Differential Gene Expression Pipeline

Seema Plaisier

11/14/23

Input data

This version of limma voom differential expression pipeline works on a merged featureCounts data file.

Sex and replicate IDs are determined from the column headings: 1) any sample column heading that contains XX will be assigned female, XY will be assigned male 2) directory name and file name suffix will be removed to assign the replicate ID (so that technical replicates can be summed)

Applying the DE Pipeline to the Placenta Dataset to Determine Sex Differences in Gene Expression

This pipeline is adapted from the limma workflow vignette on Bioconductor: https://bioconductor.org/packages/release/workflows/vignettes/RNAseq123/inst/doc/limmaWorkflow.html

Load packages

Install the necessary packages for our analysis. The function 'require' returns a true value if it is already installed, so if it is not, we need to install before loading. Certain packages are installed using install.packages, other packages are housed in the Bioconductor repository and required BiocManager::install from the BiocManager package to install.

```
if(!require(gplots)){
    install.packages("gplots")
}
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
if(!require(ggplot2)){
    install.packages("ggplot2")
}
```

```
## Loading required package: ggplot2
```

```
if(!require(Polychrome)){
    install.packages("Polychrome")
}
```

```
## Loading required package: Polychrome
```

```
if(!require(RColorBrewer)){
    install.packages("RColorBrewer")
}
```

```
## Loading required package: RColorBrewer
```

```
if(!require(ggpubr)){
    install.packages("ggpubr")
}
```

```
## Loading required package: ggpubr
```

```
if (!require("BiocManager", quietly = TRUE)) {
    install.packages("BiocManager")
}
```

```
## Bioconductor version '3.15' is out-of-date; the current release version '3.18'
##   is available with R version '4.3'; see https://bioconductor.org/install
```

```
if(!require(EnhancedVolcano)){
  BiocManager::install("EnhancedVolcano")
}
```

```
## Loading required package: EnhancedVolcano
```

```
## Loading required package: ggrepel
```

```
if(!require(limma)){
  BiocManager::install("limma")
}
```

```
## Loading required package: limma
```

```
if(!require(edgeR)){
  BiocManager::install("edgeR")
}
```

```
## Loading required package: edgeR
```

```
library(gplots)
library(ggplot2)
library(limma)
library(ggpubr)
library(edgeR)
library(Polychrome)
library(RColorBrewer)
library(EnhancedVolcano)
```

Set working directory

Your working directory is where all your output files are going to be saved. Your input directory is where the gene counts matrix file is stored.

```
# you will need to change this to the directory you are
# working in make sure to include the / at the end of the
# directory path
working_directory = "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4/"  # THIS WI
setwd(working_directory)

# see what's in your working directory files:
list.files(working_directory)
```

```
##  [1] "CompareTwo"
##  [2] "CompareTwo.pdf"
##  [3] "CompareTwo.Rmd"
##  [4] "CompareTwo_Cade.Rmd"
##  [5] "DE_Pipeline_AllData_fixPheno.Rmd"
##  [6] "DEG"
##  [7] "draft"
##  [8] "Extras.docx"
##  [9] "fake_dge_femalevsmale_all_expressed_genes.csv"
## [10] "female_vs_male"
## [11] "geneCounts"
## [12] "Instructions_DEG_trimmed.docx"
## [13] "Instructions_DEG_trimmed.txt"
## [14] "MDS_plots_top100_snippet.R"
## [15] "stats"
## [16] "Troubleshooting DE Pipeline for CURE.pdf"
```

```
# directories:
list.dirs(working_directory)
```

```
##  [1] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4/"
##  [2] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//CompareTwo"
##  [3] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//DEG"
##  [4] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//draft"
##  [5] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//draft/stats"
##  [6] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//draft/stats/trimq_0_minl
##  [7] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//draft/stats/trimq_0_minl
```

```
##  [8] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//draft/stats/trimq_0_minl
##  [9] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//draft/stats/trimq_10_min
## [10] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//draft/stats/trimq_10_min
## [11] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//draft/stats/trimq_10_min
## [12] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//draft/stats/trimq_30_min
## [13] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//draft/stats/trimq_30_min
## [14] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//draft/stats/trimq_30_min
## [15] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//draft/stats/trimq_NONE_m
## [16] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//female_vs_male"
## [17] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//female_vs_male/all_trimm
## [18] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//female_vs_male/trimq_0_m
## [19] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//female_vs_male/trimq_0_m
## [20] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//female_vs_male/trimq_0_m
## [21] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//female_vs_male/trimq_10_
## [22] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//female_vs_male/trimq_10_
## [23] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//female_vs_male/trimq_10_
## [24] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//female_vs_male/trimq_30_
## [25] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//female_vs_male/trimq_30_
## [26] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//female_vs_male/trimq_30_
## [27] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//female_vs_male/trimq_NON
## [28] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//geneCounts"
## [29] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//geneCounts/trimq_0_minle
## [30] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//geneCounts/trimq_0_minle
## [31] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//geneCounts/trimq_0_minle
## [32] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//geneCounts/trimq_10_minl
## [33] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//geneCounts/trimq_10_minl
## [34] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//geneCounts/trimq_10_minl
## [35] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//geneCounts/trimq_30_minl
## [36] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//geneCounts/trimq_30_minl
## [37] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//geneCounts/trimq_30_minl
## [38] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//geneCounts/trimq_NONE_mi
## [39] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//stats"
## [40] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//stats/trimq_0_minlen_10
## [41] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//stats/trimq_0_minlen_30
## [42] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//stats/trimq_0_minlen_75
## [43] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//stats/trimq_10_minlen_1
## [44] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//stats/trimq_10_minlen_3
## [45] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//stats/trimq_10_minlen_7
## [46] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//stats/trimq_30_minlen_1
## [47] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//stats/trimq_30_minlen_3
## [48] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//stats/trimq_30_minlen_7
## [49] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Placenta_Fall2022/Week 4//stats/trimq_NONE_minlen_
```

```
# You can now change this to trimq_NONE_minlen_NONE or any
# of the trimmed data directories
data_directory = "trimq_30_minlen_75/"

# this creates the path to the gene counts directory for
# your data
input_directory = paste0(working_directory, "geneCounts/", data_directory)

# Label for output file will be the data directory that you
# enter without the / Changed this so that no one
# mistakenly only changes the result file label without
```

```
# actually changing the input file directory
result_file_label = gsub("/", "", data_directory)
```

Read in placenta data

Read in file containing gene counts and file containing matching sample information/metadata. We will be using DGEList objects for organizing our gene expression information and passing it into functions that process gene expression data.

In our dataset there were two RNAseq samples per placenta tissue samples (technical replicates), labeled with -1 and -2 in the sample name. We store the sample name without the -1 and -2 in a sample information column called 'rep'. This sets up the sumTechReps function to sum the counts from the technical replicates before moving forward. This assures that we are not counting the technical replicates as individual samples.

```
# Save imported data as variables
counts <- read.delim(paste0(input_directory, "geneCounts_all.csv"),
    header = TRUE, sep = ",", check.names = FALSE)
# NOTICE that we are no longer getting a pheno.csv file, we
# are going to get that info from the column headers

# get the column headings for the samples in the counts
# data
samples = colnames(counts)
pheno_samples = samples[2:length(samples)]  # remove Geneid column at the front

# make a vector to serve as a place holder for the sex of
# the samples
pheno_sex = pheno_samples

# replace the sex with female/male based on if they contain
# XX/XY
for (i in 1:length(pheno_samples)) {
    if (grepl("XY", pheno_samples[i], fixed = TRUE)) {
        pheno_sex[i] = "male"
    }
    if (grepl("XX", pheno_samples[i], fixed = TRUE)) {
        pheno_sex[i] = "female"
    }
}

# make a vector to serve as a place holder for the 'rep',
# the ID of the placenta tissue
pheno_rep = pheno_samples

# To get the replicate ID: remove the untrimmed/trimmed and
# suffix of the alignment file passed into featureCounts
# depending on whether this is the untrimmed or trimmed
# data, can replace whatever is found
pheno_rep = gsub("_trimmed.XX.sort.mkdup.rdgrp.bam", "", pheno_rep)
pheno_rep = gsub("_trimmed.XY.sort.mkdup.rdgrp.bam", "", pheno_rep)
pheno_rep = gsub("_untrimmed.XX.sort.mkdup.rdgrp.bam", "", pheno_rep)
pheno_rep = gsub("_untrimmed.XY.sort.mkdup.rdgrp.bam", "", pheno_rep)
```

```
# remove the directory
pheno_rep = gsub(paste0(data_directory, "processed_bams/"), "",
    pheno_rep)

# remove the -1 or -2 for the two samples taken from the
# same placenta
pheno_rep = gsub("-1", "", pheno_rep, fixed = TRUE)
pheno_rep = gsub("-2", "", pheno_rep, fixed = TRUE)

# create a phenotype info table which we now know is in the
# right order
pheno = as.data.frame(cbind(pheno_samples, pheno_sex, pheno_rep))
colnames(pheno) = c("sample", "sex", "rep")

# before working with the counts data, make sure it is in
# the right format set row names with a unique version of
# those gene names
row.names(counts) = make.unique(counts$Geneid)
counts[1] = NULL  # remove gene name column so that you can work with the data
dim(counts)  # check out how much you have in your counts file
```

```
## [1] 57133    44
```

```
# Create a new DGEList object
x <- DGEList(counts = counts, lib.size = colSums(counts), norm.factors = rep(1,
    ncol(counts)), samples = colnames(counts), group = pheno$sex,
    genes = row.names(counts), remove.zeros = TRUE)
```

```
## Removing 13345 rows with all zero counts
```

```
dim(x)  # show how much is left after removing all-zero rows
```

```
## [1] 43788    44
```

```
# set up the groups that are possible for sex and replicate
# each sample has two technical replicates marked -1 and -2
# we will store the name of the sample in the rep column so
# we can sum technical replicates
sex = factor(pheno$sex, levels = c("female", "male"))
rep = factor(pheno$rep, levels = c("OBG0158", "OBG0116", "OBG0166",
    "OBG0126", "OBG0132", "OBG0112", "OBG0130", "OBG0111", "OBG0118",
    "OBG0120", "OBG0156", "OBG0115", "OBG0068", "OBG0170", "OBG0178",
    "OBG0133", "OBG0123", "YPOPS0006", "OBG0117", "OBG0122",
    "OBG0053", "OBG0044"))

# add sex and replicate to the samples info table
x$samples$sex = sex
x$samples$rep = rep

# sum counts of technical replicates
x = sumTechReps(x, x$samples$rep)
dim(x)  # show how much is left after combining technical replicates
```

```
## [1] 43788    22
```

```
samplenames = colnames(x)  #save sample names for plotting below

# store sample phenotypes that we are grouping samples by
# (sex)
group = as.factor(x$samples$sex)
```

Data Preprocessing

For differential expression and related analyses, gene expression is rarely considered at the level of raw counts since libraries sequenced at a greater depth will result in higher counts. Rather, it is common practice to transform raw counts onto a scale that accounts for such library size differences. Popular transformations include counts per million (CPM), log2-counts per million (log-CPM), reads per kilobase of transcript per million (RPKM), and fragments per kilobase of transcript per million (FPKM).

Here raw counts are converted to CPM and log-CPM values using the cpm function in edgeR. RPKM values are just as easily calculated as CPM values using the rpkm function in edgeR if gene lengths are available.

CPM values adjust for the fact that some genes were sequenced more than others. A CPM value of 1 for a gene equates to having 20 counts in the sample with the lowest sequencing depth or 76 counts in the sample with the greatest sequencing depth

When the parameter 'log' is set to 'TRUE', the cpm function adds an offset to the CPM values before converting to the log2-scale. By default, the offset is 2/L where 2 is the "prior count" and L is the average library size in millions, so the log-CPM values are related to the CPM values by log2(CPM + 2/L). This calculation ensures that any two read counts with identical CPM values will also have identical log-CPM values.

```
# convert raw counts to CPM and log-CPM values using cpm
# function in edgeR
cpm <- cpm(x)
lcpm <- cpm(x, log = TRUE)
head(lcpm, n = 3)  # peek at top of data after conversion to CPM
```

```
##             OBG0130  YPOPS0006    OBG0115    OBG0123   OBG0120    OBG0126
## DDX11L1   -1.709639 -1.7761956 -2.0945371 -1.5015927 -2.156030 -3.1643621
## WASH7P     3.836692  3.6945108  3.9051960  3.9929867  3.668984  3.7776277
## MIR6859-1 -1.269908 -0.5228436 -0.5730162 -0.5009787 -1.061743 -0.5352145
##              OBG0053   OBG0133    OBG0112    OBG0170    OBG0132     OBG0122
## DDX11L1   -2.2615373 -1.775469 -1.4036384 -1.4735758 -2.0076605 -1.48886294
## WASH7P     3.3330545  3.729148  4.2863650  3.4523338  3.9482792  4.24414889
## MIR6859-1 -0.4459302 -1.040521  0.1181424 -0.2067047 -0.6674575 -0.02157262
##              OBG0068    OBG0158    OBG0116    OBG0117    OBG0118    OBG0178
## DDX11L1   -1.6310545 -0.9793495 -1.2974741 -1.3748060 -1.6162983 -2.2029626
## WASH7P     3.9745365  3.4135137  3.9514332  4.0903176  3.3320196  3.7444534
## MIR6859-1 -0.8890507 -1.6907184 -0.3911365 -0.2678754 -0.9853307 -0.9031626
##             OBG0044    OBG0111   OBG0166    OBG0156
## DDX11L1   -1.947095 -1.8990900 -1.693034 -3.6171784
## WASH7P     3.302069  3.6135735  3.834375  3.7071590
## MIR6859-1 -1.526089 -0.4087761 -1.005613 -0.1784232
```

```
summary(lcpm)  # get an idea of the range of your data after transformation
```

```
##     OBG0130          YPOPS0006         OBG0115          OBG0123
##  Min.   :-4.906   Min.   :-4.906   Min.   :-4.906   Min.   :-4.906
##  1st Qu.:-4.906   1st Qu.:-4.906   1st Qu.:-4.906   1st Qu.:-4.906
##  Median :-2.485   Median :-2.771   Median :-2.479   Median :-2.726
##  Mean   :-1.103   Mean   :-1.178   Mean   :-1.132   Mean   :-1.165
##  3rd Qu.: 2.049   3rd Qu.: 2.038   3rd Qu.: 2.018   3rd Qu.: 2.038
##  Max.   :15.513   Max.   :15.582   Max.   :15.709   Max.   :15.315
##     OBG0120          OBG0126          OBG0053          OBG0133
##  Min.   :-4.906   Min.   :-4.906   Min.   :-4.906   Min.   :-4.906
##  1st Qu.:-4.906   1st Qu.:-4.906   1st Qu.:-4.906   1st Qu.:-4.906
##  Median :-2.956   Median :-2.975   Median :-2.403   Median :-2.939
##  Mean   :-1.340   Mean   :-1.323   Mean   :-1.045   Mean   :-1.347
##  3rd Qu.: 1.693   3rd Qu.: 1.828   3rd Qu.: 2.258   3rd Qu.: 1.701
##  Max.   :15.685   Max.   :15.585   Max.   :15.340   Max.   :15.417
##     OBG0112          OBG0170          OBG0132          OBG0122
##  Min.   :-4.906   Min.   :-4.906   Min.   :-4.906   Min.   :-4.906
##  1st Qu.:-4.906   1st Qu.:-4.906   1st Qu.:-4.906   1st Qu.:-4.906
##  Median :-2.641   Median :-2.471   Median :-2.744   Median :-2.562
##  Mean   :-1.158   Mean   :-1.061   Mean   :-1.208   Mean   :-1.119
##  3rd Qu.: 1.989   3rd Qu.: 2.173   3rd Qu.: 1.985   3rd Qu.: 2.132
##  Max.   :15.577   Max.   :15.025   Max.   :15.257   Max.   :15.308
##     OBG0068          OBG0158          OBG0116          OBG0117
##  Min.   :-4.906   Min.   :-4.906   Min.   :-4.906   Min.   :-4.906
##  1st Qu.:-4.906   1st Qu.:-4.906   1st Qu.:-4.906   1st Qu.:-4.906
##  Median :-2.644   Median :-2.386   Median :-2.509   Median :-2.461
##  Mean   :-1.126   Mean   :-1.004   Mean   :-1.031   Mean   :-1.065
##  3rd Qu.: 2.188   3rd Qu.: 2.237   3rd Qu.: 2.326   3rd Qu.: 2.152
##  Max.   :15.368   Max.   :15.316   Max.   :15.251   Max.   :15.281
##     OBG0118          OBG0178           OBG0044          OBG0111
##  Min.   :-4.906   Min.   :-4.9056   Min.   :-4.906   Min.   :-4.906
##  1st Qu.:-4.906   1st Qu.:-4.9056   1st Qu.:-4.906   1st Qu.:-4.906
##  Median :-2.764   Median :-2.3185   Median :-2.902   Median :-2.807
##  Mean   :-1.233   Mean   :-0.9377   Mean   :-1.212   Mean   :-1.248
##  3rd Qu.: 1.995   3rd Qu.: 2.4677   3rd Qu.: 2.036   3rd Qu.: 1.928
##  Max.   :15.411   Max.   :14.9141   Max.   :15.417   Max.   :15.693
##     OBG0166          OBG0156
##  Min.   :-4.906   Min.   :-4.906
##  1st Qu.:-4.906   1st Qu.:-4.906
##  Median :-2.608   Median :-2.492
##  Mean   :-1.136   Mean   :-1.034
##  3rd Qu.: 2.082   3rd Qu.: 2.228
##  Max.   :15.394   Max.   :14.983
```

```
# calculate the mean and median of the counts (we multiply
# by 1e-6 to represent the mean and median in millions)
L <- mean(x$samples$lib.size) * 1e-06
M <- median(x$samples$lib.size) * 1e-06
c(L, M)  # see what the mean and median are
```

```
## [1] 59.94740 60.10771
```

Removing genes that are lowly expressed

All datasets will include a mix of genes that are expressed and those that are not expressed. Whilst it is of interest to examine genes that are expressed in one condition but not in another, some genes are unexpressed throughout all samples.

Genes that do not have a worthwhile number of reads in any sample should be filtered out of the downstream analyses. There are several reasons for this. From a biological point of view, genes that not expressed at a biologically meaningful level in any condition are not of interest and are therefore best ignored. From a statistical point of view, removing low count genes allows the mean-variance relationship in the data to be estimated with greater reliability and also reduces the number of statistical tests that need to be carried out in downstream analyses looking at differential expression.

```
# filter genes using the filterByExpr function in edgeR
keep.exprs <- filterByExpr(x, group = group)
x <- x[keep.exprs, , keep.lib.sizes = FALSE]
dim(x)  # see how much remains after filtering out low expression genes
```

```
## [1] 20979    22
```

Normalising gene expression distributions

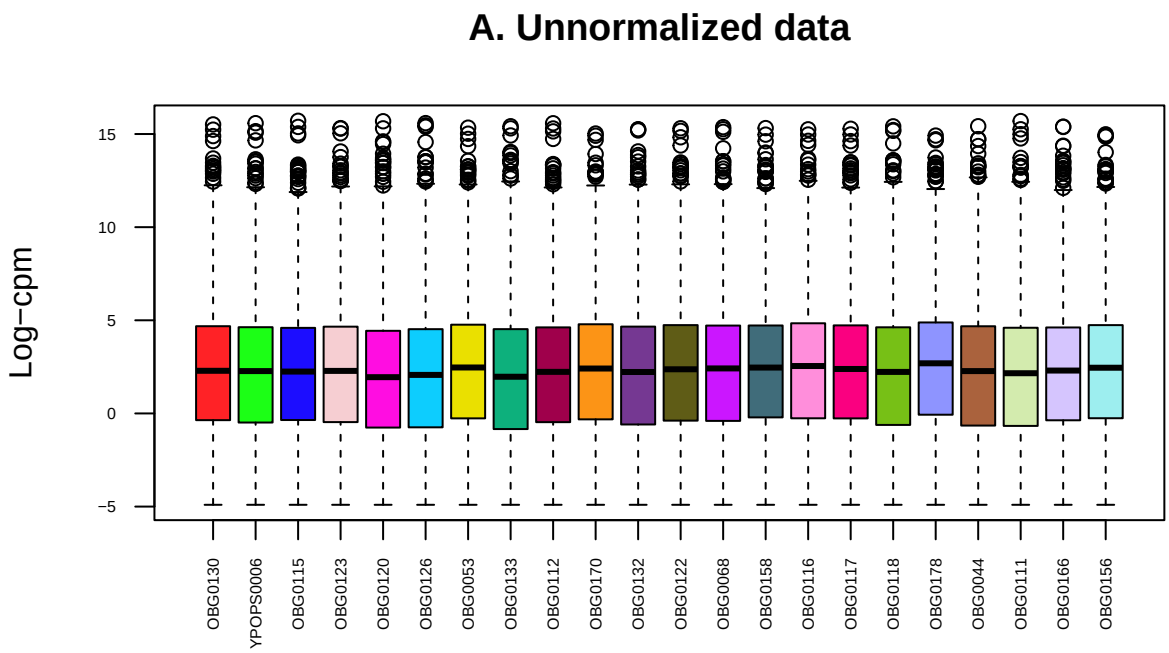
During the sample preparation or sequencing process, external factors that are not of biological interest can affect the expression of individual samples. For example, the temperature in the sequencing center might have been a bit colder on one day than the next and so there might be slight differences in the sequencing data of samples run that day. It is assumed that all samples should have a similar range and distribution of expression values. Any plot showing the per sample expression distributions, such as a density plot or box plot, is useful in determining whether any samples are dissimilar to others.

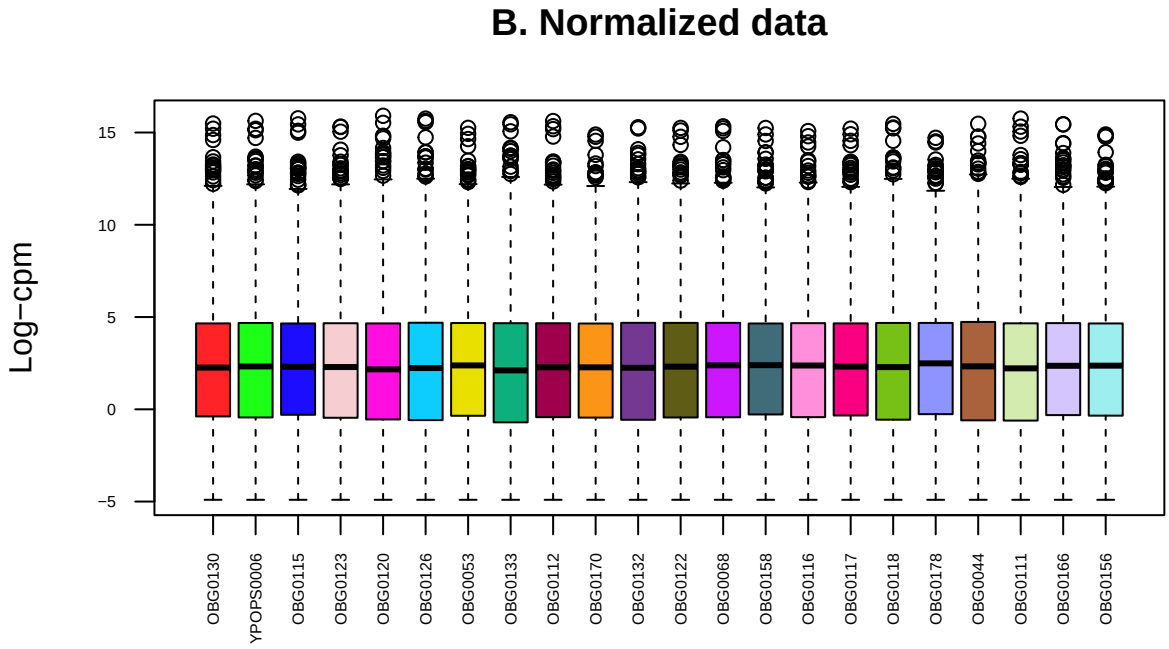
Normalization is required to ensure that the expression distributions of each sample are comparable across all the samples in the experiment. Normalization by the method of trimmed mean of M-values (TMM) (Robinson and Oshlack 2010) is performed using the calcNormFactors function in edgeR. The normalization factors calculated here are used as a scaling factor for the library sizes (the number of reads sequenced in the placenta sample). When working with DGEList objects, these normalization factors are automatically stored in norm.factors column of the sample information table. For this data set, the effect of TMM-normalization is mild. You can see that when you plot the range of data for each sample using box plots before normalization and see that the range of values doesn't fluctuate very much. You can also see this in the data alone because after normalization, the scaling factors are all pretty close to 1.

```
# use calcNormFactors function to calculate how much you
# want to multiply by to make sure the data are in
# comparable range to one another
x_norm <- calcNormFactors(x, method = "TMM")
x_norm$samples$norm.factors  # prints normalization factors
```

```
##  [1] 1.0187768 0.9659880 0.9593744 0.9952353 0.8578531 0.8894797 1.0630144
##  [8] 0.9066295 0.9697092 1.0995953 0.9822915 1.0412218 1.0230835 1.0501671
## [15] 1.1264824 1.0498745 0.9614869 1.1504894 0.9624832 0.9587749 0.9617029
## [22] 1.0637787
```

```r
# Show the expression distribution of samples unnormalized
# vs normalized using box plots

# Custom color palette
palette = createPalette(39, c("#ff0000", "#00ff00", "#0000ff"))

# convert data without normalization to log CPM
lcpm <- cpm(x, log = TRUE)

# create a panel for two box plots to be displayed
par(mfrow = c(2, 1))

# plot data without normalization (gene quantification) **
# run these next two lines together (highlight both lines,
# Run, Run selected lines)**
boxplot(lcpm, las = 2, cex.axis = 0.5, col = as.vector(palette),
    main = "")
title(main = "A. Unnormalized data", ylab = "Log-cpm")

# convert data with normalization to log CPM
lcpm_norm <- cpm(x_norm, log = TRUE)

# plot normalized CPM (gene quantification) ** run these
# next two lines together (highlight both lines, Run, Run
# selected lines)**
boxplot(lcpm_norm, las = 2, cex.axis = 0.5, col = as.vector(palette),
    main = "")
title(main = "B. Normalized data", ylab = "Log-cpm")
```


A. Unnormalized data
Log−cpm
15
10
5
0
−5
OBG0130
YPOPS0006
OBG0115
OBG0123
OBG0120
OBG0126
OBG0053
OBG0133
OBG0112
OBG0170
OBG0132
OBG0122
OBG0068
OBG0158
OBG0116
OBG0117
OBG0118
OBG0178
OBG0044
OBG0111
OBG0166
OBG0156


B. Normalized data
Log−cpm
15
10
5
0
−5
OBG0130
YPOPS0006
OBG0115
OBG0123
OBG0120
OBG0126
OBG0053
OBG0133
OBG0112
OBG0170
OBG0132
OBG0122
OBG0068
OBG0158
OBG0116
OBG0117
OBG0118
OBG0178
OBG0044
OBG0111
OBG0166
OBG0156

Unsupervised clustering of samples (MDS plot)

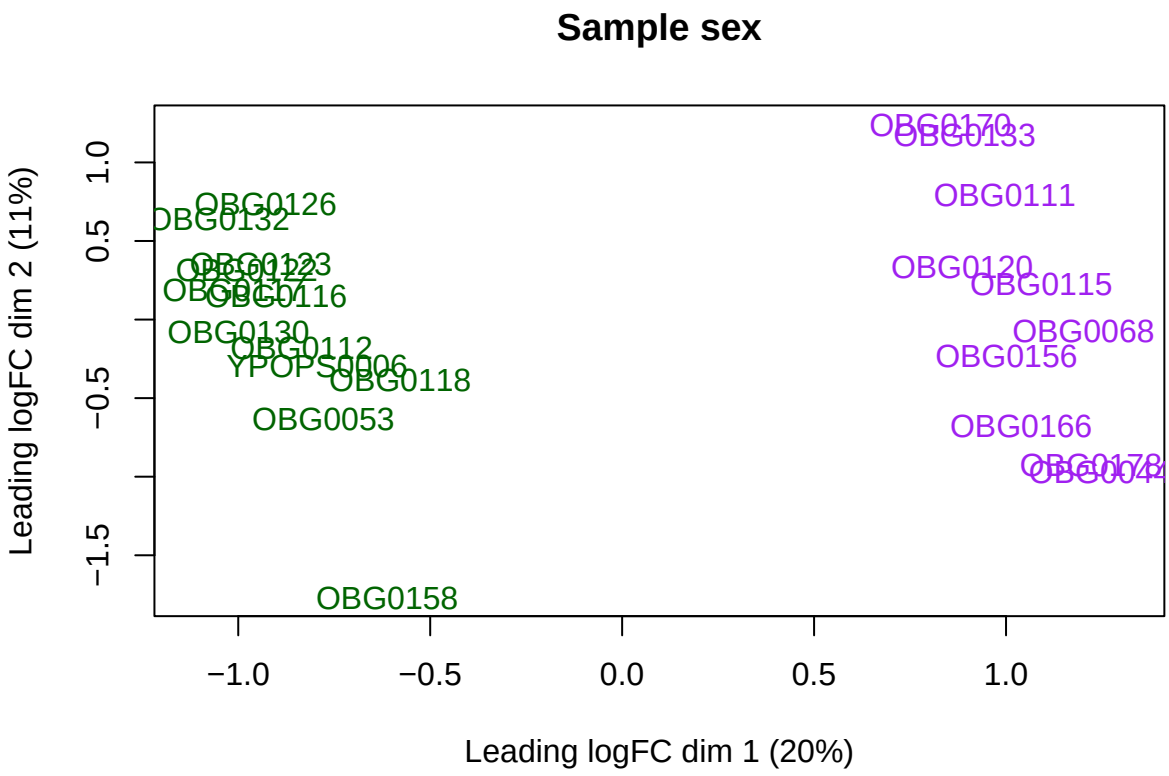
A great exploratory plot that quickly shows the main trends in your gene expression data is the multi-dimensional scaling (MDS) plot. An MDS plot uses linear combinations of genes with similar gene expression profiles to plot multidimensional data (gene expression profiles) in a 2D space, such that samples that are more close together have more similar gene expression profiles and points further apart have more distinct gene expression profiles. The plot shows similarities and dissimilarities between samples in an unsupervised manner so that one can have an idea of the extent to which differential expression can be detected before carrying out formal tests. Ideally, samples would cluster well within the primary condition of interest, and any sample straying far from its group could be identified and followed up for sources of error or extra variation. Technical replicates should be very close to one another, though in our case we summed technical replicates before doing MDS plots.

With our data, we should see the female (XX) placenta samples cluster away from the male (XY) placenta samples, indicating that there are sex differences in gene expression in the placenta in full term pregnancies. If we didn't see that the females and males separate

```
# label and color sex
col.group <- group  # group variable is set to sex above
levels(col.group)  # to confirm that we have 'female' and 'male' samples
```

```
## [1] "female" "male"
```

```
levels(col.group) = c("purple", "darkgreen")  # map colors to sex

# make MDS plot using normalized log CPM values
col.group <- as.character(col.group)
plotMDS(lcpm_norm, labels = samplenames, col = col.group)
title(main = "Sample sex")
```



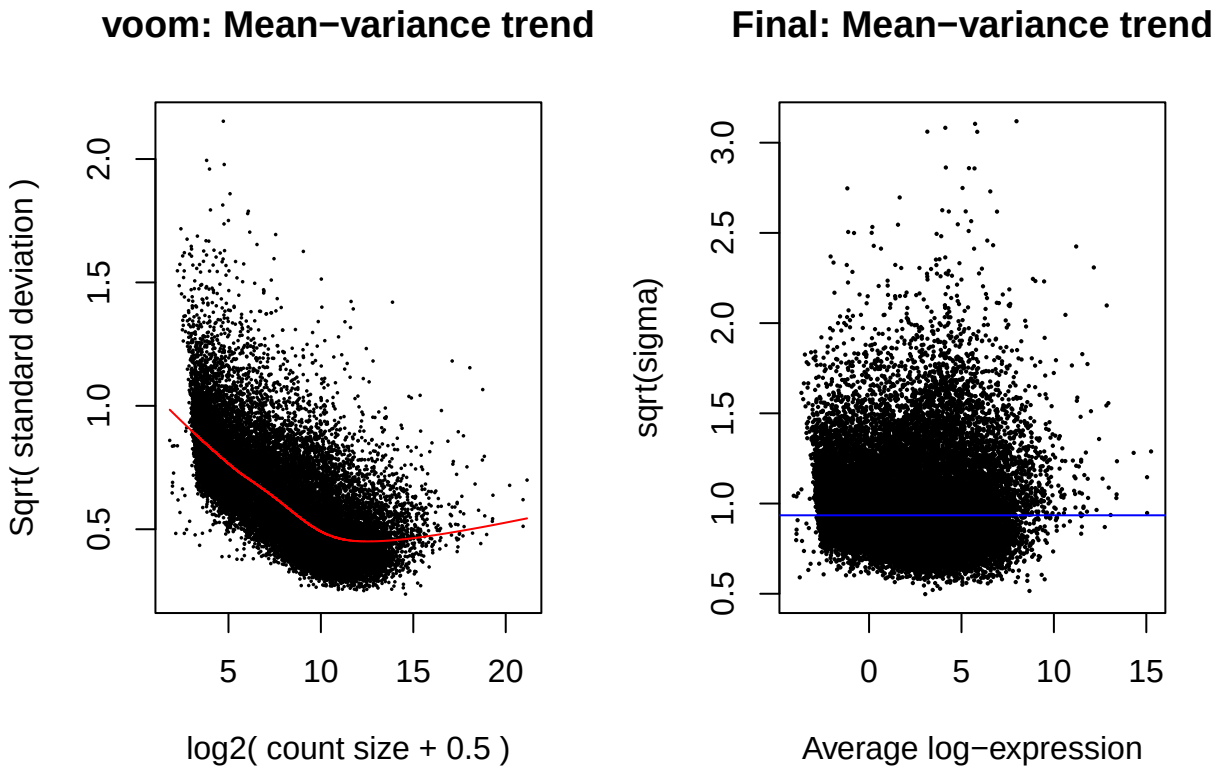
Differential Expression Analysis

In order to use linear modeling to determine which genes are differentially expressed, we first set up a design matrix relating our samples to our variable of interest (sex) and contrasts to make the comparisons we are interested in (female vs male).

In the R package limma, linear modelling is carried and assumed to be normally distributed and the mean-variance relationship is accommodated using precision weights calculated by the voom function. When operating on a DGEList object, voom converts raw counts to log-CPM values by automatically extracting library sizes and normalization factors from x itself. Voom transformation corrects for the fact that more variance is observed in genes with lower average expression so that we can get equally precise results from genes with lower expression as we do from genes with higher expression. This is important because we want to catch sex differences throughout the range of gene expression.

```
# Create the design matrix indicating sex as groups
design <- model.matrix(~0 + group)  # the variable group was set to hold sample sex above
colnames(design) <- gsub("group", "", colnames(design))

# Set up the comparisons we are interested in to determine
# sex differences
contr.matrix <- makeContrasts(FemalevsMale = female - male, levels = colnames(design))

# set up panel to show data pre and post voom
par(mfrow = c(1, 2))
```

```
# apply weights because log CMP can vary more in genes with
# low counts and display data before voom transformation
v <- voom(x_norm, design, plot = TRUE)

# fit the variance to the groups (sex)
vfit <- lmFit(v, design)
vfit <- contrasts.fit(vfit, contrasts = contr.matrix)
efit <- eBayes(vfit)

# plot fitted data post voom transformation
plotSA(efit, main = "Final: Mean-variance trend")
```




```
# write out modeling data
write.fit(efit, file = paste0(result_file_label, "_voom_results.txt"))
```

Examining individual DE genes from top to bottom.

```
# Apply modeling to get differential expression genes
dt = decideTests(efit)
summary(dt)
```

```
##        FemalevsMale
```

```
## Down             37
## NotSig        20921
## Up               21
```

```
dt_BH = decideTests(efit, adjust.method = "BH", p.value = 0.05,
    lfc = 1)
summary(dt_BH)
```

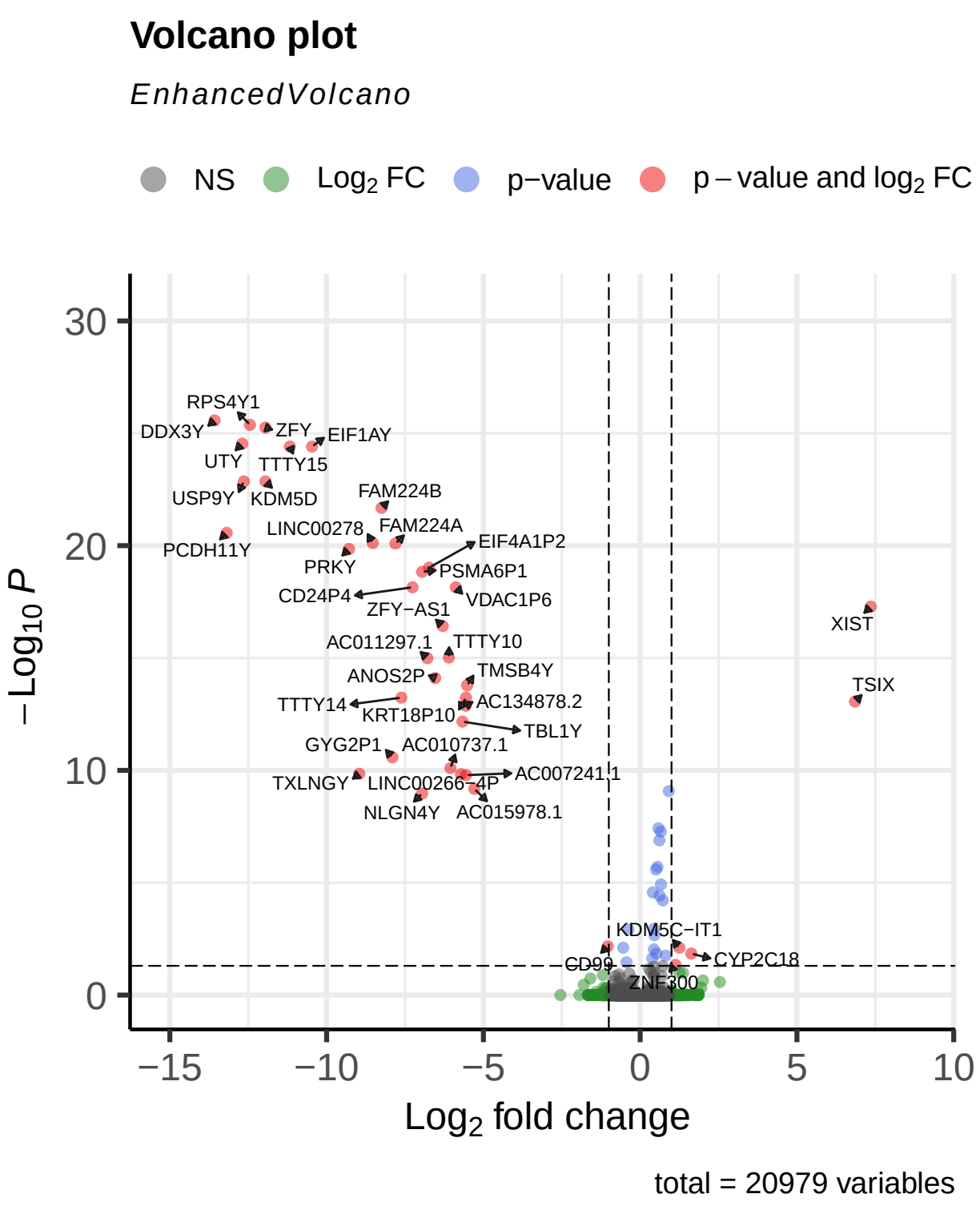
```
##        FemalevsMale
## Down             34
## NotSig        20940
## Up                5
```

```
dt_BH = as.data.frame(dt_BH)

upregulated_genes = subset(dt_BH, FemalevsMale == 1)
downregulated_genes = subset(dt_BH, FemalevsMale == -1)

# print out the number of genes that are differentially
# expressed
write.csv(data.frame(summary(decideTests(efit))), file = paste0(result_file_label,
    "_num_significant_genes_standardvoom.csv"))
write.csv(data.frame(summary(decideTests(efit, adjust.method = "BH",
    p.value = 0.05, lfc = 1))), file = paste0(result_file_label,
    "_num_significant_genes_BHcorrected_pv05_lfc1.csv"))

# calculate stats for all genes' expression differences
# between females and males
female.vs.male <- topTable(efit, coef = 1, n = Inf)
write.csv(female.vs.male, file = paste0(result_file_label, "_dge_femalevsmale_all_expressed_genes.csv"))
```

Volcano plot

A volcano plot shows log fold change on the x-axis and log p-value on the y axis. Log fold change is given a positive sign if average expression in females is higher or negative if average expression in males is higher. This brings genes that are significantly differentially expressed away from both axes. The EnhancedVolcano function shown here does a lot of labeling and coloring automatically, which is pleasing to the eye and very convenient for making figures for papers and presentations.

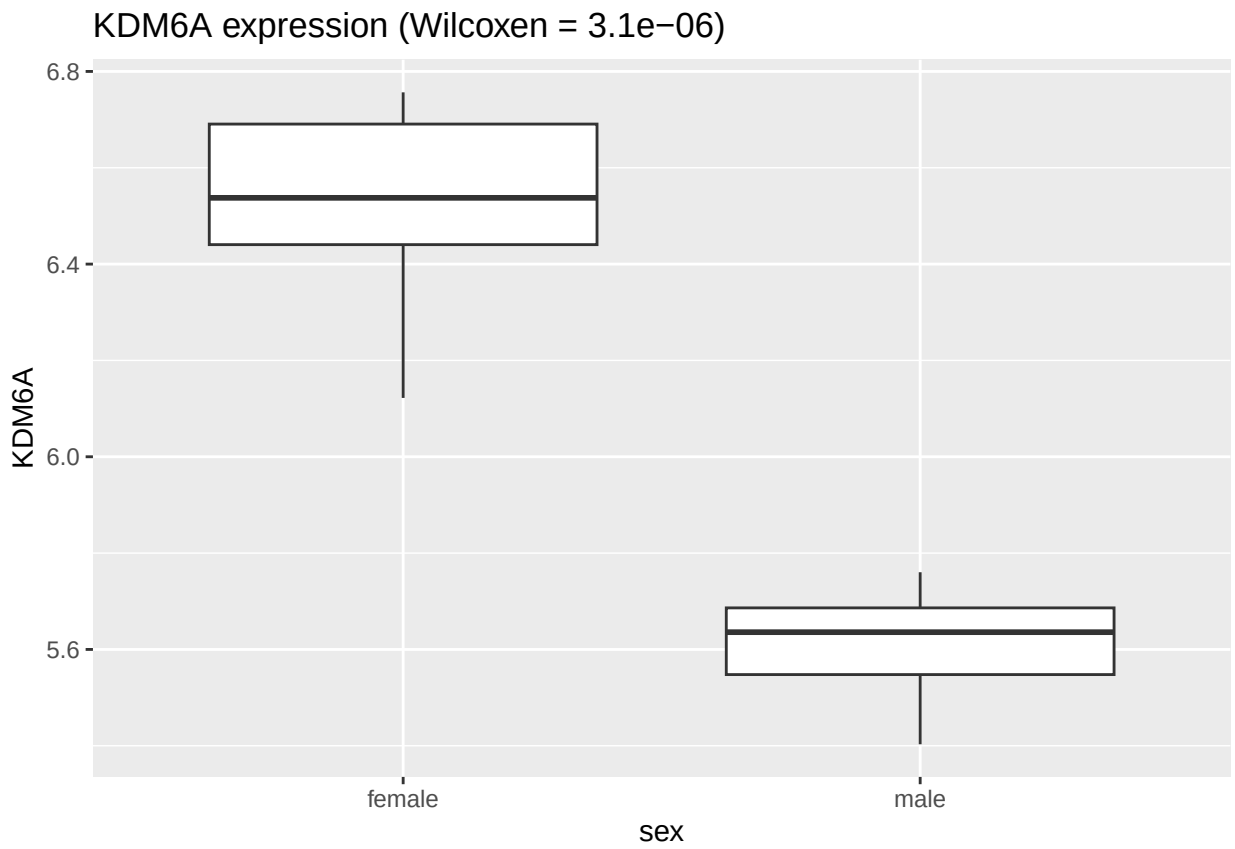
```
# volcano plot
EnhancedVolcano(female.vs.male, lab = female.vs.male$genes, x = "logFC",
    y = "adj.P.Val", labSize = 3, FCcutoff = 1, pCutoff = 0.05,
    drawConnectors = TRUE)
```


Volcano plot
EnhancedVolcano
NS
Log2 FC
p−value
p – value and log2 FC
30
20
10
0
−Log10 P
−15
−10
−5
0
5
10
Log2 fold change
RPS4Y1
DDX3Y
ZFY
EIF1AY
UTY
TTTY15
USP9Y
KDM5D
FAM224B
LINC00278
FAM224A
PCDH11Y
EIF4A1P2
PRKY
PSMA6P1
CD24P4
VDAC1P6
ZFY−AS1
AC011297.1
TTTY10
ANOS2P
TMSB4Y
TTTY14
AC134878.2
KRT18P10
TBL1Y
GYG2P1
AC010737.1
TXLNGY
LINC00266−4P
AC007241.1
NLGN4Y
AC015978.1
XIST
TSIX
KDM5C−IT1
CD99
CYP2C18
ZNF300
total = 20979 variables

Expression box plots

Here we will plot expression of specific genes in the female and male placentas so we can see to what extent they are differentially expressed. You can do this for a single gene of interest, or loop through a list of genes. You can comment out the code for a single gene or the code for a list of genes based on what you want to show.

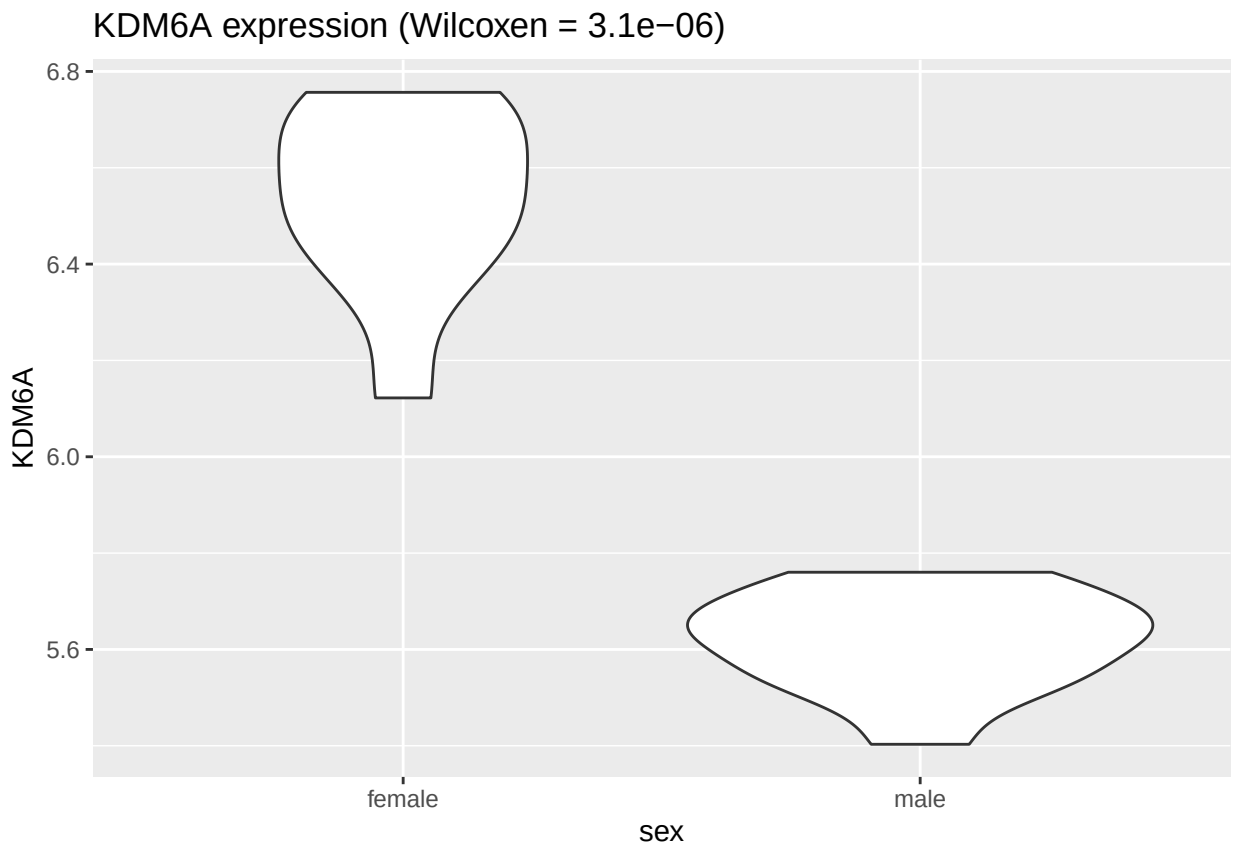
```
# boxplot of expression of single gene of interest

# store the name of the gene you want to plot
gene_of_interest = "KDM6A"

# select the normalized lcpm expression values for that
# gene
norm_data_gene = as.data.frame(lcpm_norm[rownames(lcpm_norm) ==
    gene_of_interest])
rownames(norm_data_gene) = colnames(lcpm_norm)
# set the sex so we know which samples are female vs male
norm_data_gene$sex = x_norm$samples$sex
# set the column names for our data
colnames(norm_data_gene) = c(gene_of_interest, "sex")

# calculate the p-value of the expression of the gene in
# females vs males so we can display it on our plot
calc_pval = compare_means(as.formula(paste0(gene_of_interest,
    "~ sex")), norm_data_gene)
adj_pval = calc_pval$p.adj

# use ggplot to create a box plot with the gene name and
# adjusted p-value for differential expression in the title

p = ggplot(norm_data_gene, aes(x = !!sym(gene_of_interest), y = sex)) +
    geom_boxplot() + coord_flip() + ggtitle(paste0(gene_of_interest,
    " expression (Wilcoxen = ", adj_pval, ")"))
plot(p)
```




```
# save the plot in a pdf so we can use it later on in
# papers or presentations
ggsave(plot = p, filename = paste0(result_file_label, "_boxplot_",
    gene_of_interest, ".pdf"))
```

```
## Saving 6.5 x 4.5 in image
```

```
# use ggplot to create a violin plot with the gene name and
# adjusted p-value for differential expression in the title
# (same as above but with geom_violin)

p = ggplot(norm_data_gene, aes(x = !!sym(gene_of_interest), y = sex)) +
    geom_violin() + coord_flip() + ggtitle(paste0(gene_of_interest,
    " expression (Wilcoxen = ", adj_pval, ")"))
plot(p)
```

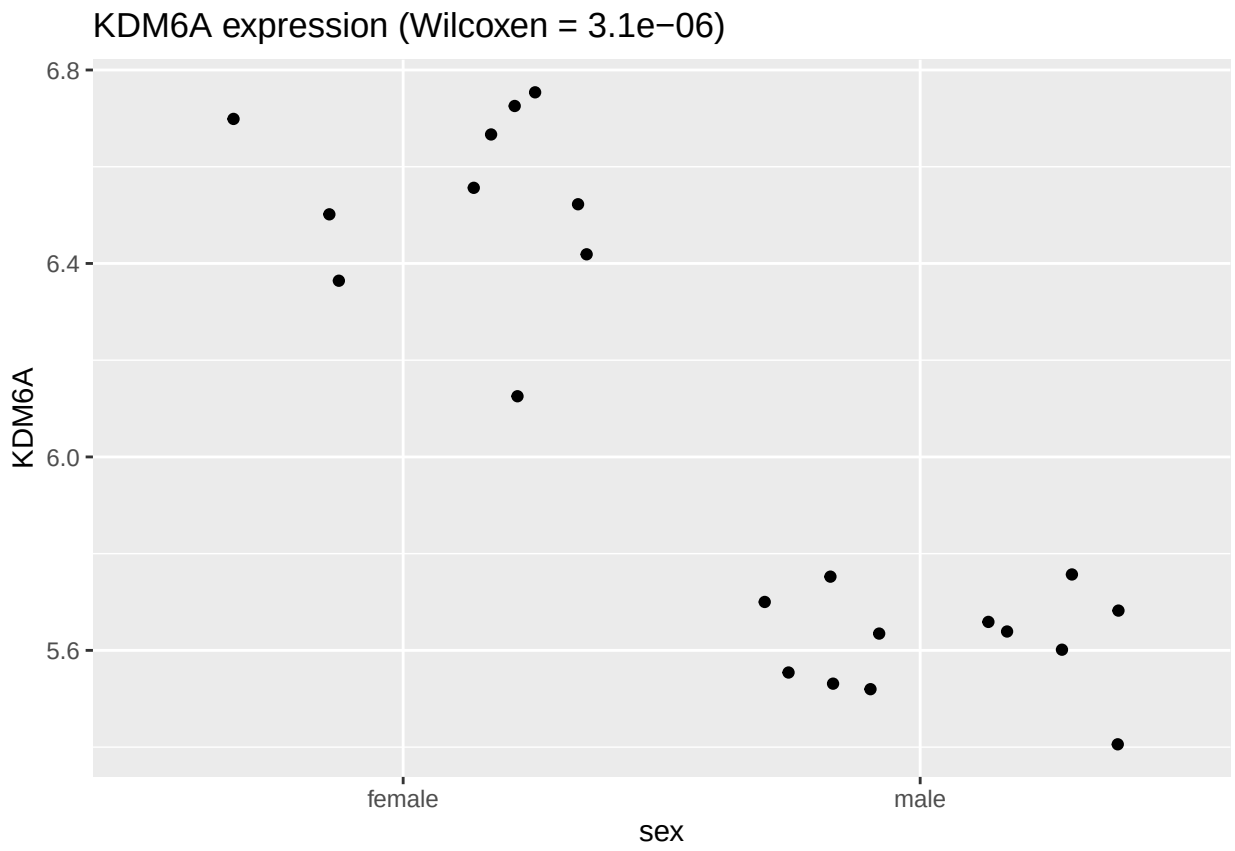



```
# add a box plot to the middle to visualize the mean and
# quartiles more easily
p = ggplot(norm_data_gene, aes(x = !!sym(gene_of_interest), y = sex)) +
    geom_violin() + coord_flip() + ggtitle(paste0(gene_of_interest,
    " expression (Wilcoxen = ", adj_pval, ")")) + geom_boxplot(width = 0.1)
plot(p)
```

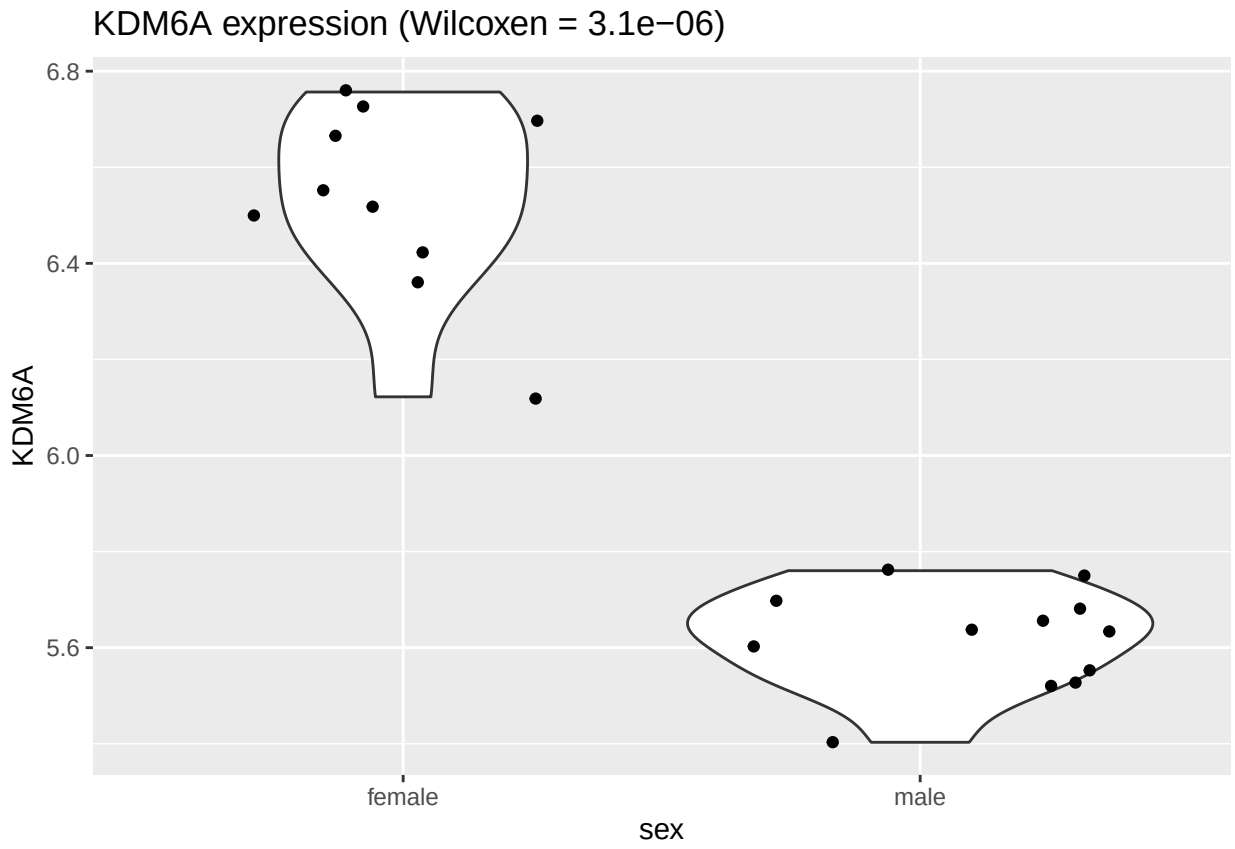

```
# save the violin plot in a pdf
ggsave(plot = p, filename = paste0(result_file_label, "_violinboxplot_",
    gene_of_interest, ".pdf"))
```

```
## Saving 6.5 x 4.5 in image
```

```
# use ggplot to create a jitter plot with the gene name and
# adjusted p-value for differential expression in the title
# (same as above but with geom_jitter)

p = ggplot(norm_data_gene, aes(x = !!sym(gene_of_interest), y = sex)) +
    geom_jitter() + coord_flip() + ggtitle(paste0(gene_of_interest,
    " expression (Wilcoxen = ", adj_pval, ")"))
plot(p)
```




```
# to make it easier to read a jitter plot, it is often
# combined with a violin plot
p = ggplot(norm_data_gene, aes(x = !!sym(gene_of_interest), y = sex)) +
    geom_violin() + geom_jitter() + coord_flip() + ggtitle(paste0(gene_of_interest,
    " expression (Wilcoxen = ", adj_pval, ")"))
plot(p)
```




```
# save the violin jitter plot in a pdf
ggsave(plot = p, filename = paste0(result_file_label, "_violinjitterplot_",
    gene_of_interest, ".pdf"))
```

```
## Saving 6.5 x 4.5 in image
```

```
# we can add a box plot to the center too if we like
p = ggplot(norm_data_gene, aes(x = !!sym(gene_of_interest), y = sex)) +
    geom_violin() + geom_jitter() + coord_flip() + ggtitle(paste0(gene_of_interest,
    " expression (Wilcoxen = ", adj_pval, ")")) + geom_boxplot(width = 0.1)
plot(p)
```

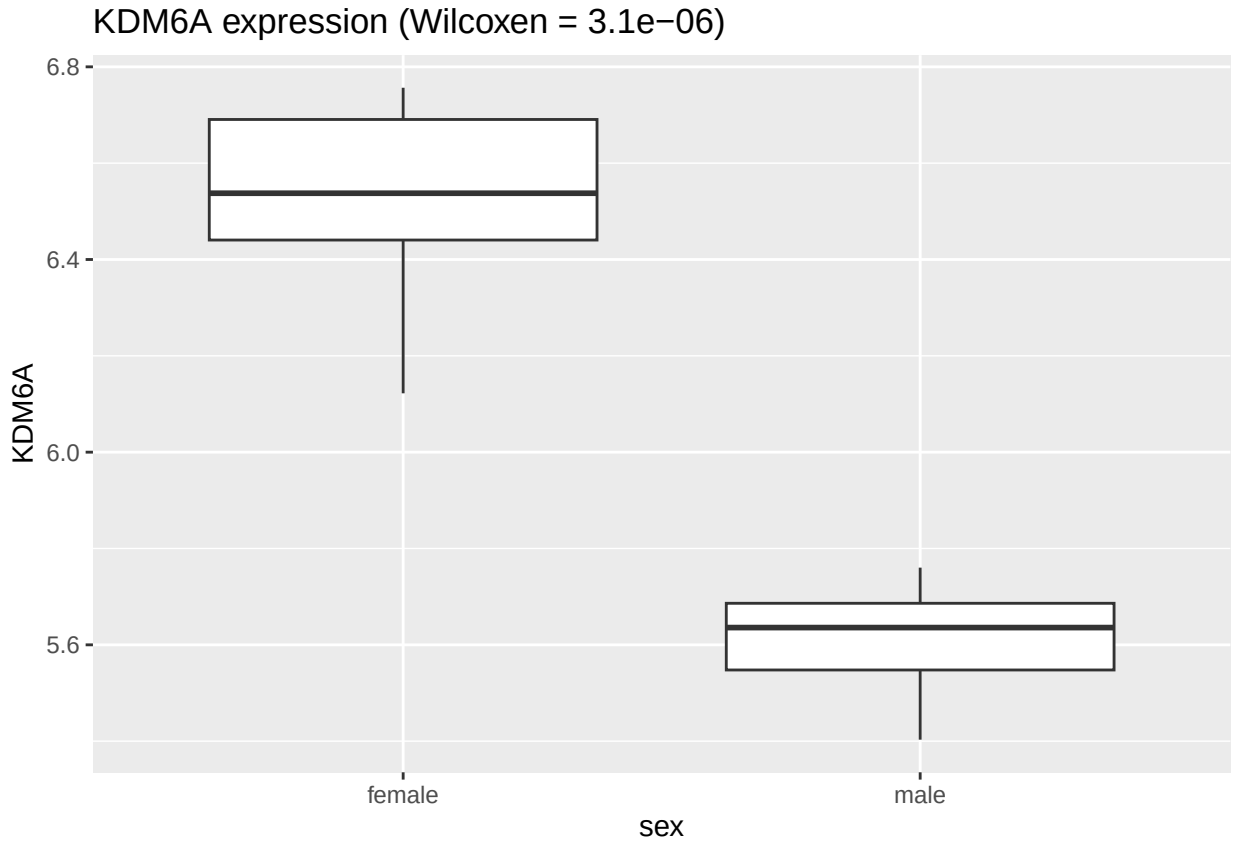

```
# save the violin jitter box plot in a pdf
ggsave(plot = p, filename = paste0(result_file_label, "_violinjitterboxplot_",
    gene_of_interest, ".pdf"))
```

```
## Saving 6.5 x 4.5 in image
```

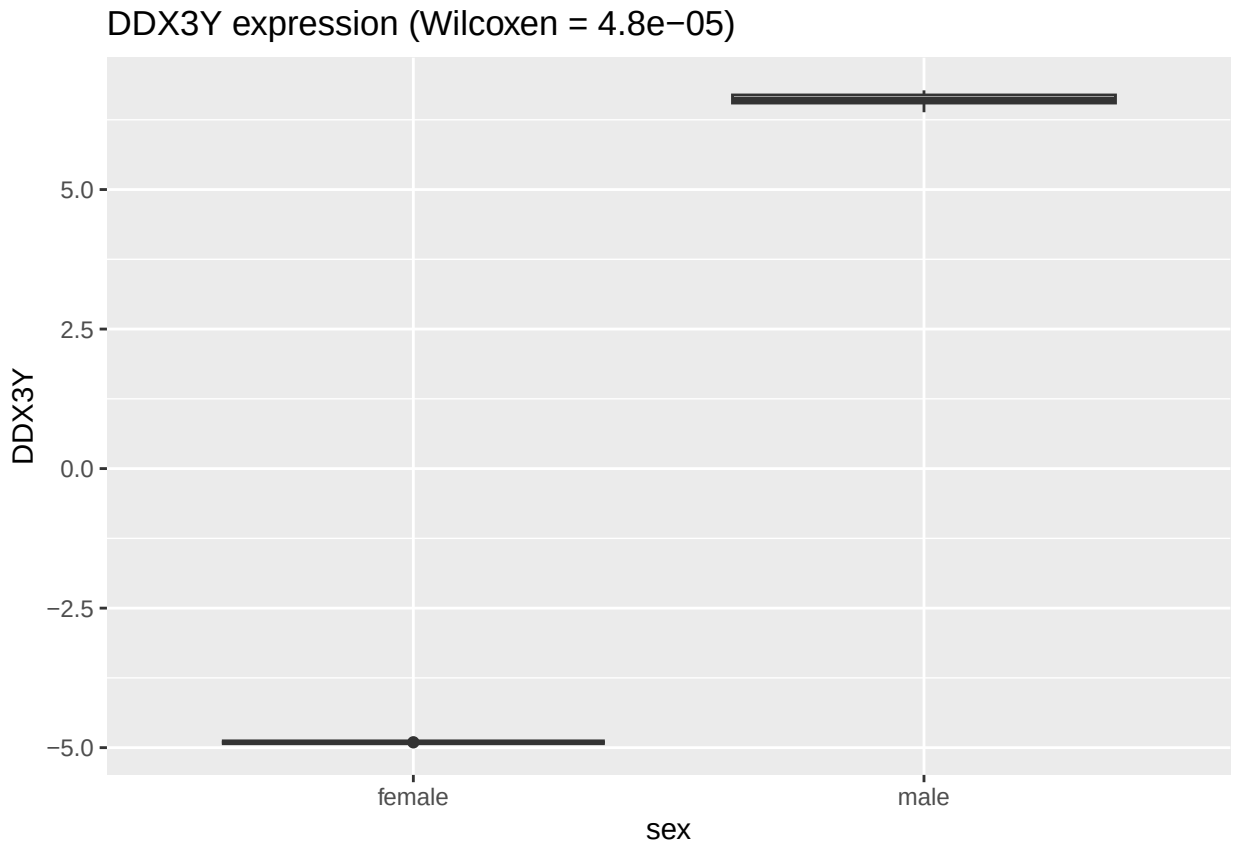
```
# To make plots of list of genes of interest, iterate
# through list of genes doing the same thing we did for one
# gene using a loop in R

# set the list of genes we are interested in looking at
genes_of_interest = c("KDM6A", "DDX3Y", "USP9Y")

# iterate over that list plotting expression as box plots
# (for example) by sex
for (gene_of_interest in genes_of_interest) {
    norm_data_gene = as.data.frame(lcpm_norm[rownames(lcpm_norm) ==
        gene_of_interest])
    rownames(norm_data_gene) = colnames(lcpm_norm)
    norm_data_gene$sex = x_norm$samples$sex
    colnames(norm_data_gene) = c(gene_of_interest, "sex")

    calc_pval = compare_means(as.formula(paste0(gene_of_interest,
        "~ sex")), norm_data_gene)
    adj_pval = calc_pval$p.adj
```

```
    p = ggplot(norm_data_gene, aes(x = !!sym(gene_of_interest),
        y = sex)) + geom_boxplot() + coord_flip() + ggtitle(paste0(gene_of_interest,
        " expression (Wilcoxen = ", adj_pval, ")"))
    plot(p)
    ggsave(plot = p, filename = paste0(result_file_label, "_boxplot_",
        gene_of_interest, ".pdf"))
}
```

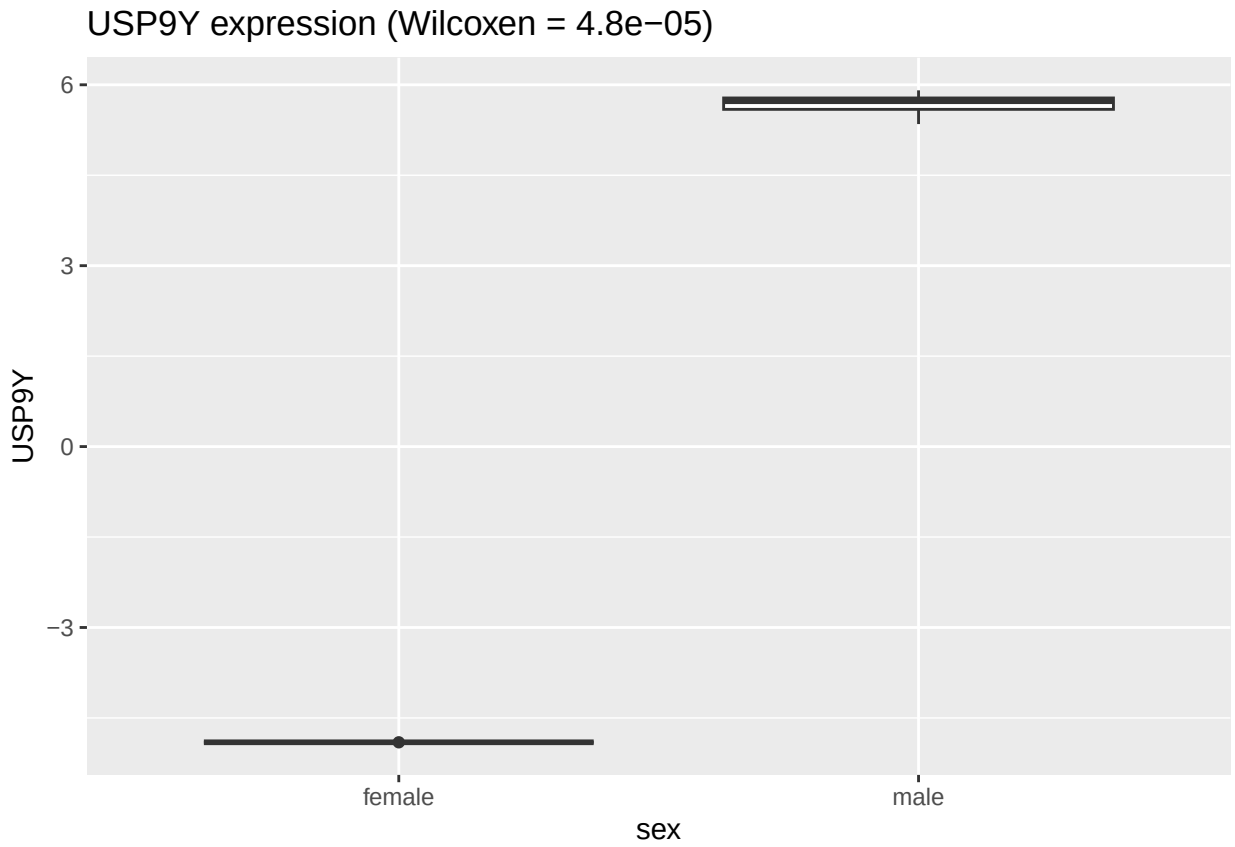
```
## Saving 6.5 x 4.5 in image
```




```
## Saving 6.5 x 4.5 in image
```




```
## Saving 6.5 x 4.5 in image
```



List all the packages used for future reference

```
sessionInfo()
```

```
## R version 4.2.1 (2022-06-23 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19044)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] edgeR_3.38.4           limma_3.52.4           EnhancedVolcano_1.14.0
##  [4] ggrepel_0.9.3          BiocManager_1.30.22    ggpubr_0.6.0
```

```
##  [7] RColorBrewer_1.1-3     Polychrome_1.5.1       ggplot2_3.4.3
## [10] gplots_3.1.3
##
## loaded via a namespace (and not attached):
##  [1] gtools_3.9.4         tidyselect_1.2.0     locfit_1.5-9.8
##  [4] xfun_0.40            purrr_1.0.2          lattice_0.21-8
##  [7] carData_3.0-5        colorspace_2.1-0     vctrs_0.6.3
## [10] generics_0.1.3       htmltools_0.5.6      yaml_2.3.7
## [13] utf8_1.2.3           rlang_1.1.1          pillar_1.9.0
## [16] glue_1.6.2           withr_2.5.0          lifecycle_1.0.3
## [19] munsell_0.5.0        ggsignif_0.6.4       gtable_0.3.3
## [22] ragg_1.2.5           caTools_1.18.2       evaluate_0.21
## [25] labeling_0.4.2       knitr_1.43           fastmap_1.1.1
## [28] fansi_1.0.4          highr_0.10           broom_1.0.5
## [31] Rcpp_1.0.11          KernSmooth_2.23-22   scales_1.2.1
## [34] backports_1.4.1      formatR_1.14         scatterplot3d_0.3-44
## [37] abind_1.4-5          systemfonts_1.0.4    farver_2.1.1
## [40] textshaping_0.3.6    digest_0.6.33        rstatix_0.7.2
## [43] dplyr_1.1.2          grid_4.2.1           cli_3.6.1
## [46] tools_4.2.1          bitops_1.0-7         magrittr_2.0.3
## [49] tibble_3.2.1         tidyr_1.3.0          car_3.1-2
## [52] pkgconfig_2.0.3      rmarkdown_2.24       rstudioapi_0.15.0
## [55] R6_2.5.1             compiler_4.2.1
```