Visualization in R ggplot2

Joann Mudge

February 3, 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 metag_ggplot2
- Open a screen.
- Make a symbolic link to the data files
  - ln -s /home/jm/unixdata/metag/97-feature-table.tsv .
  - ln -s /home/jm/unixdata/metag/metag_boxplots.txt .
- Don't forget to "source activate visualization".
- Open R by just typing "R".

1.3 Load the library

```
library(ggplot2)

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

2 Scatter plots

We'll start with simple scatter plots to learn the syntax. First take a look at the data.

Evan created a text version of the tuberculosis study otu table using the following commands:

- qiime tools export table.qza --output-dir ./
- biom convert -i feature-table.biom -o feature-table.tsv --to-tsv
- biom head -i feature-table.tsv

I then tweaked the file slightly as follows:

- Removed the first line (# Constructed from biom file)
- Removed # from the second line and changed "OTU ID" to "OTU_ID"

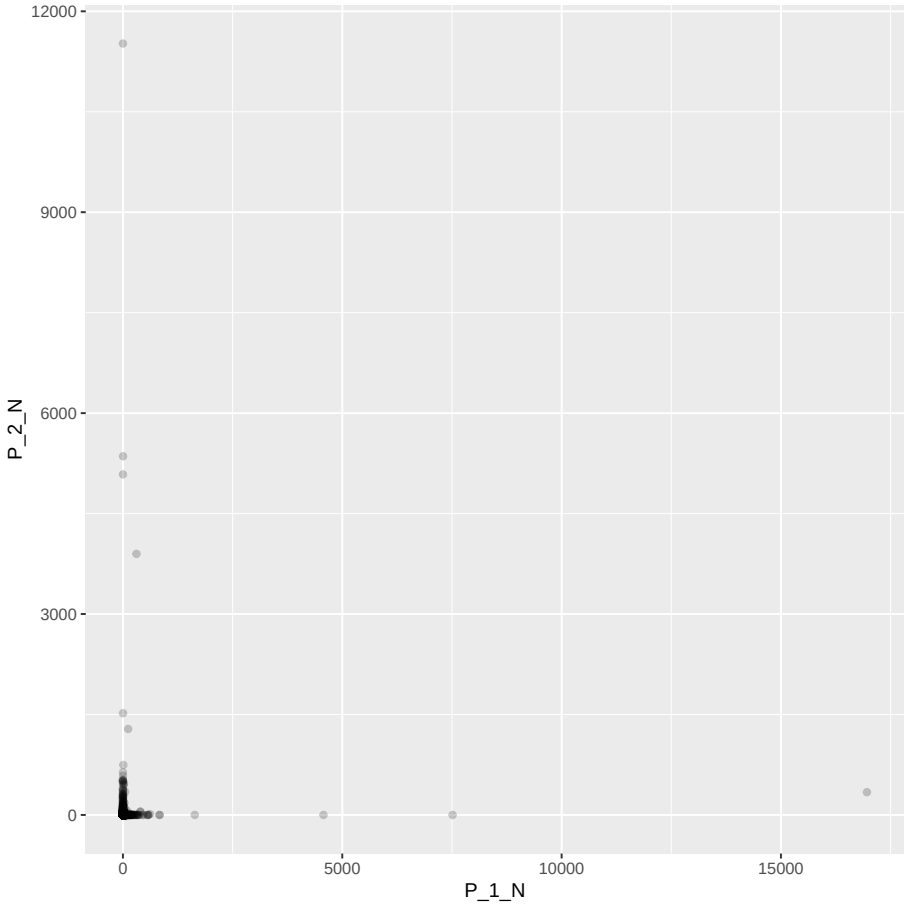
```
mg = read.table("97-feature-table.tsv", header=TRUE)

head(mg)
```

```
##                                     OTU_ID C_1_N C_1_P C_2_N C_2_P C_3_N C_3_P
## 1                                   4300554     0     0     0     0     0     0
## 2 d19cb639236fd5b84c89ea6b4aab6cefc9c877f8     0 15500     0     0     0     0
## 3 ab22afd6d23316687b71ff900f1a5c09a75f6ace   308   575     0   334    17    83
## 4 759ca279854b21dfe3192a35d285b428d2f47179     0     0     0     0     0     0
## 5 cbd65e36fa7b632c0f56c4b6e914ecbc0e430382     0     0     0     0     0     0
## 6 0a511a2d3b4e86ecddd65f83a49638837840fe4c   611     0  2328     1  4141     0
##   C_4_N C_4_P C_5_N C_5_P C_6_N C_6_P P_1_N P_1_P P_1_S P_2_N P_2_P P_2_S P_3_N
## 1     2     0     0     0     0     0     0     0     0     0     0     0     0
## 2     0     0     1     0     0     0     0     0     0     0    29     0     5
## 3    16   278     0    71     3     1     3   759 29616   280   431     0    33
## 4     0     0     0     0     0     0     0     0     0     0     0     0     0
## 5     0     0     0     0     0     0     0     0     0     0     0 41007     0
## 6   659     4  7448     0   691     0 16961     4     6   340     2     3 19959
##   P_3_P P_3_S P_4_N P_4_P P_4_S P_5_N P_5_P P_5_S P_6_N P_6_P P_6_S
## 1     0     0     0     0     0     0     0     0     0     0     0
## 2     0     0     0     0     0     0     0     0     0     3     0
## 3  1925   553     3   365   859   160     1    15   185  3002     0
## 4     0     0     0     0     0     0     0     0 28159     0     0
## 5     0     0    71     0     0     0     0     0     0     0     0
## 6     3     7    10     7     2 19005     0    10     7     3     1
```

Let's compare samples P_1_N and P_2_N (nasal samples from patient 1 and 2, respectively). 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.

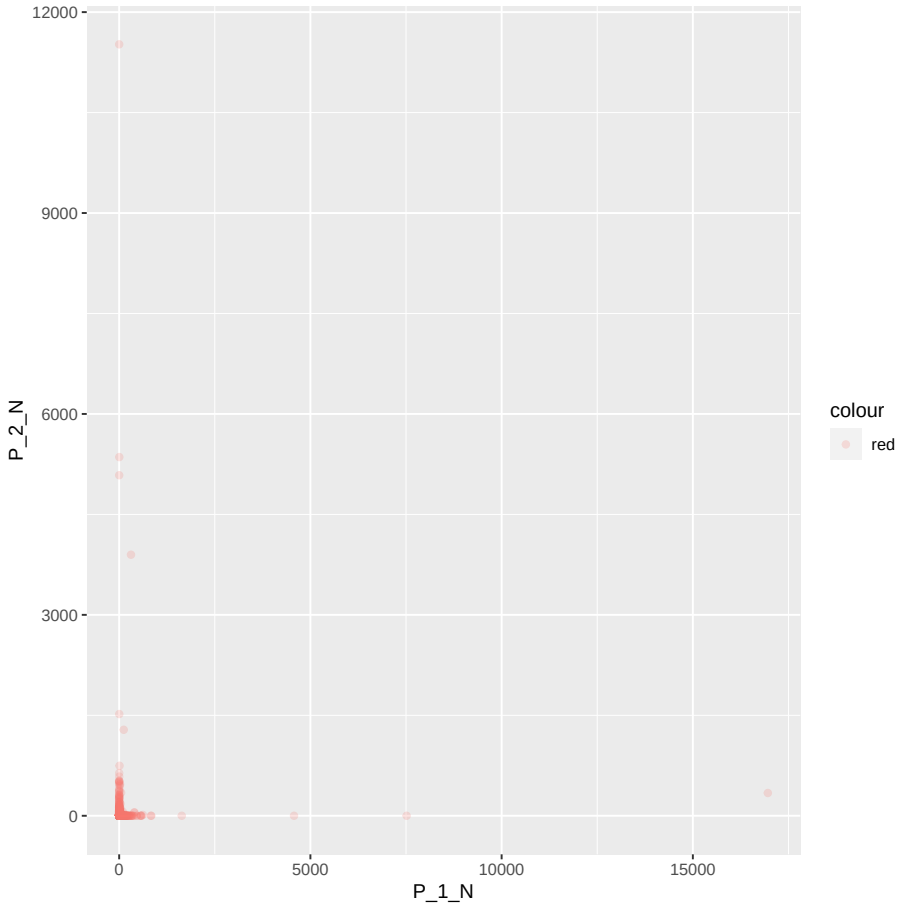
```
# We open up a pdf that we will write to
pdf("scatter.pdf")

# 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

# The alpha tells it to make the dots 80% transparent

scatmg = ggplot(mg, aes(x=P_1_N,y=P_2_N)) +
        geom_point(alpha=0.2)

# Now we actually print the plot into the PDF
scatmg

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

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



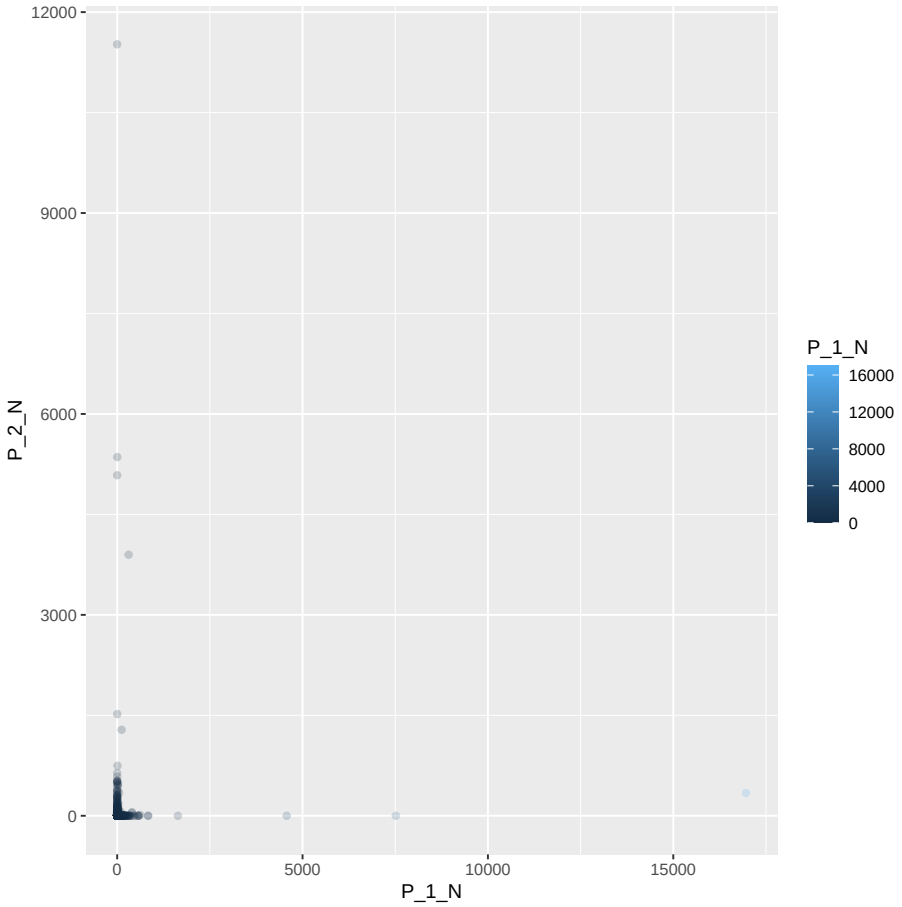
Let's add some color. We'll color all the dots red.

```
pdf("scatter_color.pdf")

scatmg2 = ggplot(mg,
        aes(x=P_1_N,y=P_2_N,color="red")) +
        geom_point(alpha=0.2)

scatmg2

dev.off()

## pdf
##   2
```



We can also color by a variable. We'll use the counts of sample P_1_N. Because P_1_N 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("scatter_color_zoom.pdf")

scatmg3 = ggplot(mg,
        aes(x=P_1_N,y=P_2_N,color=P_1_N)) +
        geom_point(alpha=0.2)

scatmg3

dev.off()

## pdf
##   2
```

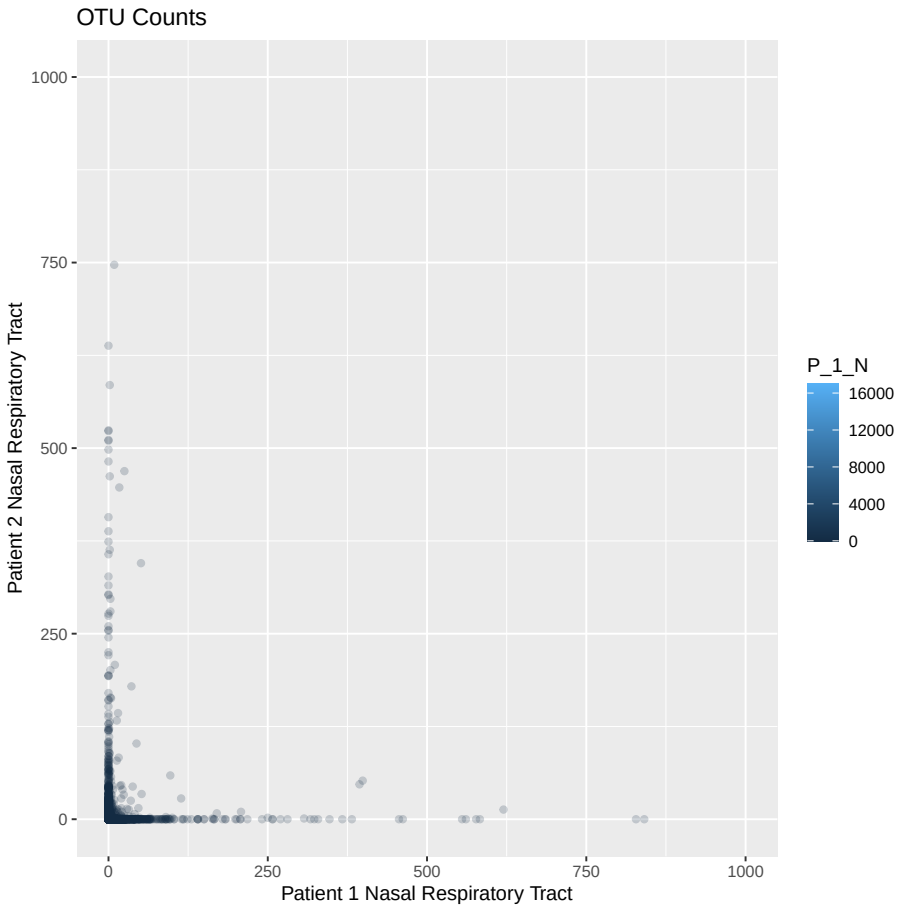


The nice thing about ggplot2 is that you can add layers. We could rewrite the full plot, but because we put the plot into a variable, we can simply add layers to the plot variable. We'll add onto that base graph some limits on the x and y axis to zoom into that bottom lefthand corner. And we'll also color by a variable. We'll use the counts of sample P_1_N. Because P_1_N 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. We'll simply write over the geom_point already contained in scatmg2.

```
pdf("scatter_color_zoom.pdf")

scatmg4 = scatmg3 +
        xlab("Patient 1 Nasal Respiratory Tract") +
        ylab("Patient 2 Nasal Respiratory Tract") +
        ggtitle("OTU Counts") +
        xlim(0,1000) +
        ylim(0,1000)
```

```
scatmg4

## Warning:  Removed 10 rows containing missing values (geom_point).

dev.off()

## pdf
##   2
```

```
## Warning:  Removed 10 rows containing missing values (geom_point).
```



1. Now choose some samples to compare.

- Try some different alpha values and colors (http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf) or color variables.
- Try to zoom in even more.

3 Boxplots

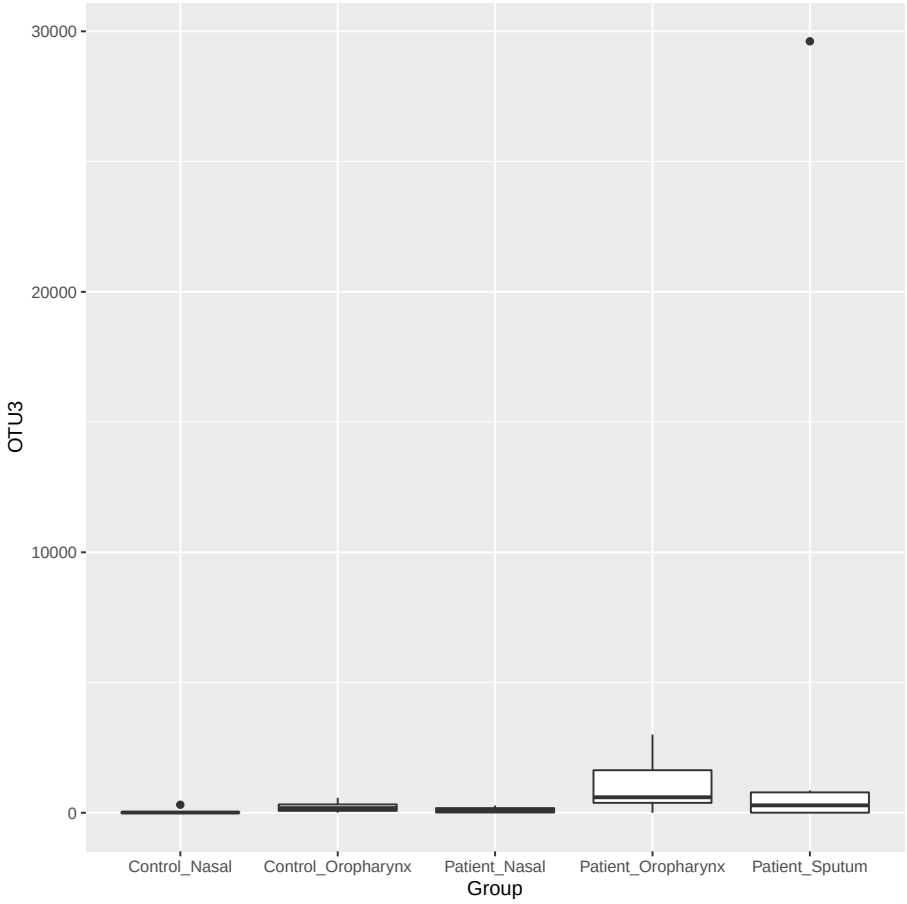
If you find an interesting OTU, you can print a boxplot. They show the mean and the interquartile range. The whiskers go to the farthest distal value from the nearest quartile that is no farther than 1.5 * IQR beyond the nearest quartile. Any outliers are plotted as individual dots. I grabbed the first 10 OTUS to practice with. I transposed the file and replaced the individual sample name with their group (ie. Control_Nasal). I also replaced the superlong OTU IDs with OTU1-OTU10. Let's import the data.

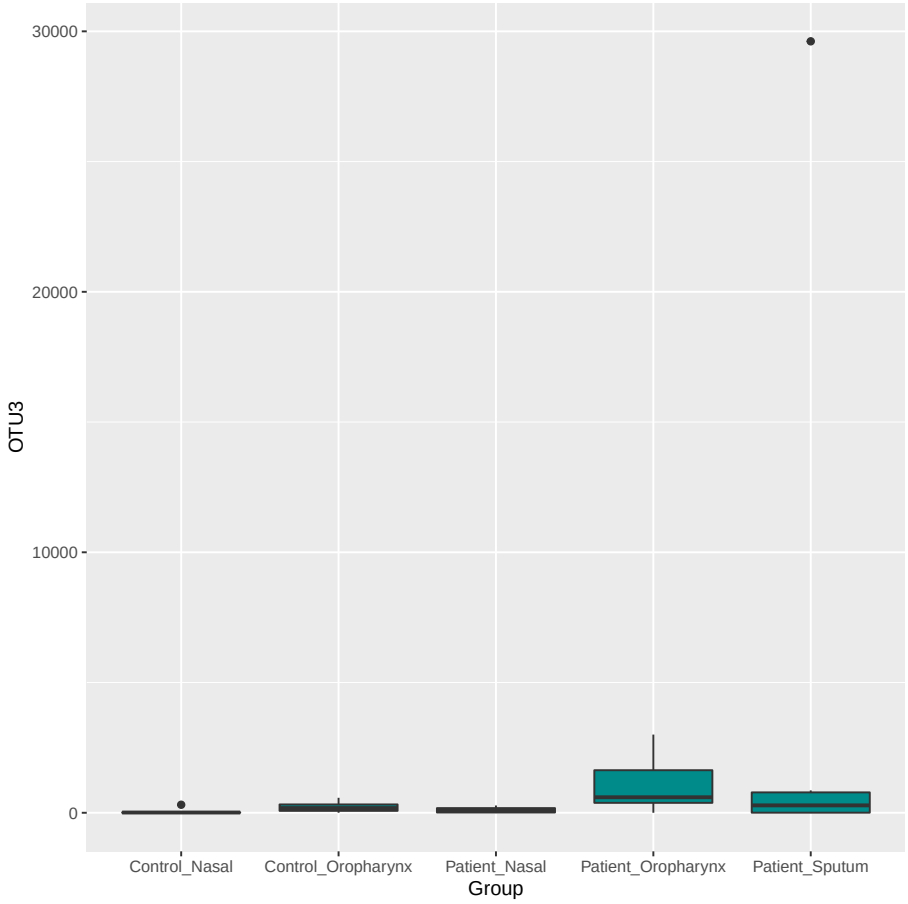
```
   box = read.table("metag_boxplots.txt", header=TRUE)

head(box)

##                Group OTU1  OTU2 OTU3 OTU4 OTU5 OTU6  OTU7 OTU8  OTU9 OTU10
## 1      Control_Nasal    0     0  308    0    0  611     0   11 11413     0
## 2 Control_Oropharynx    0 15500  575    0    0    0 30220   30     1     0
## 3      Control_Nasal    0     0    0    0    0 2328     0   35  1072     0
## 4 Control_Oropharynx    0     0  334    0    0    1     0   13     0     0
## 5      Control_Nasal    0     0   17    0    0 4141     0   44  1708     0
## 6 Control_Oropharynx    0     0   83    0    0    0     0  150     0     0
```

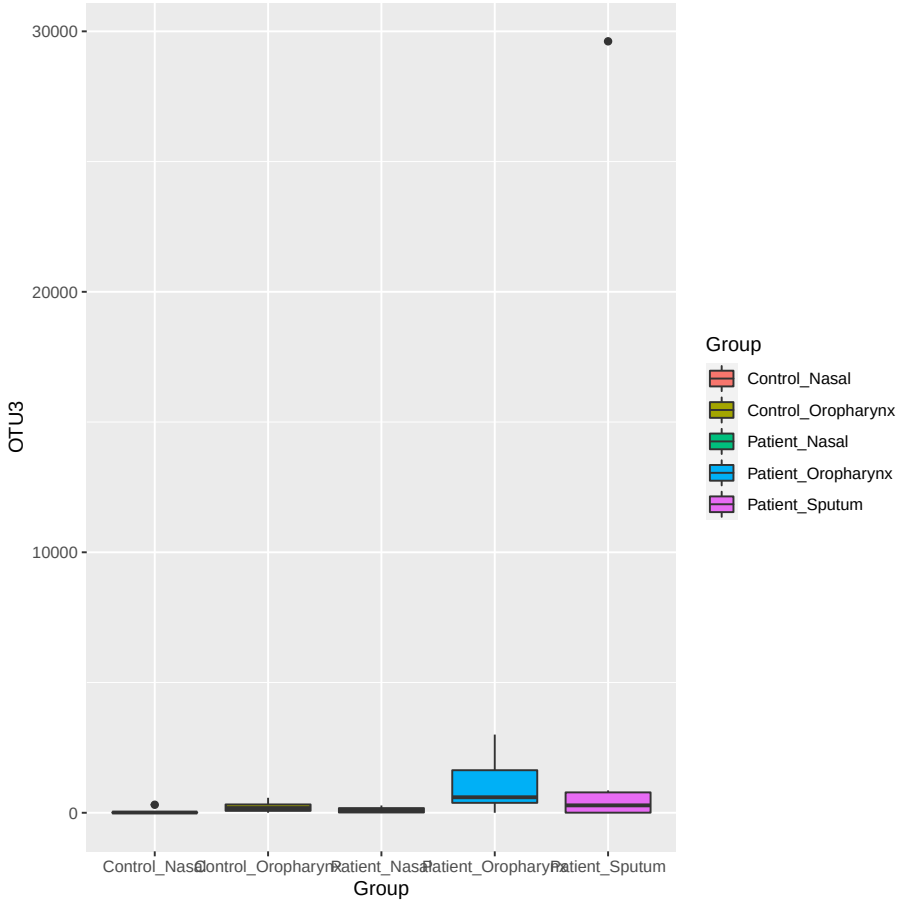
We'll use OTU3. The first part, including the aesthetics takes the same form we saw in the scatterplots. Instead of adding a geom_point() layer, we'll add a geom_boxplot() layer. We'll make a couple of different versions but put them all into a single PDF. Each plot will be on a separate page. 1) Box, 2) Color (same color for both samples), 3) Color by sample.

```
   pdf("boxplot.pdf")

# Basic boxplot

boxplot1 = ggplot(box,
        aes(x=Group,y=OTU3)) +
        geom_boxplot()
```

```r
# 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=Group,y=OTU3)) +
        geom_boxplot(fill="darkcyan")

# Color by the group
boxplot3 = ggplot(box,
        aes(x=Group,y=OTU3,fill=Group)) +
        geom_boxplot()


# print them all
boxplot1
boxplot2
boxplot2

dev.off()

## pdf
##   2
```

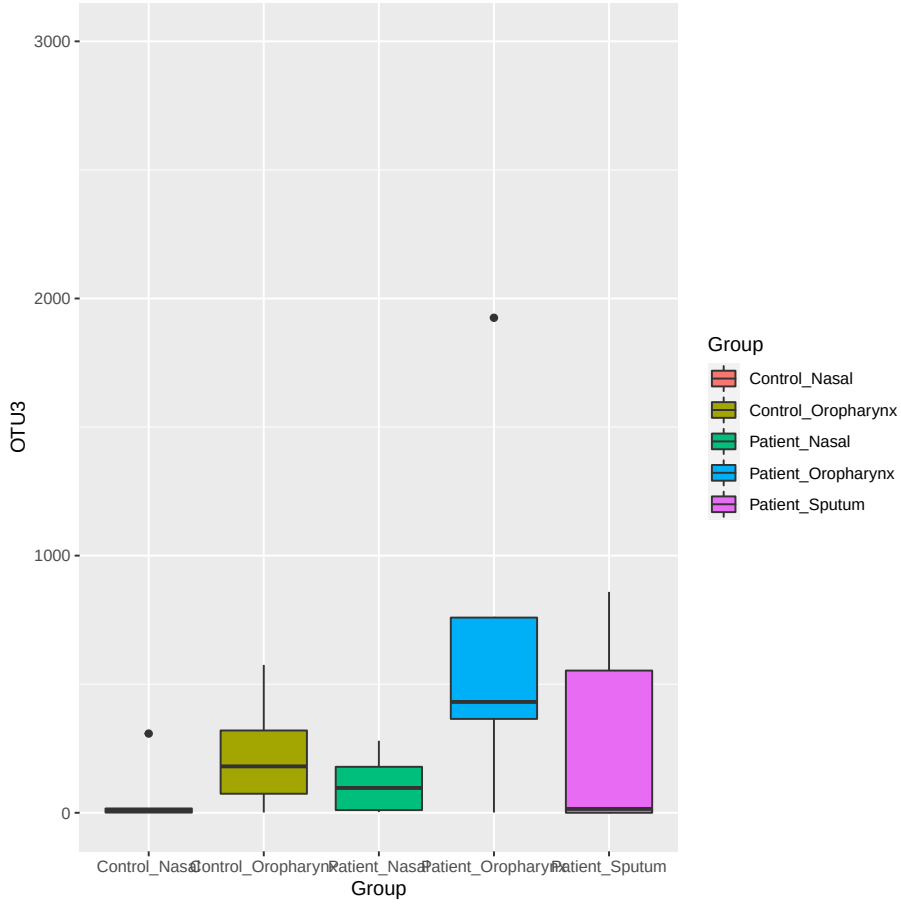

30000
20000
10000
0
OTU3
Control_Nasal
Control_Oropharynx
Patient_Nasal
Patient_Oropharynx
Patient_Sputum
Group


30000
20000
10000
0
OTU3
Control_Nasal
Control_Oropharynx
Patient_Nasal
Patient_Oropharynx
Patient_Sputum
Group



We have a really high outlier that is scrunching everything down at the bottom. See if you can figure out how to rerun it limitting the y axis to 3000. If you published a figure with the top outlier cut off, you should explain what you did and the value of the outlier.

```
## Warning:  Removed 2 rows containing non-finite values (stat_boxplot).
```



Try making boxplots from some of the other genes.

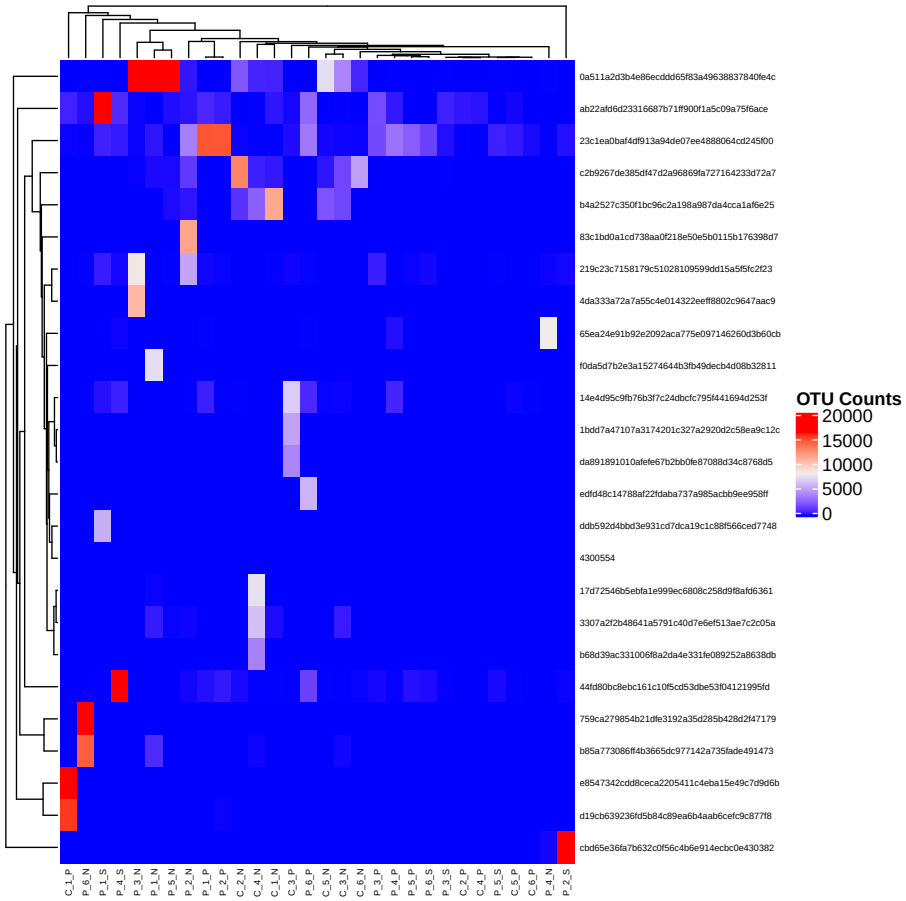
4 Heatmaps

Heatmaps are easy to make in ggplot2 using geom_tile but we often want to add a dendogram, which is a pain to do in ggplot2. So we will use the ComplexHeatmap library. More documentation is here: https://jokergoo.github.io/ComplexHeatmap-reference/book/. We'll make a heatmap with the OTU counts.

```
# Requires a matrix
# In our case we'll use a OTUs x samples matrix

library(ComplexHeatmap)

## Loading required package:  grid
```

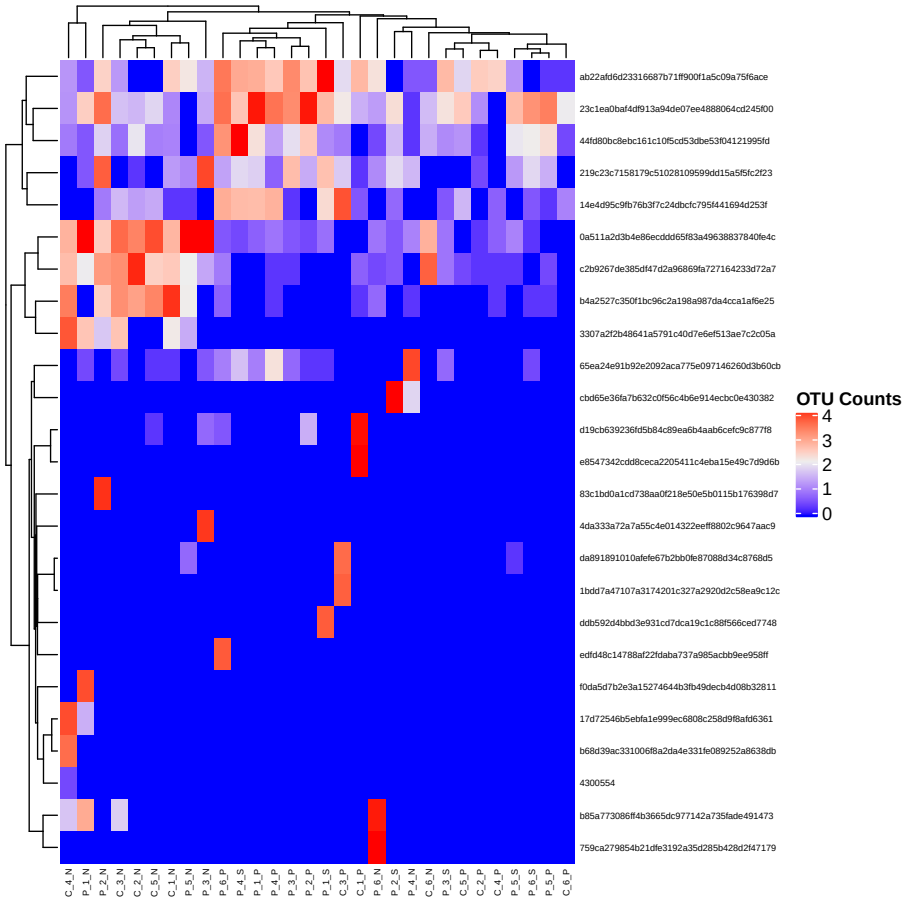
```
## ========================================
## ComplexHeatmap version 2.4.2
## Bioconductor page:  http://bioconductor.org/packages/ComplexHeatmap/
## Github page:  https://github.com/jokergoo/ComplexHeatmap
## Documentation:  http://jokergoo.github.io/ComplexHeatmap-reference
##
## If you use it 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 OTUs the rownames
rownames(mg) = mg$OTU_ID

# Then take out the column with otu IDs
# We'll also just plot the first 25 OTUs
mg25 = mg[1:25,-1]

# Check the data frames
head(mg25)

##                                          C_1_N C_1_P C_2_N C_2_P C_3_N C_3_P
## 4300554                                      0     0     0     0     0     0
## d19cb639236fd5b84c89ea6b4aab6cefc9c877f8     0 15500     0     0     0     0
## ab22afd6d23316687b71ff900f1a5c09a75f6ace   308   575     0   334    17    83
## 759ca279854b21dfe3192a35d285b428d2f47179     0     0     0     0     0     0
## cbd65e36fa7b632c0f56c4b6e914ecbc0e430382     0     0     0     0     0     0
## 0a511a2d3b4e86ecddd65f83a49638837840fe4c   611     0  2328     1  4141     0
##                                          C_4_N C_4_P C_5_N C_5_P C_6_N C_6_P
## 4300554                                      2     0     0     0     0     0
## d19cb639236fd5b84c89ea6b4aab6cefc9c877f8     0     0     1     0     0     0
## ab22afd6d23316687b71ff900f1a5c09a75f6ace    16   278     0    71     3     1
## 759ca279854b21dfe3192a35d285b428d2f47179     0     0     0     0     0     0
## cbd65e36fa7b632c0f56c4b6e914ecbc0e430382     0     0     0     0     0     0
## 0a511a2d3b4e86ecddd65f83a49638837840fe4c   659     4  7448     0   691     0
##                                          P_1_N P_1_P P_1_S P_2_N P_2_P P_2_S
## 4300554                                      0     0     0     0     0     0
## d19cb639236fd5b84c89ea6b4aab6cefc9c877f8     0     0     0     0    29     0
## ab22afd6d23316687b71ff900f1a5c09a75f6ace     3   759 29616   280   431     0
## 759ca279854b21dfe3192a35d285b428d2f47179     0     0     0     0     0     0
## cbd65e36fa7b632c0f56c4b6e914ecbc0e430382     0     0     0     0     0 41007
## 0a511a2d3b4e86ecddd65f83a49638837840fe4c 16961     4     6   340     2     3
##                                          P_3_N P_3_P P_3_S P_4_N P_4_P P_4_S
```

```
## 4300554                                        0     0     0     0     0     0
## d19cb639236fd5b84c89ea6b4aab6cefc9c877f8       5     0     0     0     0     0
## ab22afd6d23316687b71ff900f1a5c09a75f6ace      33  1925   553     3   365   859
## 759ca279854b21dfe3192a35d285b428d2f47179       0     0     0     0     0     0
## cbd65e36fa7b632c0f56c4b6e914ecbc0e430382       0     0     0    71     0     0
## 0a511a2d3b4e86ecddd65f83a49638837840fe4c   19959     3     7    10     7     2
##                                            P_5_N P_5_P P_5_S P_6_N P_6_P P_6_S
## 4300554                                        0     0     0     0     0     0
## d19cb639236fd5b84c89ea6b4aab6cefc9c877f8       0     0     0     0     3     0
## ab22afd6d23316687b71ff900f1a5c09a75f6ace     160     1    15   185  3002     0
## 759ca279854b21dfe3192a35d285b428d2f47179       0     0     0 28159     0     0
## cbd65e36fa7b632c0f56c4b6e914ecbc0e430382       0     0     0     0     0     0
## 0a511a2d3b4e86ecddd65f83a49638837840fe4c   19005     0    10     7     3     1

# Change the data frame into a matrix
mg25matrix = as.matrix(mg25)

pdf("4_heatmap.pdf",width=5,height=80)

Heatmap(mg25matrix, name = "OTU Counts", row_names_gp = gpar(fontsize = 5),
        column_names_gp = gpar(fontsize = 5) )

dev.off()

## pdf
##   2
```



There are a few OTUs with really high counts that are extending the range and making most of the OTU counts bunch up on the low end of the range. Let's try doing a log of the counts.

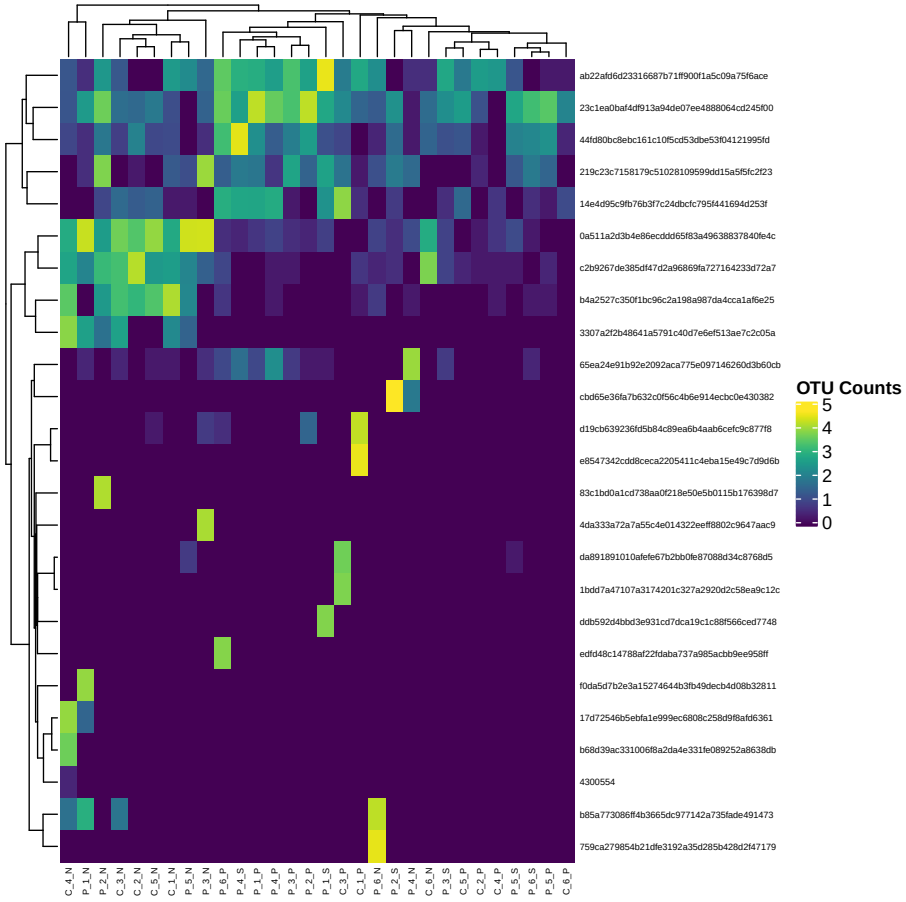
```
mg25log = log10(mg25+1)
mg25logmatrix = as.matrix(mg25log)

pdf("heatmap_log.pdf",width=5,height=80)

Heatmap(mg25logmatrix, name = "OTU Counts", row_names_gp = gpar(fontsize = 5),
        column_names_gp = gpar(fontsize = 5) )

dev.off()

## pdf
##   2
```



Let's try another color scheme, viridis (purple to yellow). According to the developers (https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html), it was developed to be:

- Colorful, spanning as wide a palette as possible so as to make differences easy to see
- Perceptually uniform, meaning that values close to each other have similar-appearing colors and values far away from each other have more different-appearing colors, consistently across the range of values
- Robust to colorblindness, so that the above properties hold true for people with common forms of colorblindness, as well as in grey scale printing, and
- Pretty, oh so pretty

```
library(viridis)

## Loading required package:  viridisLite

pdf("heatmap.viridis.pdf")

Heatmap(mg25logmatrix, name = "OTU Counts",
        col=viridis(256),
        row_names_gp = gpar(fontsize = 5),
        column_names_gp = gpar(fontsize = 5) )

dev.off()

## pdf
##   2
```



Now try to make a heatmap with the cividis spectrum. This is a color scheme based on viridis that is even better for those with color vision deficiencies. It is also in the viridis library so you just need to change the col parameter.

5 Homework

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