Visualization in R ggplot2

Joann Mudge

January 5, 2022

Contents

1 Graphing with ggplot2

1.1 Handy ggplot2 references

- ggplot2 gallery
  - http://www.r-graph-gallery.com/portfolio/ggplot2-package/
- ggplot2 cheatsheets
  - https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf
  - http://ggplot2.tidyverse.org/reference/
- ggplot2 documentation
  - https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf

1.2 Data and setup

Now get your directory setup and open R.

- Create and navigate to your working directory
  - Let's call the directory ggplot2
- Open a screen.
- Make a symbolic link to the data files
  - ln -s /home/jm/unixdata/expression/expr.txt .
  - ln -s /home/jm/unixdata/expression/boxplots.10highestFC.txt .
- Don't forget to "source activate visualization".
- Open R.

1.3 Load the library

```
library(ggplot2)

## Keep up to date with changes at https://www.tidyverse.org/blog/
```

2 Scatter plots

First import the data and take a look at it.

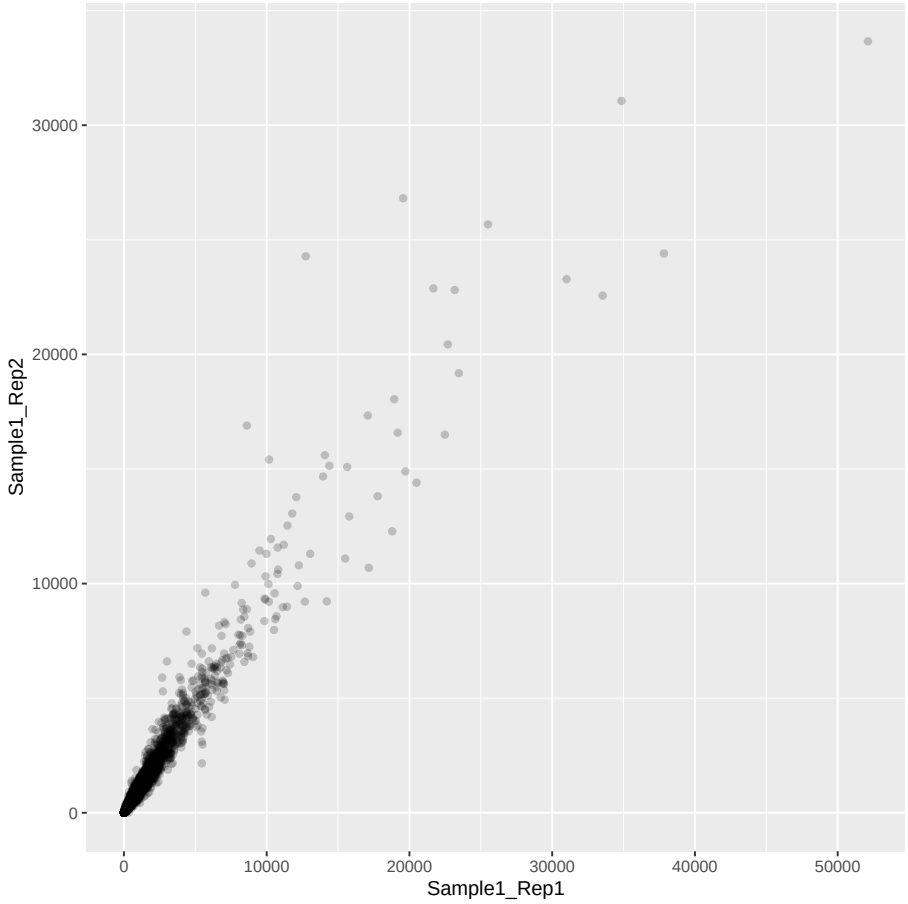
```
expr = read.table("expr.txt", header=TRUE)

head(expr)
```

```
##   Row       Gene                      Function      padj Sample1_Rep1
## 1   2     WASH7P      function=calcium_channel 0.4368729    19.348107
## 2  10 AL627309.1  function=leucine_rich_repeat 0.8411840     3.224684
## 3  13 AL627309.6           function=zinc_finger 0.8974134     8.061711
## 4  14 AL627309.7           function=dna_binding 0.9338057    11.286396
## 5  20 FO538757.1 function=transcription_factor 0.9920602    31.440673
## 6  22 AP006222.1       function=protein_kinase 0.4291107     3.224684
##   Sample2_Rep1 Sample1_Rep2 Sample2_Rep2 Sample1_Rep3 Sample2_Rep3
## 1     9.422449    12.008710    11.988421   20.8571149    11.384693
## 2     2.692128     1.200871     0.000000    0.8690465     1.138469
## 3    12.114578     9.606968    10.898564   14.7737898     5.692347
## 4    18.844899    10.207403     6.539138   14.7737898    13.661632
## 5    24.229156     6.604790    14.168133   11.2976039    11.384693
## 6    13.460642     4.803484     7.628995    2.6071394     3.415408
##   Sample1_Average Sample2_Average Fold_Change
## 1        17.40460        10.93190    0.628104
## 2         1.76487         1.27687    0.723492
## 3        10.81420         9.56850    0.884809
## 4        12.08920        13.01520    1.076600
## 5        16.44770        16.59400    1.008890
## 6         3.54510         8.16835    2.304120
```

Let's compare replicates 1 and 2 for sample 1. The aesthetics (aes) map variables onto parts of the plot. In this case, the x and y axes. We'll put the plot into a variable then print the variable. Later on this will be valuable because we can add onto plots without regenerating them. We'll print the plot to a pdf file.

```
# We open up a pdf that we will write to
pdf("1.expression_replicates.pdf")
```

```r
# Note that you can write this and other commands all on one line
#  but breaking it up makes it easier to read
#  and it doesn't run off the page

scatexpr = ggplot(expr,
        aes(x=Sample1_Rep1,y=Sample1_Rep2)) +
        geom_point(alpha=0.2)

# Now we actually print the plot into the PDF
scatexpr

# This closes off the pdf and allows us to open it (you'll have to copy it to your local com
dev.off()

## pdf
##   2
```

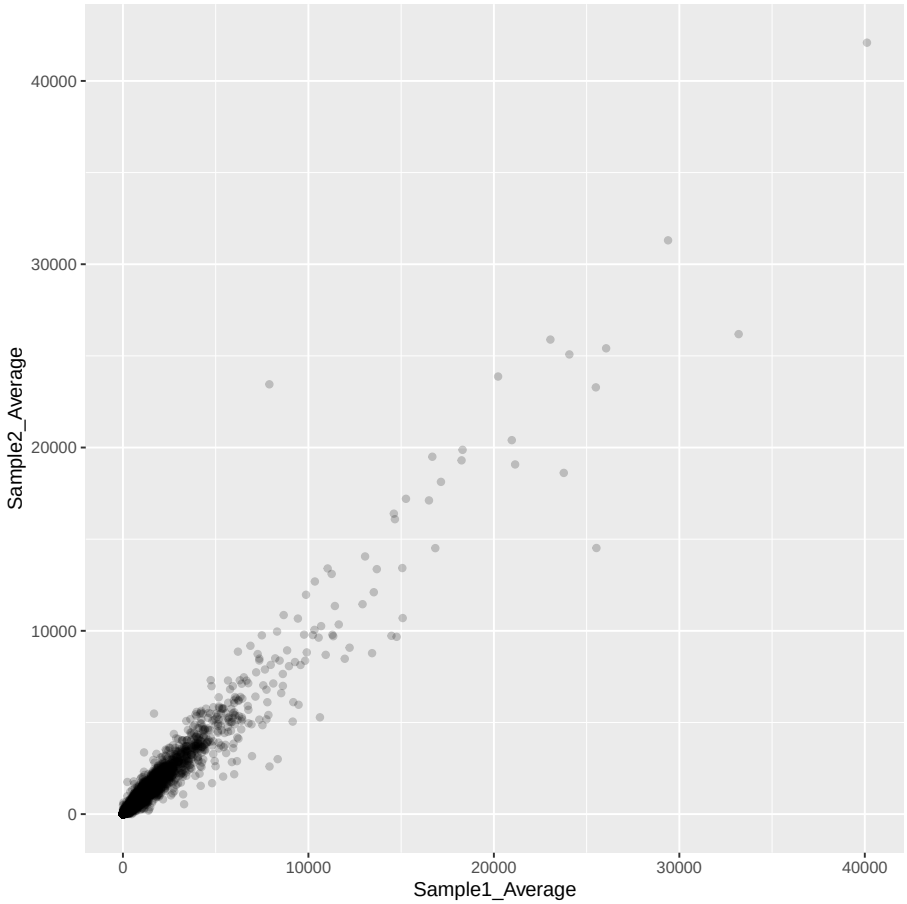


1. Compare reps 2 and 3 for sample 1.
2. Compare two of the reps for sample 2.

Now we'll compare the average of the 3 reps for Sample 1 to the average of the 3 reps for sample 2.

```
pdf("2.expression_compare_samples.pdf")

scatexpr2 = ggplot(expr,
        aes(x=Sample1_Average,y=Sample2_Average)) +
        geom_point(alpha=0.2)

scatexpr2

dev.off()

## pdf
```

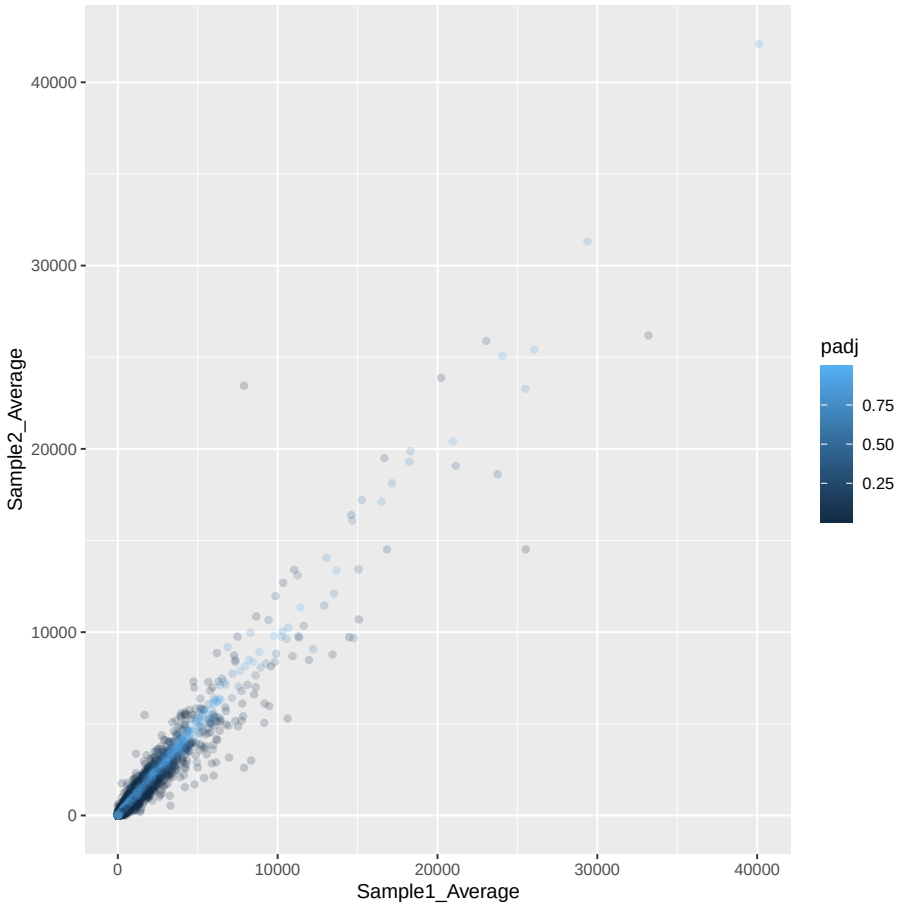
```
##   2
```



Let's add some color. We'll color based on the padj. Because padj is on a continuous scale, ggplot2 will give us a scale for color. If we had used a qualitative variable it would give us discrete colors.

```
pdf("3.expression_compare_samples_color.pdf")

scatexpr3 = ggplot(expr,
        aes(x=Sample1_Average,y=Sample2_Average,color=padj)) +
        geom_point(alpha=0.2)

scatexpr3

dev.off()

## pdf
```

```
##   2
```

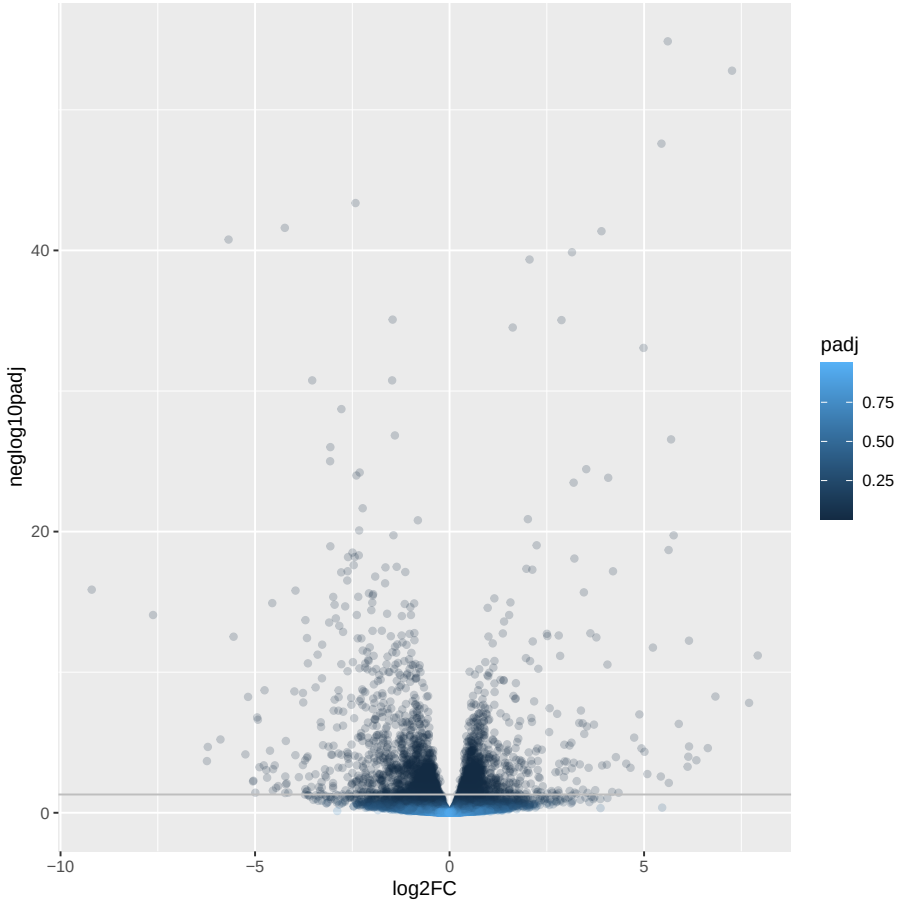


And we can even make a volcano plot in ggplot2. The x axis is the log2 of the fold change (Fold_Change). This transformation creates the symetrical "V" shape of the Volcano Plot. The y axis is the negative log10 of the p value (padj), which makes lower number (more significant) higher on the plot. We'll also add a line at p=0.05 (on the graph this will be -log10(p)=1.3). Those above the line are significant. This time, we'll add layers to the base graph we put in the "volcano" variable rather than redoing the graph from scratch.

```
pdf("4.volcano.pdf")

# Create the negative log10 of padj in another column
# expr$neglog10padj means we want to make a column called neglog10padj in the expr data fram
#    and we'll put the -log10 of the padj column from the expr data frame into it

expr$neglog10padj = -log10(expr$padj)
```

```r
# We'll create the x variable up front as well
expr$log2FC = log2(expr$Fold_Change)

volcano = ggplot(expr,
        aes(x=log2FC,y=neglog10padj,color=padj)) +
        geom_point(alpha=0.2) +
    geom_hline(yintercept=1.3, color="gray")

volcano

## Warning:  Removed 30 rows containing missing values (geom_point).

dev.off()

## pdf
##   2
```

```
## Warning:  Removed 30 rows containing missing values (geom_point).
```



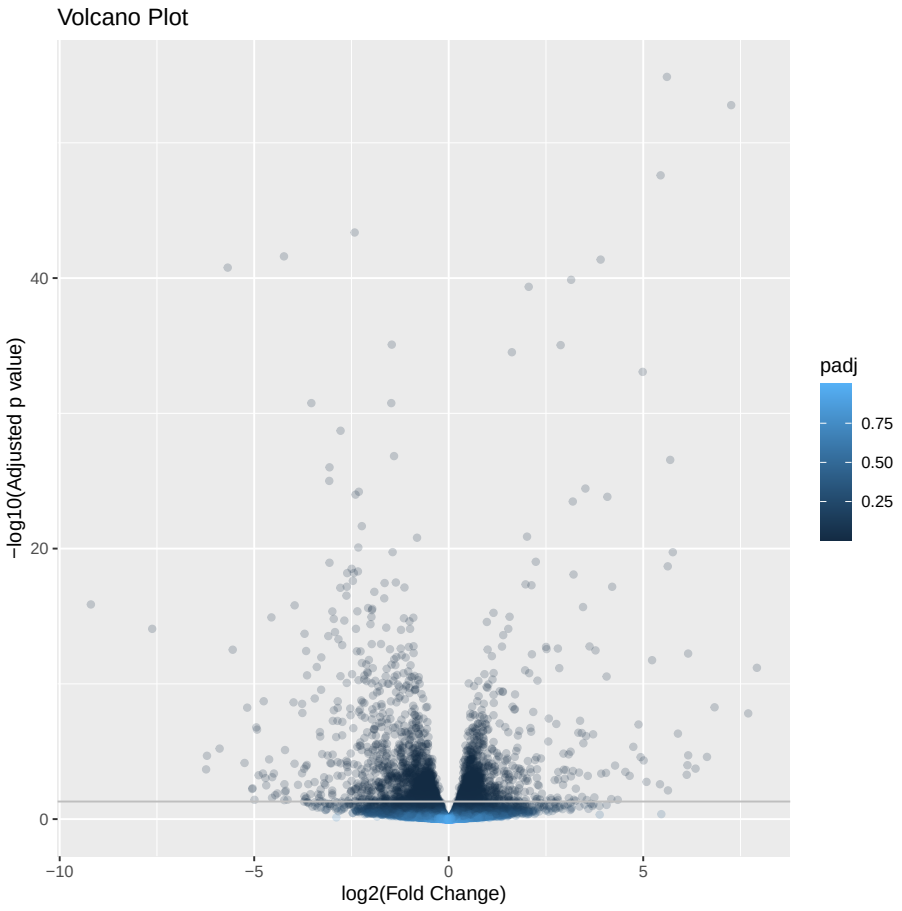
We can add layers onto this graph. We'll add a title and better axis labels.

Because we have the base plot in the "volcano" variable, we can start there. We could also rerun the plot and add these on at the end of the original code.

```
pdf("5.volcano.improved.pdf")

volcano +
        xlab("log2(Fold Change)") +
        ylab("-log10(Adjusted p value)") +
        ggtitle("Volcano Plot")

## Warning:  Removed 30 rows containing missing values (geom_point).

dev.off()
```

```
## pdf
##   2
```

```
## Warning: Removed 30 rows containing missing values (geom_point).
```



3 Boxplots

In expression data, you often want to print a boxplot of some of your interesting genes based on differential expression and/or the biology of the gene. I grabbed the 10 genes with the highest fold change to practice with. The farther to the right you go in the data columns, the higher the fold change. Let's import the data.

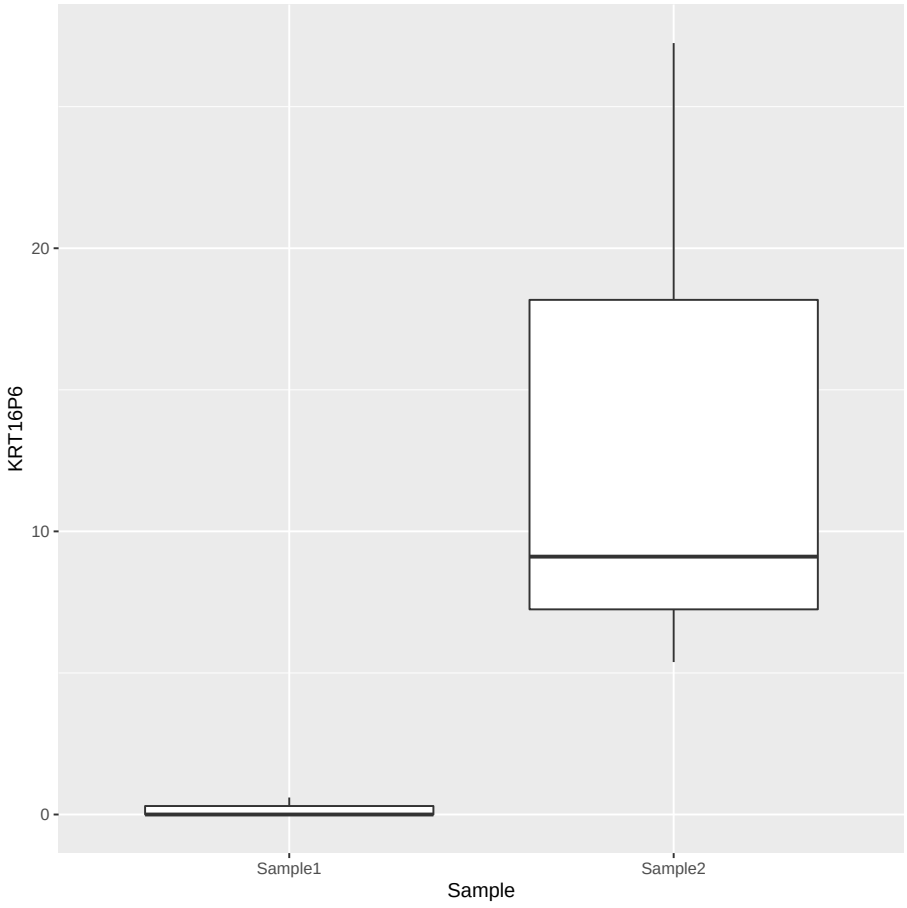
```
   box = read.table("boxplots.10highestFC.txt", header=TRUE)

head(box)
```

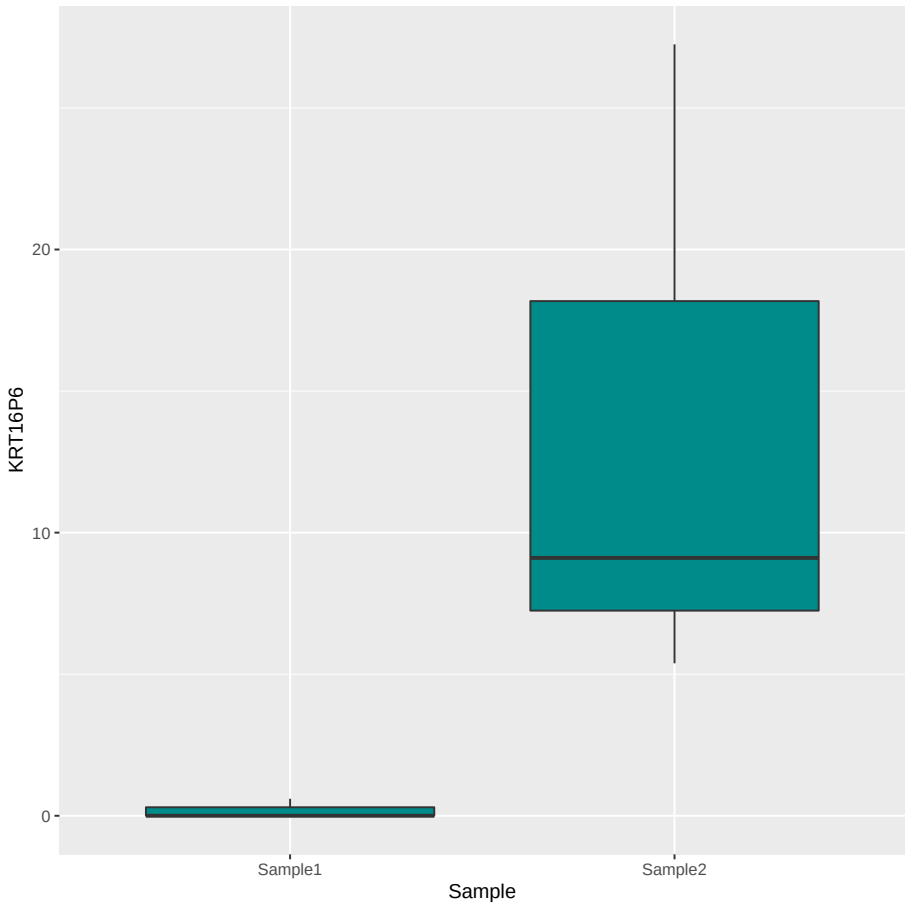
```
##    Sample    KRT16P6     ZNF711    ZNF354C BX322639.1    SLC6A14  LINC00960
## 1 Sample1  0.0000000  0.0000000  2.4185133  0.0000000  0.0000000  0.0000000
## 2 Sample2  5.3842568 12.1145778 79.4177879 10.7685136 13.4606420 18.8448988
## 3 Sample1  0.6004355  0.6004355  0.0000000  0.0000000  0.6004355  0.6004355
## 4 Sample2 27.2464104 13.0782770 77.3798056 26.1565540 23.9768412 25.0666976
## 5 Sample1  0.0000000  0.0000000  0.8690465  0.8690465  0.0000000  0.0000000
## 6 Sample2  9.1077546 17.0770398 77.4159138 25.0463250 11.3846932 15.9385705
##        NOVA1       GRM8    SLC12A7     RPS6KA6
## 1  0.8061711   5.643198  0.0000000   0.0000000
## 2 48.4583113 538.425681 52.4965039 141.3367412
## 3  0.0000000   1.200871  0.0000000   0.6004355
## 4 68.6609542 446.841131 63.2116721 101.3566467
## 5  0.8690465   2.607139  0.8690465   0.8690465
## 6 74.0005058 464.495483 64.8927512 113.8469320
```

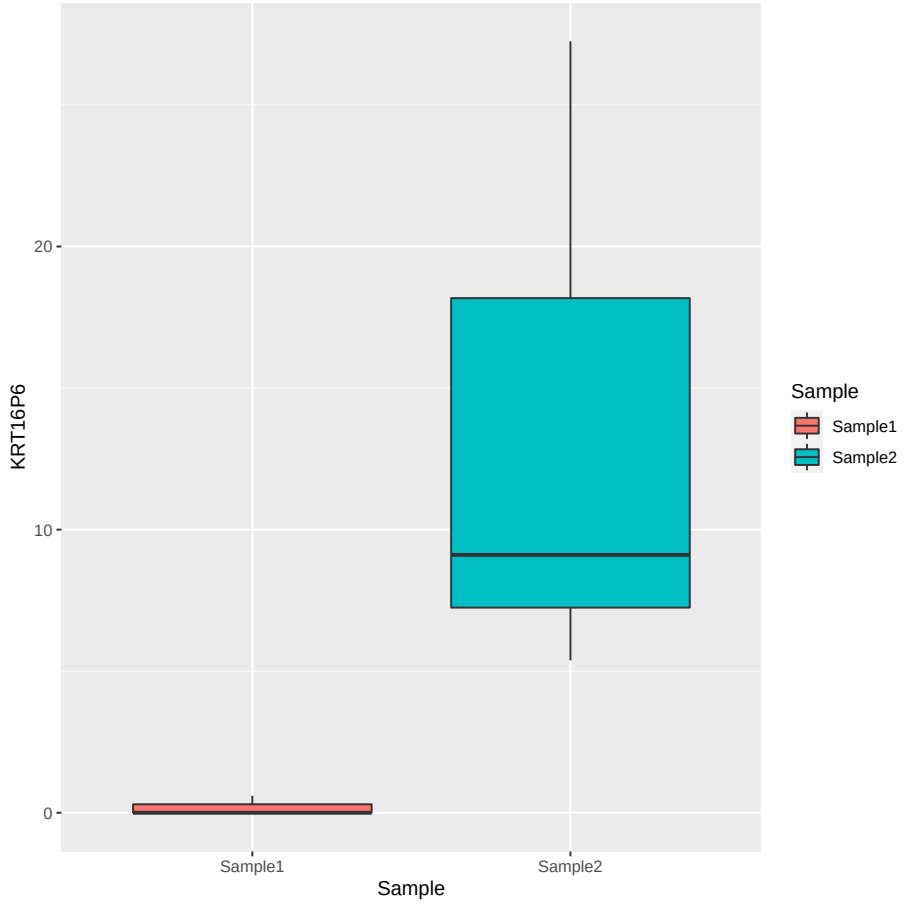
We'll use the first gene (KRT16P6). The first part, including the aesthetics takes the same form we saw in the scatterplots. Instead of adding a geom_point() layer, we'll add a geom_boxplot() layer. We'll make a couple of different versions but put them all into a single PDF. Each plot will be on a separate page. 1) Box, 2) Color (same color for both samples), 3) Color by sample.

```
   pdf("6.boxplot.pdf")

# Basic boxplot

boxplot1 = ggplot(box,
        aes(x=Sample,y=KRT16P6)) +
        geom_boxplot()

# Coloring everything the same color
# Because we need to redo the aesthetics we'll start from scratch here
# R has a lot of built in color names it recognizes (google "R colors")
#  though you can specificy colors with RGB or other color space

boxplot2 = ggplot(box,
        aes(x=Sample,y=KRT16P6)) +
        geom_boxplot(fill="darkcyan")

                # The fill can also go up in the main aesthetics
```

```
boxplot3 = ggplot(box,
        aes(x=Sample,y=KRT16P6,fill=Sample)) +
        geom_boxplot()


# print them all
boxplot1
boxplot2
boxplot2

dev.off()

## pdf
##   2
```




KRT16P6
20
10
0
Sample1
Sample2
Sample



Try making boxplots from some of the other genes.

4 Heatmaps

Heatmaps are easy to make in ggplot2. We'll use the expr dataset that we have already imported into R. But we have a lot of genes so let's do some filtering. We'll get all genes that are significant at a stringent cutoff (p<=0.01). We'll also sort by log2(fold change) which will organize the data by those most overexpressed in one sample to those most overexpressed in the other sample. We'll use the dplyr package to sort and filter.

```
library(dplyr)

##
## Attaching package:  'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##    filter, lag
## The following objects are masked from 'package:base':
##
##    intersect, setdiff, setequal, union

# How many genes do we have before filtering?
# Get the data frame dimensions which gives rows and columns
# The number of genes is the number of rows - 1 (to account for the headers)

dim(expr)

## [1] 14926    15

#Get significant genes

sigexpr = expr %>% filter(padj <= 0.01)


# Sort by log2(fold change)

sortsigexpr = arrange(sigexpr,log2FC)
```

How many significant genes do you have?

Next, we have to change the formatting. Instead of a column for each sample/replicate combination, we need those expression values to all to be in a single column (the table will get much longer). Then we'll need another column that states what the sample/replicate combination is for that row. This will make more sense once we run it and look at the outcome. We'll use the melt function in the reshape2 package.

```
library(reshape2)

# Look at the current data format

head(sortsigexpr)

##     Row       Gene                      Function         padj Sample1_Rep1
## 1  2423      FNDC7  function=leucine_rich_repeat 1.349099e-16    474.83479
## 2 40503 AC009292.2      function=calcium_channel 8.562505e-15    353.90912
## 3   272    TNFRSF9 function=transcription_factor 2.104677e-04     50.78878
## 4 52959     COL6A2          function=dna_binding 2.074331e-05     46.75792
## 5 29453      PLPP4          function=zinc_finger 6.107372e-06     53.20729
## 6 47227       CDH2      function=calcium_channel 1.683578e-41    289.41543
##   Sample2_Rep1 Sample1_Rep2 Sample2_Rep2 Sample1_Rep3 Sample2_Rep3
```

```
## 1     1.346064    433.51443     1.089856    522.29692     0.000000
## 2     0.000000    142.30321     1.089856    166.85692     2.276939
## 3     0.000000     19.21394     1.089856     12.16665     0.000000
## 4     1.346064     28.82090     0.000000     24.33330     0.000000
## 5     0.000000     34.22482     2.179713     41.71423     0.000000
## 6     5.384257    369.26783     9.808708    412.79707     5.692347
##   Sample1_Average Sample2_Average Fold_Change neglog10padj    log2FC
## 1        476.8820        0.811974  0.00170267    15.869956 -9.197985
## 2        221.0230        1.122270  0.00507762    14.067399 -7.621632
## 3         27.3898        0.363285  0.01326350     3.676814 -6.236395
## 4         33.3040        0.448688  0.01347250     4.683122 -6.213839
## 5         43.0488        0.726571  0.01687780     5.214146 -5.888729
## 6        357.1600        6.961770  0.01949200    40.773767 -5.680974
```

```
heat = melt(sortsigexpr,
        id.vars = c("Gene", "Function"),
        measure.vars =
        c("Sample1_Rep1", "Sample2_Rep1", "Sample1_Rep2",
                "Sample2_Rep2", "Sample1_Rep3", "Sample2_Rep3"))

                # The "c" above is used to combine multiple arguments


# This is the new data format

head (heat)
```

```
##         Gene                       Function     variable     value
## 1      FNDC7  function=leucine_rich_repeat Sample1_Rep1 474.83479
## 2 AC009292.2      function=calcium_channel Sample1_Rep1 353.90912
## 3    TNFRSF9 function=transcription_factor Sample1_Rep1  50.78878
## 4     COL6A2          function=dna_binding Sample1_Rep1  46.75792
## 5      PLPP4          function=zinc_finger Sample1_Rep1  53.20729
## 6       CDH2      function=calcium_channel Sample1_Rep1 289.41543
```

The melt function automatically names the new columns "variable" and "value" but that's easy to fix.

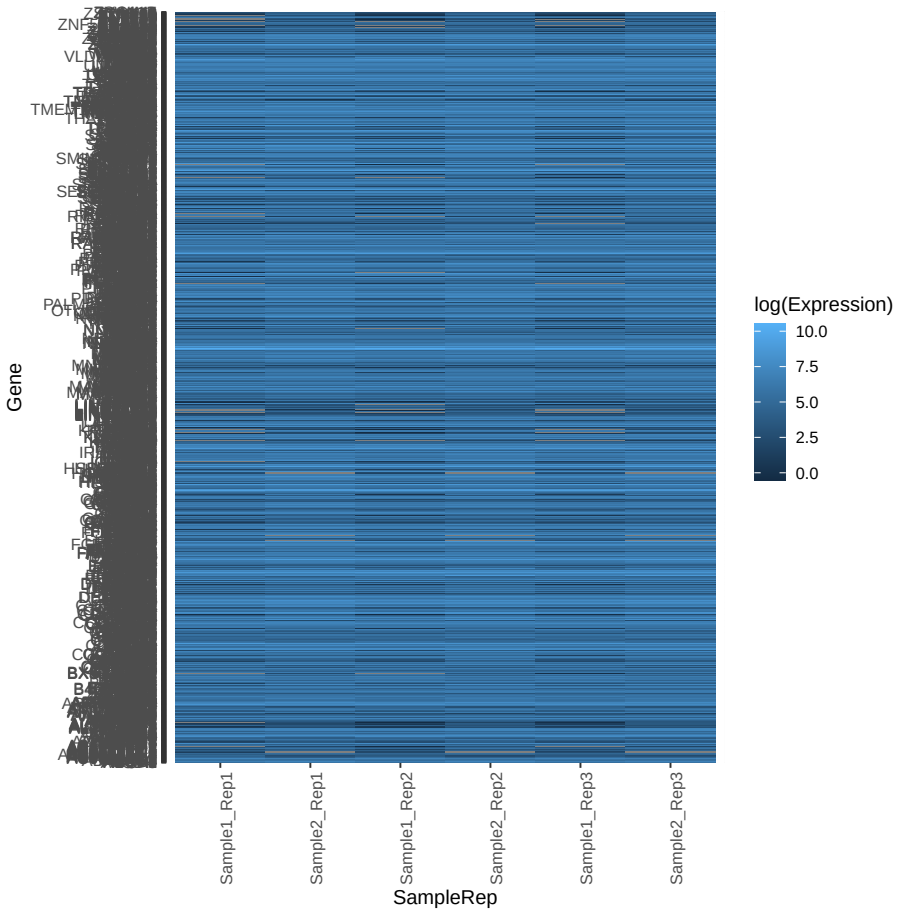
```
colnames(heat)=c("Gene","Function","SampleRep","Expression")

head(heat)
```

```
##         Gene                       Function    SampleRep Expression
## 1      FNDC7  function=leucine_rich_repeat Sample1_Rep1  474.83479
## 2 AC009292.2      function=calcium_channel Sample1_Rep1  353.90912
```

```
## 3    TNFRSF9 function=transcription_factor Sample1_Rep1   50.78878
## 4     COL6A2          function=dna_binding Sample1_Rep1   46.75792
## 5      PLPP4          function=zinc_finger Sample1_Rep1   53.20729
## 6       CDH2      function=calcium_channel Sample1_Rep1  289.41543
```

Now for the heatmap. We use geom_tile(). We still have a lot of genes so we are going to change the length of our PDF (but it will be scrunched in this document). Because there is a huge range in expression, we'll use the log.

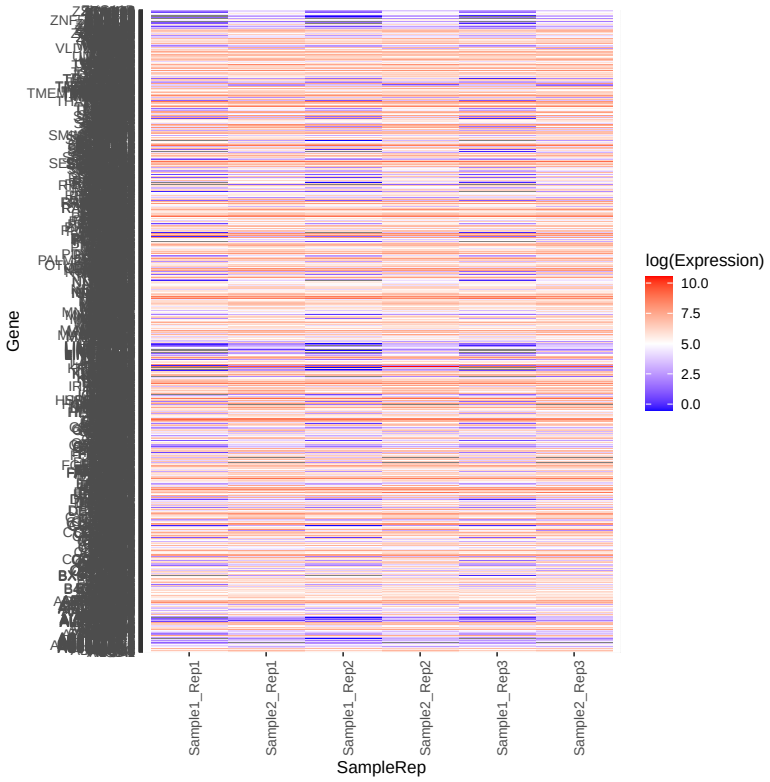
```
pdf("7.heatmap.pdf", height = 200)

heatmapplot = ggplot(heat,
        aes(x=SampleRep, y=Gene)) +
        geom_tile(aes(fill = log(Expression))) +
        theme(axis.text.x=element_text(angle=90))
                # This rotates the x axis labels

heatmapplot


dev.off()

## pdf
##   2
```

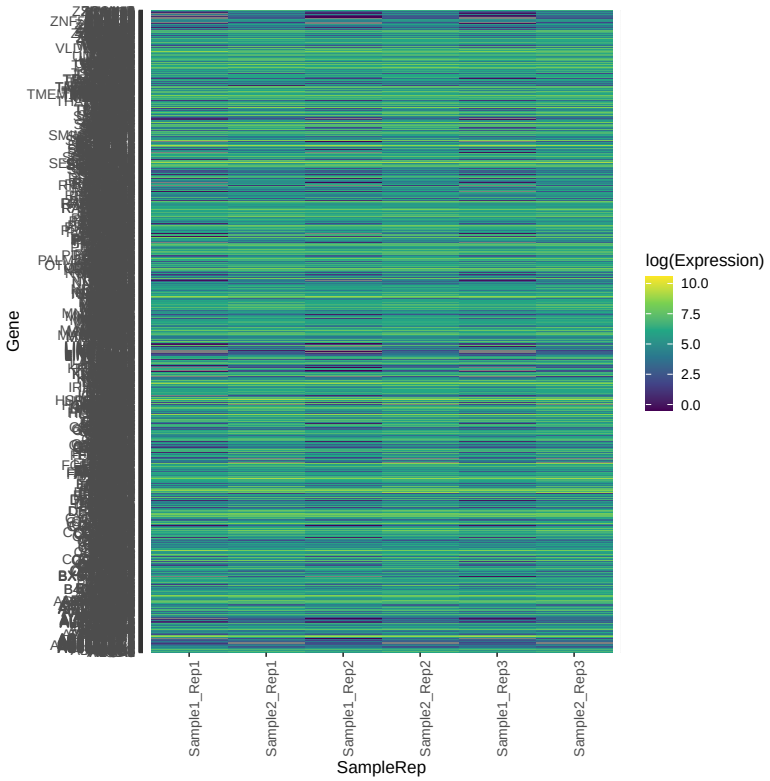


Let's try out a couple of other colorscales. We can start with the plot we put into the "heatmap" variable. Since we aren't putting the new plots into a variable, they'll plot directly to the PDF. We'll use the red-white-blue scale (red high) that is often used in expression and then the viridis color scale (purple through yellow; yellow high).

```
pdf("8.heatmaps.color.pdf", height=200)

heatmapplot +
        scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint=5)

heatmapplot +
        scale_fill_viridis_c()
                # c is for continuous scale

dev.off()
```

```
## pdf
##   2
```


Gene
SampleRep
Sample1_Rep1
Sample2_Rep1
Sample1_Rep2
Sample2_Rep2
Sample1_Rep3
Sample2_Rep3
log(Expression)
10.0
7.5
5.0
2.5
0.0



Try to make a heatmap using the Sample1 and Sample2 averages. Hint: you will have to go back to the sortsigexpr data frame to get the columns with the averges. You will have to remelt and resort and rename the columns before plotting.

If we want to add dendograms to the ggplot2 heatmap there is a way but it is a pain. Here is a better way to do heatmaps and dendograms. We'll go back to the sigexpr data frame and remove a few columns. We'll use the ComplexHeatmap library. More documentation is here: https://jokergoo.github.io/ComplexHeatmap-reference/book/. We'll make a heatmap with the expession values and another with the log10 of the expression values.

```
# Requires a matrix
# In our case we'll use a genes x sample_reps matrix

library(ComplexHeatmap)

## Loading required package:  grid
## ========================================
## ComplexHeatmap version 2.4.2
## Bioconductor page:  http://bioconductor.org/packages/ComplexHeatmap/
```

```
## Github page:  https://github.com/jokergoo/ComplexHeatmap
## Documentation:  http://jokergoo.github.io/ComplexHeatmap-reference
##
## If you use it in published research, please cite:
## Gu, Z. Complex heatmaps reveal patterns and correlations in multidimensional
##  genomic data.  Bioinformatics 2016.
##
## This message can be suppressed by:
##  suppressPackageStartupMessages(library(ComplexHeatmap))
## ========================================
```

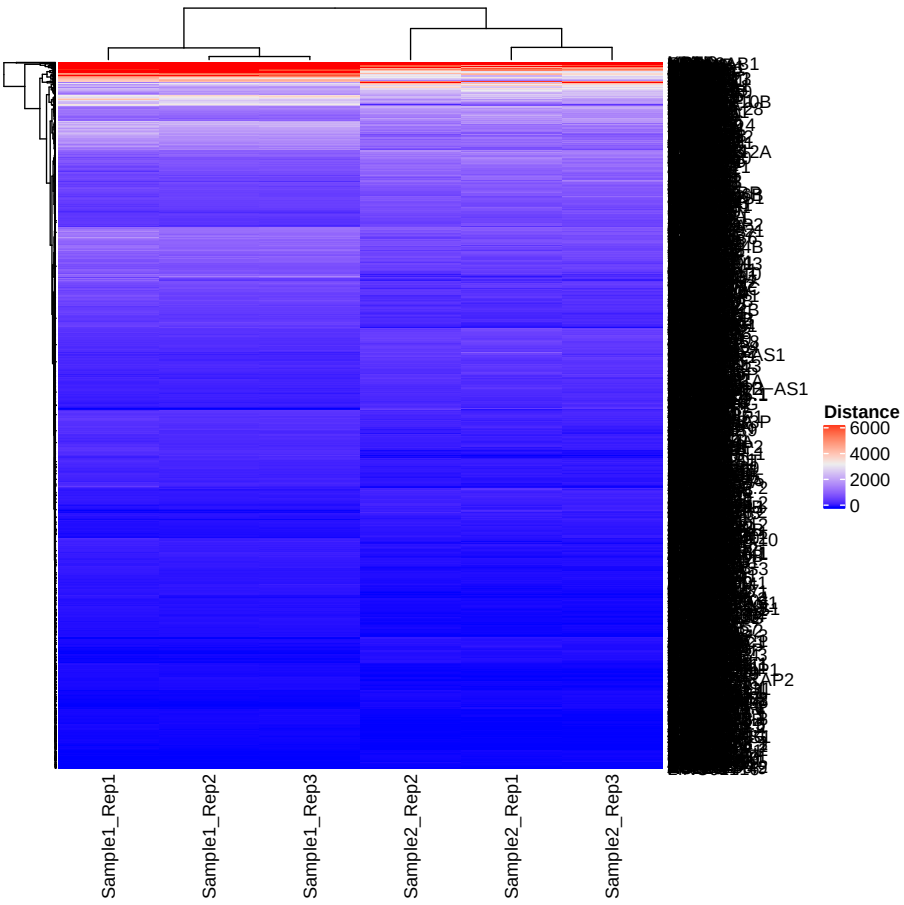
```
# Make the genes the rownames
rownames(sigexpr) = sigexpr$Gene


# Grab just the rep data
sigexpr2 = sigexpr[,5:10]

# Transform for the log version
# The +1 avoids getting -Inf for 0s, instead 0s get log(1)=0.
sigexpr2log = log10(sigexpr2+1)


# Check the data frames
head(sigexpr2)
```

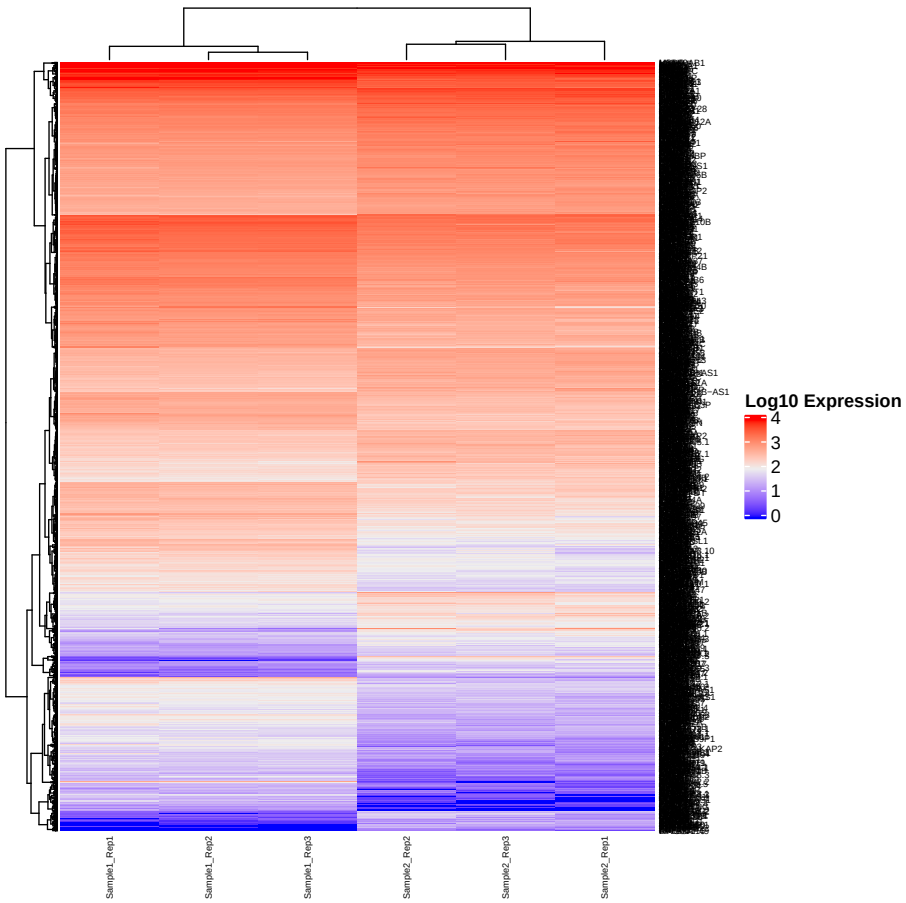
```
##                                                                  mple1_Rep3
##                                                                    179.8926
##                                                                   1188.8556
##                                                                    232.9045
##                                                                    202.4878
##                                                                    242.4640
##                                                                    244.2021
```

```
head(sigexpr2log)
```

```
##                                                                  mple1_Rep3
##                                                                    2.257421
##                                                                    3.075494
```

```
## PRKCZ        2.326780     2.515998     2.353268     2.546461     2.369038
## TP73-AS1     2.366182     2.177286     2.340361     2.186334     2.308538
## PLEKHG5      2.341380     2.157405     2.318377     2.213222     2.386435
## KLHL21       2.491030     2.292653     2.379075     2.221827     2.389524
##          Sample2_Rep3
## ISG15        1.918920
## SSU72        2.973237
## PRKCZ        2.450560
## TP73-AS1     2.176495
## PLEKHG5      2.145797
## KLHL21       2.131424

# Change the data frame into a matrix
sigexpr2matrix = as.matrix(sigexpr2)

# Make another version with the log data
sigexpr2matrixlog = as.matrix(sigexpr2log)


pdf("9.heat.dendo.pdf",width=5,height=80)

# Name is the name of the legend

# Expression values
Heatmap(sigexpr2matrix, name = "Expression", row_names_gp = gpar(fontsize = 5),
        column_names_gp = gpar(fontsize = 5) )

# Log10 of Expression values
Heatmap(sigexpr2matrixlog, name = "Log10 Expression", row_names_gp = gpar(fontsize = 5),
        column_names_gp = gpar(fontsize = 5) )

dev.off()

## pdf
##   2
```


Distance
6000
4000
2000
0
Sample1_Rep1
Sample1_Rep2
Sample1_Rep3
Sample2_Rep2
Sample2_Rep1
Sample2_Rep3



5 Homework

Use ggplot2 to plot other types of graphs
(http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html)