Comparing Range of 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(UpSetR)){
    install.packages("UpSetR")
}
```

```
## Loading required package: UpSetR
```

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

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

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

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## v purrr   0.3.5
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

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

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

```
library(ggplot2)
library(UpSetR)
library(grid)
library(tidyverse)
library()
```

Set working directory and data directories

In the SetDirectory chunk, you will set your working directory to where you want your reports and output files to be saved. You will set your untrimmed directory to where you have you data files for the untrimmed data. You will set your trimmed directory to where you will put your trimmed data. See below for what is expected in the trimmed directory.

```
# 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 5/"
setwd(working_directory)

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

```
## [1] "Extras.docx"
## [2] "fake_dge_femalevsmale_all_expressed_genes1.csv"
## [3] "fake_dge_femalevsmale_all_expressed_genes2.csv"
## [4] "fake_dge_femalevsmale_all_expressed_genes3.csv"
## [5] "FullComparison.pdf"
## [6] "FullComparison.Rmd"
## [7] "FullComparison_files"
```

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

```
## [1] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 5/"
## [2] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 5//FullComparison_files"
## [3] "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 5//FullComparison_files/figure-latex"
```

```
# paths to directory containing differential expression
# results

# set untrimmed to where your Week_3 subdirectory that
# contains
# untrimmed_dge_femalevsmale_all_expressed_genes.csv
untrimmed_directory = "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4/female_vs_male/trimq_NONE_mi
```

```r
# set to the directory containing trimmed results in class
# shared drive
trimmed_directory = "C:/Users/splaisie/Dropbox (ASU)/GenomicsCURE/Week 4/female_vs_male/all_trimmed/"

# this will be used to store full data tables for all of
# the data sets so we can iterate through them more easily
all_data = list()
```

Get sex differentially expressed genes in untrimmed placenta data

We will load the untrimmed data and filter for differentially expressed genes just like we did in the pairwise comparison in Week 4.

```r
# name of the data file that contains the untrimmed gene
# list with differential expression stats
untrimmed_data_file = "trimq_NONE_minlen_NONE_dge_femalevsmale_all_expressed_genes.csv"

# read data from untrimmed data file
untrimmed_data <- read.delim(paste0(untrimmed_directory, untrimmed_data_file),
    header = TRUE, sep = ",")

# store data for reading later
all_data$untrimmed = untrimmed_data

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

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

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

# name these gene lists with the file they came from
names(untrimmed_deg_list) = c(untrimmed_data_file)
names(untrimmed_deg_list_highfemale) = c(untrimmed_data_file)
names(untrimmed_deg_list_highmale) = c(untrimmed_data_file)
```

Get sex differentially expressed genes in trimmed data sets

We are trying to figure out which genes are differentially expressed specifically when certain trimming parameters are used and and which genes are differentially expressed regardless of which trimming parameters are used. The idea is that we would trust results more if they are robust to changes in data processing steps like trimming.

To do this, we will use result files from running the differential expression pipeline on the trimming data sets. Your instructors will put the trimmed result tables for all expressed genes in the class shared drive on Agave in a subdirectory called female_vs_male. The code below finds all files in the trimmed data directory you set above and stores the data from files containing “femalevsmale_all_expressed_genes” in the file name.

For testing purposes, we have provided three falsified data files in the Week_5 subdirectory to use as an example. We did this by adding random numbers to the differential expression results from the untrimmed data. These falsified data files have ‘fake’ in the file name. Now that we have run the actual data, let’s see what genes are in common.

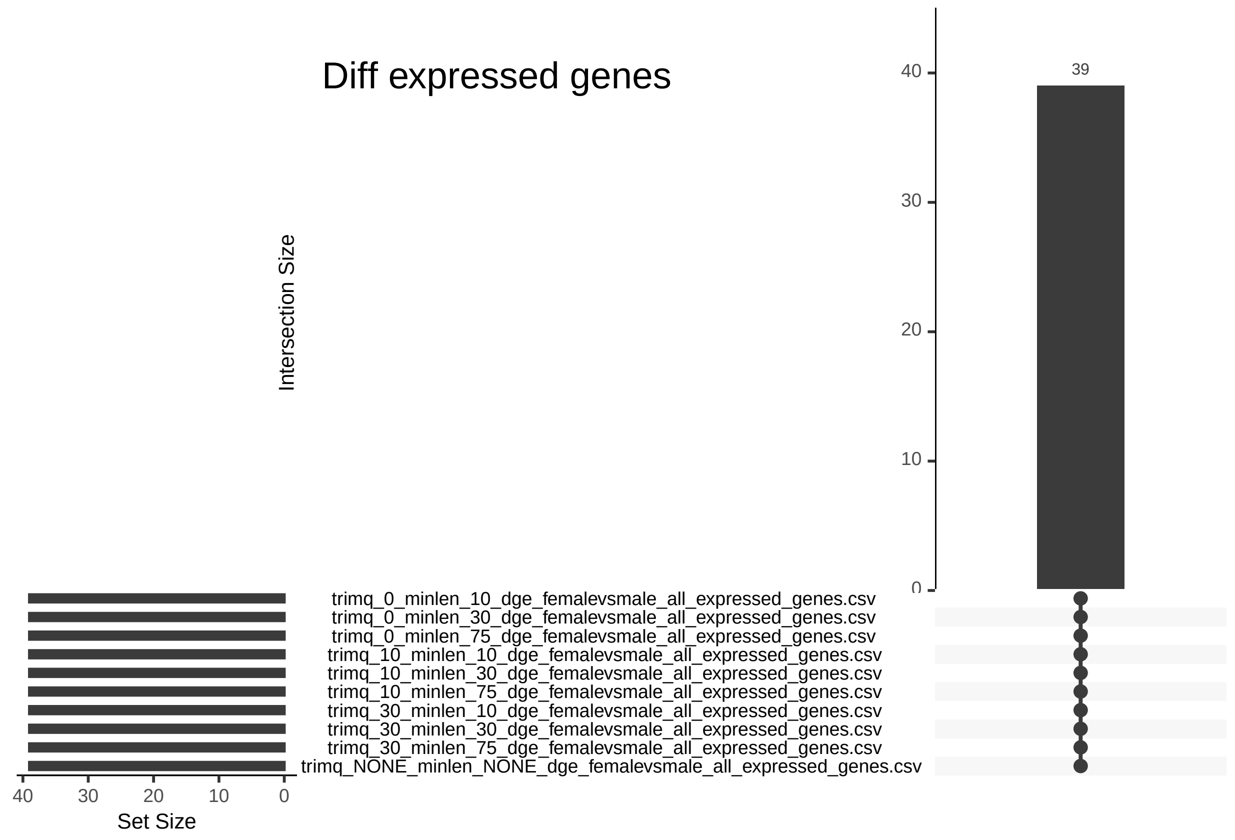
```r
# create a list of lists that contain the lists of
# differentially expressed genes

# create empty lists to hold everything we are interested
# in

# list of DEGs (adj p < 0.05) for each trimmed data file
trimmed_deg_list = list()

# list of DEGs that have higher expression in females
trimmed_deg_list_highfemale = list()

# list of DEGs that have higher expression in males
trimmed_deg_list_highmale = list()

# list of trimmed file names
data_file_names = list()

# list all the files in the trimmed directory
files = list.files(trimmed_directory)

# iterate through all the files
for (i in seq_along(files)) {

    # if the file has 'femalevsmale_all_expressed_genes' in
    # the name
    if (grepl("femalevsmale_all_expressed_genes", files[i], fixed = TRUE)) {
        # add this file name to the list of trimmed data
        # files
        data_file_names[i] = files[i]

        # read all the data in the file
        trimmed_data <- read.delim(paste0(trimmed_directory,
            files[i]), header = TRUE, sep = ",")

        # store the data for later
        all_data[[files[i]]] = trimmed_data

        # select genes where p-value < 0.05
        trimmed_deg = trimmed_data$genes[trimmed_data$adj.P.Val <
            0.05 & abs(trimmed_data$logFC) > 1]
        # store that gene list in the trimmed_deg_list list
        trimmed_deg_list[[i]] = trimmed_deg

        # select genes where p-value < 0.05 and log fold
        # change is positive, which indicates that the
        # average expression in females is higher than
        # average expression in males
        trimmed_deg_highfemale = trimmed_data$genes[trimmed_data$adj.P.Val <
            0.05 & trimmed_data$logFC > 1]
        # store that gene list in the
        # trimmed_deg_highfemale list
        trimmed_deg_list_highfemale[[i]] = trimmed_deg_highfemale

        # select genes where p-value < 0.05 and log fold
        # change is negative, which indicates that the
        # average expression in males is higher than
        # average expression in females
        trimmed_deg_highmale = trimmed_data$genes[trimmed_data$adj.P.Val <
            0.05 & trimmed_data$logFC < -1]
        # store that gene list in the trimmed_deg_highmale
        # list
        trimmed_deg_list_highmale[[i]] = trimmed_deg_highmale
    }
}

# set the names of all of our lists to the names of the
# data files upset plot function (below) takes named lists
# as input
names(trimmed_deg_list) = data_file_names
names(trimmed_deg_list_highfemale) = data_file_names
names(trimmed_deg_list_highmale) = data_file_names
```

Upset plot

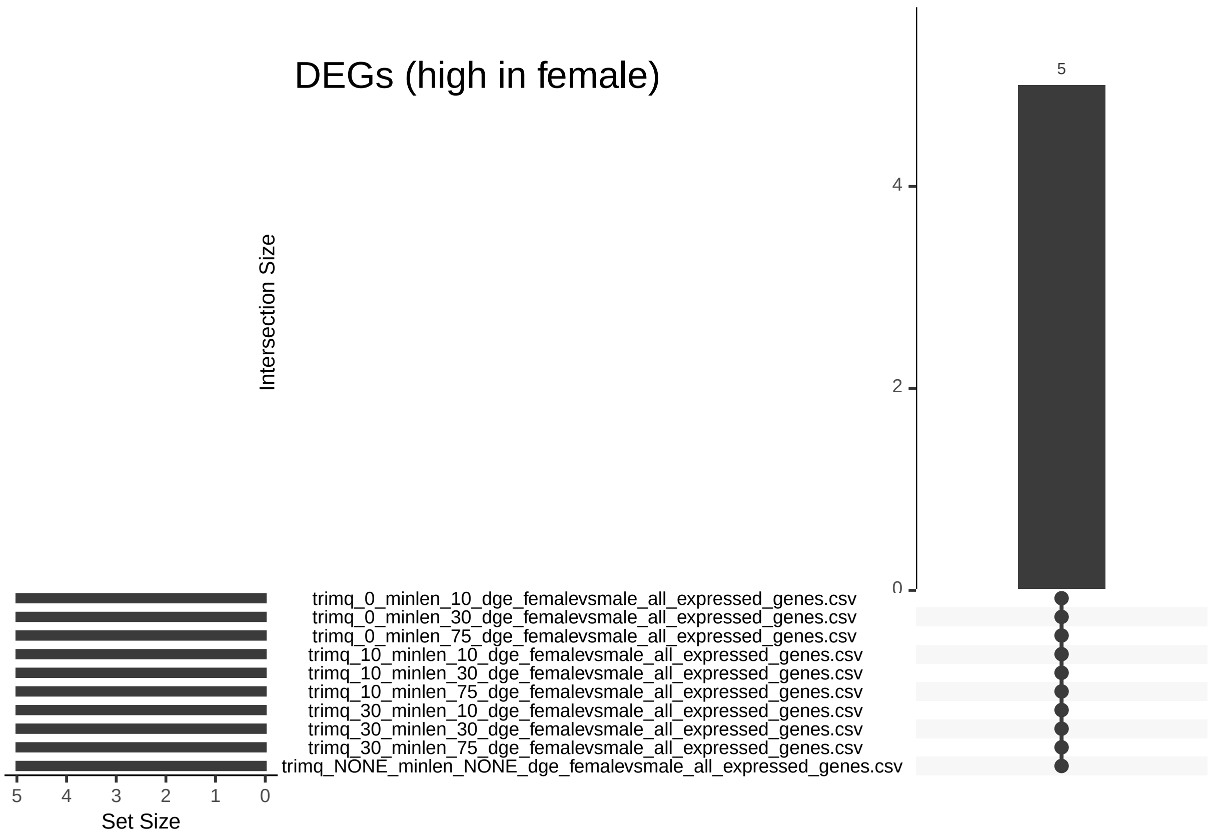
Create an upset plot of differentially expressed genes across range of trimming parameters

```r
# tack untrimmed to set trimmed data so we can view
# everything together

# to add to the list, assign to the index one greater than
# the last index (its length)
next_index = length(data_file_names) + 1

# add the untrimmed data file at the end of the trimmed
# data file list
data_file_names[next_index] = untrimmed_data_file

# add DEG list from the untrimmed results to the list of
# trimmed results
trimmed_deg_list[[next_index]] = as.vector(untrimmed_deg_list)
# name the list with file names so that you can pass a
# named list into upset plot function
names(trimmed_deg_list) = data_file_names
```

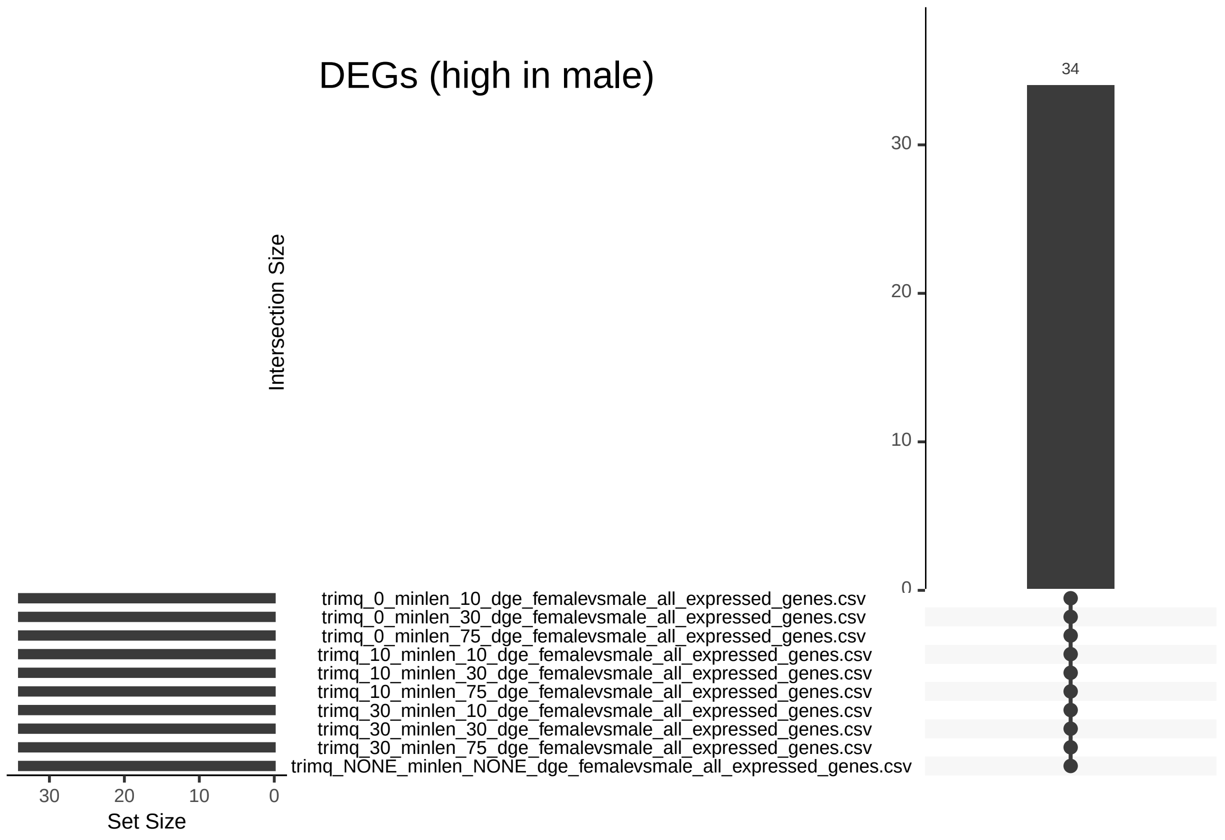
```r
# create upset plot to show the overlap
upset(fromList(trimmed_deg_list), nsets = 10)
grid.text("Diff expressed genes", x = 0.4, y = 0.9, gp = gpar(fontsize = 14))
```



```r
# repeat with DEGs with higher expression in female
trimmed_deg_list_highfemale[[next_index]] = untrimmed_deg_list_highfemale
names(trimmed_deg_list_highfemale) = data_file_names
upset(fromList(trimmed_deg_list_highfemale), nsets = 10)
grid.text("DEGs (high in female)", x = 0.4, y = 0.9, gp = gpar(fontsize = 14))
```



```r
# repeat with DEGs with higher expression in male
trimmed_deg_list_highmale[[next_index]] = untrimmed_deg_list_highmale
names(trimmed_deg_list_highmale) = data_file_names
upset(fromList(trimmed_deg_list_highmale), nsets = 10)
grid.text("DEGs (high in male)", x = 0.4, y = 0.9, gp = gpar(fontsize = 14))
```



Membership

The upset plot will tell you how many overlapping genes you have between the data sets, so now you can go in and ask what those genes are.

```
# get the overlapping genes after looking at the upset plot

# if we look at the trimmed_deg_list objects, there is a
# NULL element that I don't fully know how it was inserted,
# did a Google search to find that the compact function
# from tidyverse removes it

trimmed_deg_list = compact(trimmed_deg_list)
trimmed_deg_list_highfemale = compact(trimmed_deg_list_highfemale)
trimmed_deg_list_highmale = compact(trimmed_deg_list_highmale)
data_file_names = compact(data_file_names)

# we know from looking at the upset plot that we see that
# differntially expressed genes are completely overlapping
# in untrimmed and all 9 of the trimmed parameter cases so
# now let's list out the genes that are differentially
# expressed in all the comparisons

# we can use the Reduce function to get the intersect
# (overlap between lists)
```

```
intersecting_elements = Reduce(intersect, list(trimmed_deg_list$trimq_NONE_minlen_NONE_dge_femalevsmale_
    trimmed_deg_list$trimq_0_minlen_10_dge_femalevsmale_all_expressed_genes.csv,
    trimmed_deg_list$trimq_0_minlen_30_dge_femalevsmale_all_expressed_genes.csv,
    trimmed_deg_list$trimq_0_minlen_75_dge_femalevsmale_all_expressed_genes.csv,
    trimmed_deg_list$trimq_10_minlen_10_dge_femalevsmale_all_expressed_genes.csv,
    trimmed_deg_list$trimq_10_minlen_30_dge_femalevsmale_all_expressed_genes.csv,
    trimmed_deg_list$trimq_10_minlen_75_dge_femalevsmale_all_expressed_genes.csv,
    trimmed_deg_list$trimq_30_minlen_10_dge_femalevsmale_all_expressed_genes.csv,
    trimmed_deg_list$trimq_30_minlen_30_dge_femalevsmale_all_expressed_genes.csv,
    trimmed_deg_list$trimq_30_minlen_75_dge_femalevsmale_all_expressed_genes.csv))

# this just displays the intersecting list of genes in the
# report
intersecting_elements
```

```
##  [1] "DDX3Y"        "RPS4Y1"       "ZFY"          "UTY"          "TTTY15"
##  [6] "EIF1AY"       "USP9Y"        "KDM5D"        "FAM224B"      "XIST"
## [11] "PCDH11Y"      "LINC00278"    "PRKY"         "FAM224A"      "EIF4A1P2"
## [16] "PSMA6P1"      "CD24P4"       "VDAC1P6"      "ZFY-AS1"      "AC011297.1"
## [21] "TTTY10"       "ANOS2P"       "TMSB4Y"       "TTTY14"       "TSIX"
## [26] "KRT18P10"     "AC134878.2"   "TBL1Y"        "GYG2P1"       "AC010737.1"
## [31] "TXLNGY"       "LINC00266-4P" "AC015978.1"   "AC007241.1"   "NLGN4Y"
## [36] "KDM5C-IT1"    "CYP2C18"      "CD99"         "ZNF300"
```

```
# if the results were different between certain trimming
# parameters, we could have used this function to get the
# intersections between specific subgroups versus others
```

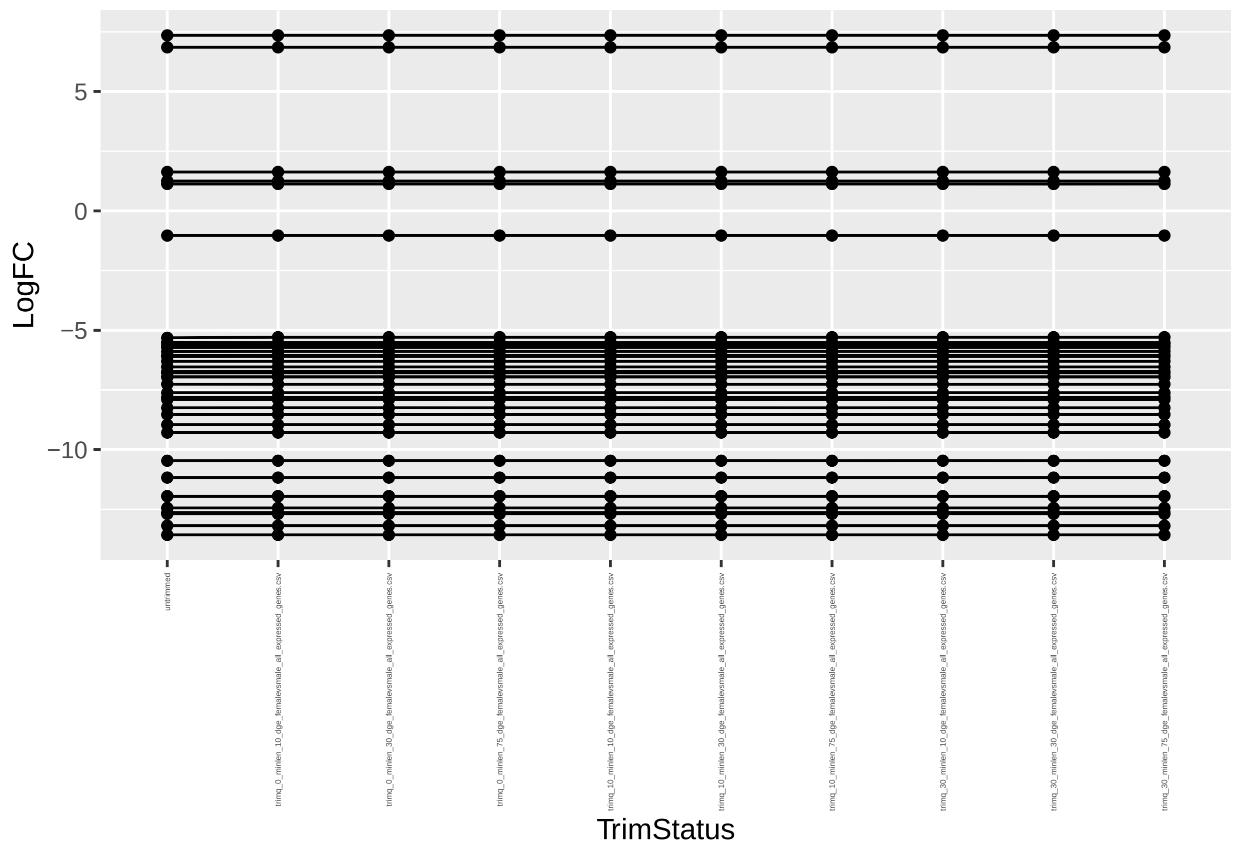
Plotting features of overlapping gene lists

Once you have a list of genes, you will want to plot them to see what features they have in common. We can use subsetting to extract data for these genes. In this chunk, we will model the line graph visualization we used for looking at the log fold change in the untrimmed data compared to one trimmed data set, but this time get the results for multiple trimmed data sets.

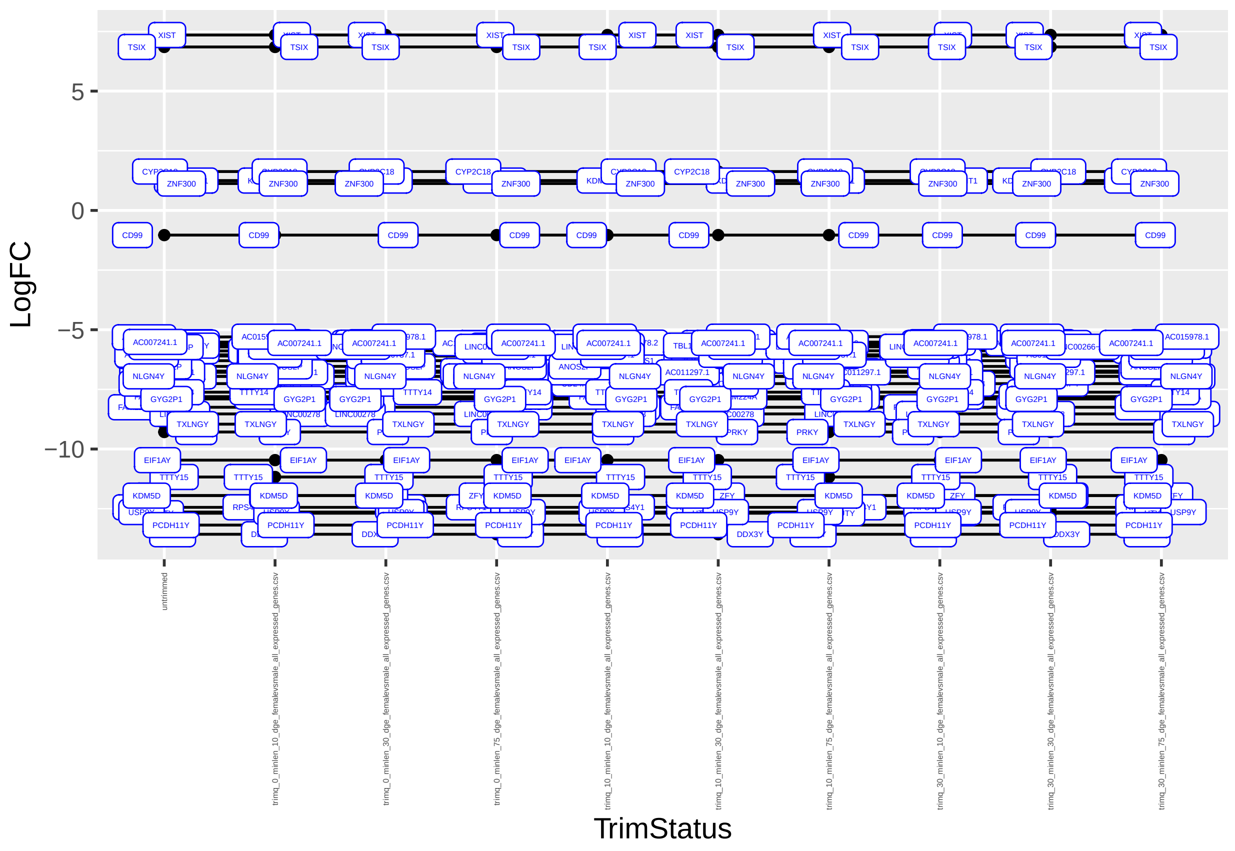
```
# we can use the FCchange chunk of CompareTwo.Rmd from
# Week_4 as a template to look across multiple trimmed data
# sets

# 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 intersecting gene, find its logFC in untrimmed
# and trimmed data to see how it has changed

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

    # find the logFC of this gene in the untrimmed data
```

```r
    untrimmedFC = 0  # set Female:Male foldchange to 0 by default
    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")

    # instead of just doing one trimmed data set, we will
    # iterate through the rest of the trimmed data sets
    # stored in all data

    # look at all the datasets in all_data
    for (trim_set in names(all_data)) {

        # skip the untrimmed since we took care of that one
        # individiually
        if (trim_set != "untrimmed") {

            trimmed_data = all_data[[trim_set]]

            # find the logFC of this gene in the trimmed
            # data
            trimmedFC = 0  # set Female:Male foldchange to 0 by default
            # 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"), trim_set)
        }
    }
}

# set order for plotting
df2$TrimStatus <- factor(df2$TrimStatus, levels = names(all_data))

# 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))

# rotate the x-axis labels, make them smaller, and justify
# them so they are easier to read
```

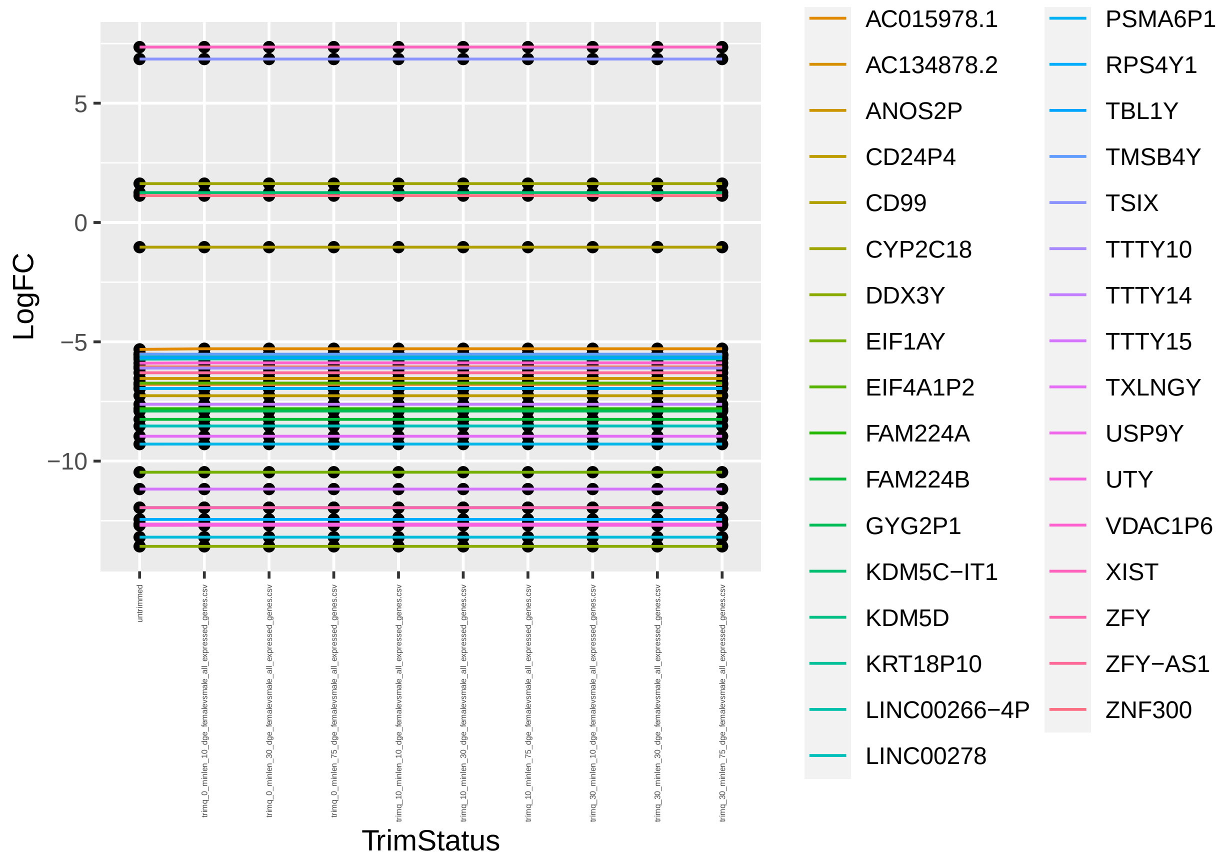
```
p2 = p2 + theme(axis.text.x = element_text(size = 3, angle = 90,
    vjust = 0.5, hjust = 1))

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

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

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



```
# label the genes so you know which one you are looking at
# used position=position_jitter to help separate the labels
# so they aren't overlapping * 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), size = 1, colour = "blue",
    position = position_jitter(width = 0.3, height = 0))

# another way to deal with overlapping labels is the
# geom_label_repel function, you can play around with that
# if you like by replacing the label line above with this
# one p2 = p2 + geom_label_repel(aes(label = gene), nudge_x
# = 0.3, size = 1, colour = 'blue', force = 0.25)

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



```
# HD labelled the genes by color instead to help make it
# easier to distinguish the genes
p2 = ggplot(data = df2, aes(x = TrimStatus, y = LogFC))
p2 = p2 + theme(axis.text.x = element_text(size = 3, angle = 90,
    vjust = 0.5, hjust = 1))
p2 = p2 + geom_point()

# this is what she changed -- give the lines unique color
# based on the gene
p2 = p2 + geom_line(aes(group = gene, color = gene))

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



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] ggrepel_0.9.1   forcats_0.5.2   stringr_1.4.1   dplyr_1.0.10
##  [5] purrr_0.3.5     readr_2.1.3     tidyr_1.2.1     tibble_3.1.8
##  [9] tidyverse_1.3.2 UpSetR_1.4.0    ggplot2_3.3.6
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.9          lubridate_1.8.0     assertthat_0.2.1
##  [4] digest_0.6.30       utf8_1.2.2          R6_2.5.1
##  [7] cellranger_1.1.0    plyr_1.8.7          backports_1.4.1
## [10] reprex_2.0.2        evaluate_0.17       highr_0.9
## [13] httr_1.4.4          pillar_1.8.1        rlang_1.0.6
## [16] googlesheets4_1.0.1 readxl_1.4.1        rstudioapi_0.14
## [19] rmarkdown_2.17      labeling_0.4.2      googledrive_2.0.0
## [22] munsell_0.5.0       broom_1.0.1         compiler_4.2.1
## [25] modelr_0.1.9        xfun_0.34           pkgconfig_2.0.3
## [28] htmltools_0.5.3     tidyselect_1.2.0    gridExtra_2.3
## [31] fansi_1.0.3         crayon_1.5.2        tzdb_0.3.0
## [34] dbplyr_2.2.1        withr_2.5.0         jsonlite_1.8.3
## [37] gtable_0.3.1        lifecycle_1.0.3     DBI_1.1.3
## [40] magrittr_2.0.3      formatR_1.12        scales_1.2.1
## [43] cli_3.3.0           stringi_1.7.8       farver_2.1.1
## [46] fs_1.5.2            xml2_1.3.3          ellipsis_0.3.2
## [49] generics_0.1.3      vctrs_0.4.1         tools_4.2.1
## [52] glue_1.6.2          hms_1.1.2           fastmap_1.1.0
## [55] yaml_2.3.6          colorspace_2.0-3    gargle_1.2.1
## [58] rvest_1.0.3         knitr_1.40          haven_2.5.1
```