Comparing Two Differential Expression Profiles

Seema Plaisier

11/21/22

Comparing untrimmed vs trimmed data results

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

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

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

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

```
## Loading required package: ggVennDiagram
```

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

```
## Loading required package: grid
```

```
library(ggplot2)
library(ggVennDiagram)
library(grid)
```

Set working directory and data directories

Your working directory is where all your output files will be stored.

You will indicate where your untrimmed results are (which directory). This will be the result of running DE_Pipeline_UntrimmedData.Rmd in Week 3.

You will indicate where your trimmed results are (which directory). This will be the result of modifying DE_Pipeline_UntrimmedData.Rmd to take your assigned trimmed data set. You can run that in the directory you are currently running out of so that your trimmed data is in the same place you are analyzing it from.

```r
# 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/Week 4/"
setwd(working_directory)

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

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

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

```
##  [1] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4/"
##  [2] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//CompareTwo"
##  [3] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//DEG"
##  [4] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//draft"
##  [5] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//draft/stats"
##  [6] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//draft/stats/trimq_0_minlen_10"
##  [7] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//draft/stats/trimq_0_minlen_30"
##  [8] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//draft/stats/trimq_0_minlen_75"
##  [9] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//draft/stats/trimq_10_minlen_10"
## [10] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//draft/stats/trimq_10_minlen_30"
## [11] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//draft/stats/trimq_10_minlen_75"
## [12] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//draft/stats/trimq_30_minlen_10"
## [13] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//draft/stats/trimq_30_minlen_30"
## [14] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//draft/stats/trimq_30_minlen_75"
## [15] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//draft/stats/trimq_NONE_minlen_NONE"
## [16] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//female_vs_male"
## [17] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//female_vs_male/all_trimmed"
## [18] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//female_vs_male/trimq_0_minlen_10"
## [19] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//female_vs_male/trimq_0_minlen_30"
## [20] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//female_vs_male/trimq_0_minlen_75"
## [21] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//female_vs_male/trimq_10_minlen_10"
## [22] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//female_vs_male/trimq_10_minlen_30"
## [23] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//female_vs_male/trimq_10_minlen_75"
## [24] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//female_vs_male/trimq_30_minlen_10"
## [25] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//female_vs_male/trimq_30_minlen_30"
## [26] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//female_vs_male/trimq_30_minlen_75"
## [27] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//female_vs_male/trimq_NONE_minlen_NONE"
## [28] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//geneCounts"
## [29] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//geneCounts/trimq_0_minlen_10"
## [30] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//geneCounts/trimq_0_minlen_30"
## [31] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//geneCounts/trimq_0_minlen_75"
## [32] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//geneCounts/trimq_10_minlen_10"
## [33] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//geneCounts/trimq_10_minlen_30"
## [34] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//geneCounts/trimq_10_minlen_75"
## [35] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//geneCounts/trimq_30_minlen_10"
## [36] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//geneCounts/trimq_30_minlen_30"
## [37] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//geneCounts/trimq_30_minlen_75"
## [38] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//geneCounts/trimq_NONE_minlen_NONE"
## [39] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//stats"
## [40] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//stats/trimq_0_minlen_10"
## [41] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//stats/trimq_0_minlen_30"
## [42] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//stats/trimq_0_minlen_75"
## [43] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//stats/trimq_10_minlen_10"
## [44] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//stats/trimq_10_minlen_30"
## [45] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//stats/trimq_10_minlen_75"
## [46] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//stats/trimq_30_minlen_10"
## [47] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//stats/trimq_30_minlen_30"
## [48] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//stats/trimq_30_minlen_75"
## [49] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4//stats/trimq_NONE_minlen_NONE"
```

```r
# set the directory where your analysis files for
# differentially expressed genes using untrimmed data is
untrimmed_directory = "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4/"

# set the directory where your analysis files for
# differentially expressed genes using your assigned
# trimmed data is
trimmed_directory = "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4/"

# set the directories that contain the stats for alignment
# of the untrimmed data set (trimq_NONE_minlen_NONE) and
# your assigned trimmed data set

untrimmed_stats_directory = paste0(untrimmed_directory, "stats/trimq_NONE_minlen_NONE/")
trimmed_stats_directory = paste0(trimmed_directory, "stats/trimq_30_minlen_75/")
```

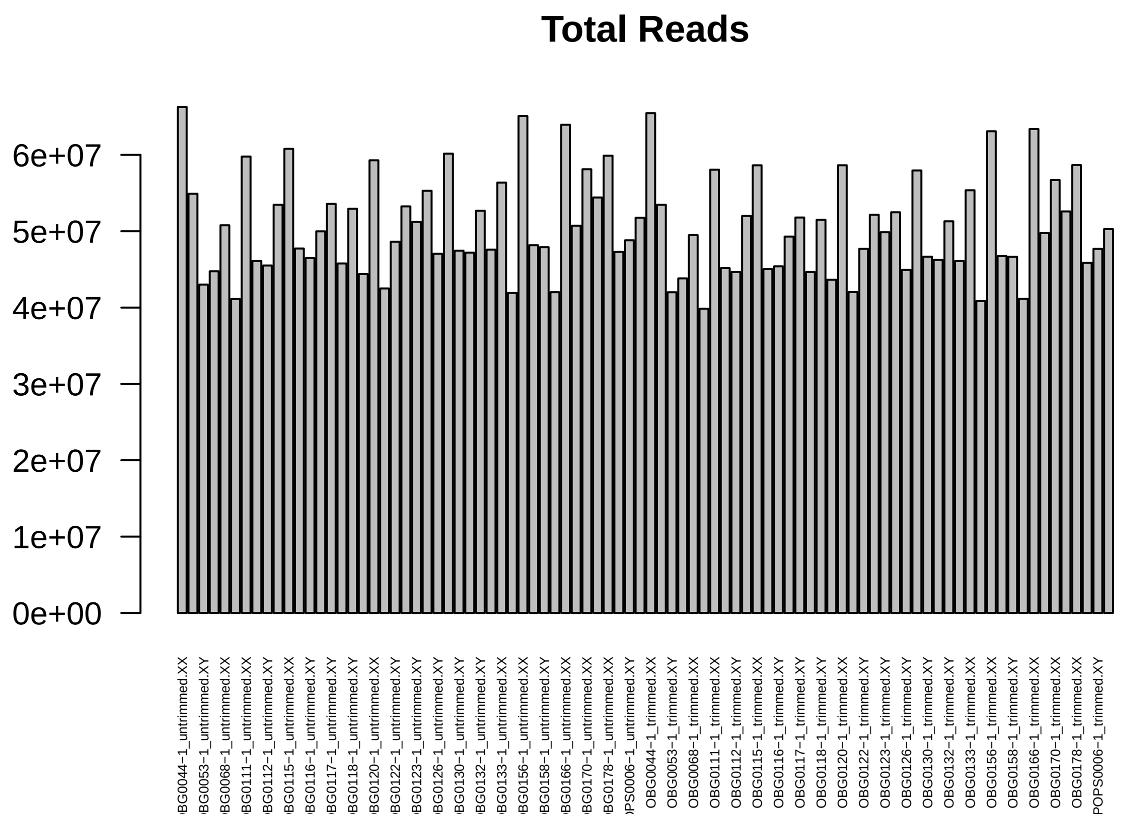
Comparing total number of reads after trimming

We use trimming to remove low quality reads and adapter sequences. There will be a certain amount of loss with any amount of trimming. We want to assure that the number of reads we have remaining gives us sufficient read coverage. The Wilson lab recommends at least 40 million reads for human transcriptome RNAseq samples. In this chunk, we will read mapping statistics files acquired using alignment analysis tools ('samtools flagstat') where the number of reads is reported as 'read1'. If you go into the stats directory inside the untrimmed data directory (trimq_NONE_minlen_NONE) or your assigned trimmed data directory, you will find files containing "flagstats" in the file name. Open one of those files to get an idea of what we are reading here. The 'samtools flagstat' tool counts flags set during alignment, details on these flags are given here if you are curious: http://www.htslib.org/doc/samtools-flagstat.html

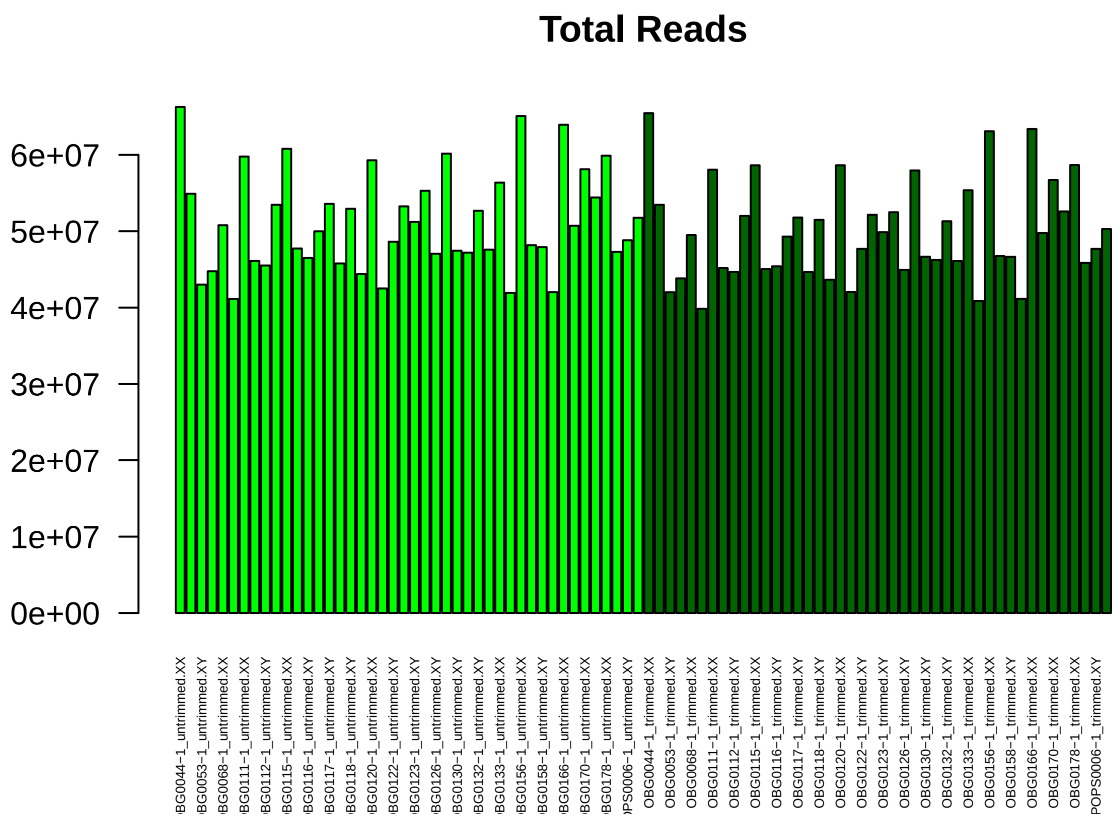
For this application, it is important to know that the total number of reads we are interested in is given by the line containing ‘read1’, which will be equal to ‘read2’ since our samples are paired-end reads.

```r
# starting with the untrimmed data...

# create a list to store the total number of reads for each
# stats file made after alignment
total_reads_untrimmed = list()

# keep a list of sample names for the x-axis
sample_list_untrimmed = list()

# list the files in the stats directory for the untrimmed
# data
untrimmed_stats_files = list.files(untrimmed_stats_directory)

# find the ones that contain rdgrp.flagstats.txt these were
# generated after alignment to help us see how many reads
# were available, how many of those mapped well to the
# reference genome, and scan reads for duplication or
# mismatched forward and reverse reads
rd_stats_files_untrimmed = untrimmed_stats_files[grepl("rdgrp.flagstats.txt",
    untrimmed_stats_files)]

# read through all the rdgrp.flagstats.txt files
for (rd_stats_file in rd_stats_files_untrimmed) {

    # read the lines of text in each file need to add the
    # directory so R can find the file outside your working
    # directory
    stats_info = readLines(paste0(untrimmed_stats_directory,
        rd_stats_file))

    # find the entry for 'Total reads'
    total_reads = stats_info[grepl("read1", stats_info)]

    # replace the text ' + 0 read1' in the line containing
    # total reads
    total_reads_found = strsplit(total_reads[1], " ")
    total_reads_found = total_reads_found[[1]][1]

    # convert it to a number and add it to our list
    total_reads_untrimmed = append(total_reads_untrimmed, as.numeric(total_reads_found))

    # shorten the file name and add that to our list
    sample_list_untrimmed = append(sample_list_untrimmed, gsub(".sort.mkdup.rdgrp.flagstats.txt",
        "", rd_stats_file))
}

# do the same thing for the stats files for the trimmed
# data
total_reads_trimmed = list()
sample_list_trimmed = list()
```

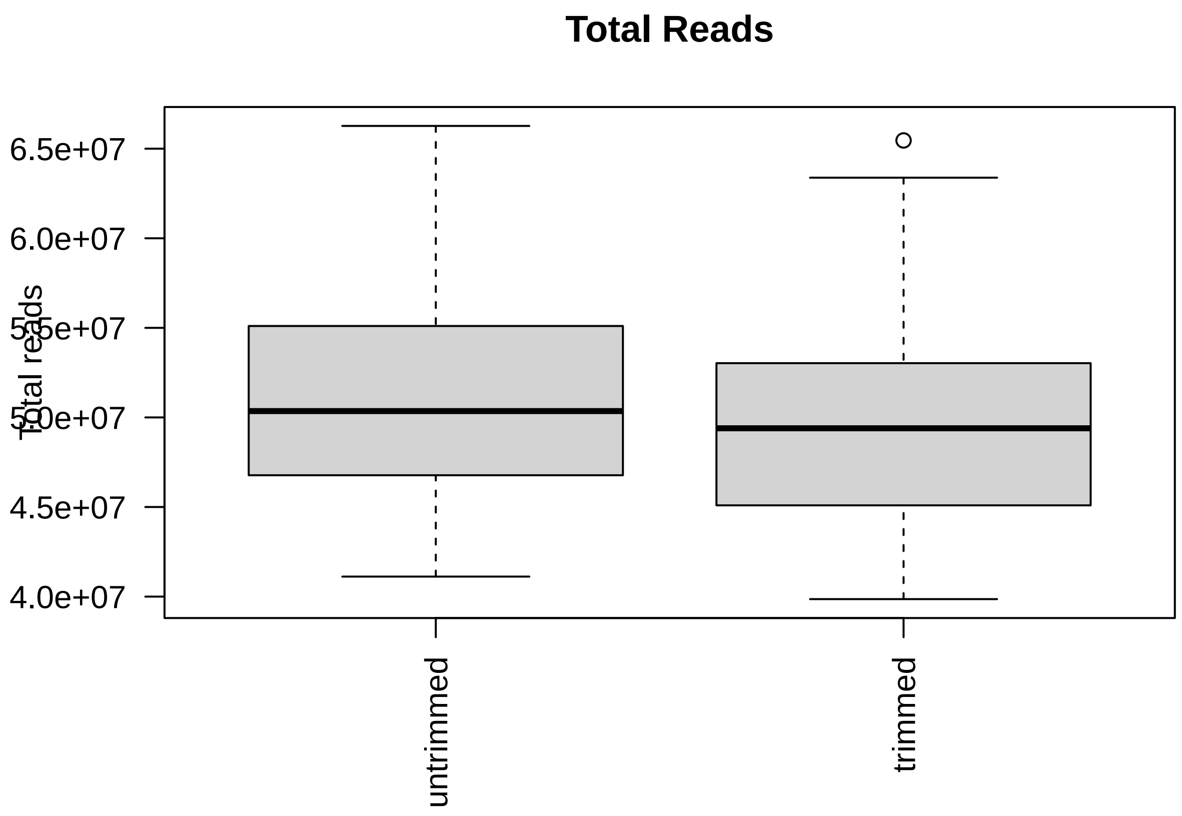
```r
trimmed_stats_files = list.files(trimmed_stats_directory)
rd_stats_files_trimmed = trimmed_stats_files[grepl("rdgrp.flagstats.txt",
    trimmed_stats_files)]

for (rd_stats_file in rd_stats_files_trimmed) {

    # read the lines of text in each file need to add the
    # directory so R can find the file outside your working
    # directory
    stats_info = readLines(paste0(trimmed_stats_directory, rd_stats_file))

    # find the entry for 'Total reads'
    total_reads = stats_info[grepl("read1", stats_info)]

    # replace the text ' + 0 read1' in the line containing
    # total reads
    total_reads_found = strsplit(total_reads[1], " ")
    total_reads_found = total_reads_found[[1]][1]

    # convert it to a number and add it to our list
    total_reads_trimmed = append(total_reads_trimmed, as.numeric(total_reads_found))

    # shorten the file name and add that to our list
    sample_list_trimmed = append(sample_list_trimmed, gsub(".sort.mkdup.rdgrp.flagstats.txt",
        "", rd_stats_file))
}

# to make a bar plot of all the sampes together, make a
# list of the untrimmed data combined with the trimmed data
total_reads_list = append(total_reads_untrimmed, total_reads_trimmed)
sample_list = append(sample_list_untrimmed, sample_list_trimmed)

# use a basic barplot to show the amount of total reads
# before and after trimming the 'unlist' function makes the
# object into a simple list (without names)
barplot(unlist(total_reads_list), names.arg = sample_list, cex.names = 0.4,
    las = 2, main = "Total Reads")
```



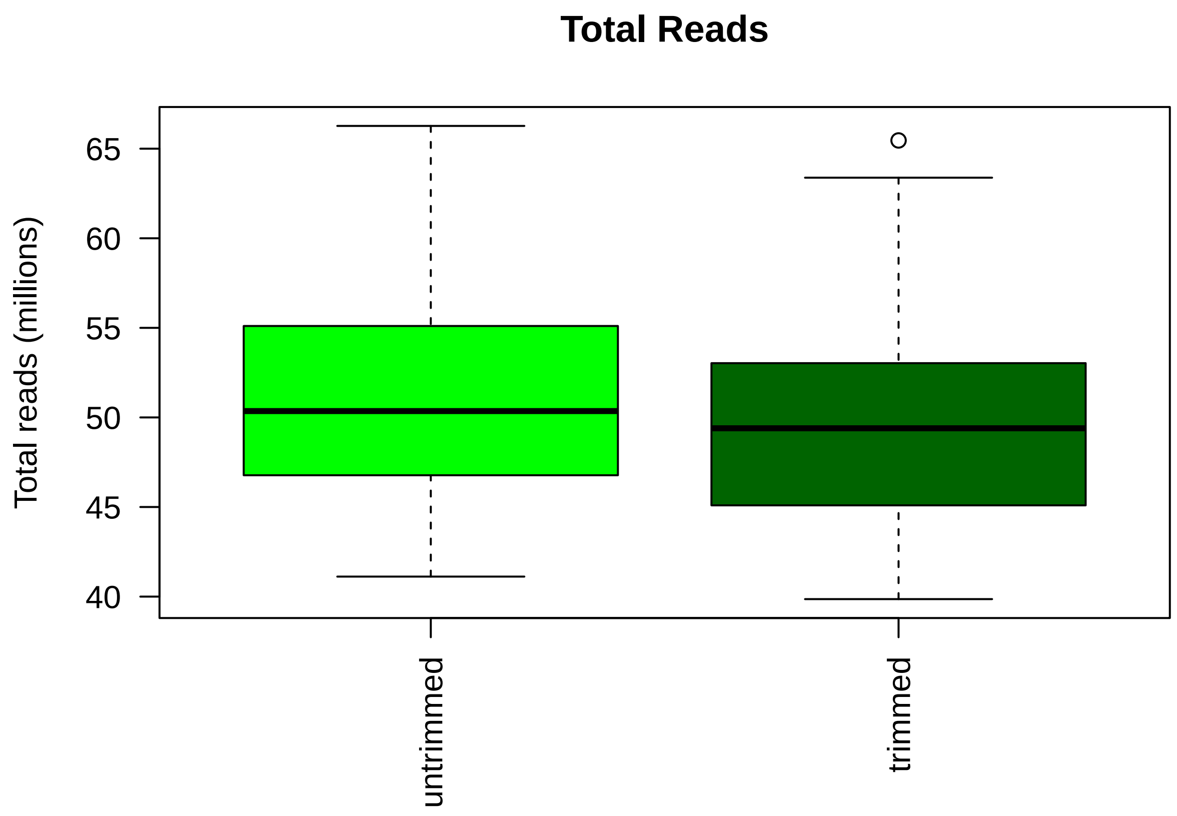
```r
# add colors to the bars to make it easier to tell
# untrimmed from trimmed
bar_colors = c(rep("green", length(total_reads_untrimmed)), rep("darkgreen",
    length(total_reads_trimmed)))
barplot(unlist(total_reads_list), names.arg = sample_list, cex.names = 0.4,
    las = 2, main = "Total Reads", col = bar_colors)
```



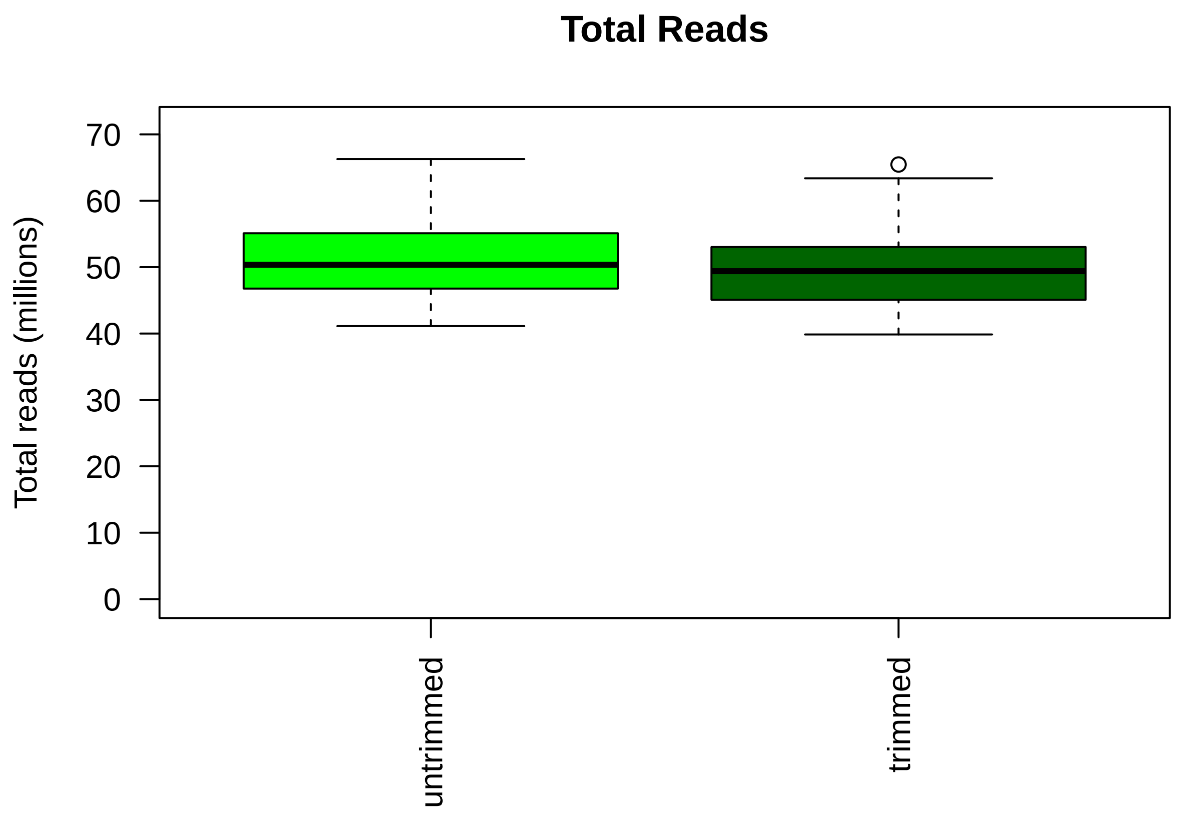
```r
# since there are a lot of samples, let's try a box plot
total_reads_list2 = list(untrimmed = unlist(total_reads_untrimmed),
    trimmed = unlist(total_reads_trimmed))
boxplot(total_reads_list2, names = names(total_reads_list2),
    cex.names = 0.4, las = 2, main = "Total Reads", ylab = "Total reads")
```



```
# see that the y-axis numbers are long and overrunning the
# y-axis label to fix that, let's express the total reads
# in millions (dividing by 1,000,000) when you apply a
# mathematical operation on a list in R, that operation is
# done on each element of that list and let's add colors
# for funsies
total_reads_list3 = list(untrimmed = unlist(total_reads_untrimmed)/1e+06,
    trimmed = unlist(total_reads_trimmed)/1e+06)
boxplot(total_reads_list3, names = names(total_reads_list3),
    cex.names = 0.4, las = 2, main = "Total Reads", ylab = "Total reads (millions)",
    col = c("green", "darkgreen"))
```



```
# additionally sometimes the difference between two groups
# is exaggerated when you don't have your minimum set to 0
# in the y-axis, so let's set the minimim to 0 and a
# maximum just over the max of the untrimmed data to see
boxplot(total_reads_list3, names = names(total_reads_list3),
    cex.names = 0.4, las = 2, main = "Total Reads", ylab = "Total reads (millions)",
    col = c("green", "darkgreen"), ylim = c(0, max(total_reads_list3$untrimmed) +
        5))
```



```
# now you can see how there isn't much difference between
# untrimmed and trimmed

# if you see that you are under 40 million reads, your
# trimming parameters could be too stringent

# get an estimate of how much data loss we have after
# trimming

# get the mean of the total reads of all the untrimmed
# samples and trimmed samples
untrimmed_avg = mean(unlist(total_reads_untrimmed))
trimmed_avg = mean(unlist(total_reads_trimmed))

# report the ratio of trimmed to untrimmed for average
# total number of reads
print(paste0("The ratio of average total read count of trimmed files to untrimmed files is ",
    trimmed_avg/untrimmed_avg))
```

```
## [1] "The ratio of average total read count of trimmed files to untrimmed files is 0.974540797586753"
```

```
# report the percent loss due to trimming, 'floor' function
# used to round down for easy reading
print(paste0("The percent loss in average total read count of trimmed files to untrimmed files is ",
    floor((untrimmed_avg - trimmed_avg)/untrimmed_avg * 100),
    "%"))
```

```
## [1] "The percent loss in average total read count of trimmed files to untrimmed files is 2%"
```

```
# if you see greater than 25% loss, trimming parameters
# might be too stringent which could mean you are throwing
# out acceptable quality reads along with bad quality reads
```

Get sex differentially expressed genes in untrimmed placenta data

We will filter the genes based on their adjusted p-value (p-value adjusted for multiple hypothesis testing) and the sign of their log fold change (positive indicates higher average expression in females, negative higher in males). We will store those lists so that we can compare them in the next chunks.

In this chunk we will do the sex differentially expressed genes using the untrimmed data.

```
# set name of the data file that contains the untrimmed
# gene list with differential expression stats this was set
# in the code in DE_Pipeline_UntrimmedData.Rmd
untrimmed_data_file = "trimq_NONE_minlen_NONE_dge_femalevsmale_all_expressed_genes.csv"

# read data from untrimmed data file
untrimmed_data <- read.delim(paste0("C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4/female_vs_male
    untrimmed_data_file), header = TRUE, sep = ",")

# select and store genes where adjusted p-value < 0.05
untrimmed_deg_list = untrimmed_data$genes[untrimmed_data$adj.P.Val <
    0.05 & abs(untrimmed_data$logFC) > 1]

# select and store genes where adjusted p-value < 0.05 and
# log fold change female vs male is positive, indicating
# that the average expression in females is higher than in
# males
untrimmed_deg_highfemale = untrimmed_data$genes[untrimmed_data$adj.P.Val <
    0.05 & untrimmed_data$logFC > 1]

# select and store genes where adjusted p-value < 0.05 and
# log fold change female vs male is negative, indicating
# that the average expression in males is higher than in
# females
untrimmed_deg_highmale = untrimmed_data$genes[untrimmed_data$adj.P.Val <
    0.05 & untrimmed_data$logFC < -1]
```

Get sex differentially expressed genes in trimmed placenta data

In this chunk we will do the sex differentially expressed genes using the trimmed data using the same methods as for the untrimmed.

```
# do the same thing we did for DEGs from untrimmed data for
# the results with your assigned trimmed data set

# have provided you with a falsified data file to use for
```

```r
# testing/learning purposes will need to switch that with
# the real results after running the differential
# expression Rmd with your assigned trimming data
trimmed_data <- read.delim(paste0(trimmed_directory, "female_vs_male/trimq_30_minlen_75/trimq_30_minlen
    header = TRUE, sep = ",")
trimmed_deg_list = trimmed_data$genes[trimmed_data$adj.P.Val <
    0.05 & abs(trimmed_data$logFC) > 1]
trimmed_deg_highfemale = trimmed_data$genes[trimmed_data$adj.P.Val <
    0.05 & trimmed_data$logFC > 1]
trimmed_deg_highmale = trimmed_data$genes[trimmed_data$adj.P.Val <
    0.05 & trimmed_data$logFC < -1]
```

Plan for how to compare data sets in terms of sex differentially expressed genes

First, we will compare number of differentially expressed genes using a bar graph

Second, we will use a venn diagram to show the overlap and unique sex differentially expressed genes between with untrimmed and trimmed data. We will use a hypergeometric p-value to assess the probability of seeing the observed number of overlapping genes or more. We will also look at the correlation between various features of the genes with untrimmed data versus trimmed data.

Third, we will assess the change in fold changes specific genes differentially expressed in females vs males before and after trimming. We will show this with a scatter plot with lines connecting the fold change of the same gene both trimming conditions.

Bar graph of number of DEG

A bar graph is a simple data visualization showing the value we are interested in studying on the y-axis and the items we are interested in as columns displayed along the x-axis.

R has functions to make lots of different flavors of bar graphs, for a taste check out this wiki: http://www.sthda.com/english/wiki/ggplot2-barplots-quick-start-guide-r-software-and-data-visualization

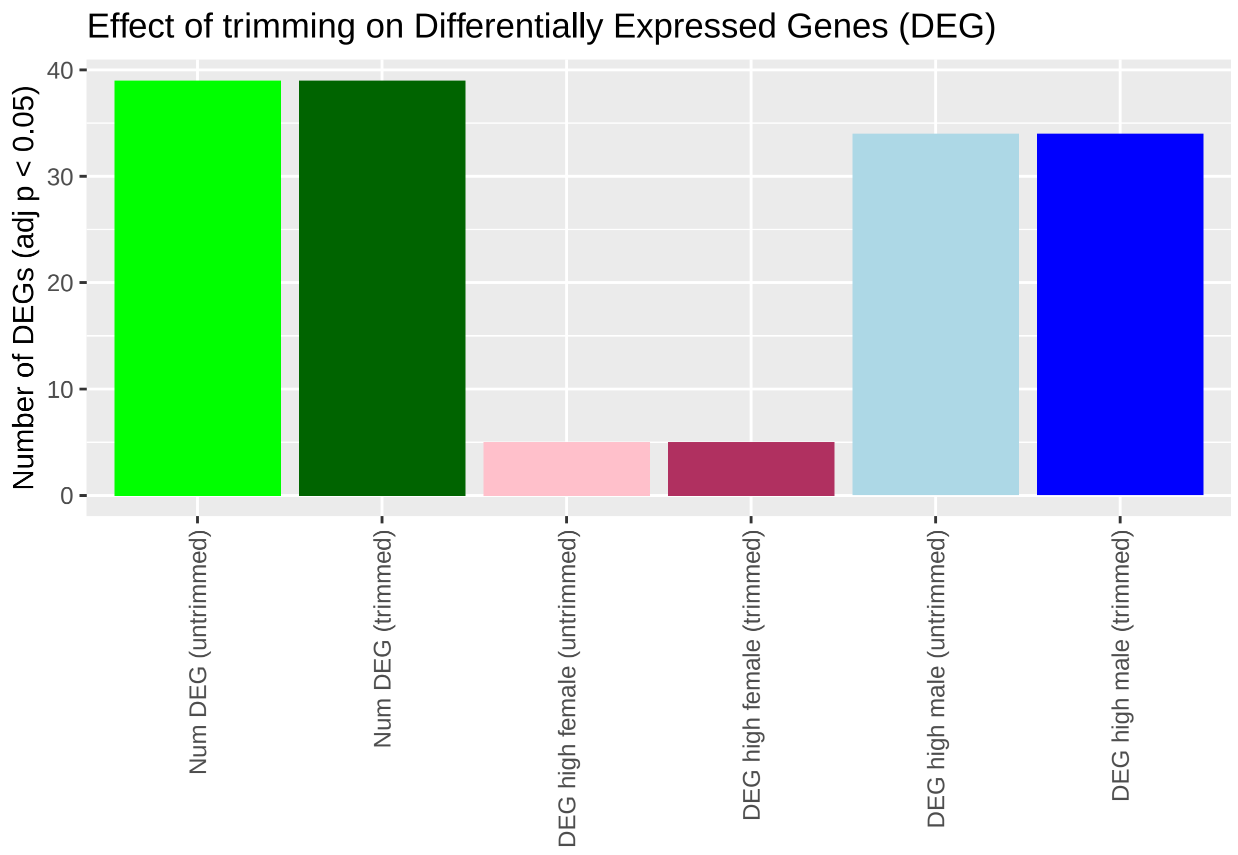
We will use ggplot to make bar graphs to compare the number of differentially expressed genes.

Here are the functions that are used to make the plot: 1) ggplot sets the x and y axis variables 2) geom_bar specifies that the data is displayed as a bar graph 3) theme sets the background and axis labels 4) labs sets the text of the main title and axis labels 5) scale_x_discrete used to make sure the graph is not reordered alphabetically (default)

R colors can be viewed here: https://bookdown.org/hneth/ds4psy/D-3-apx-colors-basics.html

```r
# determine the number of differentially expressed genes in
# untrimmed and trimmed
num_untrimmed_deg = length(untrimmed_deg_list)
num_trimmed_deg = length(trimmed_deg_list)

# determine the number of DEGs that have higher expression
# in females
num_untrimmed_deg_highfemale = length(untrimmed_deg_highfemale)
num_trimmed_deg_highfemale = length(trimmed_deg_highfemale)

# determine the number of DEGs that have higher expression
# in males
num_untrimmed_deg_highmale = length(untrimmed_deg_highmale)
```

```r
num_trimmed_deg_highmale = length(trimmed_deg_highmale)

# create a data frame to hold all the data we are going to
# plot in a bar graph 'headings' will be used to hold the
# x-axis labels we will use to identify each bar 'numbers'
# will be the counts we just accumulated 'colors' will be
# used to set the color of the bars so we can have related
# sets have similar color

df = data.frame(headings = c("Num DEG (untrimmed)", "Num DEG (trimmed)",
    "DEG high female (untrimmed)", "DEG high female (trimmed)",
    "DEG high male (untrimmed)", "DEG high male (trimmed)"),
    counts = c(num_untrimmed_deg, num_trimmed_deg, num_untrimmed_deg_highfemale,
        num_trimmed_deg_highfemale, num_untrimmed_deg_highmale,
        num_trimmed_deg_highmale), colors = c("green", "darkgreen",
        "pink", "maroon", "lightblue", "blue"))

# create the bar graph
p = ggplot(data = df, aes(x = headings, y = counts))  # set the data
p = p + geom_bar(stat = "identity", fill = df$colors)  # specify bar chart
p = p + theme(axis.title.x = element_blank(), axis.text.x = element_text(angle = 90,
    vjust = 0.5, hjust = 1))  # set the background and size of axis labels
p = p + labs(title = "Effect of trimming on Differentially Expressed Genes (DEG)",
    y = "Number of DEGs (adj p < 0.05)")  # change the main title and the y-axis label
p = p + scale_x_discrete(limits = df$headings)  # change the x-axis labels to be more descriptive
plot(p)  # display the plot in the report
```



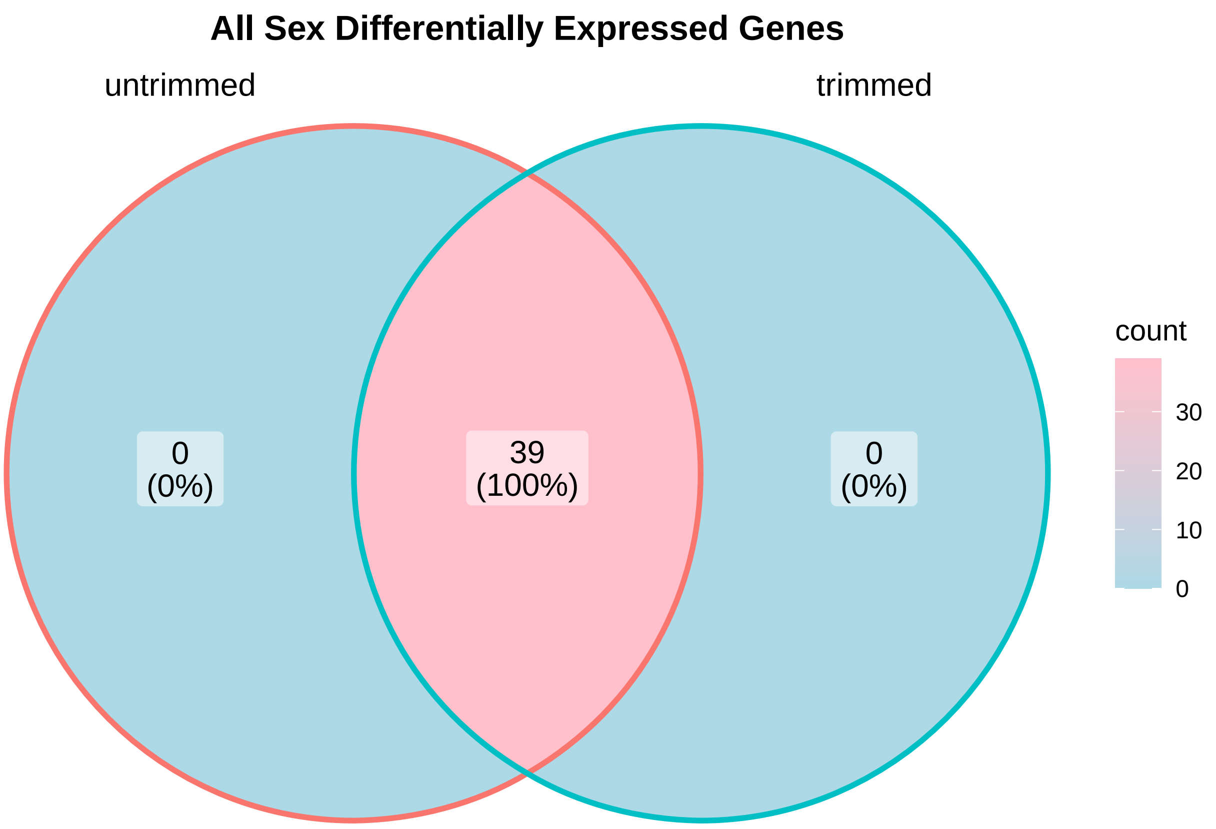
Venn diagram

A venn diagram a visualization where there are two overlapping circles representing two sets of items such that the number in the overlapping area is the number of overlapping items between the two sets and the number in the non-overlapping areas are unique to one set. The ggVennDiagram in the ggplot2 package can be used to create a Venn diagram and it will be colorized to bring attention to the area with the highest number of items (be that overlapping or unique).

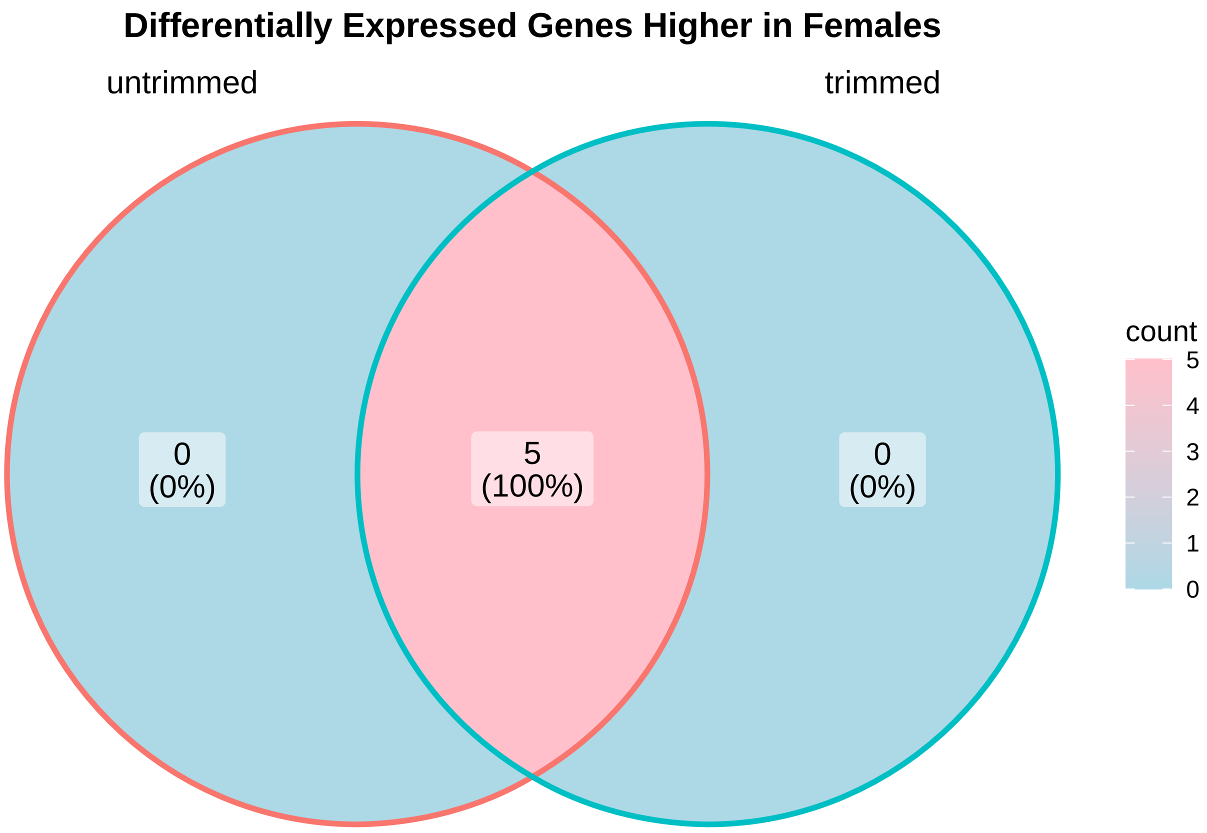
Here are the functions that are used to make the plot: 1) list of named lists to represent the gene lists being compared 2) ggVennDiagram takes the input lists of lists 3) scale_fill_gradient sets the continuous color gradient from blue (low number) to red (high number) so we can easily tell which area of the venn diagram has the most elements inside 4) ggtitle sets the title and subtitle of the graph (which we will use to hold the comparsion we are making and the statistical significance of the observed overlap) 5) theme will be used to center align and format the title and subtitle

```
# make venn diagrams to show overlap set the colors to
# match the bar graph above for easier interpretation

# pass in the full list of DEGs from untrimmed and trimmed,
# no matter whether the expression is high in females or
# males
allDEGs = list(untrimmed = untrimmed_deg_list, trimmed = trimmed_deg_list)  # set up data
v_DEGs = ggVennDiagram(allDEGs)  #specify venn diagram with input data
v_DEGs = v_DEGs + scale_fill_gradient(low = "lightblue", high = "pink")  # set color scale
```

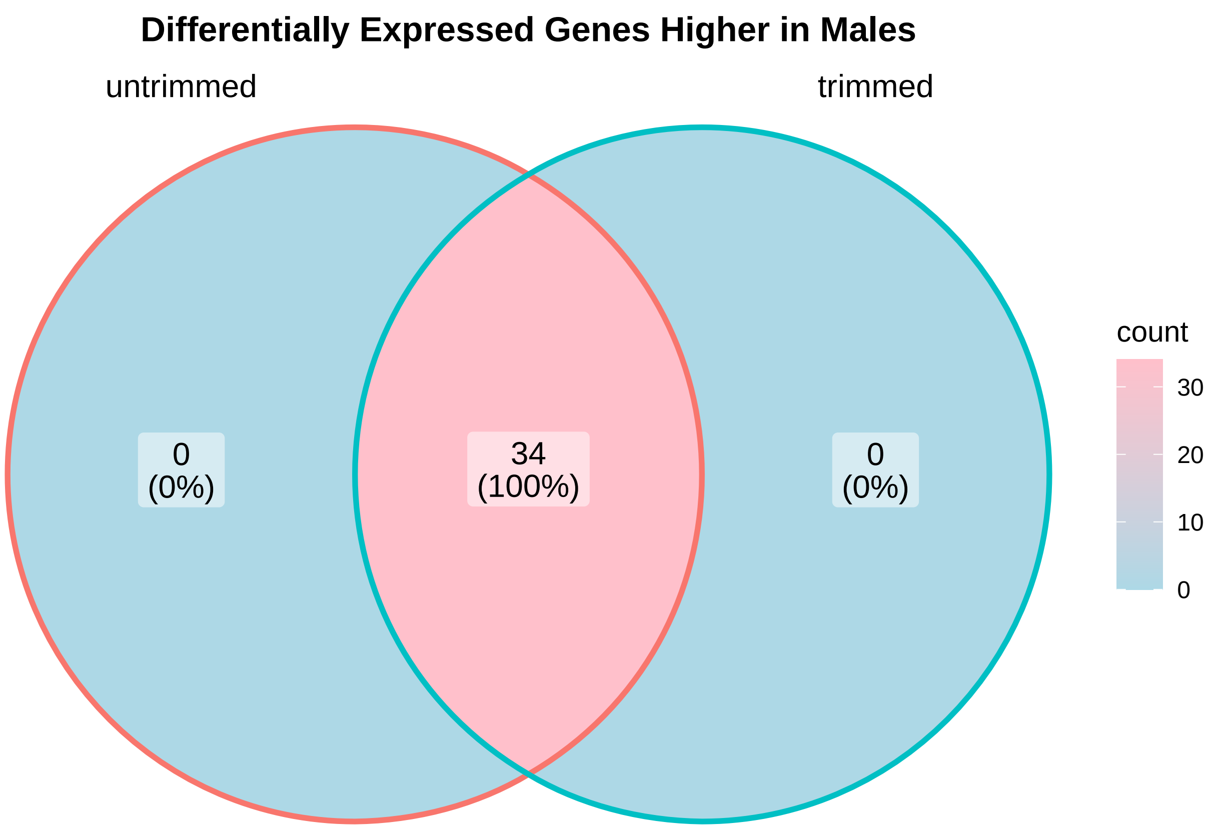
```r
# list overlap
allDEGs_overlap = Reduce(intersect, list(untrimmed_deg_list,
    trimmed_deg_list))  # get intersection of two input lists

# add title indicating data sets we are plotting as title
# and overlap p-value in subtitle
v_DEGs = v_DEGs + ggtitle(label = "All Sex Differentially Expressed Genes") +
    theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5))  # print title center bold, subtitle center

# display plot
plot(v_DEGs)
```



```r
# repeat with the DEGs that have higher expression in
# females from untrimmed and trimmed data
DEGs_female = list(untrimmed = untrimmed_deg_highfemale, trimmed = trimmed_deg_highfemale)
v_DEGs_female = ggVennDiagram(DEGs_female)
v_DEGs_female = v_DEGs_female + scale_fill_gradient(low = "lightblue",
    high = "pink")

DEGs_female_overlap = Reduce(intersect, list(untrimmed_deg_highfemale,
    trimmed_deg_highfemale))

v_DEGs_female = v_DEGs_female + ggtitle("Differentially Expressed Genes Higher in Females") +
    theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5))  # print title center bold, subtitle center

plot(v_DEGs_female)
```



```
# repeat with the DEGs that have higher expression in males
# from untrimmed and trimmed data
DEGs_male = list(untrimmed = untrimmed_deg_highmale, trimmed = trimmed_deg_highmale)
v_DEGs_male = ggVennDiagram(DEGs_male)
v_DEGs_male = v_DEGs_male + scale_fill_gradient(low = "lightblue",
    high = "pink")

DEGs_male_overlap = Reduce(intersect, list(untrimmed_deg_highmale,
    trimmed_deg_highmale))

v_DEGs_male = v_DEGs_male + ggtitle("Differentially Expressed Genes Higher in Males") +
    theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5))  # print title center bold, subtitle center

plot(v_DEGs_male)
```

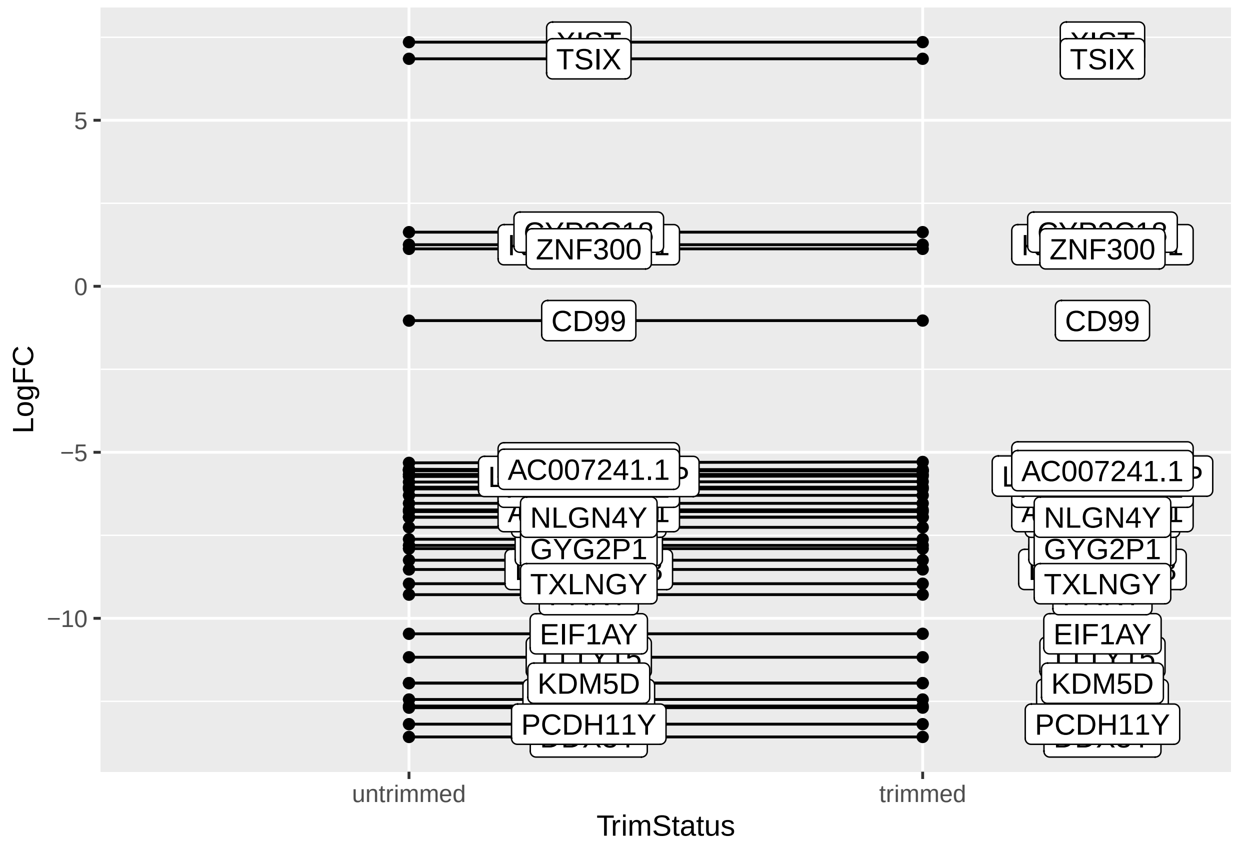


Visualizing change in fold change between sexes

One way of visualizing trends between two data sets to show pairs of values. In this visualization, we will enter a list of specific genes and plot their female:male fold change (logFC) in the untrimmed data and the trimmed data. The fold change will be represented as a labelled point and a line is drawn to show the points for the same gene in both data sets. If we see many lines tilting upwards in the positive range and downwards in the negative range, that indicates that trimming leads to quantification that amplifies fold change between the sexes. If we see lines staying level, that would indicate that trimming has little to no effect on the measured fold change. If we see the lines tilting inward toward y=0, trimming is reducing the fold change we see, which we would want to look at more carefully to see whether that is making the signal more reliable or if that is a downside of trimming.

```r
# enter a list of genes that are interesting
# genes_of_interest = c('KDM6A','DDX3Y','USP9Y')

# in the original template, we just had three genes that we
# knew had shown sex differences, but let's look at all of
# our overlapping genes between untrimmed and trimmed
genes_of_interest = allDEGs_overlap

# create an empty data frame to look up and store fold
# changes female vs male pre and post trimming
df2 = data.frame(matrix(ncol = 3, nrow = 0))
colnames(df2) = c("gene", "LogFC", "TrimStatus")
```

```
# for each gene of interest, find its logFC in untrimmed
# and trimmed data to see how it has changed

for (i in 1:length(genes_of_interest)) {
    current_gene = genes_of_interest[i]

    # find the logFC of this gene in the untrimmed data
    untrimmedFC = 0  # set Female:Male foldchange to 0 by default
    if (current_gene %in% untrimmed_deg_list) {
        # replace with logFC in untrimmed data if found
        untrimmedFC = untrimmed_data$logFC[untrimmed_data$genes ==
            current_gene]
    }

    # add untrimmed FC to the data frame we will use to
    # plot sprintf sets the precision so we don't have long
    # decimals on our y-axis
    df2[nrow(df2) + 1, ] = c(current_gene, sprintf(untrimmedFC,
        fmt = "%#.3f"), "untrimmed")

    # find the logFC of this gene in the trimmed data
    trimmedFC = 0  # set Female:Male foldchange to 0 by default
    if (current_gene %in% trimmed_deg_list) {
        # replace with logFC in untrimmed data if found
        trimmedFC = trimmed_data$logFC[trimmed_data$genes ==
            current_gene]
    }

    # add trimmed FC to the data frame we will use to plot
    df2[nrow(df2) + 1, ] = c(current_gene, sprintf(trimmedFC,
        fmt = "%#.3f"), "trimmed")
}

# set order for plotting
df2$TrimStatus <- factor(df2$TrimStatus, levels = c("untrimmed",
    "trimmed"))

# tell R that the LogFC values are numeric values (not
# strings)
df2$LogFC = as.numeric(df2$LogFC)

# do a paired scatter plot to see how the fold changes of
# select genes change after trimming

# tell ggplot that you are plotting which data set on the
# x-axis and the LogFC on the y-axis
p2 = ggplot(data = df2, aes(x = TrimStatus, y = LogFC))

# add the lines that connect the same genes in the trimmed
# and untrimmed
p2 = p2 + geom_line(aes(group = gene))

# label the genes so you know which one you are looking at
# * genes for which no LogFC was found will sit on top of
# each other at 0 so you won't be able to see them all
p2 = p2 + geom_label(aes(label = gene), nudge_x = 0.35, size = 4)

# add the points for each gene
p2 = p2 + geom_point()

# make the plot in your report
plot(p2)
```



```
# we can see that they are all exactly the same in the
# untrimmed vs trimmed!
```

Overall Correlation of Data before and after trimming

In this section we will create x-y scatter plots to compare various features of the entire data set before trimming and after trimming. We can calculate the correlation between the feature in both data sets using the 'cor.test' function in R and visualize the correlation using a scatter plot.

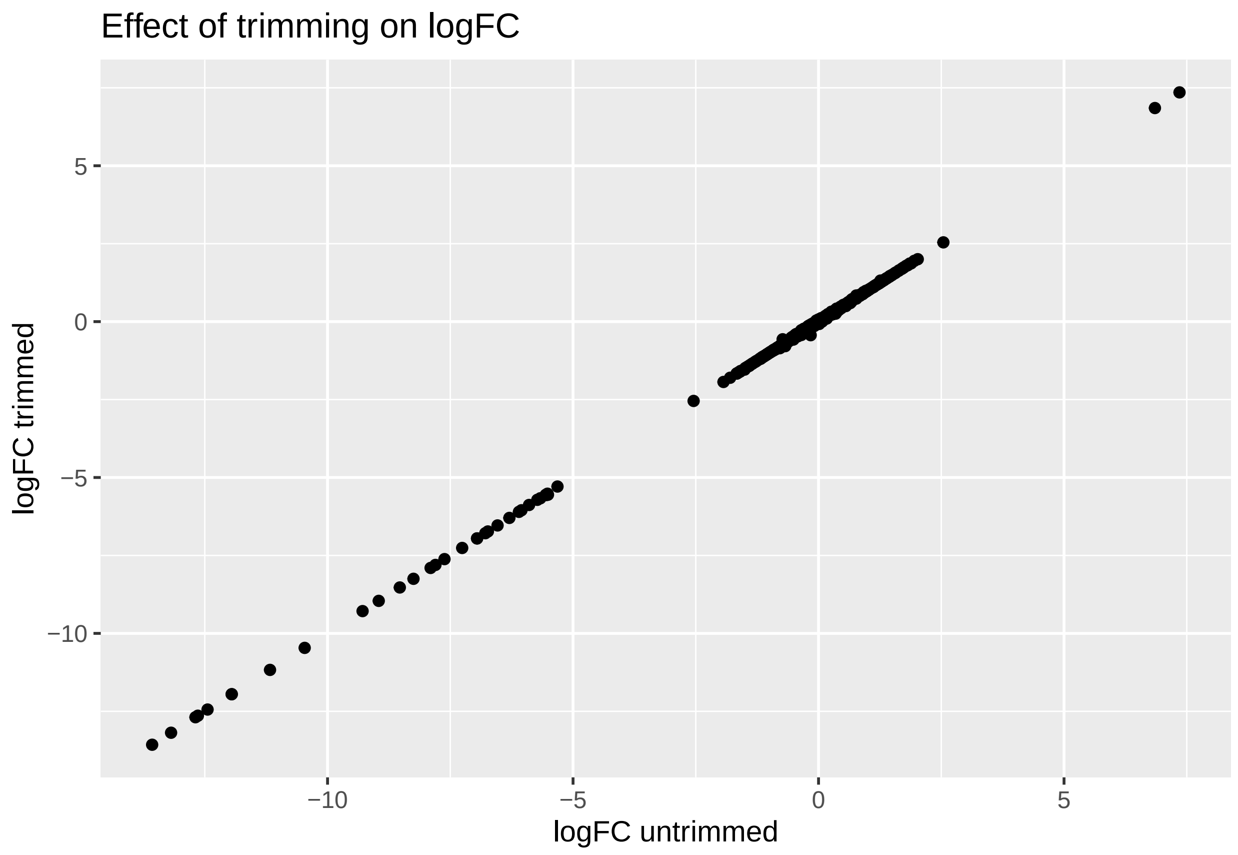
For more information about correlation between two variables: http://www.sthda.com/english/wiki/correlation-test-between-two-variables-in-r

For more ways to customize scatter plots, check this out: http://www.sthda.com/english/wiki/ggplot2-scatter-plots-quick-start-guide-r-software-and-data-visualization

```r
# Comparing the logFC (log fold change) of XX:XY placenta
# in untrimmed vs trimmed

merged_data = merge(untrimmed_data, trimmed_data, by = "genes")  # put all data into one data frame
str(merged_data)  # take a peek at the merged data to see how the data was merged (.x and .y appended t
```

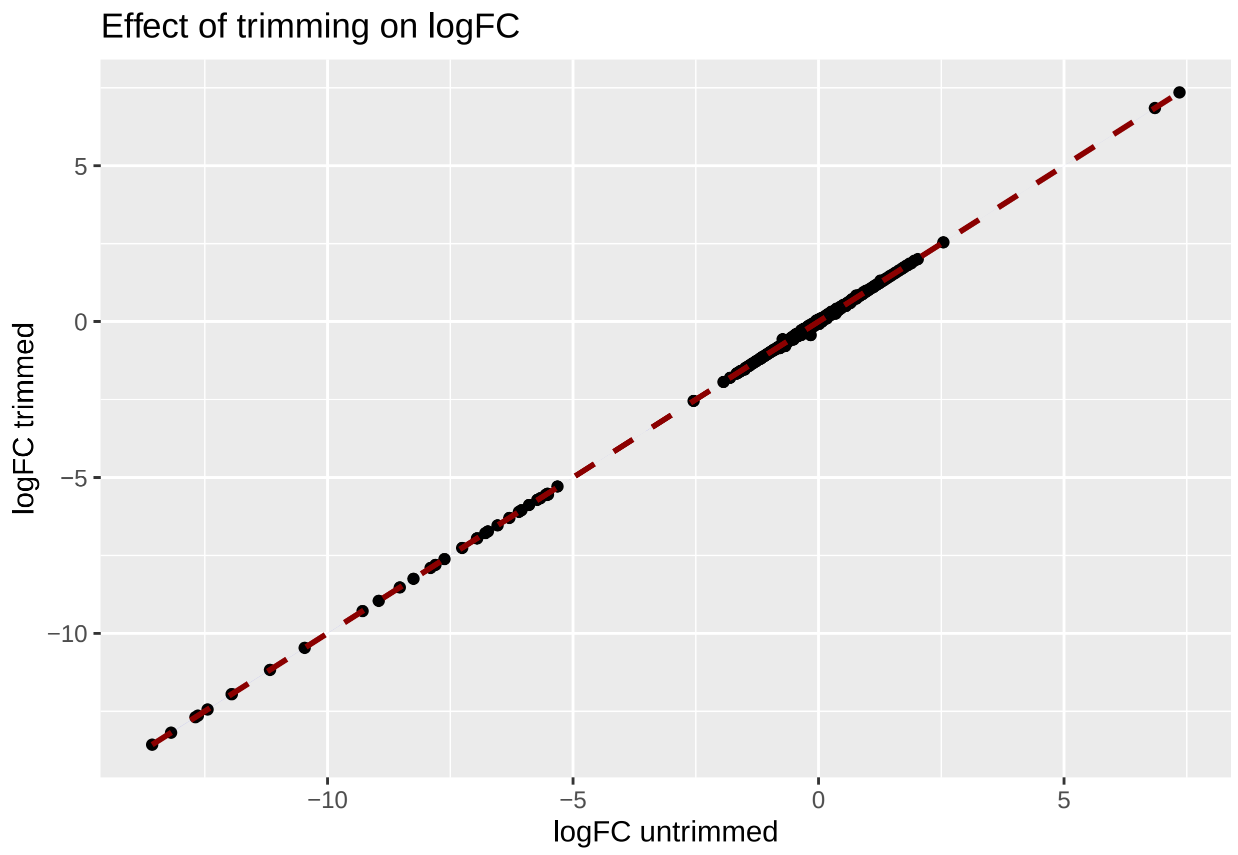
```
## 'data.frame':    20975 obs. of  15 variables:
##  $ genes      : chr  "A1BG-AS1" "A2M" "A2M-AS1" "A2ML1" ...
##  $ X.x        : chr  "A1BG-AS1" "A2M" "A2M-AS1" "A2ML1" ...
##  $ logFC.x    : num  -0.1612 -0.0561 0.0755 -0.2413 -0.0663 ...
##  $ AveExpr.x  : num  -0.0125 7.9615 -1.3605 0.2438 -2.5924 ...
##  $ t.x        : num  -0.496 -0.268 0.384 -0.916 -0.215 ...
##  $ P.Value.x  : num  0.625 0.791 0.705 0.369 0.832 ...
##  $ adj.P.Val.x: num  1 1 1 1 1 ...
##  $ B.x        : num  -6.92 -7.91 -6.76 -6.65 -6.61 ...
##  $ X.y        : chr  "A1BG-AS1" "A2M" "A2M-AS1" "A2ML1" ...
##  $ logFC.y    : num  -0.1612 -0.0561 0.0755 -0.2413 -0.0664 ...
##  $ AveExpr.y  : num  -0.0124 7.9615 -1.3603 0.2439 -2.5923 ...
##  $ t.y        : num  -0.496 -0.268 0.384 -0.917 -0.215 ...
##  $ P.Value.y  : num  0.625 0.791 0.705 0.369 0.832 ...
##  $ adj.P.Val.y: num  1 1 1 1 1 ...
##  $ B.y        : num  -6.92 -7.91 -6.76 -6.65 -6.62 ...
```

```r
# plot logFC in untrimmed vs trimmed
p3 = ggplot(data = merged_data, aes(x = logFC.x, y = logFC.y)) +
    geom_point()
p3 = p3 + labs(title = "Effect of trimming on logFC", x = "logFC untrimmed",
    y = "logFC trimmed")  # change the main title and axis labels
plot(p3)
```



```r
# we can see a near perfect correlation any points that are
# not exactly on the diagonal are just barely off of it
# showing a small decrease in read counts due to trimming

# add trendline
p3 = p3 + geom_smooth(method = lm, linetype = "dashed", color = "darkred",
    fill = "blue")
plot(p3)
```

```
## ‘geom_smooth()‘ using formula ’y ~ x’
```



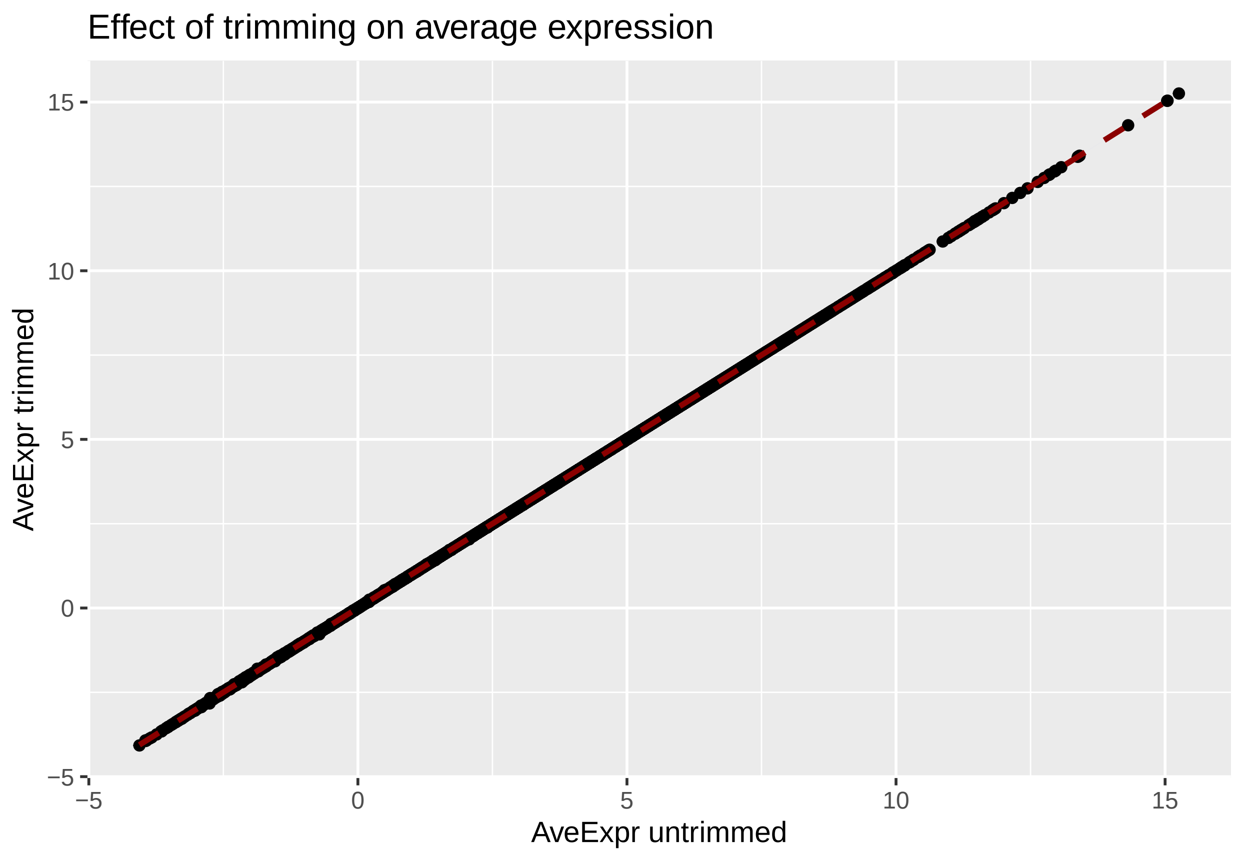
```r
# calculate the correlation between logFC in untrimmed vs
# trimmed
cor.test(merged_data$logFC.x, merged_data$logFC.y, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  merged_data$logFC.x and merged_data$logFC.y
## t = 10702, df = 20973, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9999059 0.9999109
## sample estimates:
##       cor
## 0.9999085
```

```r
# if p-value < 0.05, we can accept the alternative
# hypothesis that the two variables are significantly
# correlated

# do the same thing with average expression
p4 = ggplot(data = merged_data, aes(x = AveExpr.x, y = AveExpr.y)) +
    geom_point()
p4 = p4 + labs(title = "Effect of trimming on average expression",
    x = "AveExpr untrimmed", y = "AveExpr trimmed")
```

```
p4 = p4 + geom_smooth(method = lm, linetype = "dashed", color = "darkred",
    fill = "blue")
plot(p4)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



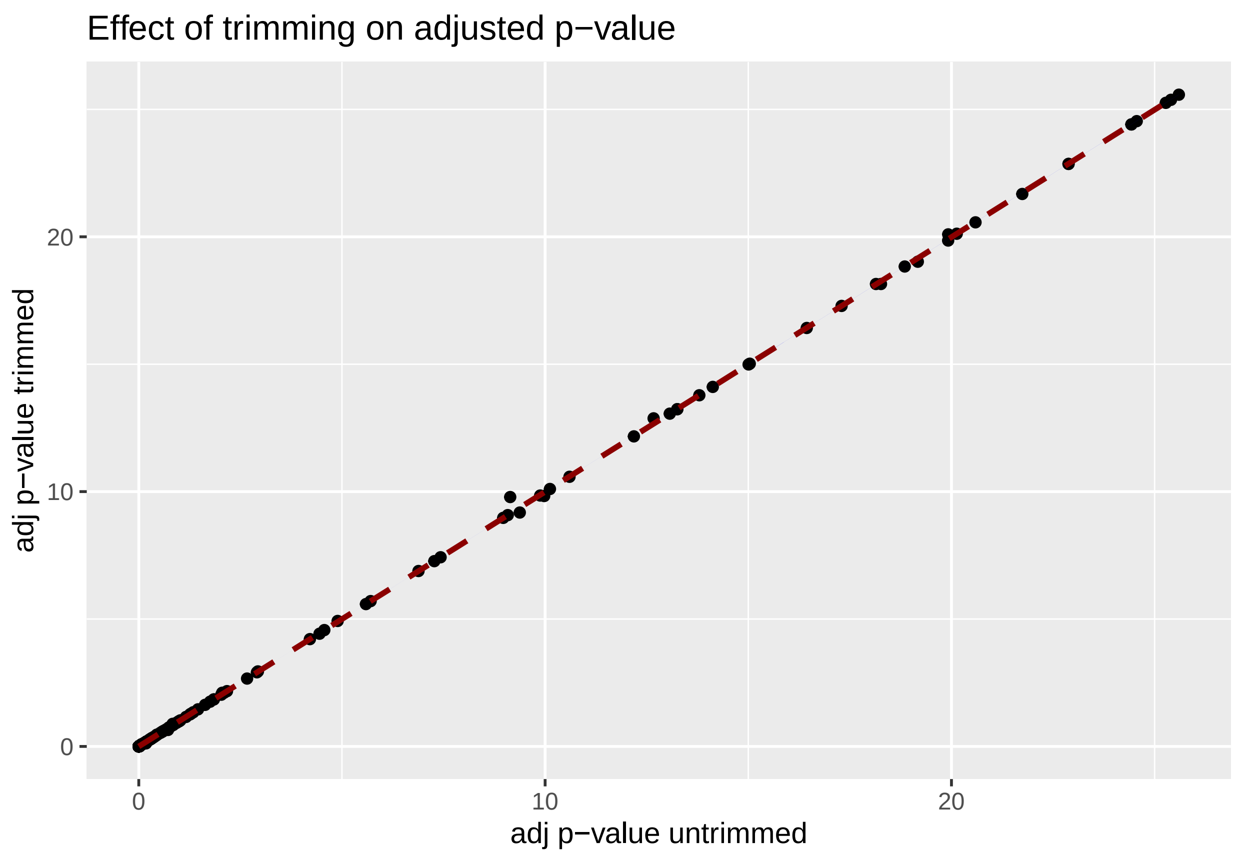
```
cor.test(merged_data$AveExpr.x, merged_data$AveExpr.y, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  merged_data$AveExpr.x and merged_data$AveExpr.y
## t = 165148, df = 20973, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9999996 0.9999996
## sample estimates:
##       cor
## 0.9999996
```

```
# do the same thing with adjusted p-value (log transform to
# pull significant genes away from the origin)
merged_data$logadjP.x = -1 * log(merged_data$adj.P.Val.x, base = 10)  # log transform untrimmed adj p-v
```

```
merged_data$logadjP.y = -1 * log(merged_data$adj.P.Val.y, base = 10)  # log transform trimmed adj p-val
p5 = ggplot(data = merged_data, aes(x = logadjP.x, y = logadjP.y)) +
    geom_point()
p5 = p5 + labs(title = "Effect of trimming on adjusted p-value",
    x = "adj p-value untrimmed", y = "adj p-value trimmed")
p5 = p5 + geom_smooth(method = lm, linetype = "dashed", color = "darkred",
    fill = "blue")
plot(p5)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



```
cor.test(merged_data$logadjP.x, merged_data$logadjP.y, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  merged_data$logadjP.x and merged_data$logadjP.y
## t = 19811, df = 20973, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9999725 0.9999740
## sample estimates:
##       cor
## 0.9999733
```

```
# the correlation coefficients are very close to 1 but not
# exactly 1 this means that there were a few differences
# observed if we had seen that there were no differences at
# all, we would have wanted to check that we didn't
# accidently correlate the data to itself when when we
# meant to compare two different data sets
```

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] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] ggVennDiagram_1.2.2 ggplot2_3.3.6
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.2.0   xfun_0.34          purrr_0.3.5        sf_1.0-8
##  [5] lattice_0.20-45    splines_4.2.1      colorspace_2.0-3   vctrs_0.4.1
##  [9] generics_0.1.3     htmltools_0.5.3    s2_1.1.0           yaml_2.3.6
## [13] mgcv_1.8-40        utf8_1.2.2         rlang_1.0.6        e1071_1.7-12
## [17] pillar_1.8.1       RVenn_1.1.0        glue_1.6.2         withr_2.5.0
## [21] DBI_1.1.3          wk_0.7.0           lifecycle_1.0.3    stringr_1.4.1
## [25] munsell_0.5.0      gtable_0.3.1       evaluate_0.17      labeling_0.4.2
## [29] knitr_1.40         fastmap_1.1.0      class_7.3-20       fansi_1.0.3
## [33] highr_0.9          Rcpp_1.0.9         KernSmooth_2.23-20 scales_1.2.1
## [37] classInt_0.4-8     formatR_1.12       farver_2.1.1       digest_0.6.30
## [41] stringi_1.7.8      dplyr_1.0.10       cli_3.3.0          tools_4.2.1
## [45] magrittr_2.0.3     proxy_0.4-27       tibble_3.1.8       pkgconfig_2.0.3
## [49] Matrix_1.5-1       assertthat_0.2.1   rmarkdown_2.17     rstudioapi_0.14
## [53] R6_2.5.1           nlme_3.1-157       units_0.8-0        compiler_4.2.1
```