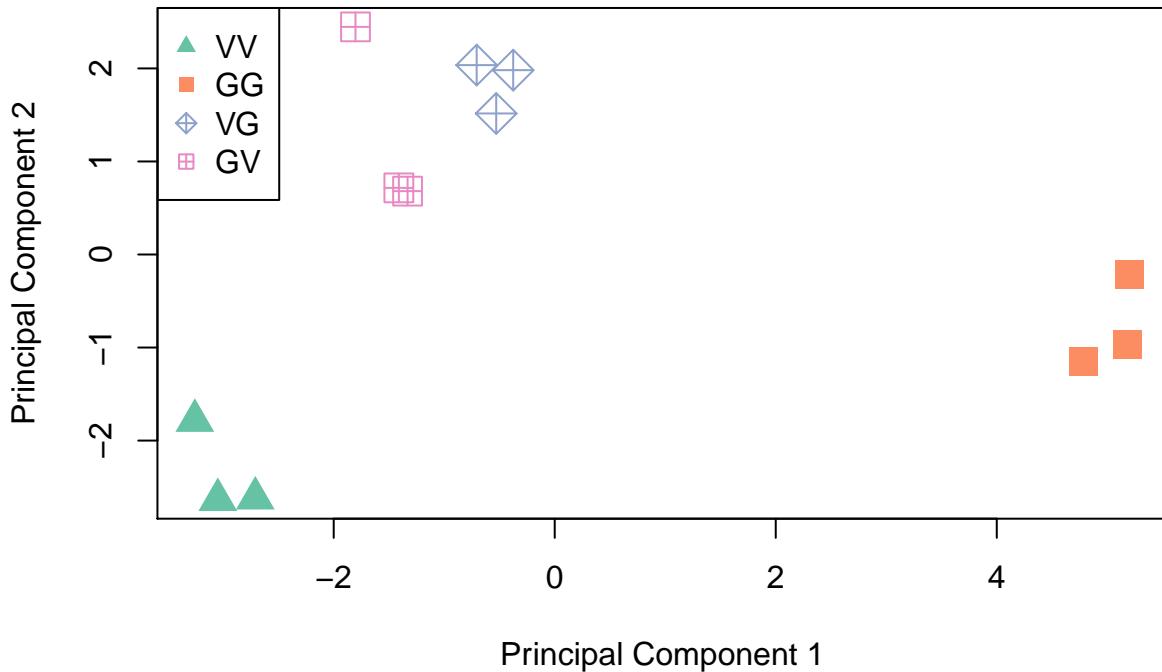
```
# MDS plot of the first two dimensions using top 50 genes.
# The dimensions are in the mds object.
# MDS plot of the first two dimensions using top50 genes. The dimensions are in the mds object.
#png(filename = "figures/wilson_AvgREFs_mds_1and2.png",
#     width = 450,
#     height = 450)
mds <-
  plotMDS(
    v,
    top = 50,
    ndim = 10,
    dim.plot = c(1, 2),
    plot = TRUE,
    cex = 2,
    pch = ifelse(
      v$targets$strain %in% c("VV"),
      17,
      ifelse(
        v$targets$strain %in% c("GG"),
        15,
```

```

        ifelse(v$targets$strain %in% c("VG"), 9, 12)
    )
),
col = ifelse(
  v$targets$strain == "VV",
  VV_Color,
  ifelse(
    v$targets$strain == "GG",
    GG_Color,
    ifelse(v$targets$strain == "VG",
      VG_Color, GV_Color)
  )
),
gene.selection = "common"
)

# Adding legends
legend(
  "topleft",
  pch = c(17, 15, 9, 12),
  col = c(VV_Color, GG_Color, VG_Color, GV_Color),
  legend = c("VV", "GG", "VG", "GV")
)

```



```

# dev.off()
# dev.off()
# MDS plot of the first two dimensions using top50 genes. The dimensions are in the mds object.
#png(filename = "figures/wilson_AvgREFs_mds_2and3.png",
#     width = 450,
#     height = 450)
mds <-
plotMDS(

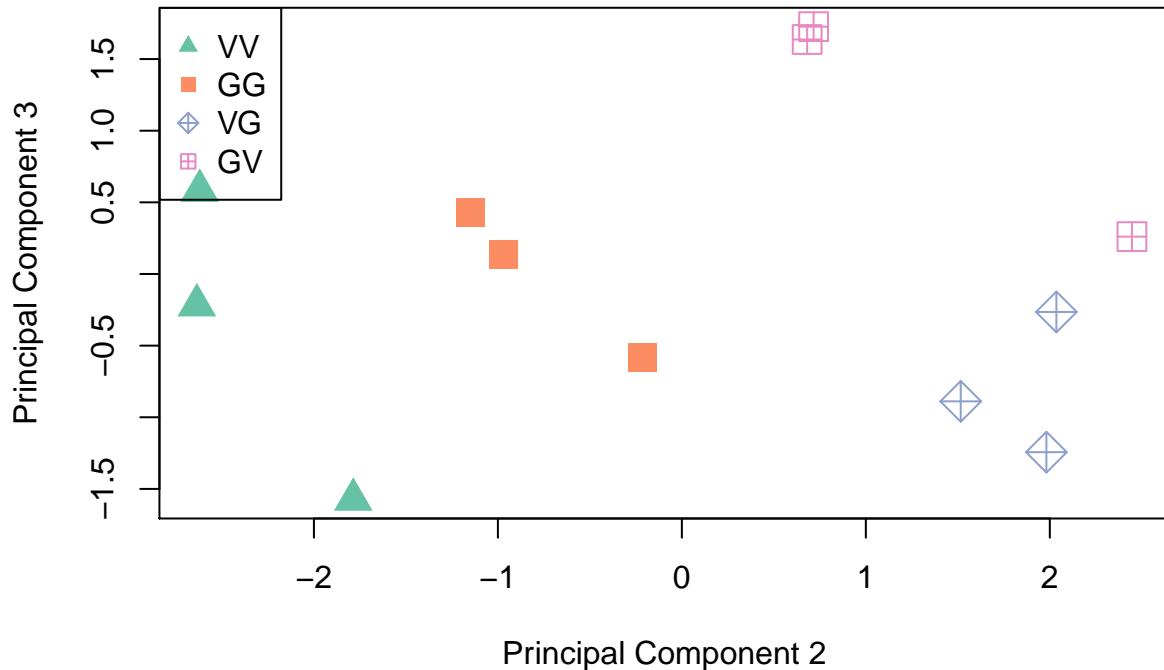
```

```

v,
top = 50,
ndim = 10,
dim.plot = c(2, 3),
plot = TRUE,
cex = 2,
pch = ifelse(
  v$targets$strain %in% c("VV"),
  17,
  ifelse(
    v$targets$strain %in% c("GG"),
    15,
    ifelse(v$targets$strain %in% c("VG"), 9, 12)
  )
),
col = ifelse(
  v$targets$strain == "VV",
  VV_Color,
  ifelse(
    v$targets$strain == "GG",
    GG_Color,
    ifelse(v$targets$strain == "VG",
      VG_Color, GV_Color)
  )
),
gene.selection = "common"
)

# Adding legends
legend(
  "topleft",
  pch = c(17, 15, 9, 12),
  col = c(VV_Color, GG_Color, VG_Color, GV_Color),
  legend = c("VV", "GG", "VG", "GV")
)

```



```
# dev.off()
# dev.off()
```

PCA on all/the most variable genes

Select most variable genes based on the biological coefficient of variance

```
# Voom transformed counts
# Voom transformed counts
voomCounts <- v$E
voomCountsMatrix <- data.matrix(voomCounts, rownames.force = NA)

# Setting the N of genes to use
ntop = length(dge$genes$Geneid)

# Sorting by the coefficient of variance
means <- rowMeans(voomCountsMatrix)
Pvars <- rowVars(voomCountsMatrix)
cv2 <- Pvars / means ^ 2
select <-
  order(cv2, decreasing = TRUE)[seq_len(min(ntop, length(cv2)))]
head(select)

## [1] 226 7861 752 90 6552 5493

highly_variable_exp <- ((voomCountsMatrix)[select,])
dim(highly_variable_exp)

## [1] 10737 12

# Running PCA
pca_exp <- prcomp(t(highly_variable_exp), scale = F, center = T)
# scale a logical value indicating whether the variables should be scaled
# to have unit variance before the analysis takes place.
```

```

# a logical value indicating whether the variables should be shifted to be zero centered.
head(pca_exp$x) [, 1:3]

##          PC1       PC2       PC3
## X014444 -74.33269 23.43750 12.396915
## X014445 -70.78432 43.53518  7.561028
## X014446 -67.25900 45.58106 -3.847198
## X014447  92.10340 17.07399 16.738050
## X014448  82.44734 29.79583 -14.606503
## X014449  91.73104 24.19623 -3.533677

summary(pca_exp)

## Importance of components:
##          PC1       PC2       PC3       PC4       PC5       PC6
## Standard deviation   59.8741 34.7929 24.82739 18.93885 15.84645 11.07168
## Proportion of Variance 0.5481 0.1851 0.09425 0.05484 0.03839 0.01874
## Cumulative Proportion 0.5481 0.7332 0.82748 0.88232 0.92072 0.93946
##          PC7       PC8       PC9       PC10      PC11      PC12
## Standard deviation   10.4518 9.57889 8.69706 8.01891 7.41708 8.469e-14
## Proportion of Variance 0.0167 0.01403 0.01157 0.00983 0.00841 0.000e+00
## Cumulative Proportion 0.9562 0.97019 0.98176 0.99159 1.00000 1.000e+00

# Dataframe with the first 10 PCs
dim1_10 <- data.frame(pca_exp$x[, 1:10])
# Adding metadata
pcaWithMetadata <- merge(dim1_10, dge$samples, by = 0, all = TRUE)
pcaWithMetadata$strain <- factor(pcaWithMetadata$strain,
                                   levels = c("VV", "GG", "VG", "GV", NA))

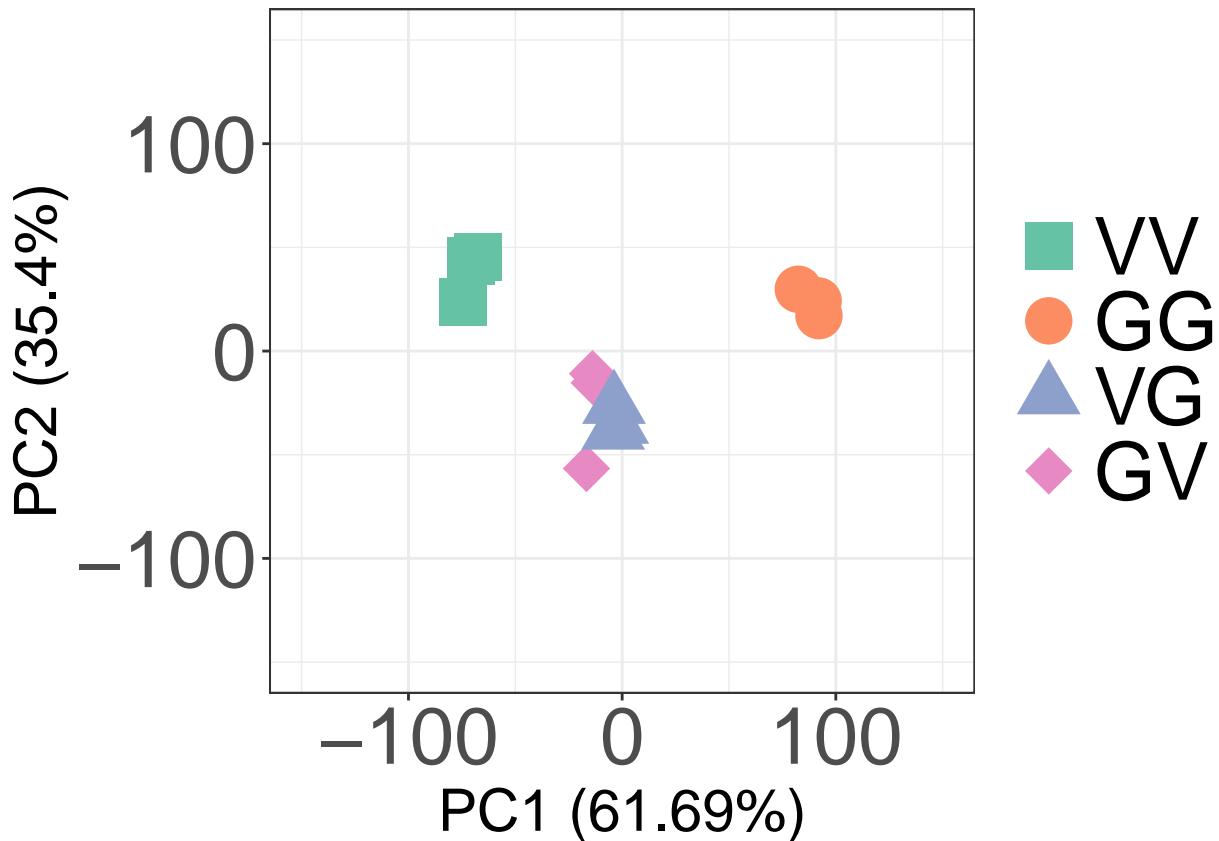
# Plotting
#png(filename = "figures/wilson_AvgREFs_PCA.png",
#     width = 650,
#     height = 650)
ggplot(data = pcaWithMetadata, aes(
  x = PC1,
  y = PC2,
  shape = strain,
  color = strain
)) +
  geom_point(size = 8) +
  theme_bw() +
  xlim(-150, 150) +
  ylim(-150, 150) +
  scale_color_manual(values = c(VV_Color, GG_Color,
                                VG_Color, GV_Color,
                                "azure3")) +
  scale_shape_manual(values = c(VV_Shape, GG_Shape,
                                VG_Shape, GV_Shape)) +
  theme(
    plot.title = element_text(size = 12, face = "bold"),
    legend.title = element_text(size = 30),
    legend.text = element_text(size = 30),
    axis.text.x = element_text(size = 30),
    axis.text.y = element_text(size = 30),

```

```

    axis.title.x = element_text(size = 22),
    axis.title.y = element_text(size = 22)
) +
#guides(color = guide_legend(order = 2)) +
theme(legend.title = element_blank()) +
xlab("PC1 (61.69%)") +
ylab("PC2 (35.4%)")

```



```

# dev.off()
# dev.off()
# samples cluster by strain

```

Differential expression analysis with limma

```

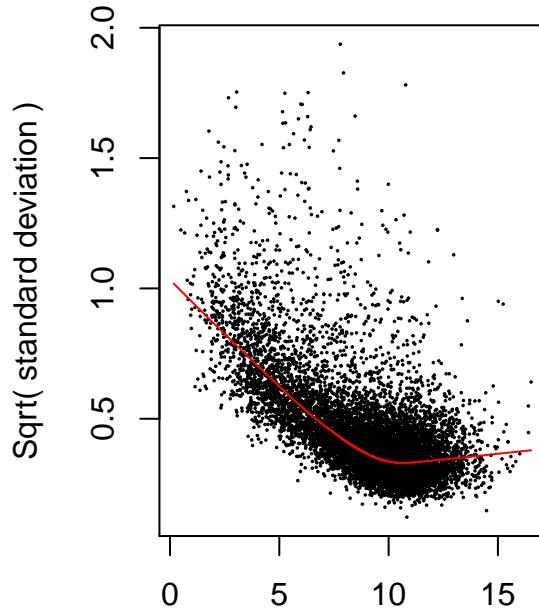
design <- model.matrix(~ 0 + v$targets$strain)
colnames(design) <-
  gsub("v\\$targets\\$strain", "", colnames(design))
head(design)

##   GG  GV  VG  VV
## 1  0  0  0  1
## 2  0  0  0  1
## 3  0  0  0  1
## 4  1  0  0  0
## 5  1  0  0  0
## 6  1  0  0  0

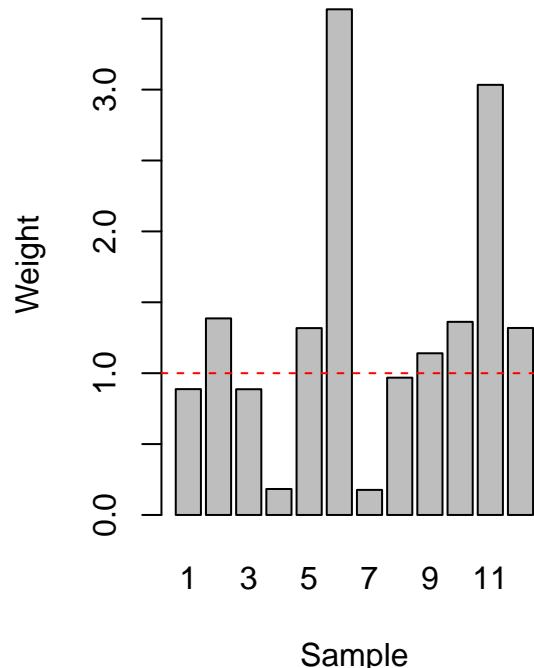
```

```
# Running voom again with the new design matrix.
v <- voomWithQualityWeights(dge, design, plot = TRUE)
```

voom: Mean–variance trend



Sample-specific weights



```
fit <- lmFit(v, design)
# Contrast design for differential expression
# Defining pairwise comparisons
contrasts <- makeContrasts(
  VV_vs_GG = VV - GG,
  VV_vs_VG = VV - VG,
  VV_vs_GV = VV - GV,
  GG_vs_VG = GG - VG,
  GG_vs_GV = GG - GV,
  VG_vs_GV = VG - GV,
  levels = colnames(design)
)
head(contrasts)

##          Contrasts
## Levels VV_vs_GG VV_vs_VG VV_vs_GV GG_vs_VG GG_vs_GV VG_vs_GV
##   GG      -1       0       0       1       1       0
##   GV      0       0      -1       0      -1      -1
##   VG      0      -1       0      -1       0       1
##   VV      1       1       1       0       0       0

# Assigning all comparisons to a vector for later
allComparisons <- colnames(contrasts)

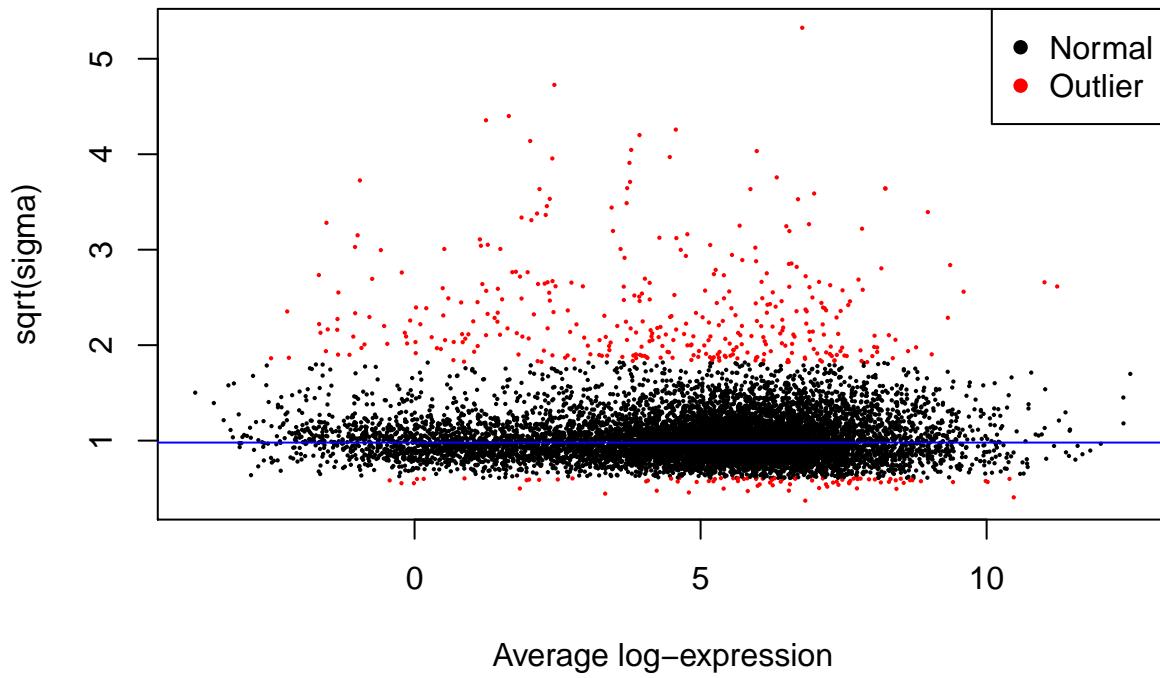
# Running contrast analysis
vfit <- contrasts.fit(fit, contrasts = contrasts)
# Looking at N of DEGs with adj. p <0.01 and log2FC>2
```

```

sumTable <-
  summary(decideTests(
    vfit,
    adjust.method = "BH",
    p.value = 0.01,
    lfc = 2
  ))
# Computing differential expression based on the empirical Bayes moderation of
# the standard errors towards a common value. Robust = should the estimation of
# the empirical Bayes prior parameters be robustified against outlier sample
# variances?
veBayesFit <- eBayes(vfit, robust = TRUE)
plotSA(veBayesFit, main = "Final model: Mean-variance trend")

```

Final model: Mean–variance trend



```

# Getting the DEGs. Log2FC of 1 is equivalent to linear fold change of 2.
# Getting summary statistics for all genes
# Getting the DEGs. Log2FC of 1 is equivalent to linear fold change of 2.
# Getting summary statistics for all genes
coef = 1
for (i in allComparisons) {
  vTopTableAll <-
    topTable(
      veBayesFit,
      coef = coef,
      n = Inf,
      p.value = 1,
      lfc = 0
    )
}

```

```

path <-
  paste("./DEGs/DEGs_wilson_AvgREFs_fpkm05_",
    i,
    "_fdr1_lfc0.txt",
    sep = "")
write.table(vTopTableAll, path, sep = "\t")

# Adj.p<0.05, log2FC>1
vTopTable1 <-
  topTable(
    veBayesFit,
    coef = coef,
    n = Inf,
    p.value = 0.05,
    lfc = 1
  )
path <-
  paste("./DEGs/DEGs_wilson_AvgREFs_fpkm05_",
    i,
    "_fdr05_lfc1.txt",
    sep = "")
write.table(vTopTable1, path, sep = "\t")

# Adj.p<0.01, log2FC>0
vTopTable1 <-
  topTable(
    veBayesFit,
    coef = coef,
    n = Inf,
    p.value = 0.01,
    lfc = 0
  )
path <-
  paste("./DEGs/DEGs_wilson_AvgREFs_fpkm05_",
    i,
    "_fdr001_lfc0.txt",
    sep = "")
write.table(vTopTable1, path, sep = "\t")

# Adj.p<0.01, log2FC>1
vTopTable2 <-
  topTable(
    veBayesFit,
    coef = coef,
    n = Inf,
    p.value = 0.01,
    lfc = 1
  )
path <-
  paste("./DEGs/DEGs_wilson_AvgREFs_fpkm05_",
    i,
    "_fdr001_lfc1.txt",
    sep = "")

```

```

write.table(vTopTable2, path, sep = "\t")

# Adj.p<0.01, log2FC>2
vTopTable3 <-
  topTable(
    veBayesFit,
    coef = coef,
    n = Inf,
    p.value = 0.01,
    lfc = 2
  )
path <-
  paste("./DEGs/DEGs_wilson_AvgREFs_fpkm05_",
    i,
    "_fdr001_lfc2.txt",
    sep = "")
write.table(vTopTable3, path, sep = "\t")

coef = coef + 1
}

VV_GG_DEG <-
  rownames(read.table(
    "./DEGs/DEGs_wilson_AvgREFs_fpkm05_VV_vs_GG_fdr001_lfc2.txt"
  ))
VV_VG_DEG <-
  rownames(read.table(
    "./DEGs/DEGs_wilson_AvgREFs_fpkm05_VV_vs_VG_fdr001_lfc2.txt"
  ))
VV_GV_DEG <-
  rownames(read.table(
    "./DEGs/DEGs_wilson_AvgREFs_fpkm05_VV_vs_GV_fdr001_lfc2.txt"
  ))

```

Volcano plots of differential expression analysis for all pairwise comparisons

```

VV_vs_GG_DEG <-
  read.table(
    "./DEGs/DEGs_wilson_AvgREFs_fpkm05_VV_vs_GG_fdr1_lfc0.txt",
    header = TRUE,
    sep = "\t",
    stringsAsFactors = F
  )
VV_vs_GG_DF <-
  data.frame(VV_vs_GG_DEG$adj.P.Val,
    VV_vs_GG_DEG$logFC,
    VV_vs_GG_DEG$Chr,
    VV_vs_GG_DEG$Geneid)
colnames(VV_vs_GG_DF) <- c("adj.P.Val", "logFC", "Chr", "Geneid")

VV_vs_GG_DF_Sig <

```

```

VV_vs_GG_DF[(abs(VV_vs_GG_DF$logFC) >= 2 &
              VV_vs_GG_DF$adj.P.Val <= 0.01), ]$Geneid

# Finding stain bias genes, assigning color values
nonSig <- subset(VV_vs_GG_DF, !(Geneid %in% VV_vs_GG_DF_Sig))
nonSig <- cbind(nonSig , rep(1, nrow(nonSig)))
colnames(nonSig)[5] <- "Color"

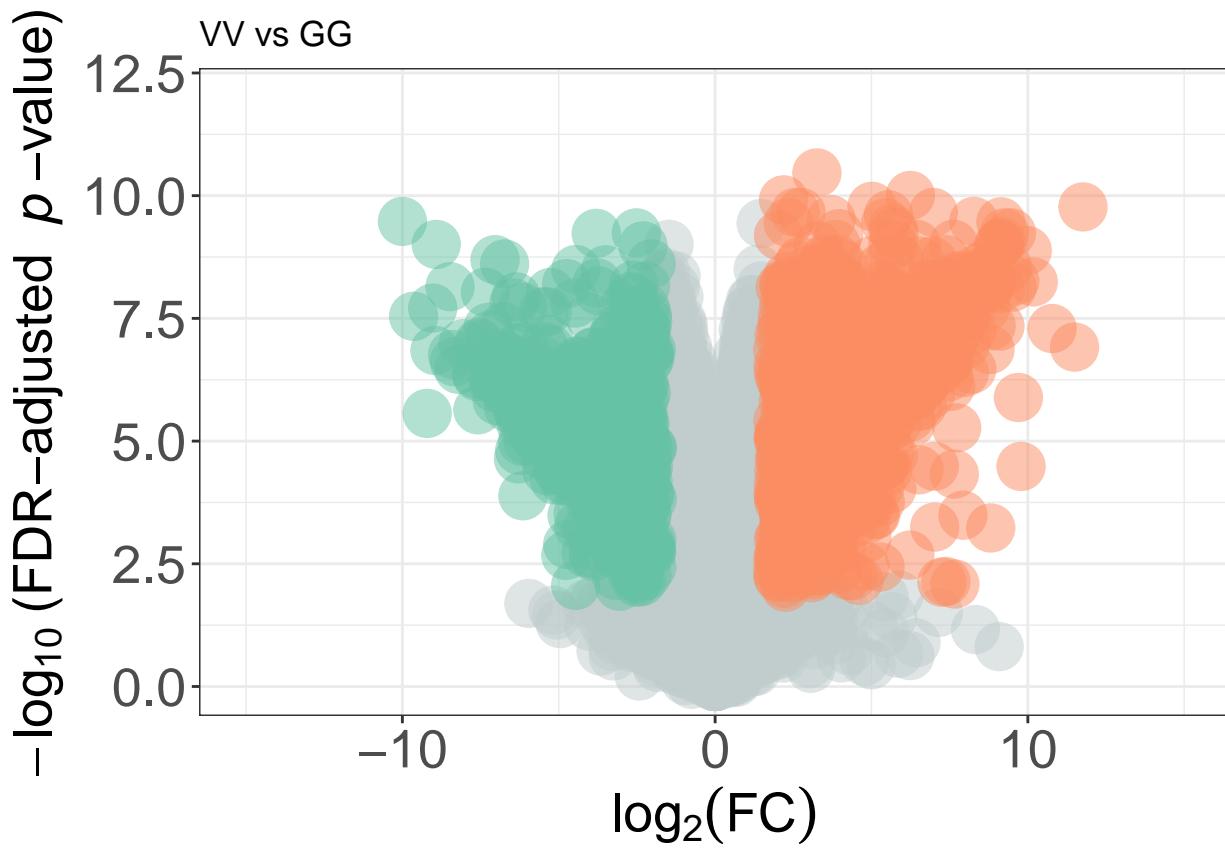
up_VV <-
  subset(
    VV_vs_GG_DF,
    VV_vs_GG_DF$logFC >= 2 &
      VV_vs_GG_DF$adj.P.Val <= 0.01 & (Geneid %in% VV_vs_GG_DF_Sig)
  )
up_VV <- cbind(up_VV , rep(2, nrow(up_VV)))
colnames(up_VV) [5] <- "Color"

up_GG <-
  subset(
    VV_vs_GG_DF,
    VV_vs_GG_DF$logFC <= -2 &
      VV_vs_GG_DF$adj.P.Val <= 0.01 & (Geneid %in% VV_vs_GG_DF_Sig)
  )
up_GG <- cbind(up_GG , rep(3, nrow(up_GG)))
colnames(up_GG) [5] <- "Color"

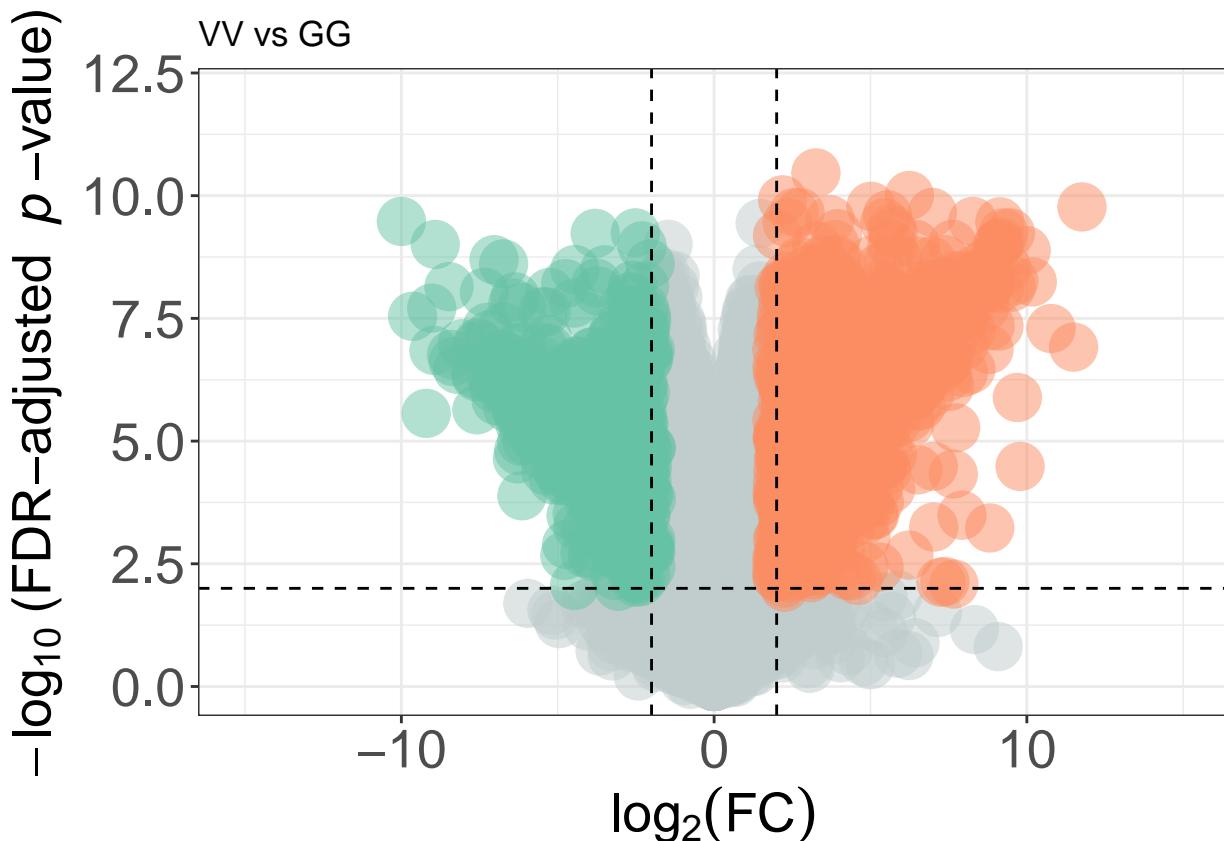
dfPlot <- rbind(nonSig, up_VV, up_GG)
dfPlot$Color <- as.factor(dfPlot$Color)

# Constructing the plot object.
p <-
  ggplot(data = dfPlot, aes(
    x = logFC,
    y = -log10(adj.P.Val),
    color = Color
  )) +
  geom_point(alpha = 0.5, size = 8) +
  theme_bw() +
  theme(legend.position = "none") +
  xlim(c(-15, 15)) + ylim(c(0, 12)) +
  scale_color_manual(values = c("azure3", GG_Color, VV_Color)) +
  labs(
    title = "VV vs GG",
    x = expression(log[2](FC)),
    y = expression(-log[10] ~ "(FDR-adjusted " ~ italic("p") ~ "-value)")
  ) +
  theme(axis.title.x = element_text(size = 20),
        axis.text.x = element_text(size = 20)) +
  theme(axis.title.y = element_text(size = 20),
        axis.text.y = element_text(size = 20))
# Adding lines for significance thresholds
p

```



```
#png(filename = "figures/wilson_AvgREFs_VV_vs_GG_valcano.png",
#     width = 650,
#     height = 650)
p + geom_hline(yintercept = 2,
                 colour = "#000000",
                 linetype = "dashed") +
  geom_vline(xintercept = 2,
             colour = "#000000",
             linetype = "dashed") +
  geom_vline(xintercept = -2,
             colour = "#000000",
             linetype = "dashed")
```



```

# dev.off()
# dev.off()

# VV vs VG
VV_vs_VG_DEG <-
  read.table(
    "./DEGs/DEGs_wilson_AvgREFs_fpkm05_VV_vs_VG_fdr1_lfc0.txt",
    header = TRUE,
    sep = "\t",
    stringsAsFactors = F
  )
VV_vs_VG_DF <-
  data.frame(VV_vs_VG_DEG$adj.P.Val,
             VV_vs_VG_DEG$logFC,
             VV_vs_VG_DEG$Chr,
             VV_vs_VG_DEG$Geneid)
colnames(VV_vs_VG_DF) <- c("adj.P.Val", "logFC", "Chr", "Geneid")

VV_vs_VG_DF_Sig <-
  VV_vs_VG_DF[(abs(VV_vs_VG_DF$logFC) >= 2 &
                VV_vs_VG_DF$adj.P.Val <= 0.01), ]$Geneid

# Finding stain bias genes, assigning color values
nonSig <- subset(VV_vs_VG_DF, !(Geneid %in% VV_vs_VG_DF_Sig))
nonSig <- cbind(nonSig, rep(1, nrow(nonSig)))
colnames(nonSig)[5] <- "Color"

```

```

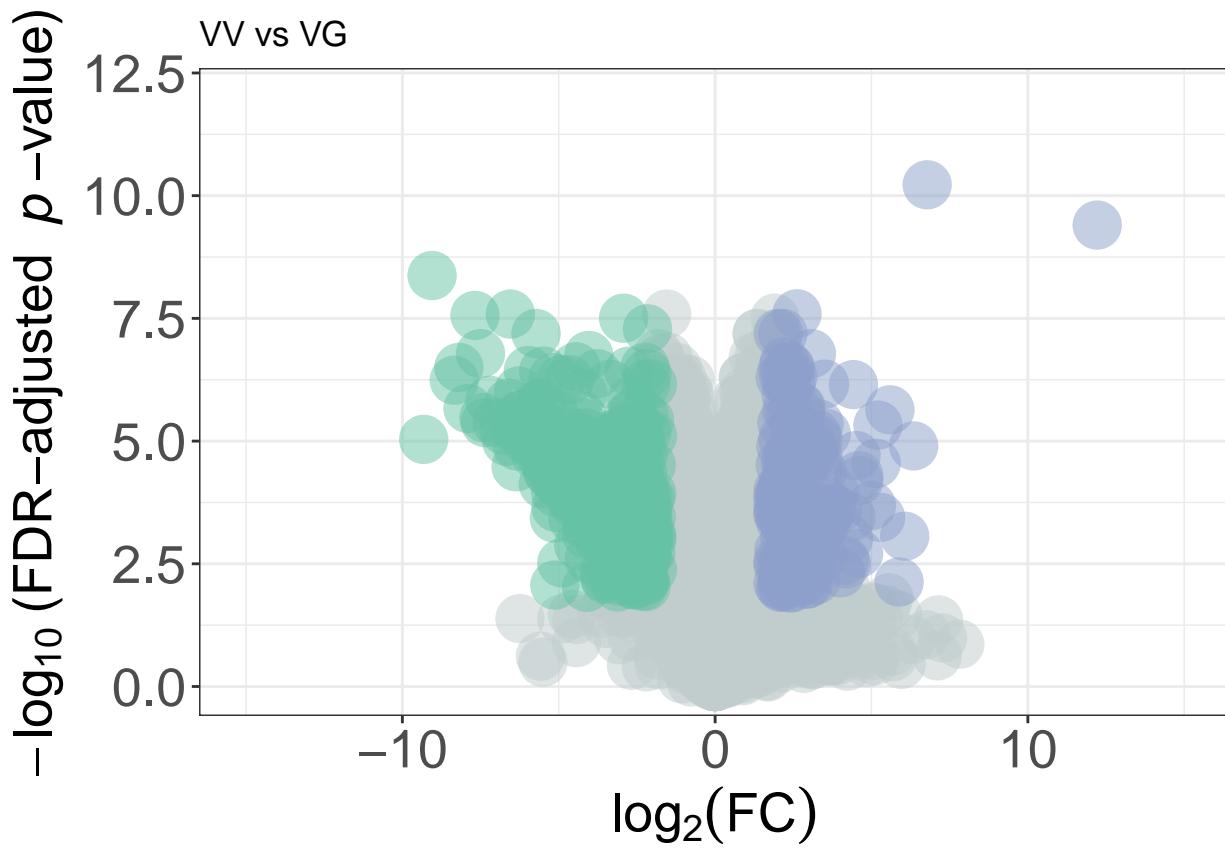
up_VV <-
  subset(
    VV_vs_VG_DF,
    VV_vs_VG_DF$logFC >= 2 &
      VV_vs_VG_DF$adj.P.Val <= 0.01 & (Geneid %in% VV_vs_VG_DF_Sig)
  )
up_VV <- cbind(up_VV , rep(2, nrow(up_VV)))
colnames(up_VV) [5] <- "Color"

up_VG <-
  subset(
    VV_vs_VG_DF,
    VV_vs_VG_DF$logFC <= -2 &
      VV_vs_VG_DF$adj.P.Val <= 0.01 & (Geneid %in% VV_vs_VG_DF_Sig)
  )
up_VG <- cbind(up_VG , rep(3, nrow(up_VG)))
colnames(up_VG) [5] <- "Color"

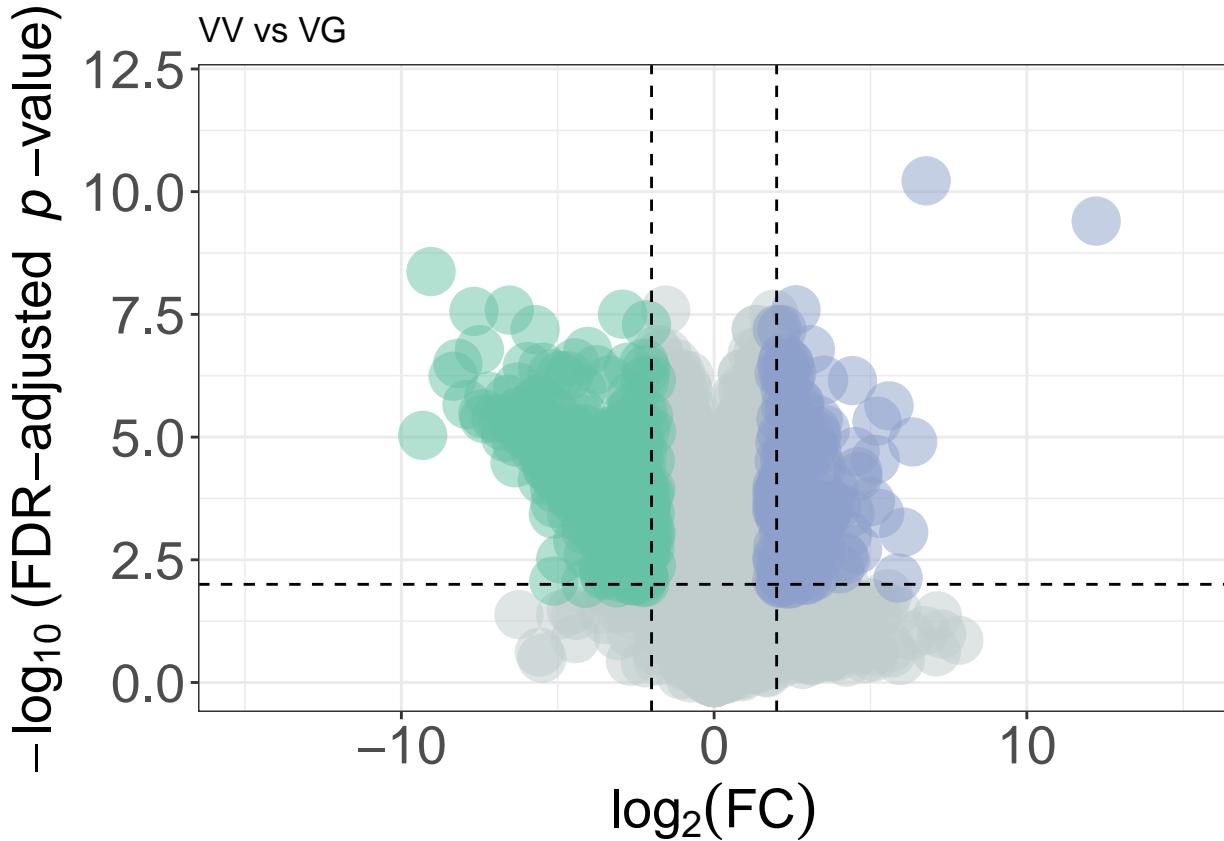
dfPlot <- rbind(nonSig, up_VV, up_VG)
dfPlot$Color <- as.factor(dfPlot$Color)

# Constructing the plot object.
p <-
  ggplot(data = dfPlot, aes(
    x = logFC,
    y = -log10(adj.P.Val),
    color = Color
  )) +
  geom_point(alpha = 0.5, size = 8) +
  theme_bw() +
  theme(legend.position = "none") +
  xlim(c(-15, 15)) + ylim(c(0, 12)) +
  scale_color_manual(values = c("azure3", VG_Color, VV_Color)) +
  labs(
    title = "VV vs VG",
    x = expression(log[2](FC)),
    y = expression(-log[10] ~ "(FDR-adjusted " ~ italic("p") ~ "-value)")
  ) +
  theme(axis.title.x = element_text(size = 20),
        axis.text.x = element_text(size = 20)) +
  theme(axis.title.y = element_text(size = 20),
        axis.text.y = element_text(size = 20))
# Adding lines for significance thresholds
p

```



```
#png(filename = "figures/wilson_AvgREFs_VV_vs_VG_valcano.png",
#     width = 650,
#     height = 650)
p + geom_hline(yintercept = 2,
                 colour = "#000000",
                 linetype = "dashed") +
  geom_vline(xintercept = 2,
             colour = "#000000",
             linetype = "dashed") +
  geom_vline(xintercept = -2,
             colour = "#000000",
             linetype = "dashed")
```



```

# dev.off()
# dev.off()

# VV vs GV
VV_vs_GV_DEG <-
  read.table(
    "./DEGs/DEGs_wilson_AvgREFs_fpkm05_VV_vs_GV_fdr1_lfc0.txt",
    header = TRUE,
    sep = "\t",
    stringsAsFactors = F
  )
VV_vs_GV_DF <-
  data.frame(VV_vs_GV_DEG$adj.P.Val,
             VV_vs_GV_DEG$logFC,
             VV_vs_GV_DEG$Chr,
             VV_vs_GV_DEG$Geneid)
colnames(VV_vs_GV_DF) <- c("adj.P.Val", "logFC", "Chr", "Geneid")

VV_vs_GV_DF_Sig <-
  VV_vs_GV_DF[(abs(VV_vs_GV_DF$logFC) >= 2 &
                VV_vs_GV_DF$adj.P.Val <= 0.01), ]$Geneid

# Finding stain bias genes, assigning color values
nonSig <- subset(VV_vs_GV_DF, !(Geneid %in% VV_vs_GV_DF_Sig))
nonSig <- cbind(nonSig, rep(1, nrow(nonSig)))
colnames(nonSig)[5] <- "Color"

```

```

up_VV <-
  subset(
    VV_vs_GV_DF,
    VV_vs_GV_DF$logFC >= 2 &
      VV_vs_GV_DF$adj.P.Val <= 0.01 & (Geneid %in% VV_vs_GV_DF_Sig)
  )
up_VV <- cbind(up_VV , rep(2, nrow(up_VV)))
colnames(up_VV) [5] <- "Color"

up_GV <-
  subset(
    VV_vs_GV_DF,
    VV_vs_GV_DF$logFC <= -2 &
      VV_vs_GV_DF$adj.P.Val <= 0.01 & (Geneid %in% VV_vs_GV_DF_Sig)
  )
up_GV <- cbind(up_GV , rep(3, nrow(up_GV)))
colnames(up_GV) [5] <- "Color"

dfPlot <- rbind(nonSig, up_VV, up_GV)
dfPlot$Color <- as.factor(dfPlot$Color)

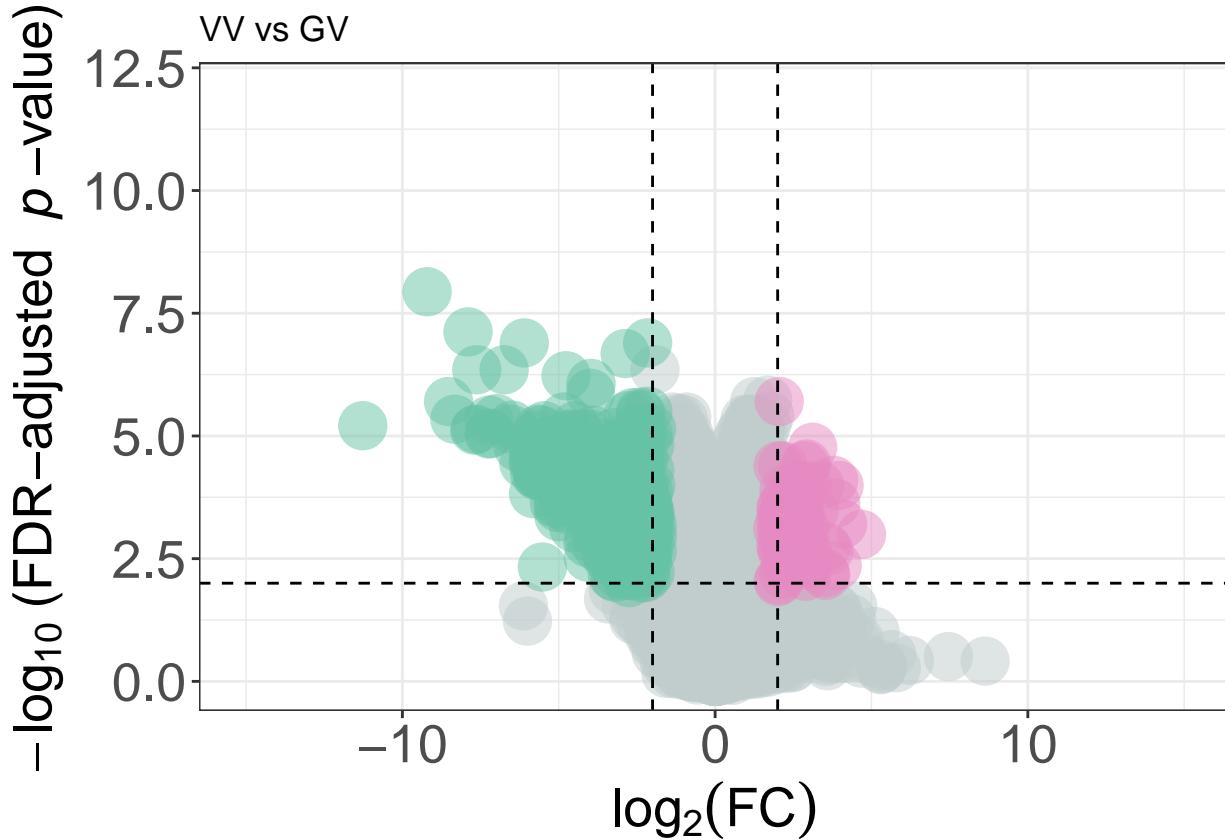
# Constructing the plot object.
p <-
  ggplot(data = dfPlot, aes(
    x = logFC,
    y = -log10(adj.P.Val),
    color = Color
  )) +
  geom_point(alpha = 0.5, size = 8) +
  theme_bw() +
  theme(legend.position = "none") +
  xlim(c(-15, 15)) + ylim(c(0, 12)) +
  scale_color_manual(values = c("azure3", GV_Color, VV_Color)) +
  labs(
    title = "VV vs GV",
    x = expression(log[2](FC)),
    y = expression(-log[10] ~ "(FDR-adjusted " ~ italic("p") ~ "-value)")
  ) +
  theme(axis.title.x = element_text(size = 20),
        axis.text.x = element_text(size = 20)) +
  theme(axis.title.y = element_text(size = 20),
        axis.text.y = element_text(size = 20))
# Adding lines for significance thresholds
#png(filename = "figures/wilson_AvgREFs_VV_vs_GV_valcano.png",
#     width = 650,
#     height = 650)
p + geom_hline(yintercept = 2,
                 colour = "#000000",
                 linetype = "dashed") +
  geom_vline(xintercept = 2,
             colour = "#000000",
             linetype = "dashed") +
  geom_vline(xintercept = -2,

```

```

  colour = "#000000",
  linetype = "dashed")

```



```

# dev.off()
# dev.off()

# GG vs VG
GG_vs_VG_DEG <-
  read.table(
    "./DEGs/DEGs_wilson_AvgREFs_fpkm05_GG_vs_VG_fdr1_lfc0.txt",
    header = TRUE,
    sep = "\t",
    stringsAsFactors = F
  )
GG_vs_VG_DF <-
  data.frame(GG_vs_VG_DEG$adj.P.Val,
             GG_vs_VG_DEG$logFC,
             GG_vs_VG_DEG$Chr,
             GG_vs_VG_DEG$Geneid)
colnames(GG_vs_VG_DF) <- c("adj.P.Val", "logFC", "Chr", "Geneid")

GG_vs_VG_DF_Sig <-
  GG_vs_VG_DF[(abs(GG_vs_VG_DF$logFC) >= 2 &
                GG_vs_VG_DF$adj.P.Val <= 0.01), ]$Geneid

# Finding stain bias genes, assigning color values
nonSig <- subset(GG_vs_VG_DF, !(Geneid %in% GG_vs_VG_DF_Sig))

```

```

nonSig <- cbind(nonSig , rep(1, nrow(nonSig)))
colnames(nonSig)[5] <- "Color"

up_GG <-
subset(
  GG_vs_VG_DF,
  GG_vs_VG_DF$logFC >= 2 &
  GG_vs_VG_DF$adj.P.Val <= 0.01 & (Geneid %in% GG_vs_VG_DF_Sig)
)
up_GG <- cbind(up_GG , rep(2, nrow(up_GG)))
colnames(up_GG)[5] <- "Color"

up_VG <-
subset(
  GG_vs_VG_DF,
  GG_vs_VG_DF$logFC <= -2 &
  GG_vs_VG_DF$adj.P.Val <= 0.01 & (Geneid %in% GG_vs_VG_DF_Sig)
)
up_VG <- cbind(up_VG , rep(3, nrow(up_VG)))
colnames(up_VG)[5] <- "Color"

dfPlot <- rbind(nonSig, up_GG, up_VG)
dfPlot$Color <- as.factor(dfPlot$Color)

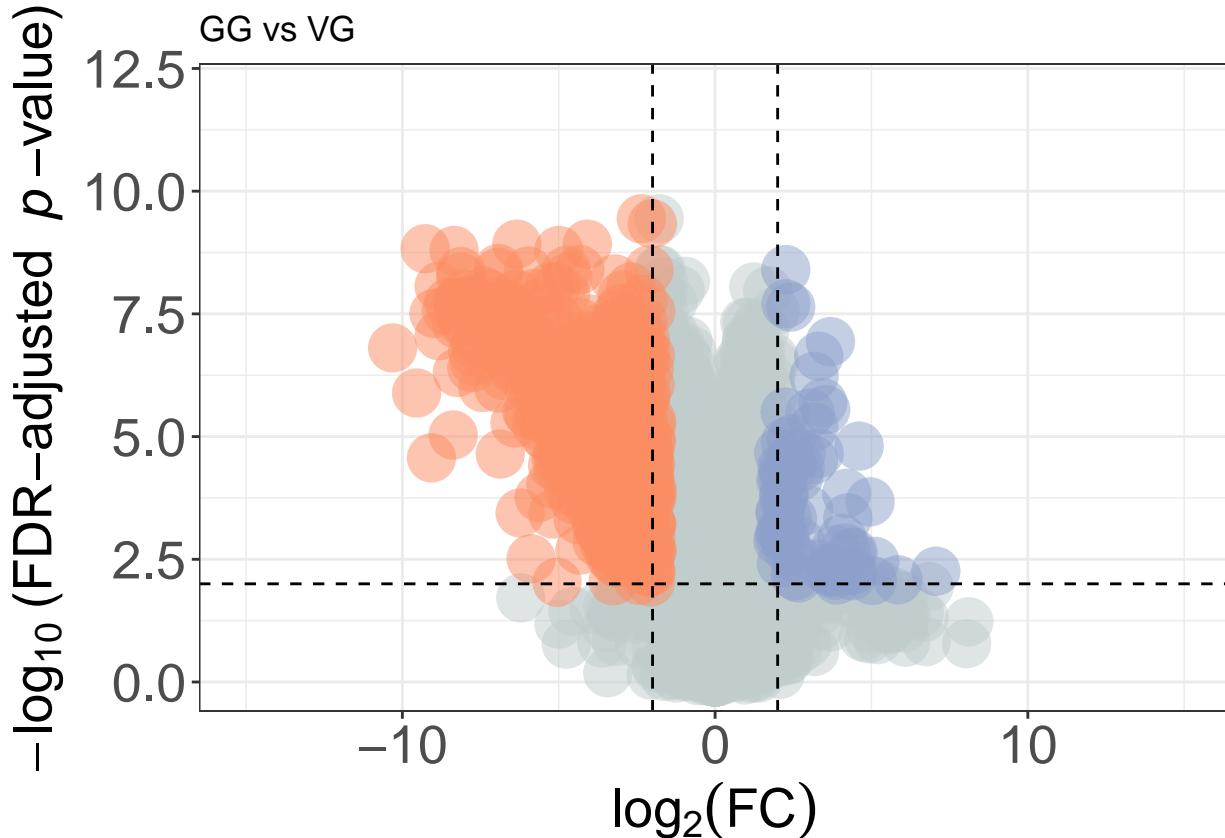
# Constructing the plot object.
p <-
ggplot(data = dfPlot, aes(
  x = logFC,
  y = -log10(adj.P.Val),
  color = Color
)) +
geom_point(alpha = 0.5, size = 8) +
theme_bw() +
theme(legend.position = "none") +
xlim(c(-15, 15)) + ylim(c(0, 12)) +
scale_color_manual(values = c("azure3", VG_Color, GG_Color)) +
labs(
  title = "GG vs VG",
  x = expression(log[2](FC)),
  y = expression(-log[10] ~ "(FDR-adjusted " ~ italic("p") ~ "-value)")
) +
theme(axis.title.x = element_text(size = 20),
      axis.text.x = element_text(size = 20)) +
theme(axis.title.y = element_text(size = 20),
      axis.text.y = element_text(size = 20))
# Adding lines for significance thresholds
#png(filename = "figures/wilson_AvgREFs_GG_vs_VG_valcano.png",
#     width = 650,
#     height = 650)
p + geom_hline(yintercept = 2,
                colour = "#000000",
                linetype = "dashed") +
geom_vline(xintercept = 2,

```

```

        colour = "#000000",
        linetype = "dashed") +
geom_vline(xintercept = -2,
            colour = "#000000",
            linetype = "dashed")

```



```

# dev.off()
# dev.off()

# GG vs GV
GG_vs_GV_DEG <-
  read.table(
    "./DEGs/DEGs_wilson_AvgREFs_fpkm05_GG_vs_GV_fdr1_lfc0.txt",
    header = TRUE,
    sep = "\t",
    stringsAsFactors = F
  )
GG_vs_GV_DF <-
  data.frame(GG_vs_GV_DEG$adj.P.Val,
             GG_vs_GV_DEG$logFC,
             GG_vs_GV_DEG$Chr,
             GG_vs_GV_DEG$Geneid)
colnames(GG_vs_GV_DF) <- c("adj.P.Val", "logFC", "Chr", "Geneid")

GG_vs_GV_DF_Sig <-
  GG_vs_GV_DF[(abs(GG_vs_GV_DF$logFC) >= 2 &
                GG_vs_GV_DF$adj.P.Val <= 0.01), ]$Geneid

```

```

# Finding stain bias genes, assigning color values
nonSig <- subset(GG_vs_GV_DF, !(Geneid %in% GG_vs_GV_DF_Sig))
nonSig <- cbind(nonSig , rep(1, nrow(nonSig)))
colnames(nonSig)[5] <- "Color"

up_GG <-
  subset(
    GG_vs_GV_DF,
    GG_vs_GV_DF$logFC >= 2 &
      GG_vs_GV_DF$adj.P.Val <= 0.01 & (Geneid %in% GG_vs_GV_DF_Sig)
  )
up_GG <- cbind(up_GG , rep(2, nrow(up_GG)))
colnames(up_GG) [5] <- "Color"

up_GV <-
  subset(
    GG_vs_GV_DF,
    GG_vs_GV_DF$logFC <= -2 &
      GG_vs_GV_DF$adj.P.Val <= 0.01 & (Geneid %in% GG_vs_GV_DF_Sig)
  )
up_GV <- cbind(up_GV , rep(3, nrow(up_GV)))
colnames(up_GV) [5] <- "Color"

dfPlot <- rbind(nonSig, up_GG, up_GV)
dfPlot$Color <- as.factor(dfPlot$Color)

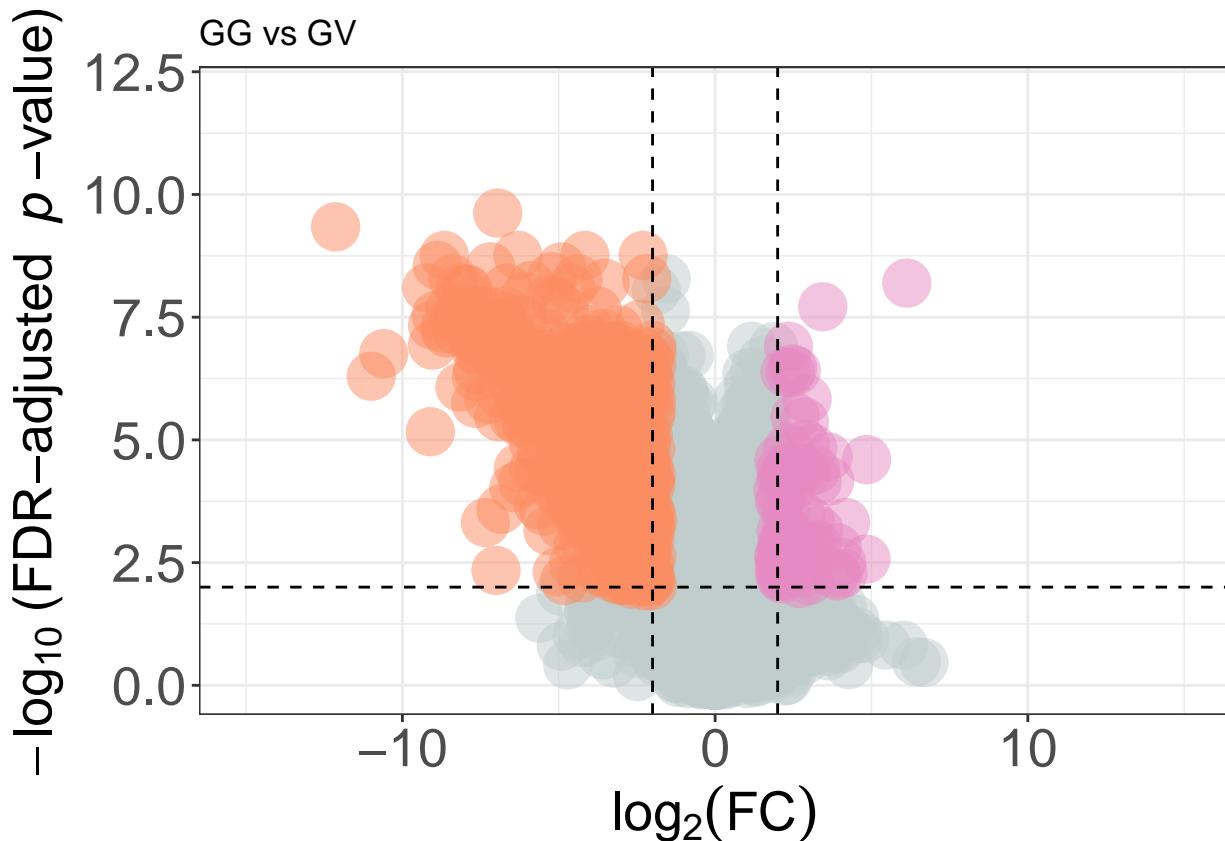
# Constructing the plot object.
p <-
  ggplot(data = dfPlot, aes(
    x = logFC,
    y = -log10(adj.P.Val),
    color = Color
  )) +
  geom_point(alpha = 0.5, size = 8) +
  theme_bw() +
  theme(legend.position = "none") +
  xlim(c(-15, 15)) + ylim(c(0, 12)) +
  scale_color_manual(values = c("azure3", GV_Color, GG_Color)) +
  labs(
    title = "GG vs GV",
    x = expression(log[2](FC)),
    y = expression(-log[10] ~ "(FDR-adjusted " ~ italic("p") ~ "-value)")
  ) +
  theme(axis.title.x = element_text(size = 20),
        axis.text.x = element_text(size = 20)) +
  theme(axis.title.y = element_text(size = 20),
        axis.text.y = element_text(size = 20))
# Adding lines for significance thresholds
#png(filename = "figures/wilson_AvgREFs_GG_vs_GV_valcano.png",
#     width = 650,
#     height = 650)
p + geom_hline(yintercept = 2,
                colour = "#000000",

```

```

        linetype = "dashed") +
geom_vline(xintercept = 2,
            colour = "#000000",
            linetype = "dashed") +
geom_vline(xintercept = -2,
            colour = "#000000",
            linetype = "dashed")

```



```

# dev.off()
# dev.off()

# VG vs GV
VG_vs_GV_DEG <-
read.table(
  "./DEGs/DEGs_wilson_AvgREFs_fpkm05_VG_vs_GV_fdr1_lfc0.txt",
  header = TRUE,
  sep = "\t",
  stringsAsFactors = F
)
VG_vs_GV_DF <-
  data.frame(VG_vs_GV_DEG$adj.P.Val,
             VG_vs_GV_DEG$logFC,
             VG_vs_GV_DEG$Chr,
             VG_vs_GV_DEG$Geneid)
colnames(VG_vs_GV_DF) <- c("adj.P.Val", "logFC", "Chr", "Geneid")

VG_vs_GV_DF_Sig <

```

```

VG_vs_GV_DF[(abs(VG_vs_GV_DF$logFC) >= 2 &
              VG_vs_GV_DF$adj.P.Val <= 0.01), ]$Geneid

# Finding stain bias genes, assigning color values
nonSig <- subset(VG_vs_GV_DF, !(Geneid %in% VG_vs_GV_DF_Sig))
nonSig <- cbind(nonSig, rep(1, nrow(nonSig)))
colnames(nonSig)[5] <- "Color"

up_VG <-
  subset(
    VG_vs_GV_DF,
    VG_vs_GV_DF$logFC >= 2 &
      VG_vs_GV_DF$adj.P.Val <= 0.01 & (Geneid %in% VG_vs_GV_DF_Sig)
  )
up_VG <- cbind(up_VG, rep(2, nrow(up_VG)))
colnames(up_VG)[5] <- "Color"

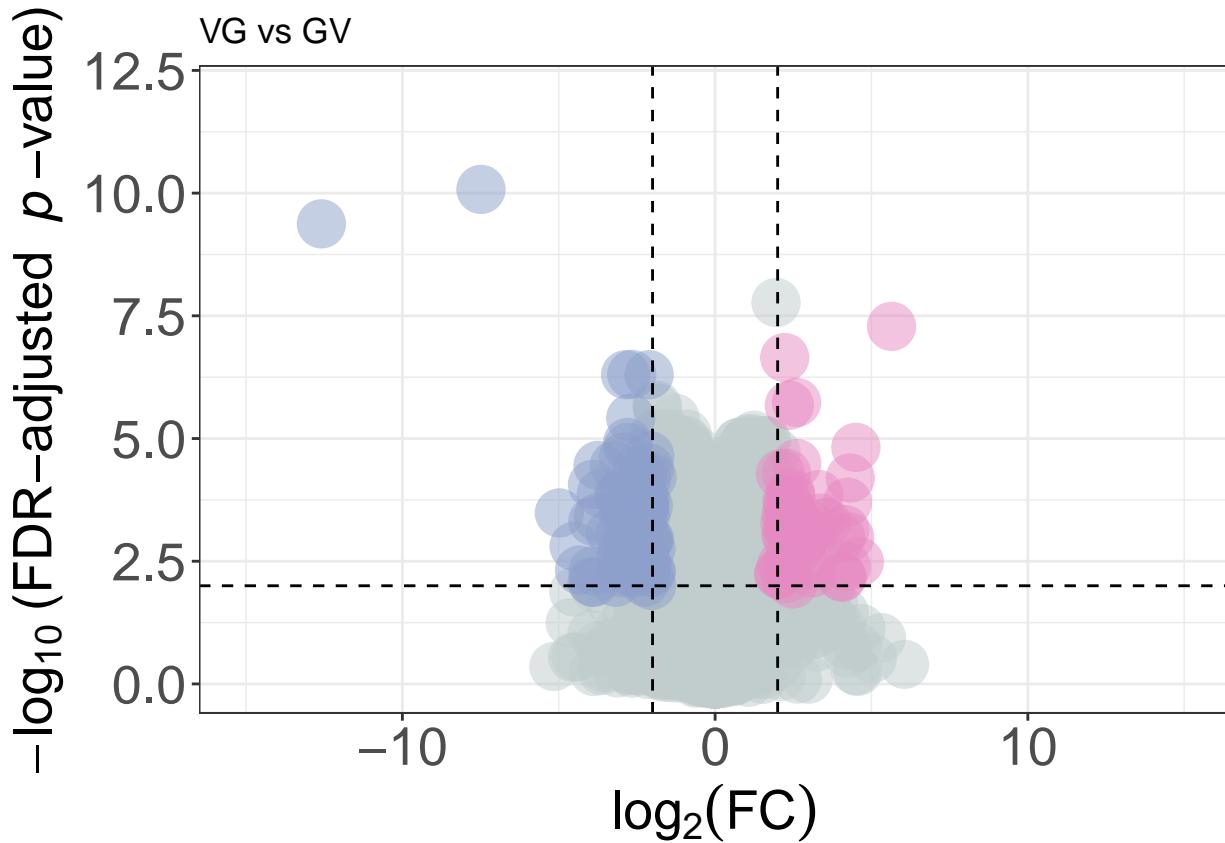
up_GV <-
  subset(
    VG_vs_GV_DF,
    VG_vs_GV_DF$logFC <= -2 &
      VG_vs_GV_DF$adj.P.Val <= 0.01 & (Geneid %in% VG_vs_GV_DF_Sig)
  )
up_GV <- cbind(up_GV, rep(3, nrow(up_GV)))
colnames(up_GV)[5] <- "Color"

dfPlot <- rbind(nonSig, up_VG, up_GV)
dfPlot$Color <- as.factor(dfPlot$Color)

# Constructing the plot object.
p <-
  ggplot(data = dfPlot, aes(
    x = logFC,
    y = -log10(adj.P.Val),
    color = Color
  )) +
  geom_point(alpha = 0.5, size = 8) +
  theme_bw() +
  theme(legend.position = "none") +
  xlim(c(-15, 15)) + ylim(c(0, 12)) +
  scale_color_manual(values = c("azure3", GV_Color, VG_Color)) +
  labs(
    title = "VG vs GV",
    x = expression(log[2](FC)),
    y = expression(-log[10] ~ "(FDR-adjusted " ~ italic("p") ~ "-value)")
  ) +
  theme(axis.title.x = element_text(size = 20),
        axis.text.x = element_text(size = 20)) +
  theme(axis.title.y = element_text(size = 20),
        axis.text.y = element_text(size = 20))
# Adding lines for significance thresholds
#png(filename = "figures/wilson_AvgREFs_VG_vs_GV_valcano.png",
#     width = 650,

```

```
#     height = 650)
p + geom_hline(yintercept = 2,
                 colour = "#000000",
                 linetype = "dashed") +
  geom_vline(xintercept = 2,
             colour = "#000000",
             linetype = "dashed") +
  geom_vline(xintercept = -2,
             colour = "#000000",
             linetype = "dashed")
```



```
# dev.off()
# dev.off()
```