DESEQ

James Adler

9/7/2021

1. Loading Data into DESEQ2

```
# read into dataframes
countdata = read.csv("Gacu_gut_counts.tsv",
                     sep="\t",
                     row.names=1,
                     stringsAsFactors=TRUE
)

coldata = read.csv("Gacu_gut_metadata.tsv",
                   sep="\t",
                   row.names=1,
                   stringsAsFactors=TRUE
)
```

```
# deseq matrix
ddsMat = DESeqDataSetFromMatrix(
  countData = countdata,
  colData = coldata,
  design = ~ Population + Sex + Treatment,
)

# deseq2 object
dds = DESeq(ddsMat)
```

Desribe DESeqDataSetFromMatrix() Output

The ddsMat object is made up of seven elements:

1. **design:** the design of the experiment that we passed in above.
2. **dispersionFunction:** this element contains an environment element that contains all of the data passed in above. And additionally contains elements such as newdata, stats_df, and stats_res.
3. **rowRanges:** this is an S4 object of class compressedGRangesList. There are many objects that exist within this rowRanges object.
4. **colData:** this is s4 object of class DataFrame. This object contains the metadata information that we passed into the function.
5. **assays:** this is an S4 object of class SimpleASsays. There is just one subobject that states element type is 'ANY'.

6. **elementMetadata:** this contains metadata about the ddsMat object.
7. **metadata:** this contains information on the version of the package used to create the ddsMat object.

Describe the DESeq() output

The output of dds object appears to be about nearly identical to the ddsMat object, though now contains additional elements within the 'dispersionFunction'.

1. **formats:** contains one element 'q' that appears to be missing.
2. **body:** contains elements of type symbol, language, and double, where the values are symbols, letters, numbers, and coefs.

2. Pre-filter the Dataset

```
# print rows pre-filter
nrow(dds)

# keep only counts greater than 1
keep = rowSums(counts(dds)) > 1
dds = dds[keep,]

# print rows post-filter
nrow(dds)
```

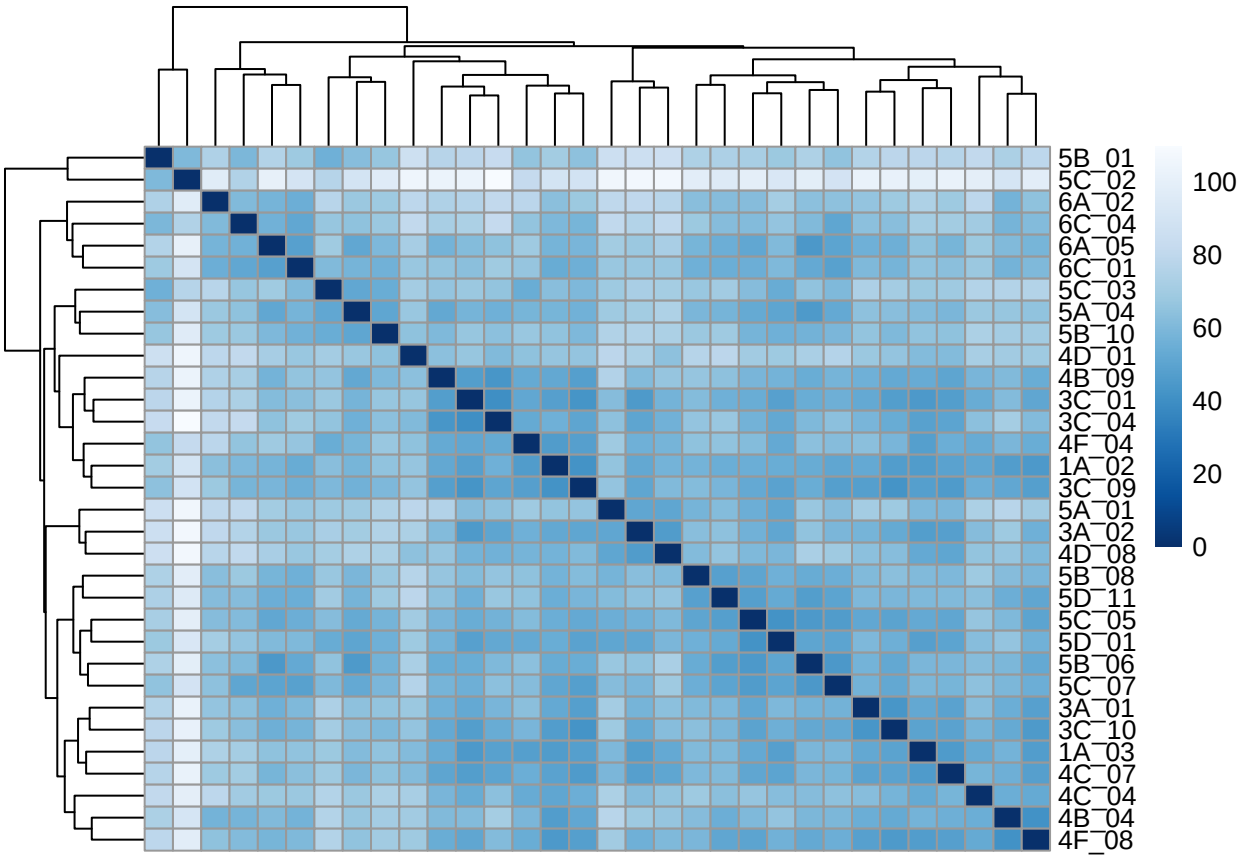
How many genes were removed? 22456 - 20797 = **1659** genes were removed

3. rlog Transformation

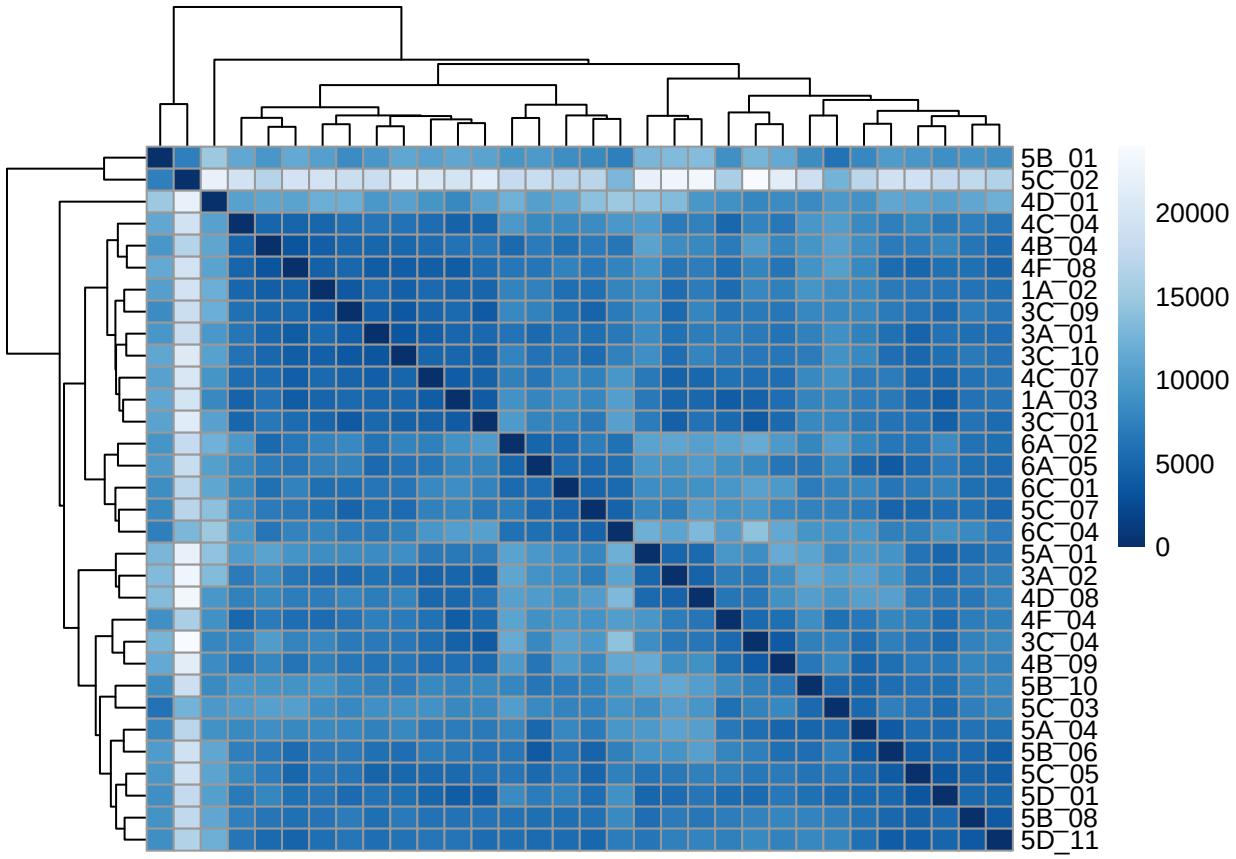
Why did we choose the rlog over the VST transform? The rlog works well on small datasets when there is a wide range of sequenceing depth across the samples.

4. Generate Sample Distance Heatmaps with Dendrograms

```
# calculate euclidean distances between samples
sampleDists = dist(t(assay(rld))) # transpose so genes are columns

# generate matrix and heatmap
sampleDistMatrix = as.matrix(sampleDists)
rownames(sampleDistMatrix) = colnames(rld)
colnames(sampleDistMatrix) = NULL
colors = colorRampPalette(rev(brewer.pal(9,"Blues")))(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows=sampleDists,
         clustering_distance_cols=sampleDists,
         col=colors
)
```




```
# calculate poisson distance
poisd = PoissonDistance(t(counts(dds))) # must transpose so genes are columns

# generate matrix and heatmap
samplePoisDistMatrix = as.matrix(poisd$dd)
rownames(samplePoisDistMatrix) = colnames(rld)
colnames(samplePoisDistMatrix) = NULL
pheatmap(samplePoisDistMatrix,
         clustering_distance_rows=poisd$dd,
         clustering_distance_cols=poisd$dd,
         col=colors
)
```



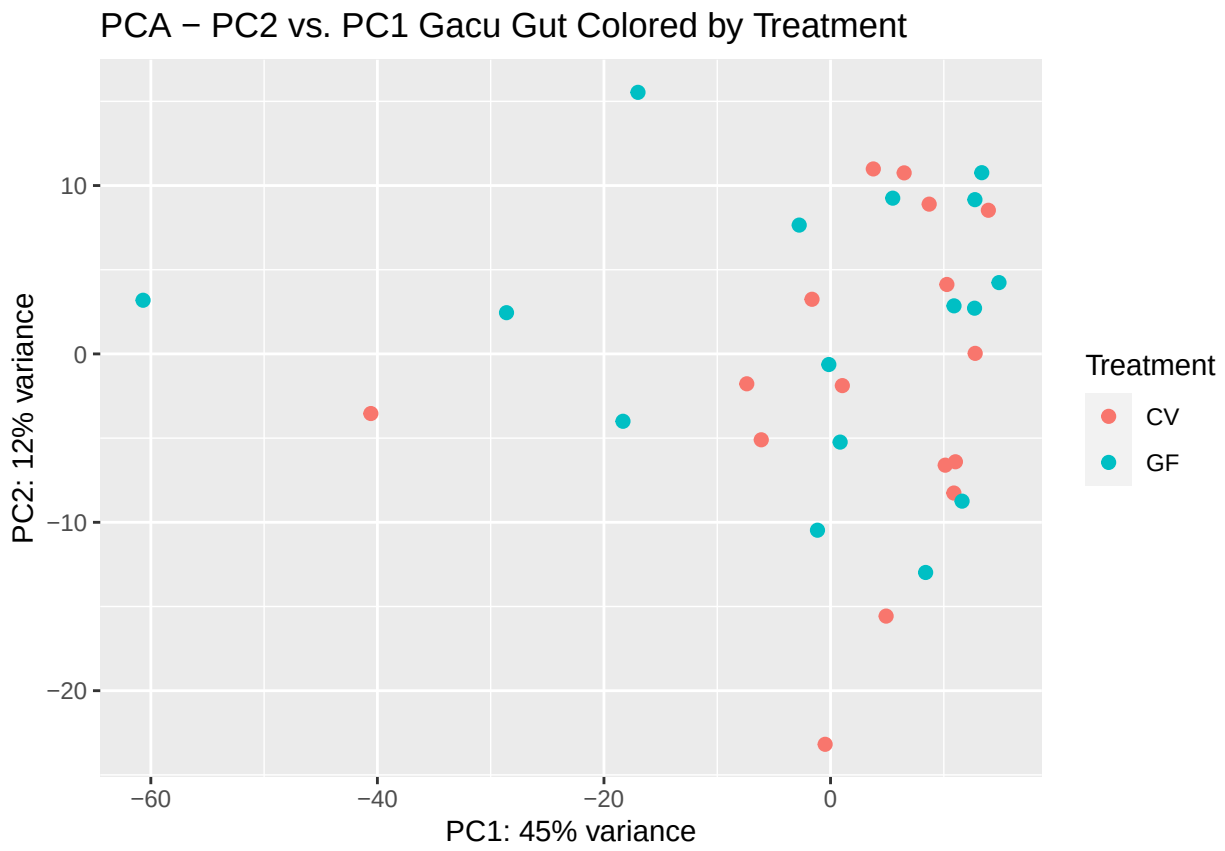
Do there appear to be outliers? Yes, the 5C_02 sample in both graphs appears to be an outlier relative to the other samples.

Do the plots agree? No, the two plots do not agree. There are differences in the structure of the dendrogram and clustering of each of the samples. Euclidean is measuring the similarity, while poisson measuring dissimilarity.

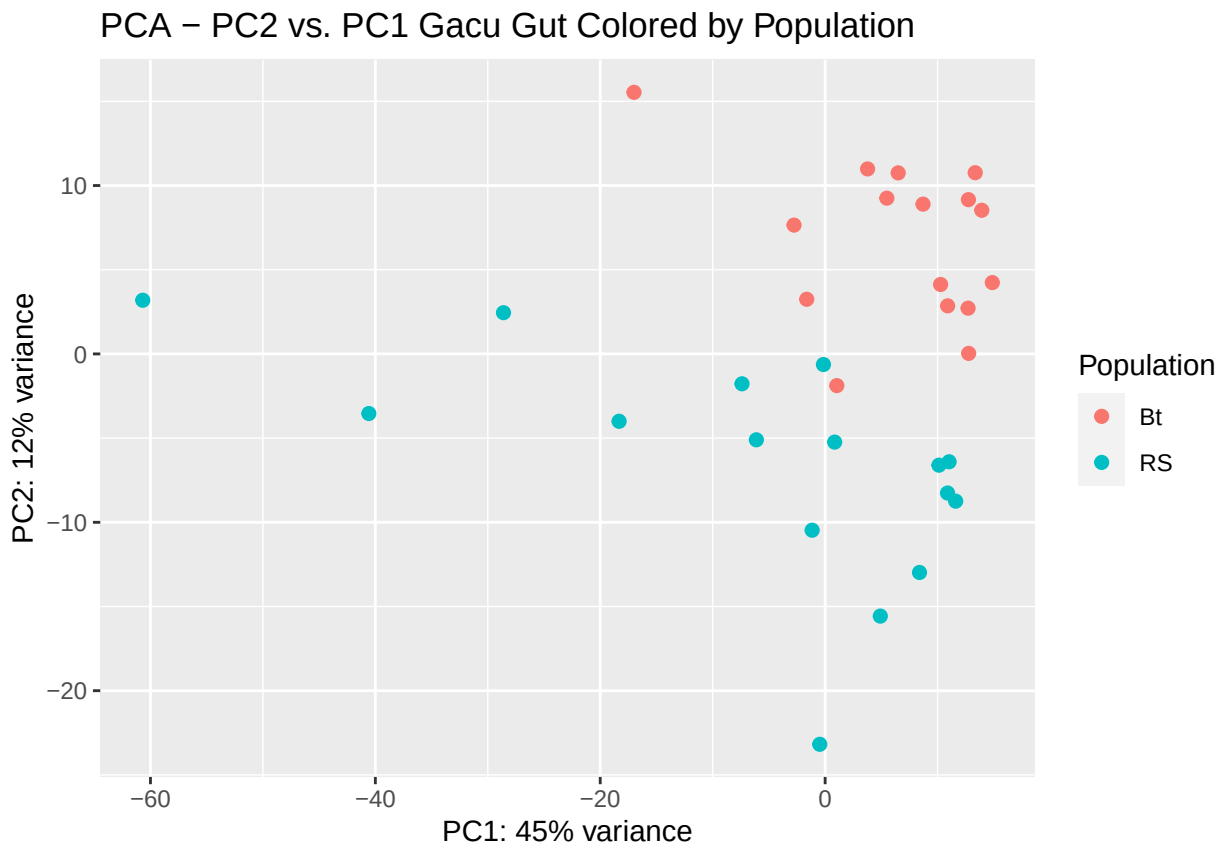
5. Generate PCA, Generalized PCA, and MDS Plots

PCA

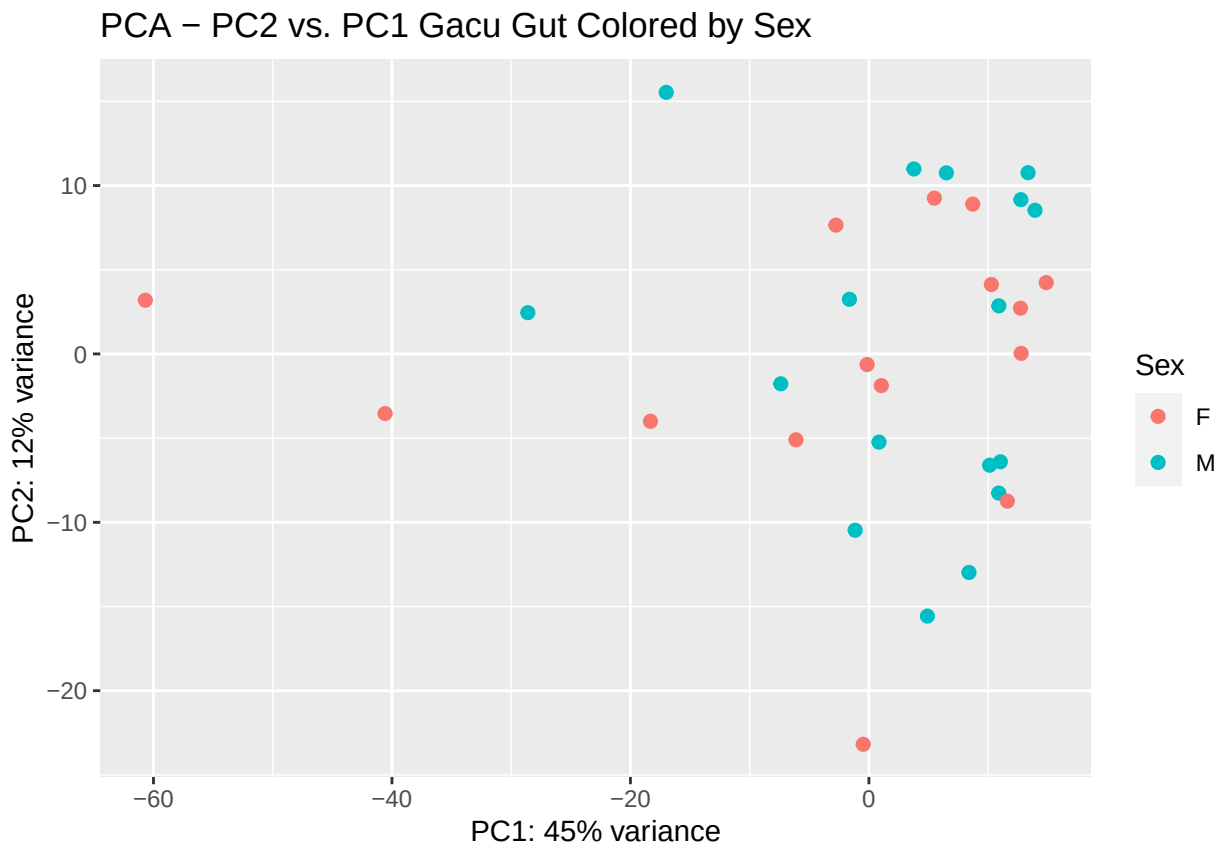
```
# perform pca
colData(dds)
pcaData = plotPCA(rld,intgroup=c("Population","Sex","Treatment"),returnData=TRUE)
pcaData

percentVar = round(100 * attr(pcaData, "percentVar"))

# generate plot
ggplot(pcaData, aes(x=PC1,y=PC2,color=Treatment)) +
  geom_point(size=2) +
  xlab(paste0("PC1: ", percentVar[1], "% variance")) +
  ylab(paste0("PC2: ", percentVar[2], "% variance")) +
  ggtitle("PCA - PC2 vs. PC1 Gacu Gut Colored by Treatment")
```




```
# generate plot
ggplot(pcaData, aes(x=PC1,y=PC2,color=Population)) +
  geom_point(size=2) +
  xlab(paste0("PC1: ", percentVar[1], "% variance")) +
  ylab(paste0("PC2: ", percentVar[2], "% variance")) +
  ggtitle("PCA - PC2 vs. PC1 Gacu Gut Colored by Population")
```

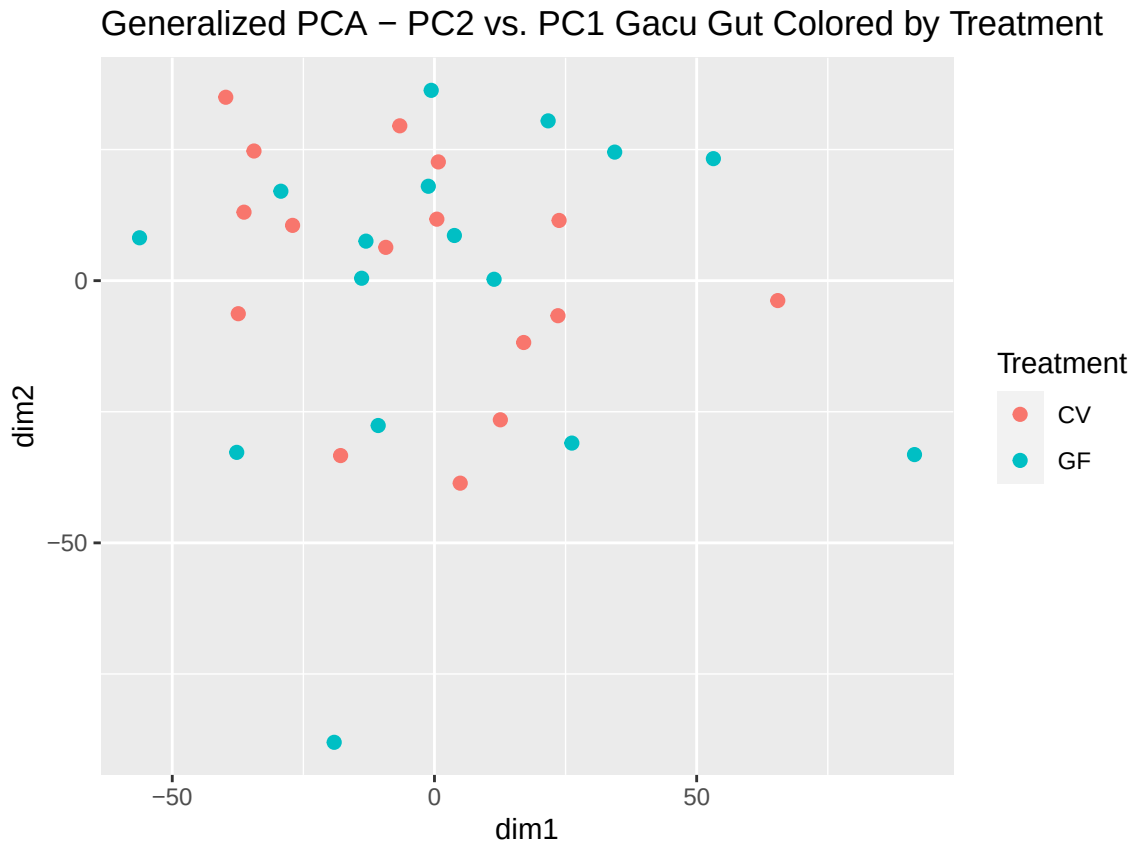



```
# generate plot
ggplot(pcaData, aes(x=PC1,y=PC2,color=Sex)) +
  geom_point(size=2) +
  xlab(paste0("PC1: ", percentVar[1], "% variance")) +
  ylab(paste0("PC2: ", percentVar[2], "% variance")) +
  ggtitle("PCA - PC2 vs. PC1 Gacu Gut Colored by Sex")
```

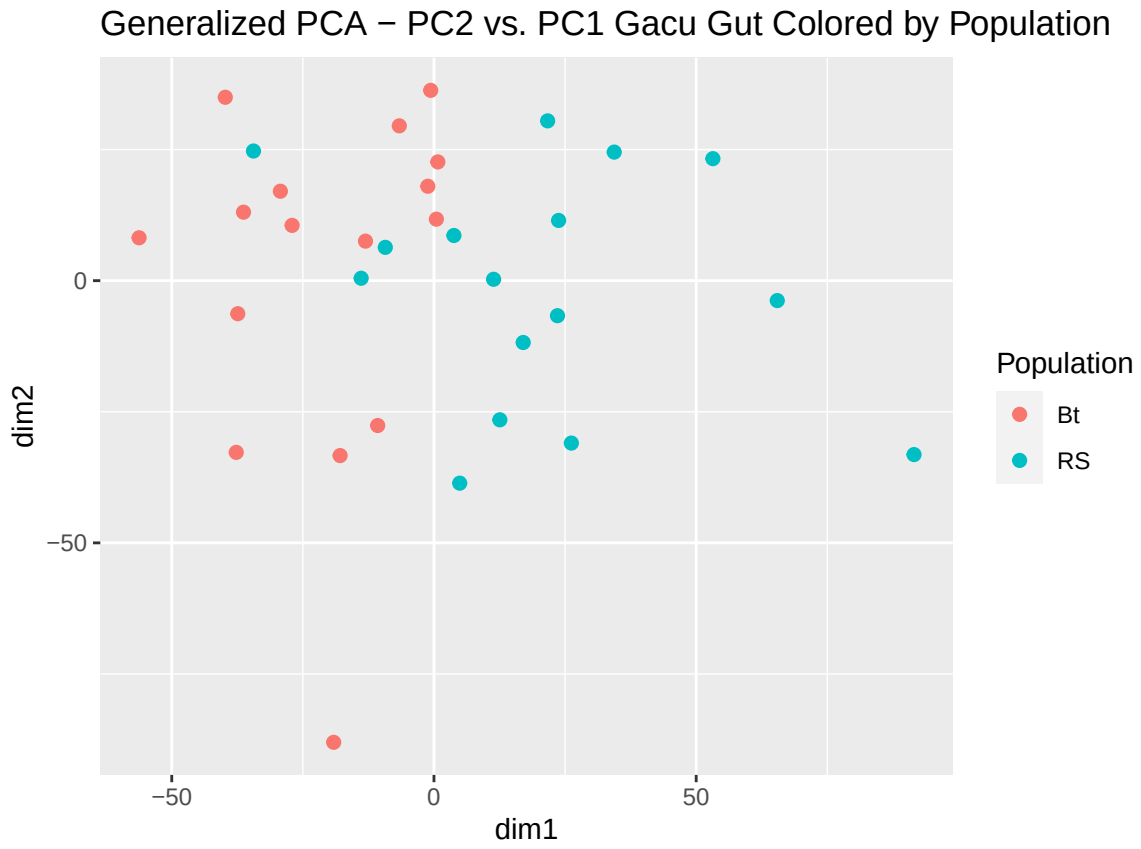


Generalized PCA

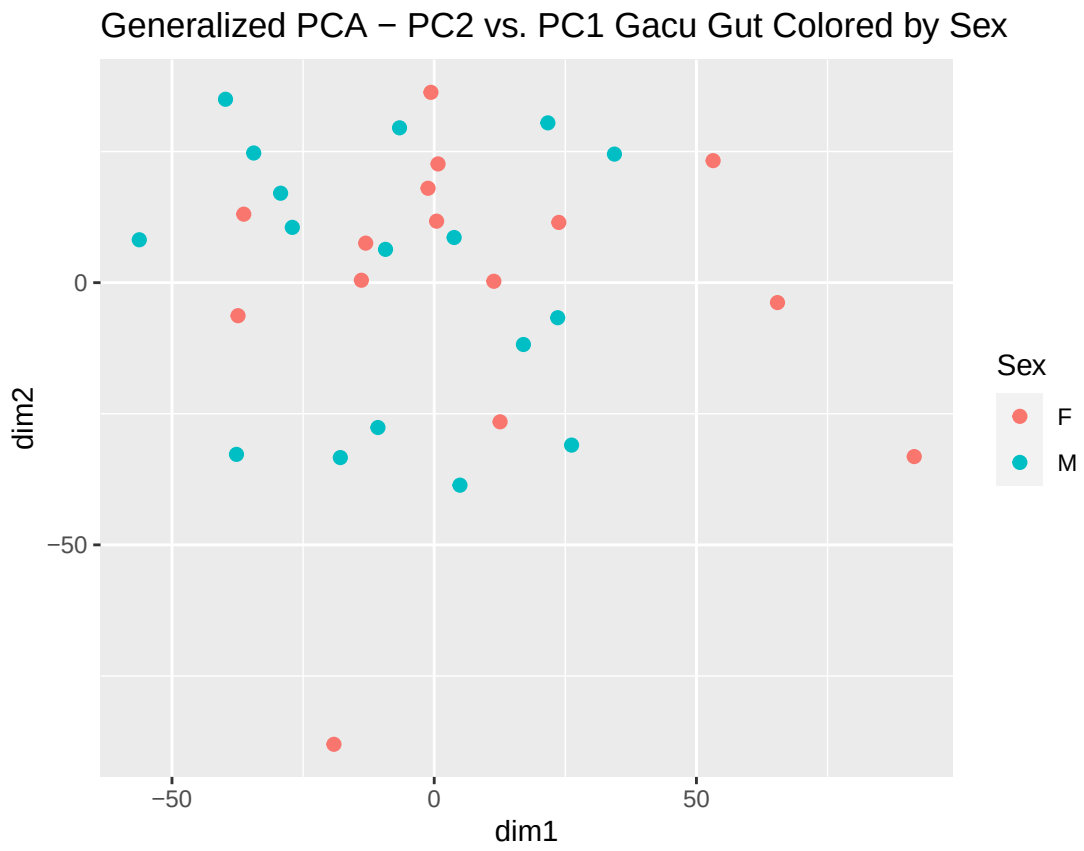
```
# perfom generalized pca
gpca = glmpca(counts(dds), L=2)
gpca.dat = gpca$factors
gpca.dat$Treatment = dds$Treatment
gpca.dat$Population = dds$Population
gpca.dat$Sex = dds$Sex

# plot pca
ggplot(gpca.dat, aes(x=dim1, y=dim2, color=Treatment)) +
  geom_point(size=2) +
  ggtitle("Generalized PCA - PC2 vs. PC1 Gacu Gut Colored by Treatment") +
  coord_fixed()
```




```
# plot pca
ggplot(gpca.dat, aes(x=dim1, y=dim2, color=Population)) +
  geom_point(size=2) +
  ggtitle("Generalized PCA - PC2 vs. PC1 Gacu Gut Colored by Population") +
  coord_fixed()
```

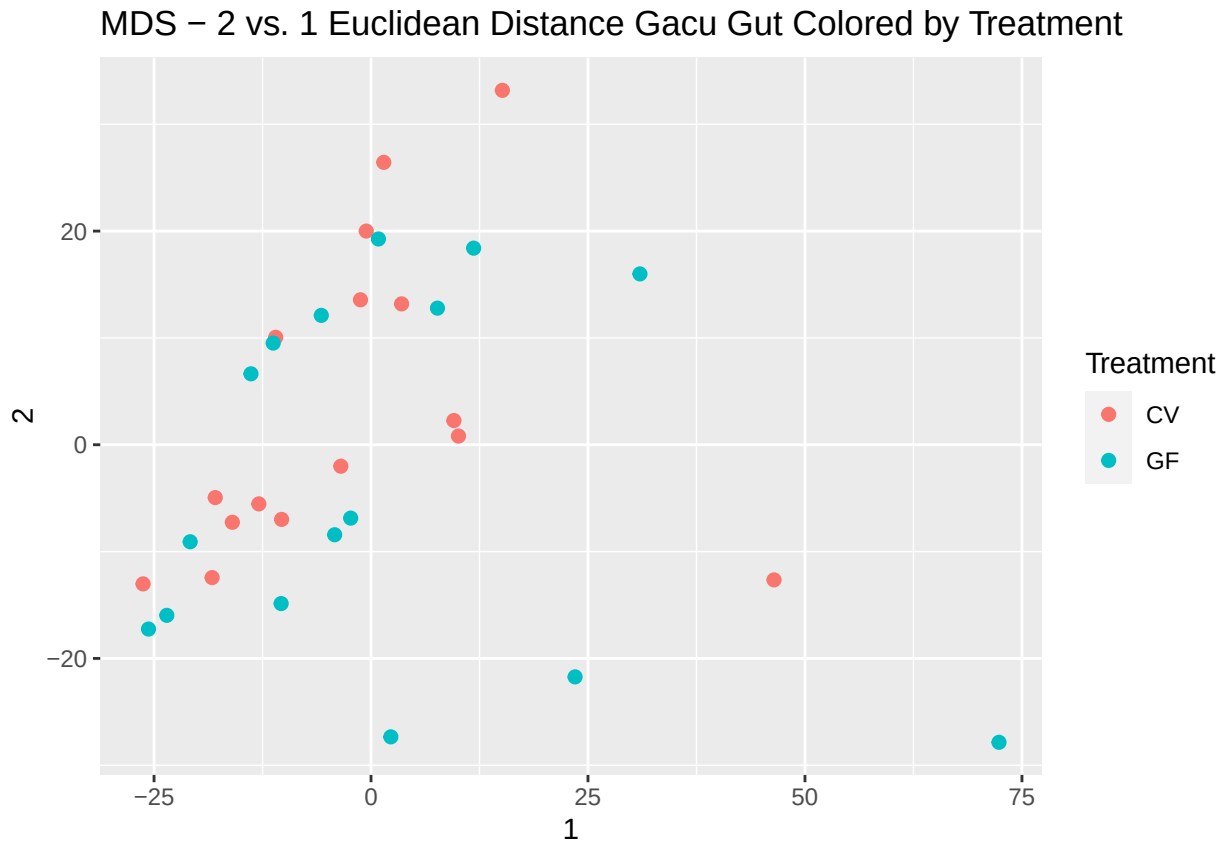



```
# plot pca
ggplot(gpca.dat, aes(x=dim1, y=dim2, color=Sex)) +
  geom_point(size=2) +
  ggtitle("Generalized PCA - PC2 vs. PC1 Gacu Gut Colored by Sex") +
  coord_fixed()
```

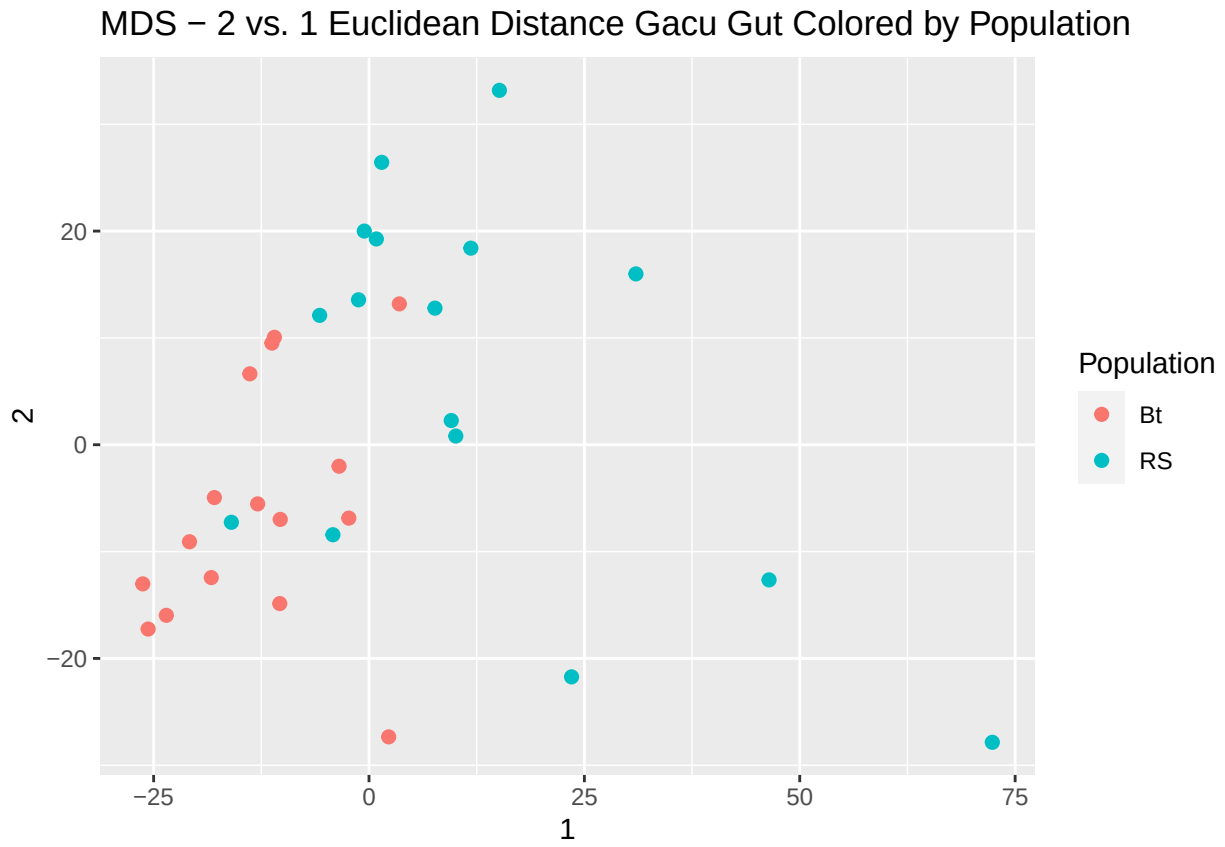


MDS Plot - Multidimensional Scaling

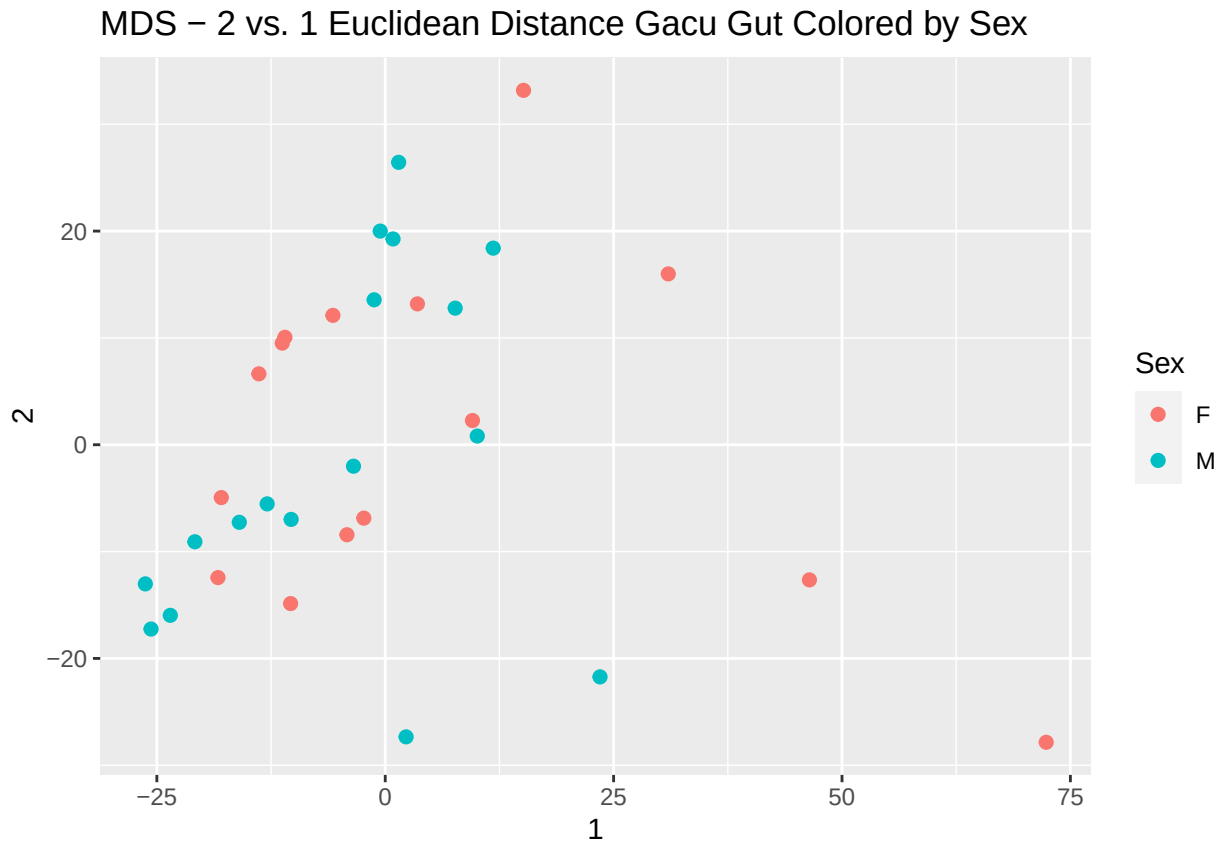
```
# calculate mds
mds = as.data.frame(colData(rld)) %>%
  cbind(cmdscale(sampleDistMatrix))

# plot euclidean
ggplot(mds, aes(x=`1`,y=`2`,color=Treatment)) +
  geom_point(size=2) +
  ggtitle("MDS - 2 vs. 1 Euclidean Distance Gacu Gut Colored by Treatment")
```



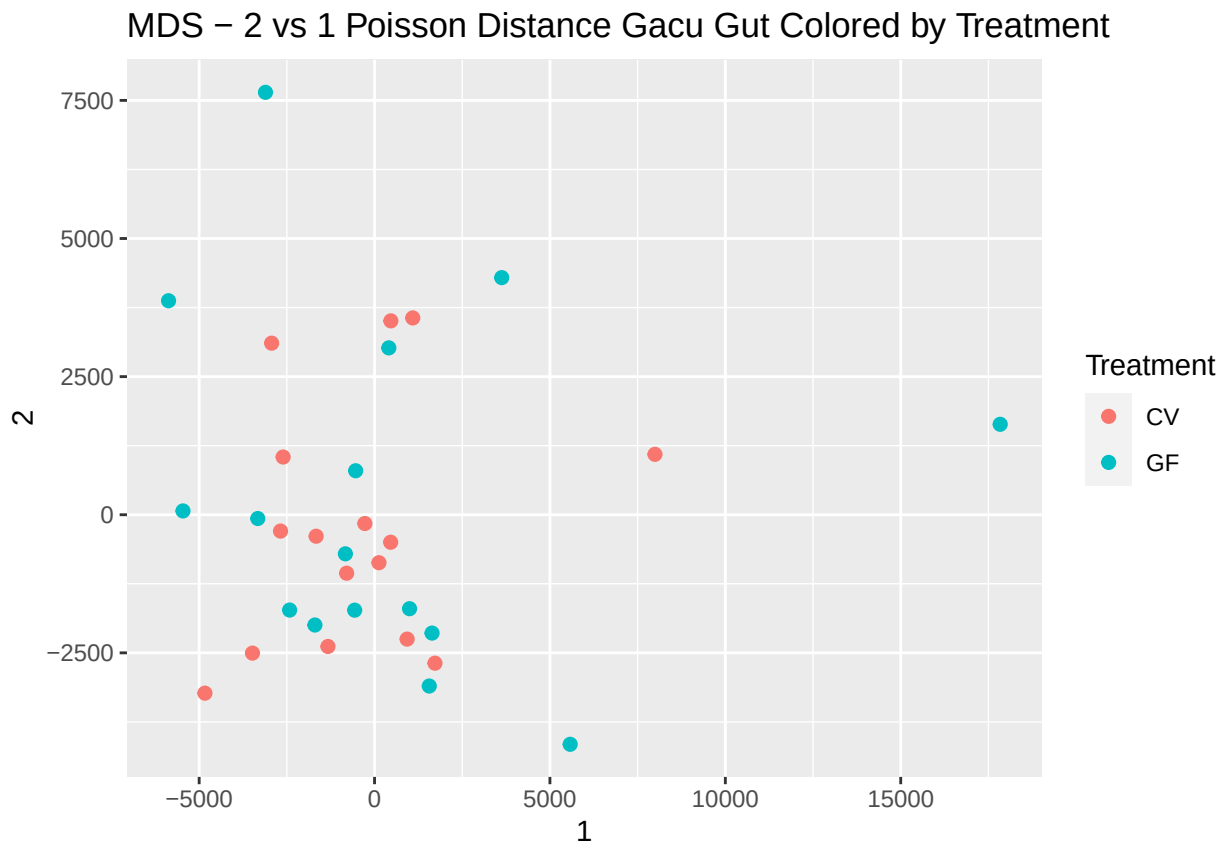

```
# plot euclidean
ggplot(mds, aes(x=`1`,y=`2`,color=Population)) +
  geom_point(size=2) +
  ggtitle("MDS - 2 vs. 1 Euclidean Distance Gacu Gut Colored by Population")
```



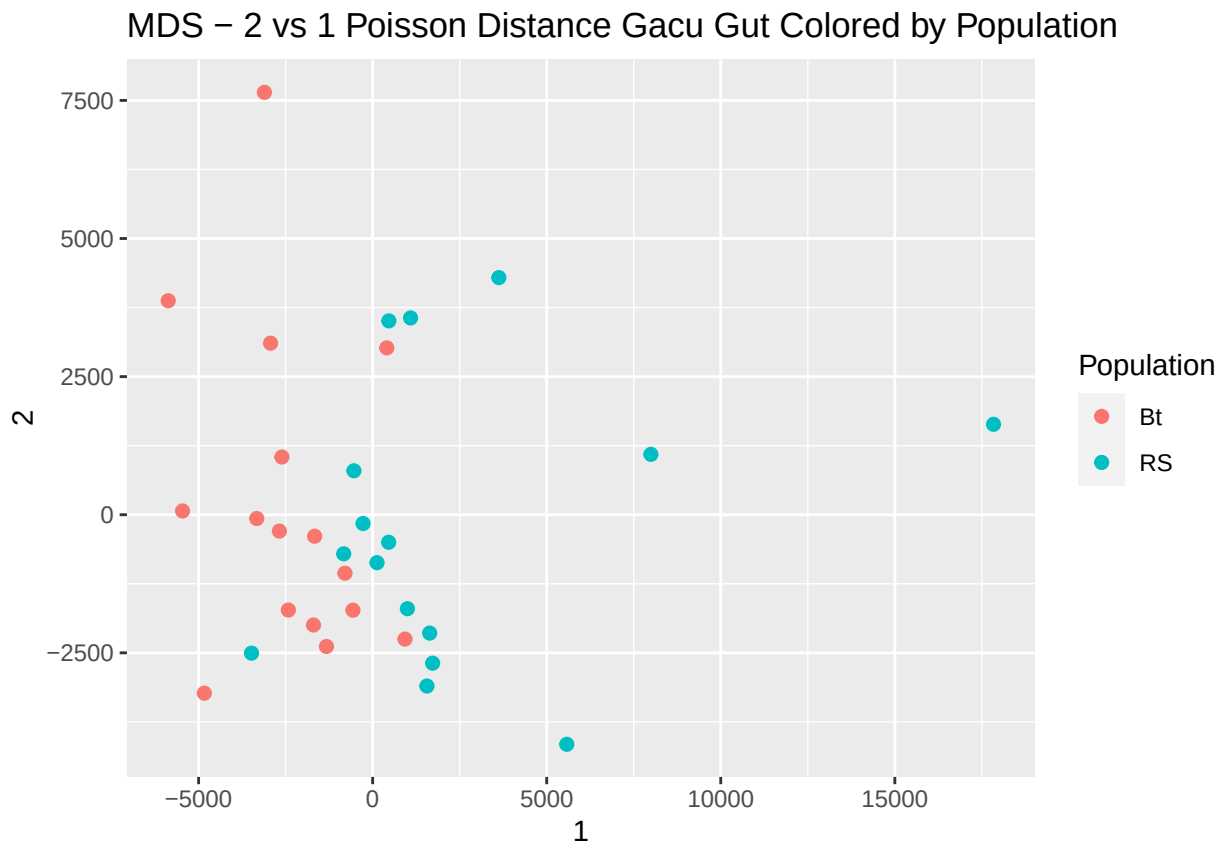

```
# plot euclidean
ggplot(mds, aes(x=`1`,y=`2`,color=Sex)) +
  geom_point(size=2) +
  ggtitle("MDS - 2 vs. 1 Euclidean Distance Gacu Gut Colored by Sex")
```



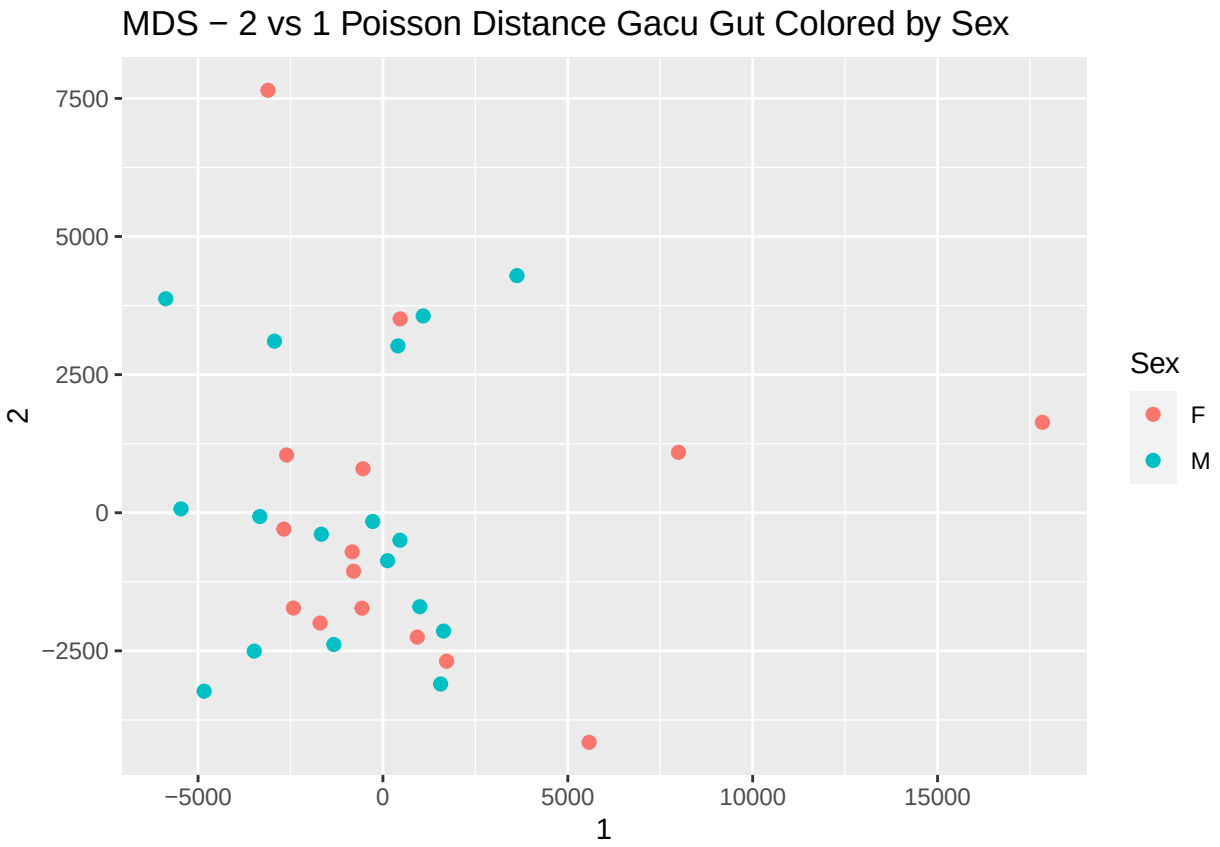

```
# calculate mds poisson
mdsPois = as.data.frame(colData(dds)) %>%
  cbind(cmdscale(samplePoisDistMatrix))

# plot poisson
ggplot(mdsPois, aes(x=`1`,y=`2`,color=Treatment)) +
  geom_point(size=2) +
  ggtitle("MDS - 2 vs 1 Poisson Distance Gacu Gut Colored by Treatment")
```




```
# plot poisson
ggplot(mdsPois, aes(x=`1`,y=`2`,color=Population)) +
  geom_point(size=2) +
  ggtitle("MDS - 2 vs 1 Poisson Distance Gacu Gut Colored by Population")
```




```
# plot poisson
ggplot(mdsPois, aes(x=`1`,y=`2`,color=Sex)) +
  geom_point(size=2) +
  ggtitle("MDS - 2 vs 1 Poisson Distance Gacu Gut Colored by Sex")
```



Describe PCA: PCA flattens the data into 2 dimensions. The first dimension, plotted along the x-axis, is the direction that describes the greatest difference in the data. The second dimension, plotted along the y-axis, is the direction that separates the data second most (must be orthoganol to the first dimension).

Describe Generalized PCA Plot: generalized PCA is best for situations where the data is not normally distributed, such as when data are exponentially distributed. This type of PCA can be beneficial for count data.

Describe MDS - Euclidean: MDS stands for multidimensional sampling. These plots are most useful when we only have a matrix of distances and not a matrix of data. In this version of the MDS plot, we utilized the euclidian distances to generate our plot

Describe MDS - Poisson: MDS stands for multidimensional sampling. These plots are most useful when we only have a matrix of distances and not a matrix of data. In this version of the MDS plot, we utilized the poisson distances to generate our plot

Which of these dimensional reduction methods seem least appropriate? PCA seems least appropriate in this situation, because our data are not normally distributed. Generalized PCA is specifically intended for data that is not not normally distributed.

6. Generate a Results Table

```
# generate results table
res = results(dds)
res
```

```
# print summary
summary(res)
```

Describe Each Column of the Results Table:

1. **baseMean:** average of the normalized count values, divided by the size factors, taken over all samples in the DESeqDataSet.
2. **log2FoldChange:** this value reports the change in the effect variable as a result of the changes in the independent variables.
3. **lfcSE:** standard error for the log2FoldChange.
4. **stat:** the Wald statistic.
5. **pvalue:** the probability that a log fold chance equal to or greater than the reported value would be observed under conditions described by the null hypothesis.
6. **padj:** BH adjusted pvalue. This is a method to account for false discovery rate.

```
# calculate number of genes with pvalue less than 0.1
sum(res$pvalue < 0.1, na.rm=TRUE)

# calculate number of genes with a padj less than 0.1
sum(res$padj < 0.1, na.rm=TRUE)
```

1892 genes with p value less than 0.1

74 genes with padj less than 0.1

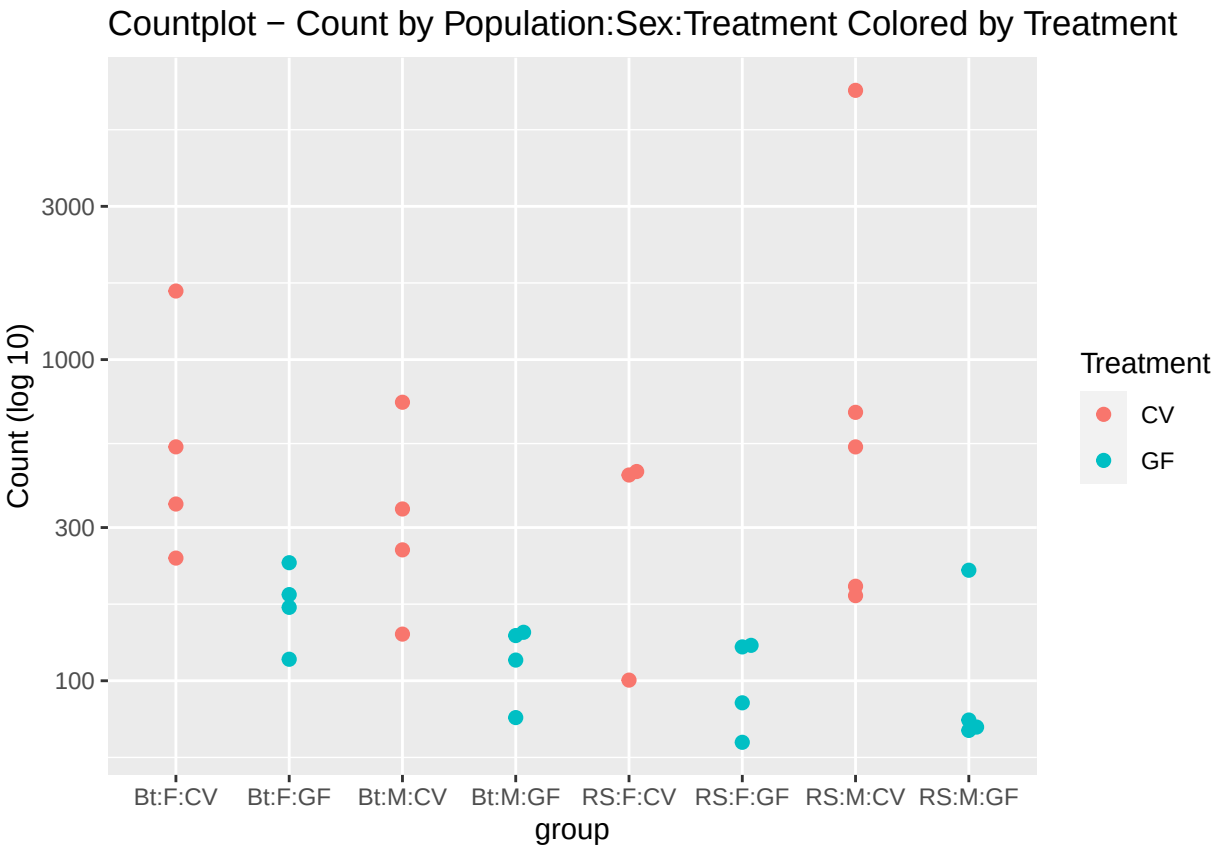
Generate Subset of Genes with padj < 0.1:

```
# generate subset
padj_mask = which(res$padj < 0.1)
res_subset = res[padj_mask,] # may need to do this a different way actually

# print to console
print(res_subset)
```

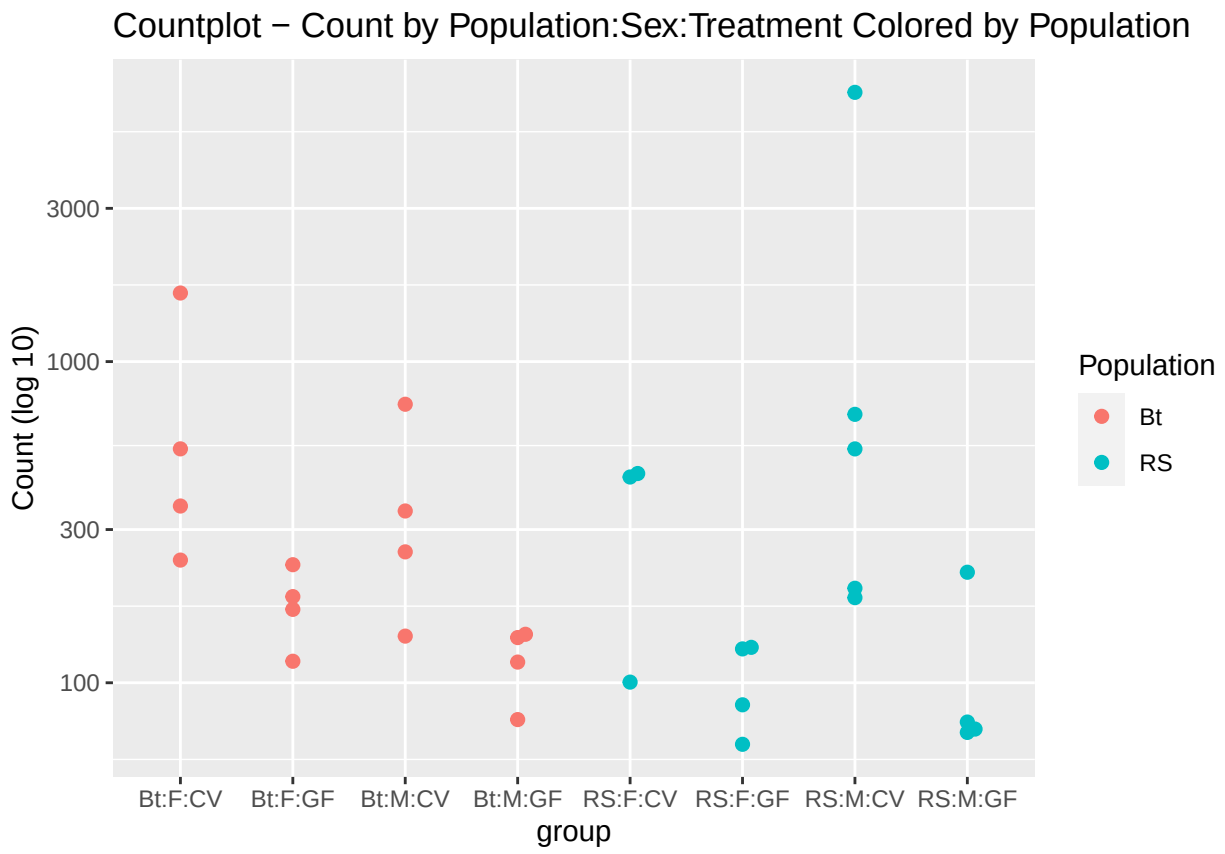
7. Generate Plots

Beeswarm Plot

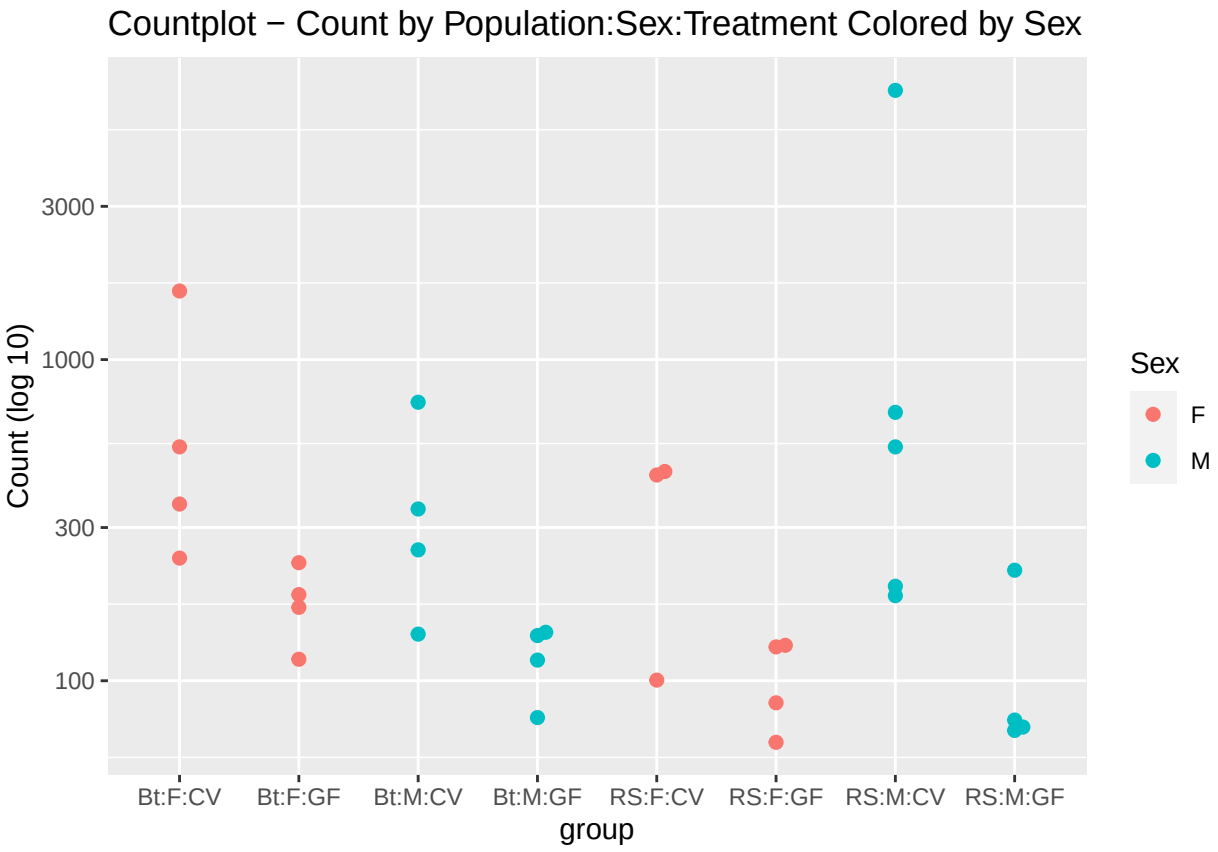
```
# determine gene with lowest padjust value
topGene = rownames(res)[which.min(res$padj)]

# plot
count_plot = plotCounts(dds, gene=topGene, intgroup=c("Population","Sex","Treatment"), returnData=TRUE)
count_plot$group = pcaData$group

ggplot(count_plot, aes(x=group, y=count, color=Treatment)) +
  geom_beeswarm(size=2) +
  scale_y_log10() +
  ggtitle("Countplot - Count by Population:Sex:Treatment Colored by Treatment") +
  ylab("Count (log 10)")
```




```
ggplot(count_plot, aes(x=group, y=count, color=Population)) +
  geom_beeswarm(size=2) +
  scale_y_log10() +
  ggtitle("Countplot - Count by Population:Sex:Treatment Colored by Population") +
  ylab("Count (log 10)")
```




```
ggplot(count_plot, aes(x=group, y=count, color=Sex)) +
  geom_beeswarm(size=2) +
  scale_y_log10() +
  ggtitle("Countplot - Count by Population:Sex:Treatment Colored by Sex") +
  ylab("Count (log 10)")
```

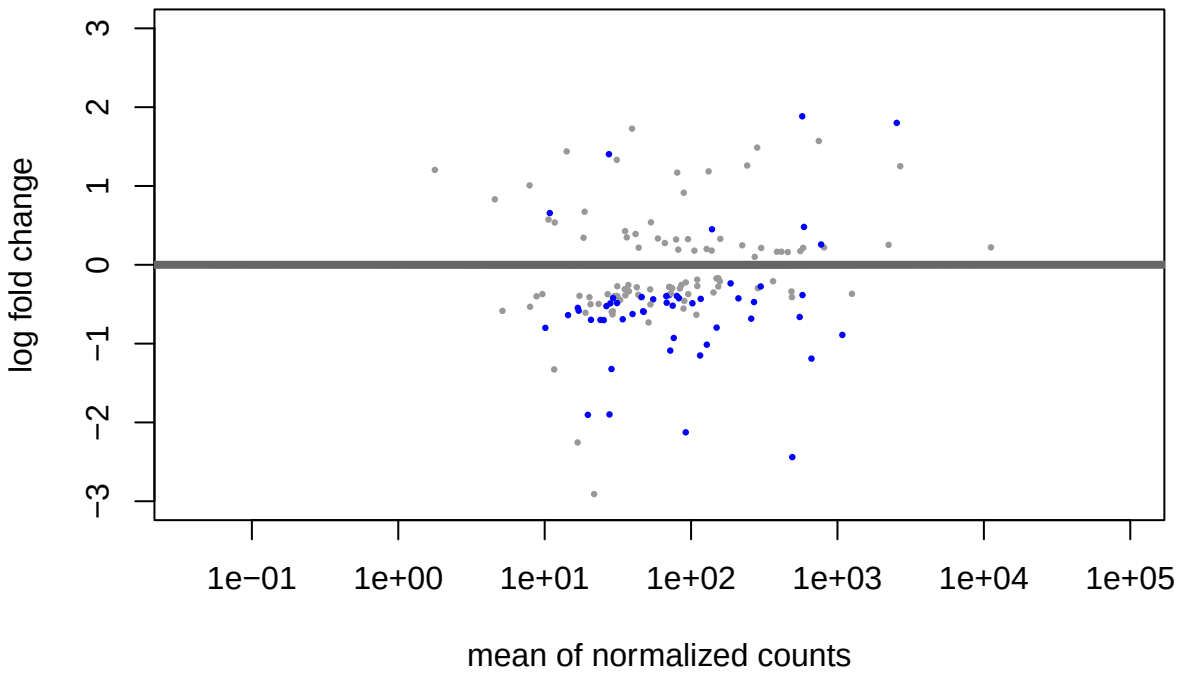


MA Plot with shrink

```
resultsNames(dds)

res_ma = lfcShrink(dds, coef="Treatment_GF_vs_CV", type="apeglm")
```

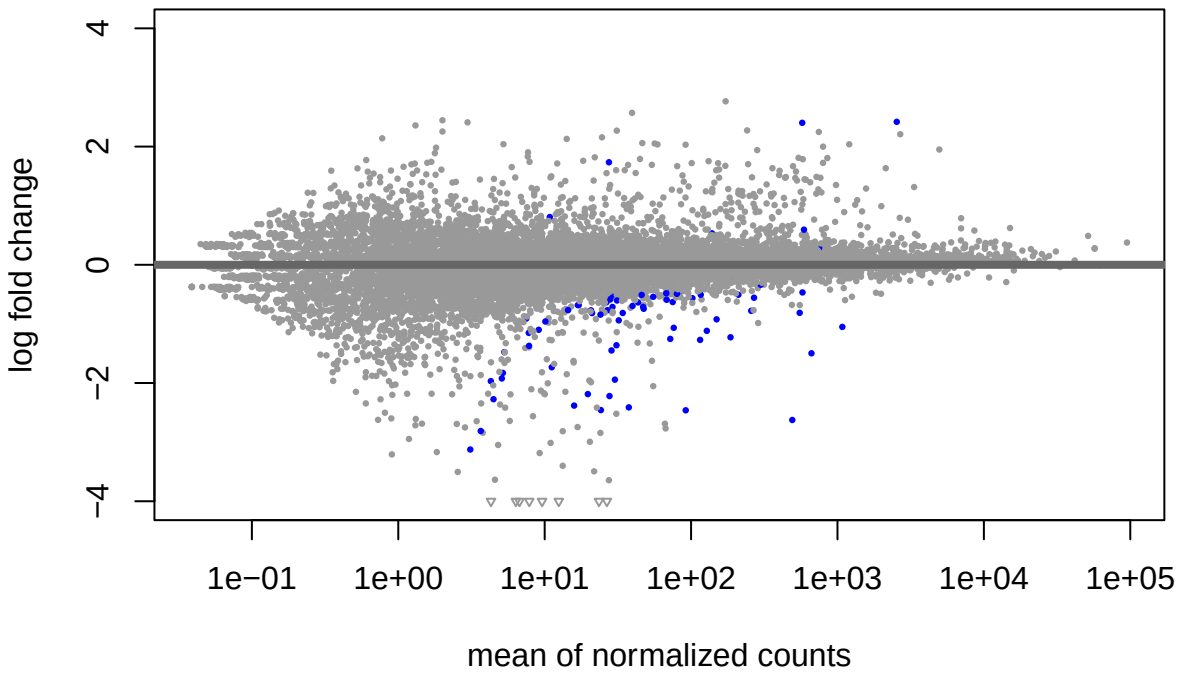
```
## using 'apeglm' for LFC shrinkage. If used in published research, please cite:
##     Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
##     sequence count data: removing the noise and preserving large differences.
##     Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895
```

```
plotMA(res_ma,ylim = c(-3, 3))
```



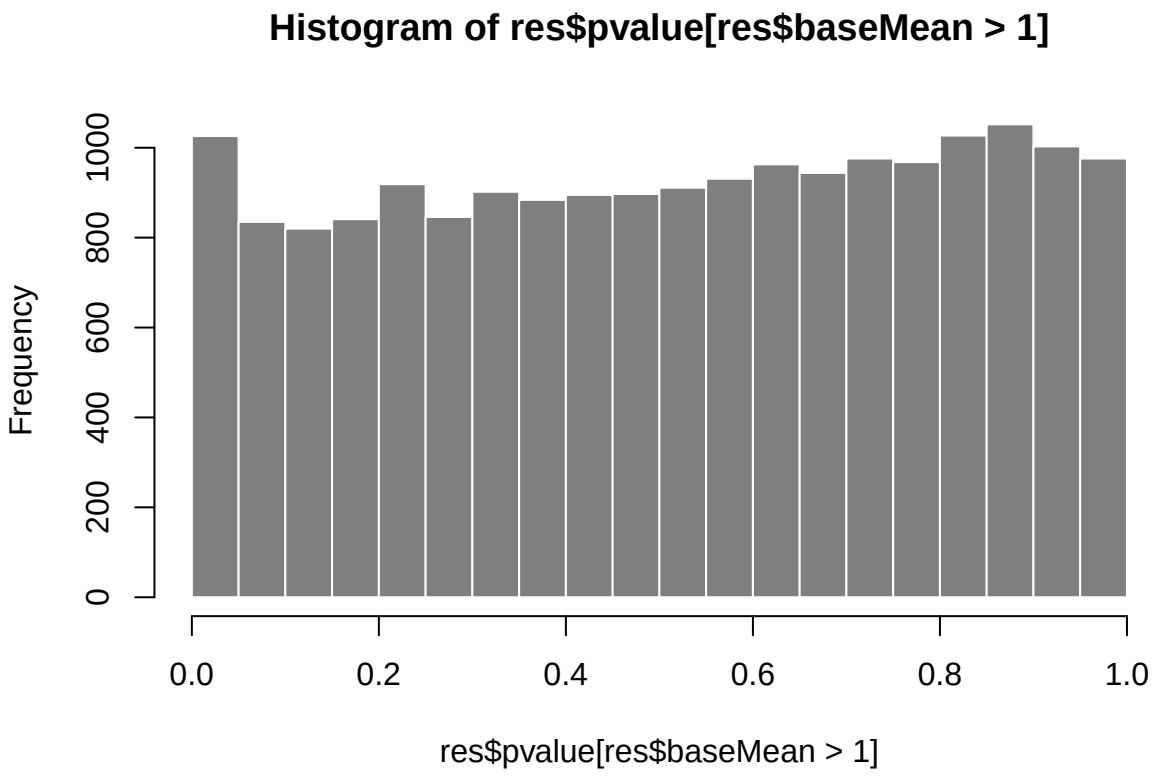
MA Plot without Shrink

```
res_ma_noshr = results(dds, name="Treatment_GF_vs_CV")

plotMA(res_ma_noshr,ylim = c(-4, 4))
```



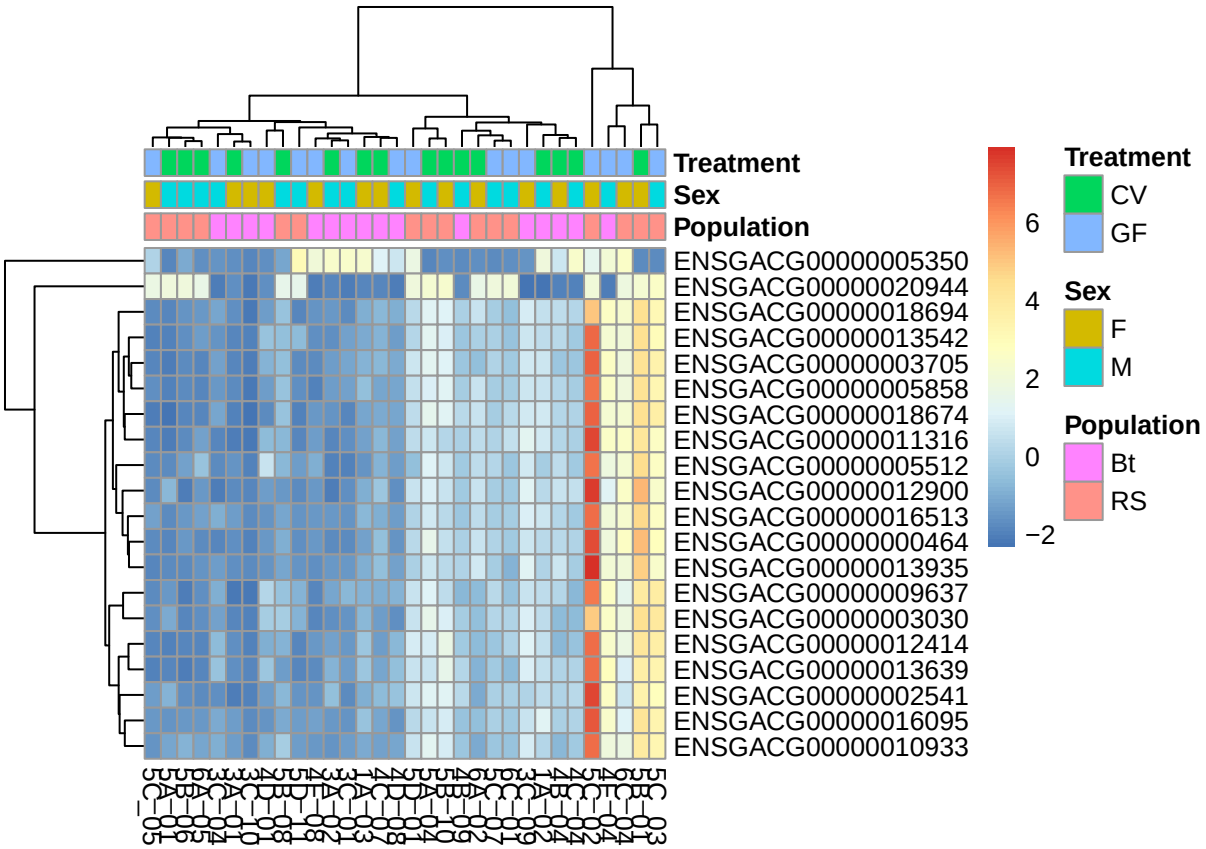
Histogram of pvalues - Excluding Low Counts

```
hist(res$pvalue[res$baseMean > 1], breaks = 0:20/20,
     col = "grey50", border = "white")
```



Gene Clustering

```
# isolate top varying genes
topVarGenes = head(order(rowVars(assay(rld)), decreasing = TRUE), 20)
mat = assay(rld)[topVarGenes,]

# mean normalize
mat = mat - rowMeans(mat)

# dataframe of each effect variable
anno = as.data.frame(colData(rld))[,c("Population","Sex","Treatment")]

# plot heatmap
pheatmap(mat, annotation_col = anno)
```

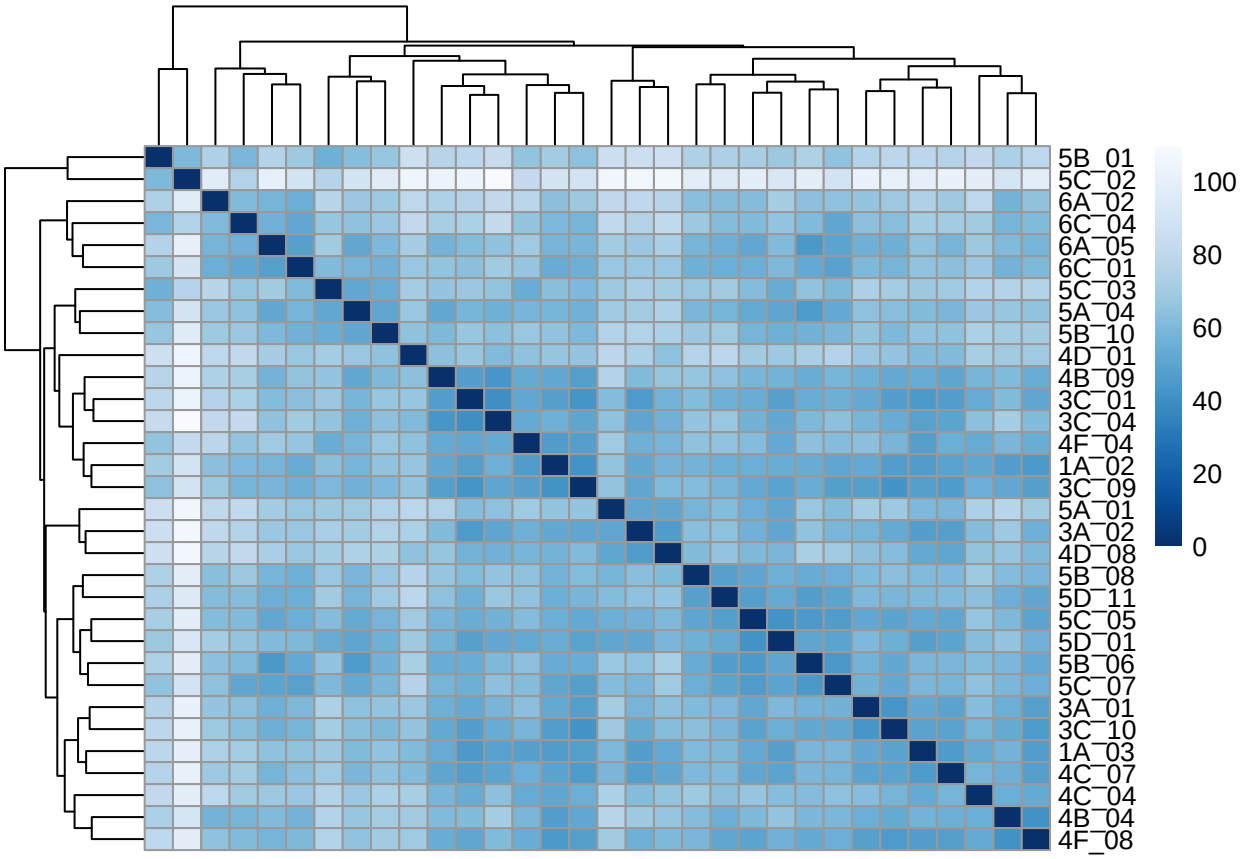


Rerun with Fewer Factors

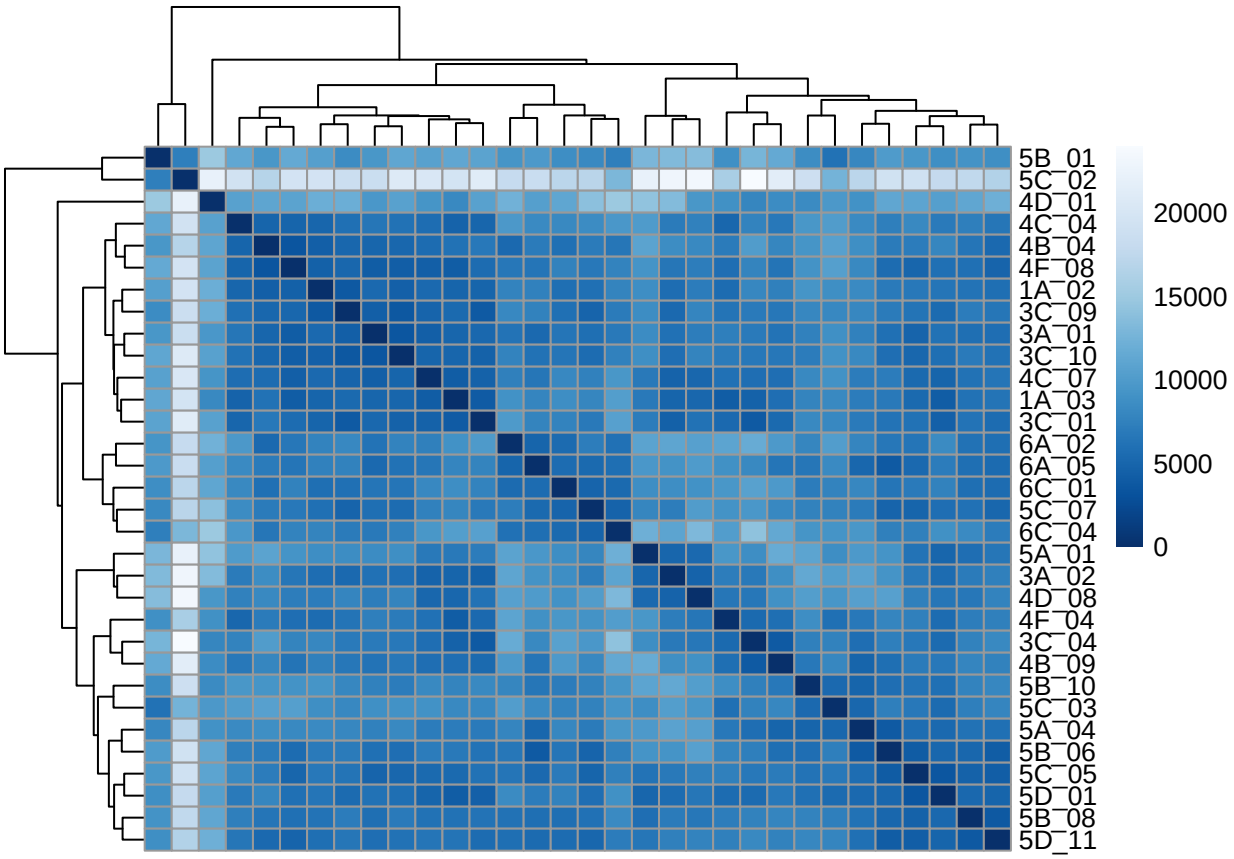
8. Rerun with Fewer Factors - No Population in Design

```
# deseq matrix
ddsMat = DESeqDataSetFromMatrix(
  countData = countdata,
  colData = coldata,
  design = ~ Sex + Treatment,
)

# deseq2 object
dds = DESeq(ddsMat)

# keep only counts greater than 1
keep = rowSums(counts(dds)) > 1
dds = dds[keep,]

# perform transformation
rld <- rlog(dds, blind = TRUE)

# calculate euclidean distances between samples
sampleDists = dist(t(assay(rld))) # transpose so genes are columns
```

```
# generate matrix and heatmap
sampleDistMatrix = as.matrix(sampleDists)
rownames(sampleDistMatrix) = colnames(rld)
colnames(sampleDistMatrix) = NULL
colors = colorRampPalette(rev(brewer.pal(9,"Blues")))(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows=sampleDists,
         clustering_distance_cols=sampleDists,
         col=colors
)
```

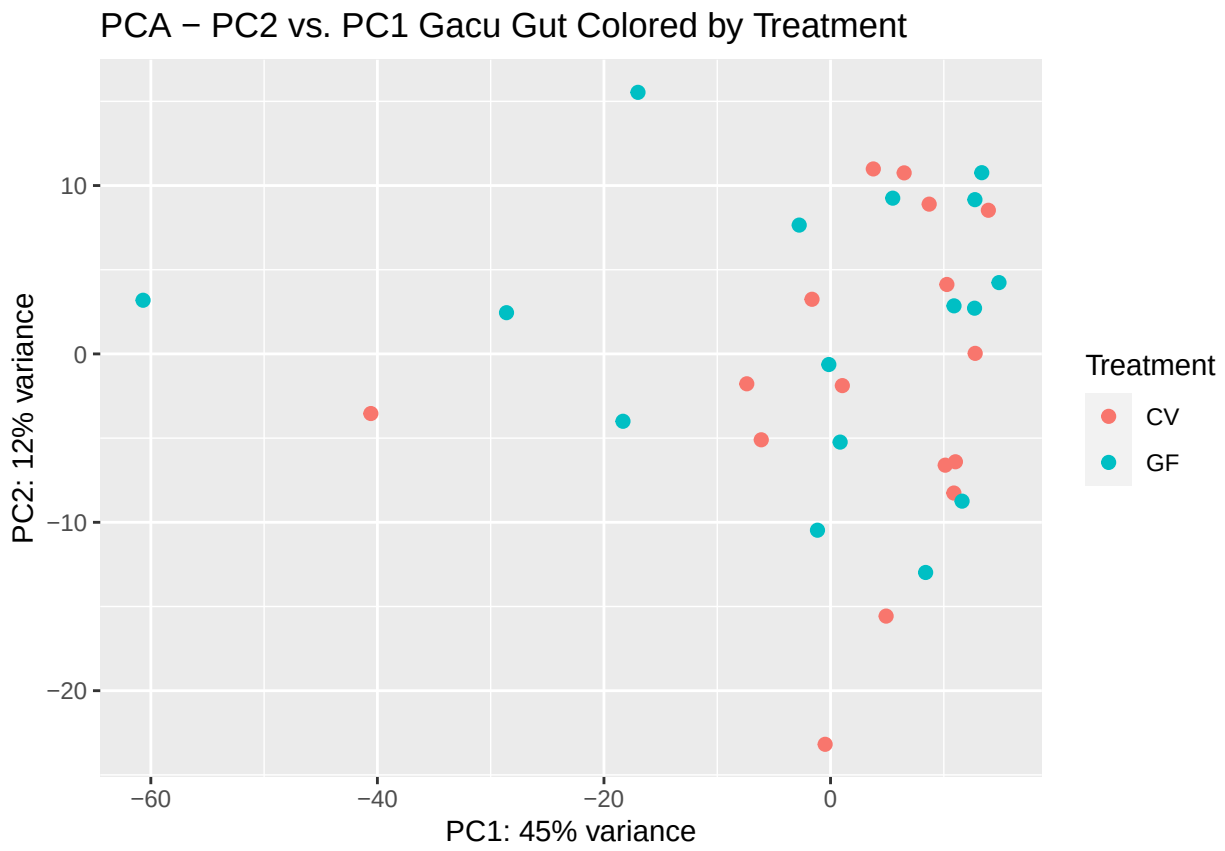



```
# calculate poisson distance
poisd = PoissonDistance(t(counts(dds))) # must transpose so genes are columns

# generate matrix and heatmap
samplePoisDistMatrix = as.matrix(poisd$dd)
rownames(samplePoisDistMatrix) = colnames(rld)
colnames(samplePoisDistMatrix) = NULL
pheatmap(samplePoisDistMatrix,
         clustering_distance_rows=poisd$dd,
         clustering_distance_cols=poisd$dd,
         col=colors
)
```

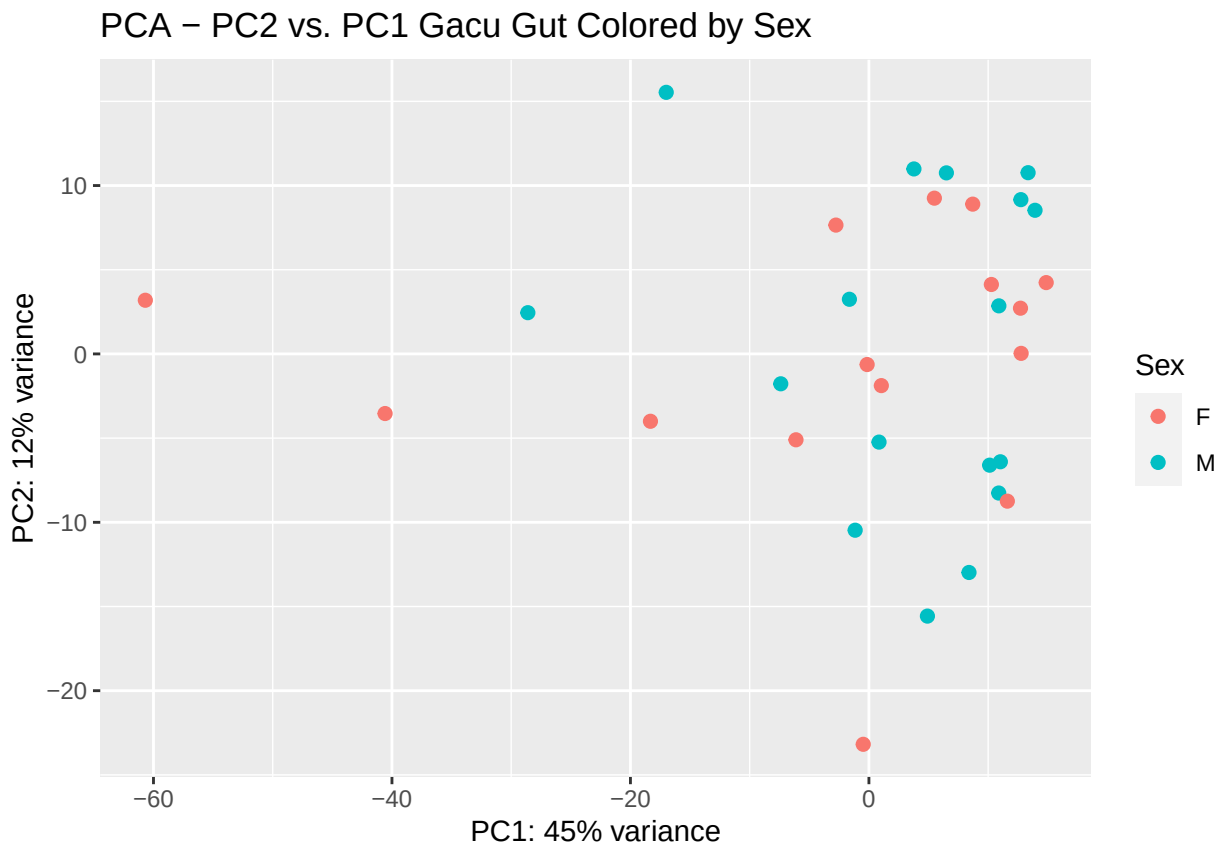



```
# perform pca
colData(dds)
pcaData = plotPCA(rld,intgroup=c("Sex","Treatment"),returnData=TRUE)
pcaData

percentVar = round(100 * attr(pcaData, "percentVar"))
```

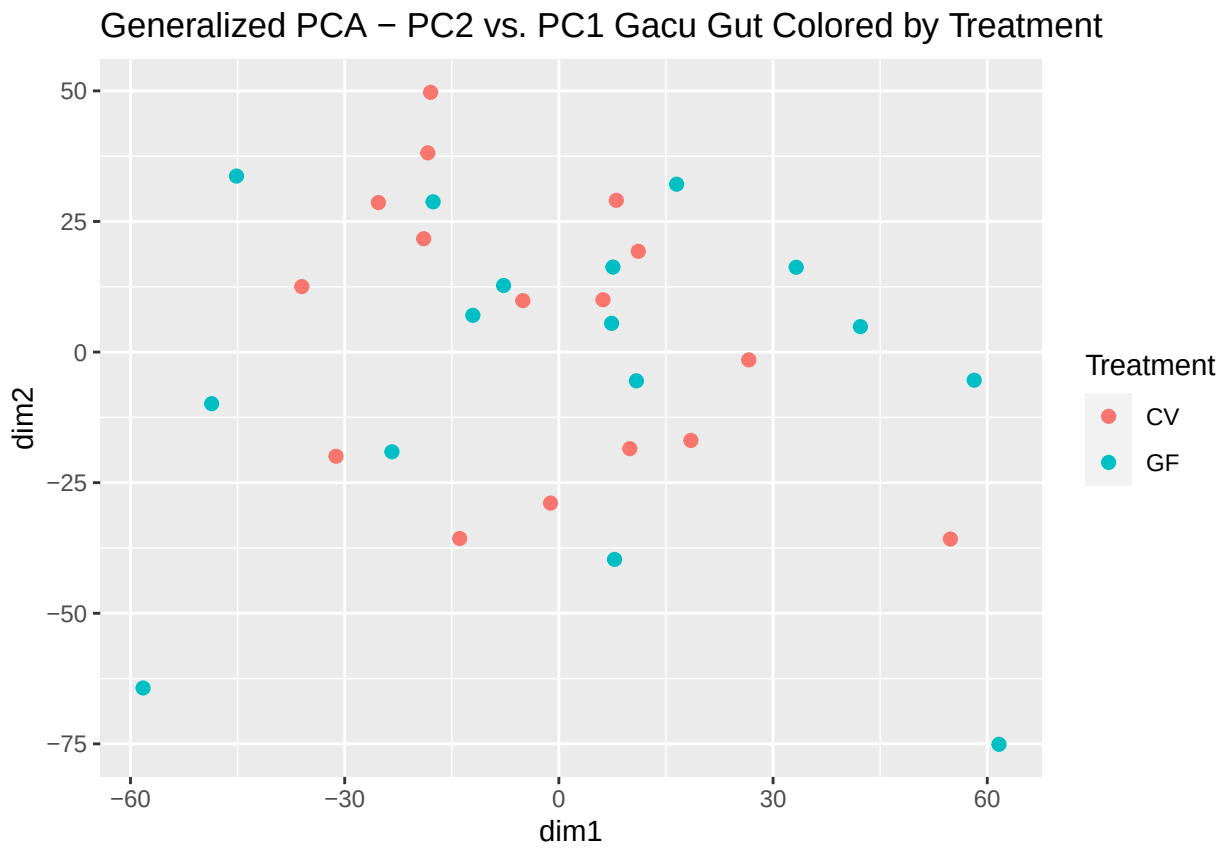
```
# generate plot
ggplot(pcaData, aes(x=PC1,y=PC2,color=Treatment)) +
  geom_point(size=2) +
  xlab(paste0("PC1: ", percentVar[1], "% variance")) +
  ylab(paste0("PC2: ", percentVar[2], "% variance")) +
  ggtitle("PCA - PC2 vs. PC1 Gacu Gut Colored by Treatment")
```



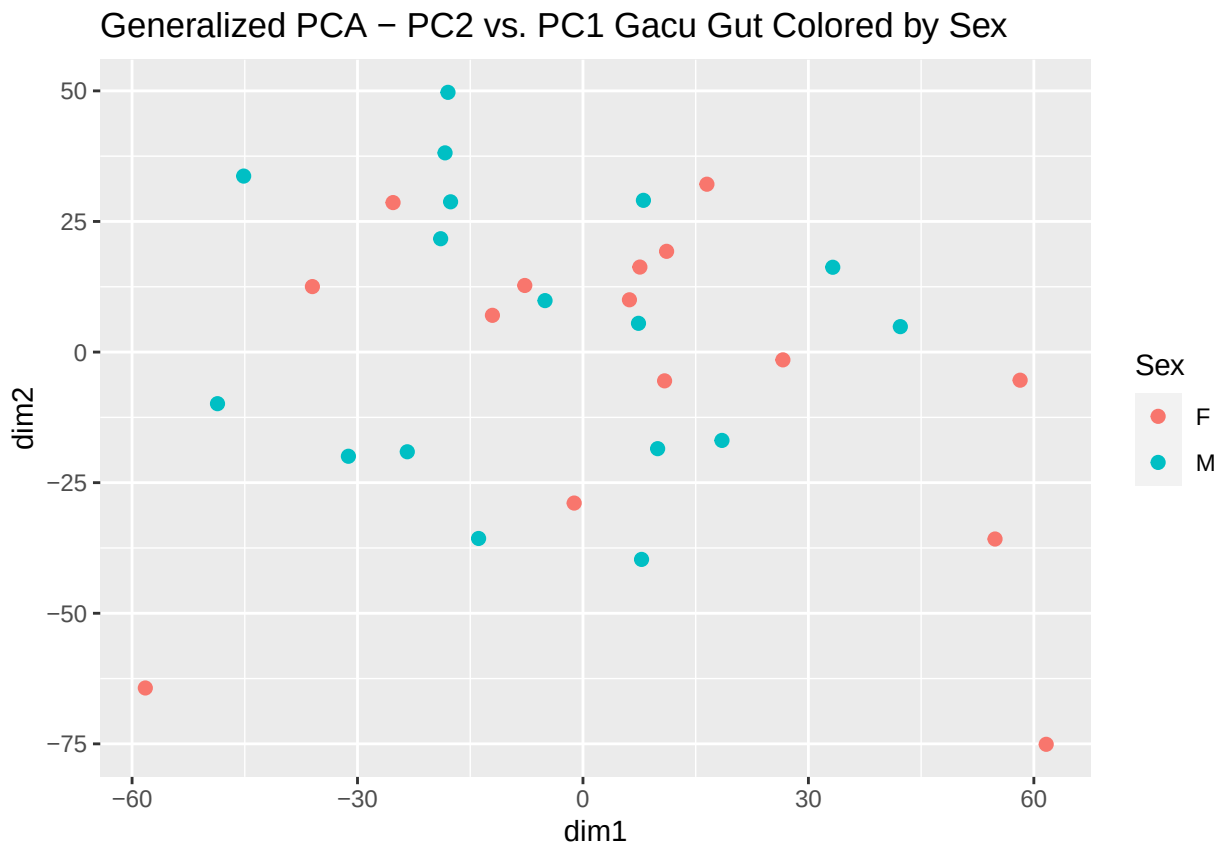

```
ggplot(pcaData, aes(x=PC1,y=PC2,color=Sex)) +
  geom_point(size=2) +
  xlab(paste0("PC1: ", percentVar[1], "% variance")) +
  ylab(paste0("PC2: ", percentVar[2], "% variance")) +
  ggtitle("PCA - PC2 vs. PC1 Gacu Gut Colored by Sex")
```



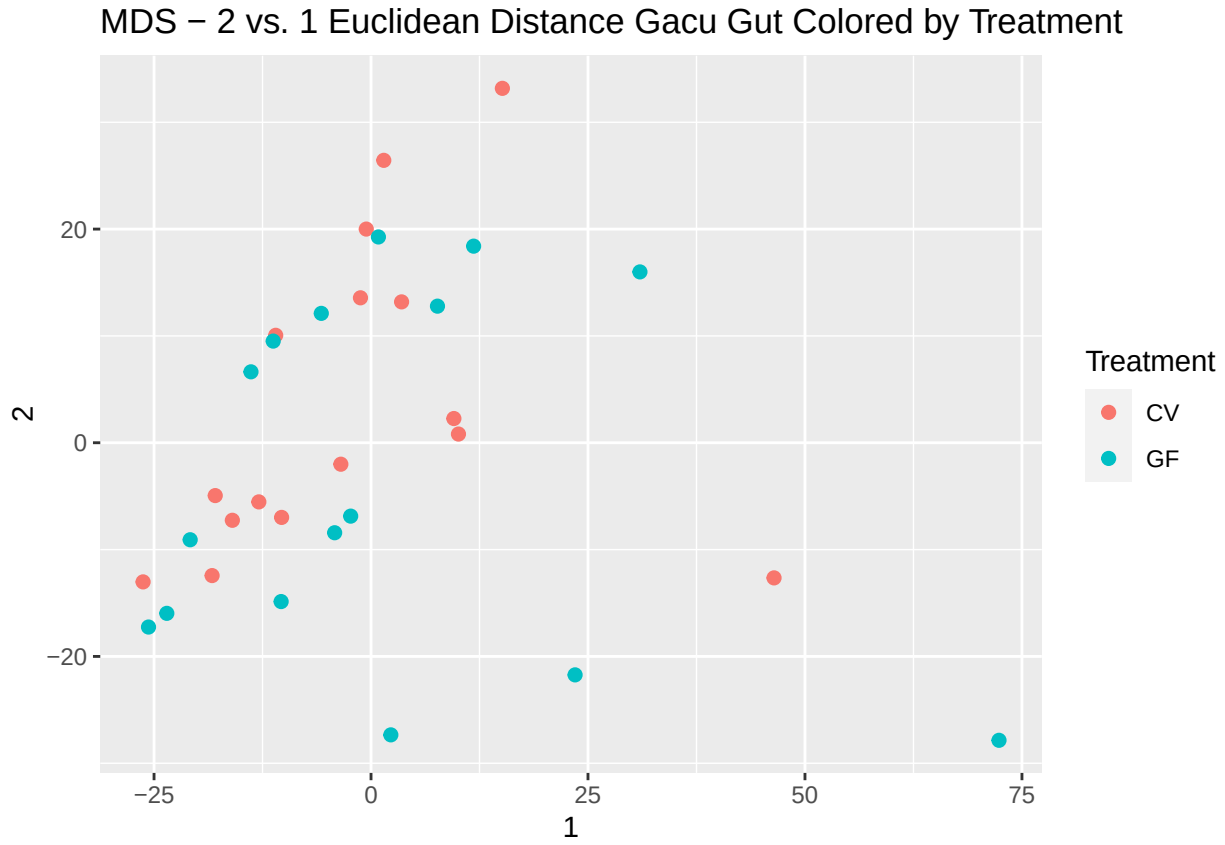

```
# perfom generalized pca
gpca = glmpca(counts(dds), L=2)
gpca.dat = gpca$factors
gpca.dat$Treatment = dds$Treatment
gpca.dat$Sex = dds$Sex

# plot generalized pca
ggplot(gpca.dat, aes(x=dim1, y=dim2, color=Treatment)) +
  geom_point(size=2) +
  ggtitle("Generalized PCA - PC2 vs. PC1 Gacu Gut Colored by Treatment")
```



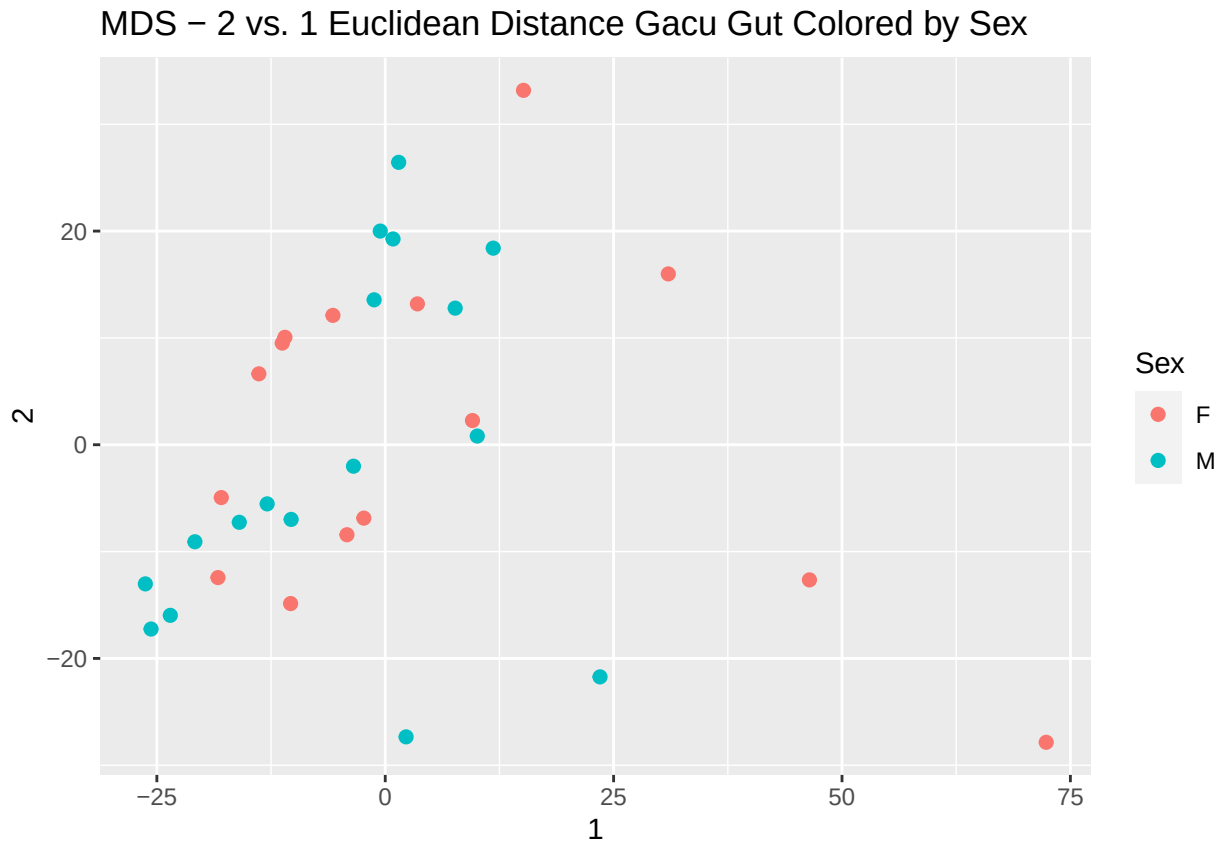

```
ggplot(gpca.dat, aes(x=dim1, y=dim2, color=Sex)) +
  geom_point(size=2) +
  ggtitle("Generalized PCA - PC2 vs. PC1 Gacu Gut Colored by Sex")
```



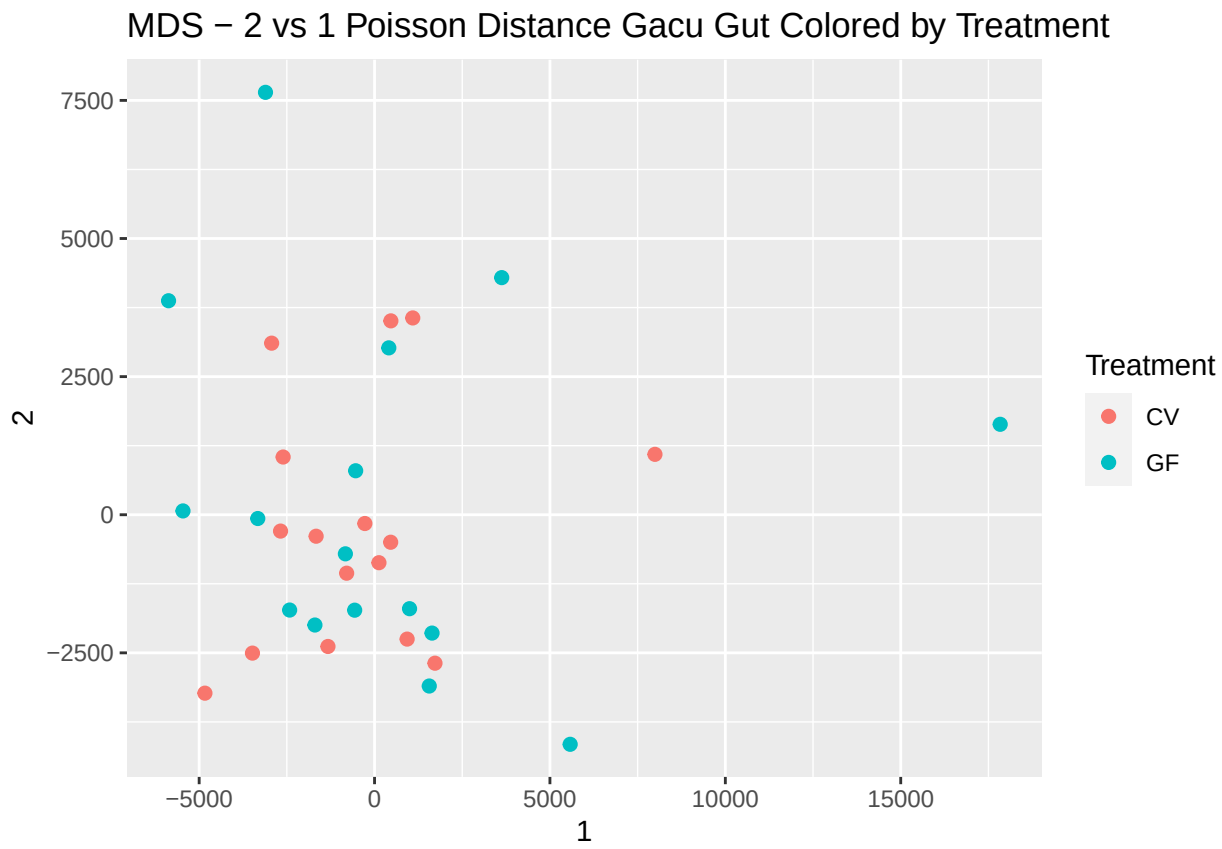

```
# calculate mds
mds = as.data.frame(colData(rld)) %>%
  cbind(cmdscale(sampleDistMatrix))

# plot euclidean
ggplot(mds, aes(x=`1`,y=`2`,color=Treatment)) +
  geom_point(size=2) +
  ggtitle("MDS - 2 vs. 1 Euclidean Distance Gacu Gut Colored by Treatment")
```



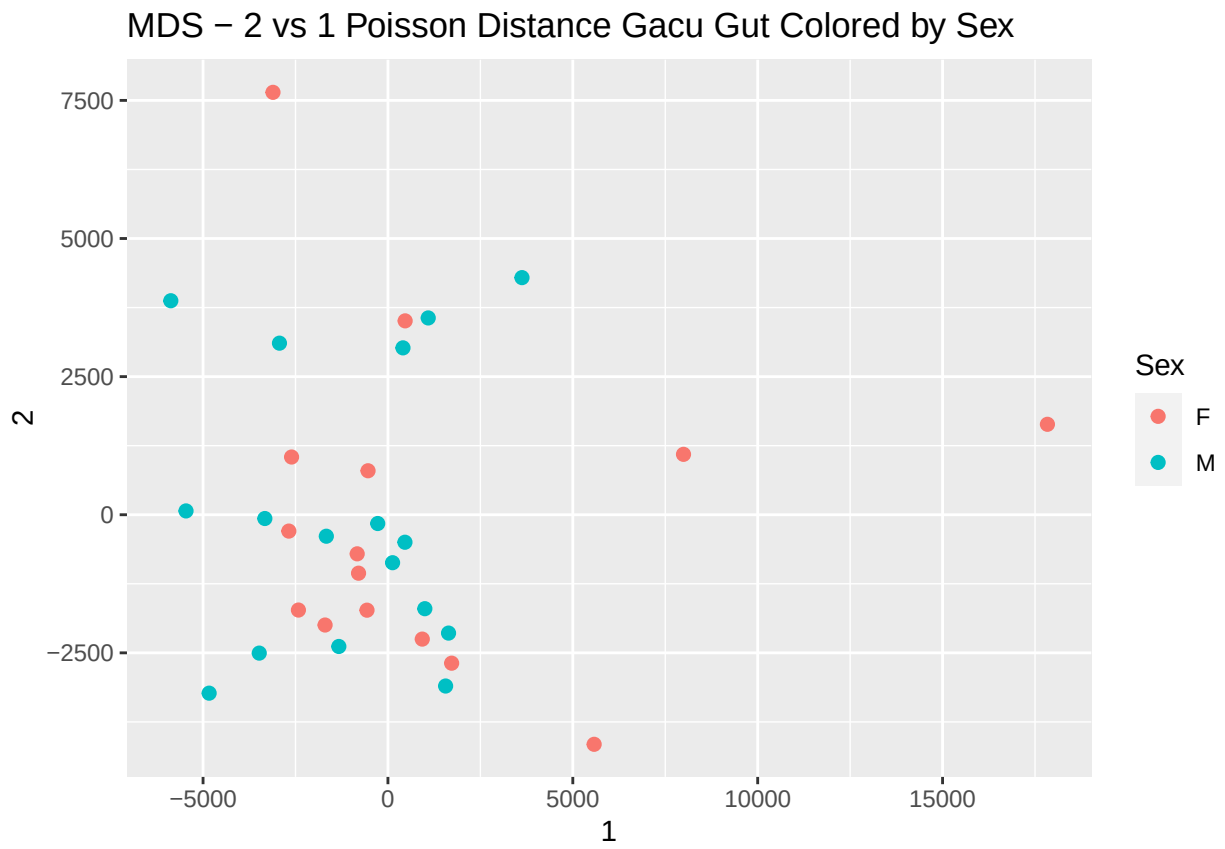

```
ggplot(mds, aes(x=`1`,y=`2`,color=Sex)) +
  geom_point(size=2) +
  ggtitle("MDS - 2 vs. 1 Euclidean Distance Gacu Gut Colored by Sex")
```




```
# calculate mds poisson
mdsPois = as.data.frame(colData(dds)) %>%
  cbind(cmdscale(samplePoisDistMatrix))

# plot poisson
ggplot(mdsPois, aes(x=`1`,y=`2`,color=Treatment)) +
  geom_point(size=2) +
  ggtitle("MDS - 2 vs 1 Poisson Distance Gacu Gut Colored by Treatment")
```

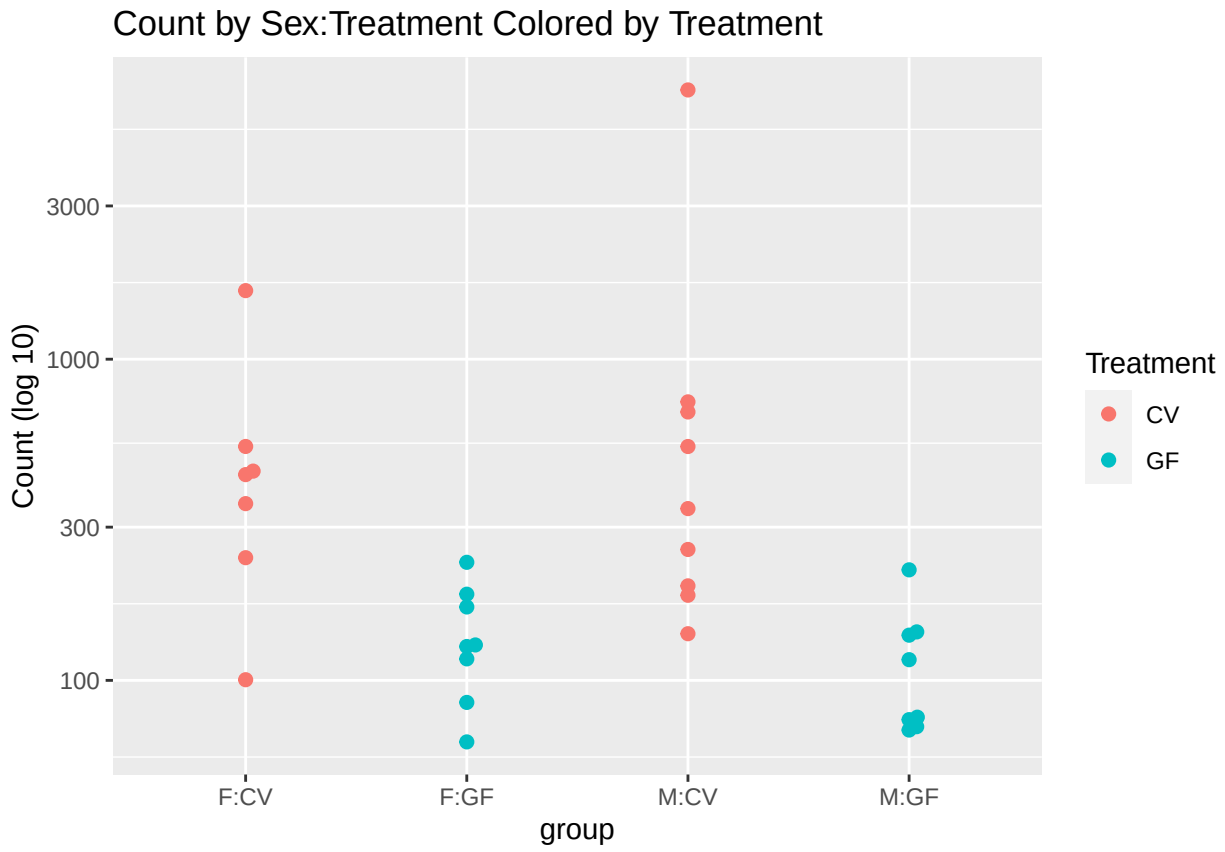



```
ggplot(mdsPois, aes(x=`1`,y=`2`,color=Sex)) +
  geom_point(size=2) +
  ggtitle("MDS - 2 vs 1 Poisson Distance Gacu Gut Colored by Sex")
```

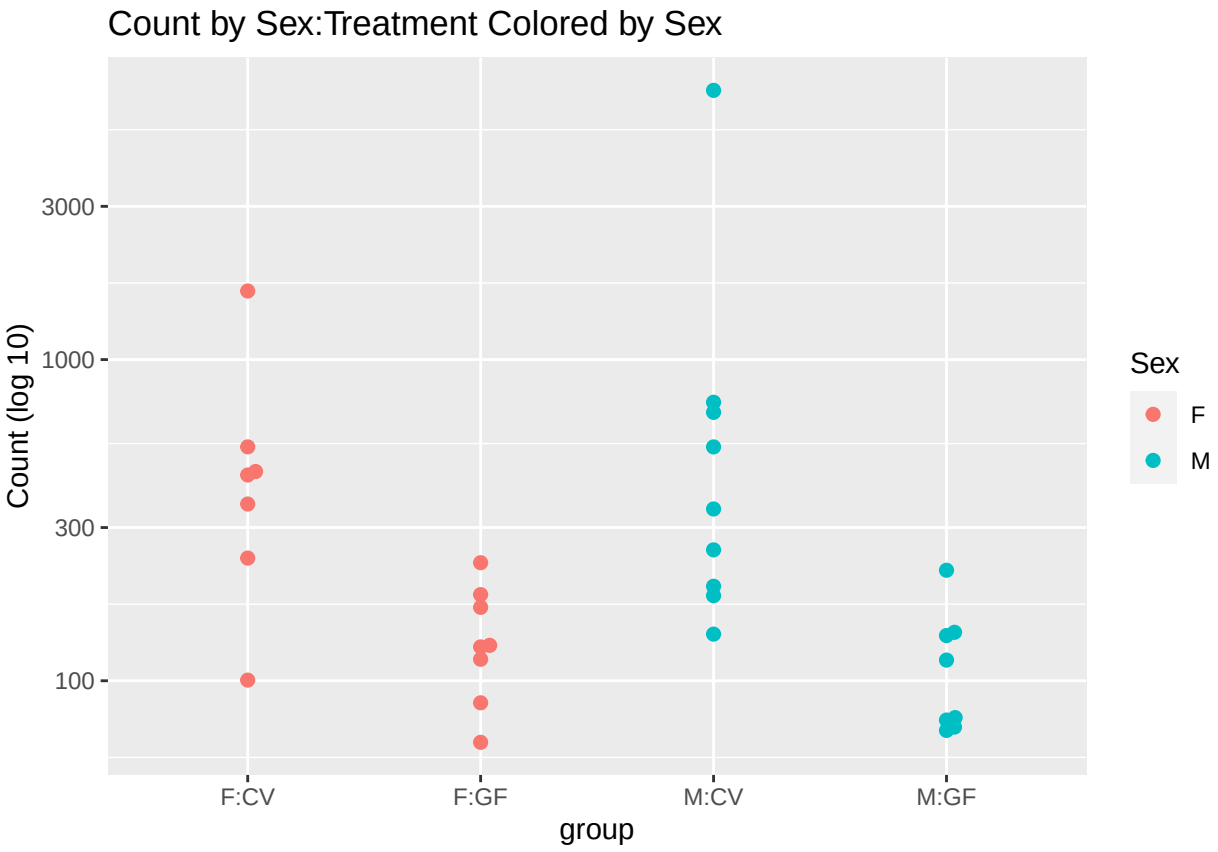



```
# generate results table
res = results(dds)
res

# print summary
summary(res)

# calculate number of genes with pvalue less than 0.1
sum(res$pvalue < 0.1, na.rm=TRUE)

# calculate number of genes with a padj less than 0.1
sum(res$padj < 0.1, na.rm=TRUE)

# generate subset
padj_mask = which(res$padj < 0.1)
res_subset = res[padj_mask,] # may need to do this a different way actually

# print to console
print(res_subset)

# determine gene with lowest padjust value
topGene = rownames(res)[which.min(res$padj)]

# plot
count_plot = plotCounts(dds, gene=topGene, intgroup=c("Sex","Treatment"), returnData=TRUE)
count_plot$group = pcaData$group
```

```
# count plot (beeswarm)
ggplot(count_plot, aes(x=group, y=count, color=Treatment)) +
  geom_beeswarm(size=2) +
  scale_y_log10() +
  ggtitle("Count by Sex:Treatment Colored by Treatment") +
  ylab("Count (log 10)")
```



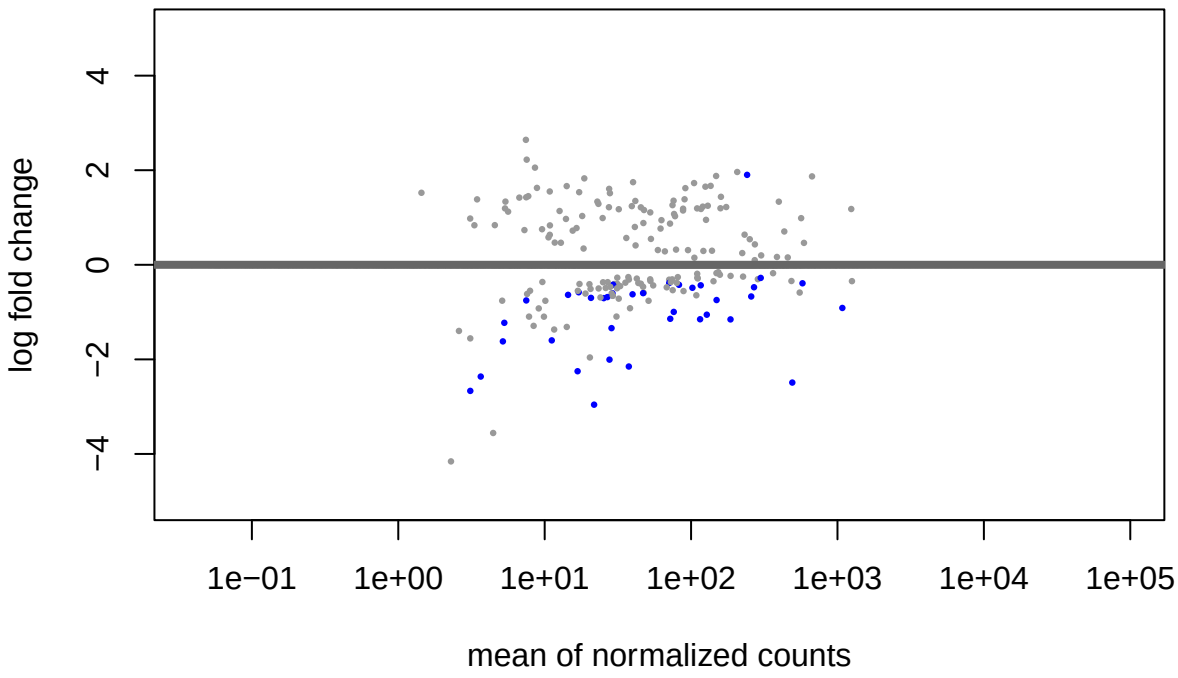

```
ggplot(count_plot, aes(x=group, y=count, color=Sex)) +
  geom_beeswarm(size=2) +
  scale_y_log10() +
  ggtitle("Count by Sex:Treatment Colored by Sex") +
  ylab("Count (log 10)")
```



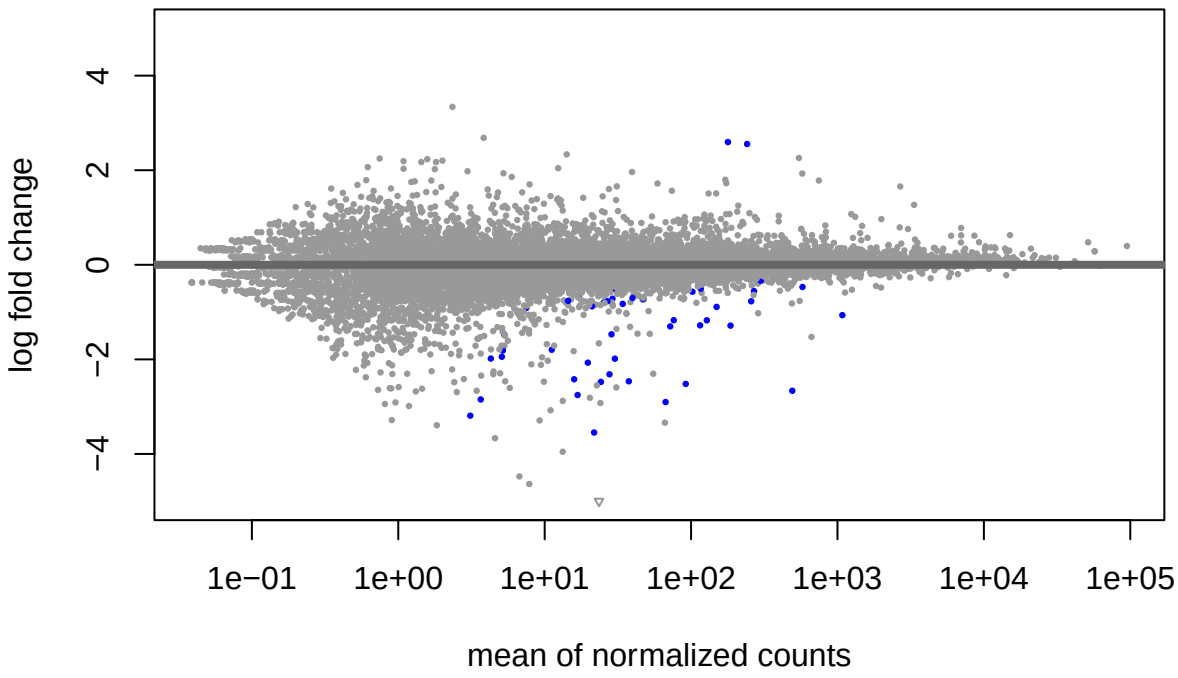

```
# MA plot
res_ma = lfcShrink(dds, coef="Treatment_GF_vs_CV", type="apeglm")
```

```
## using 'apeglm' for LFC shrinkage. If used in published research, please cite:
##     Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
##     sequence count data: removing the noise and preserving large differences.
##     Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895
```

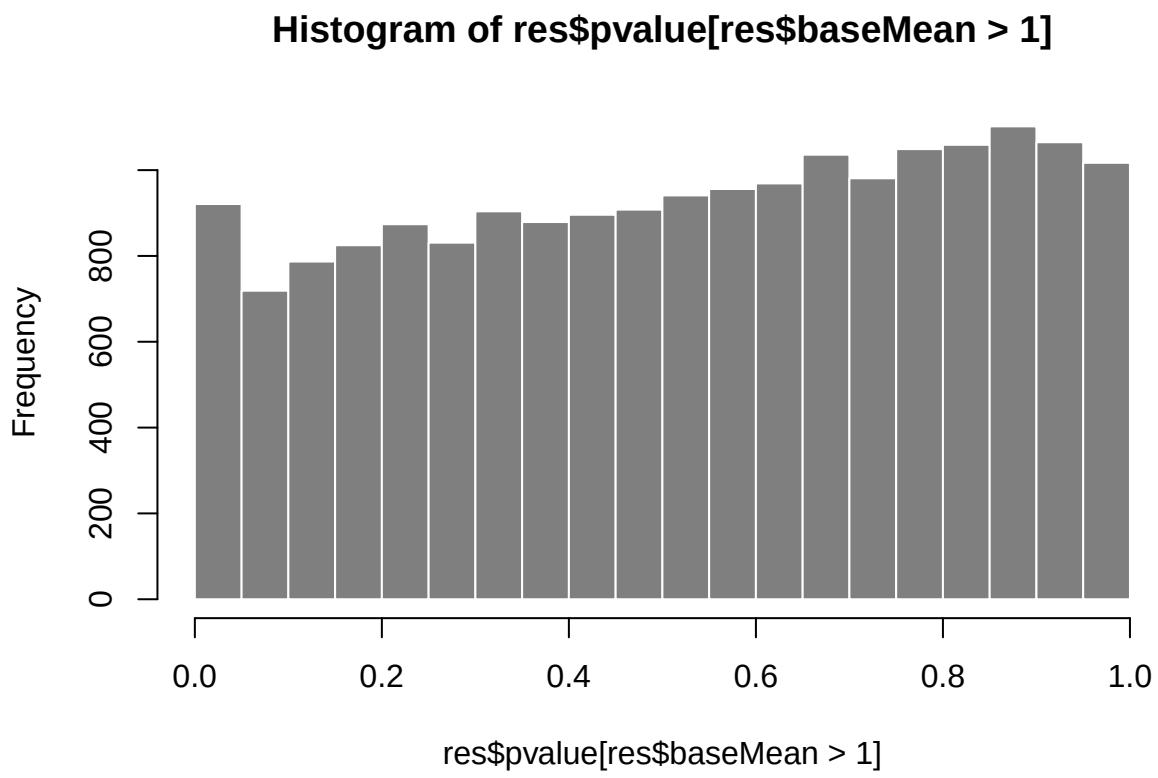
```
plotMA(res_ma,ylim = c(-5, 5))
```



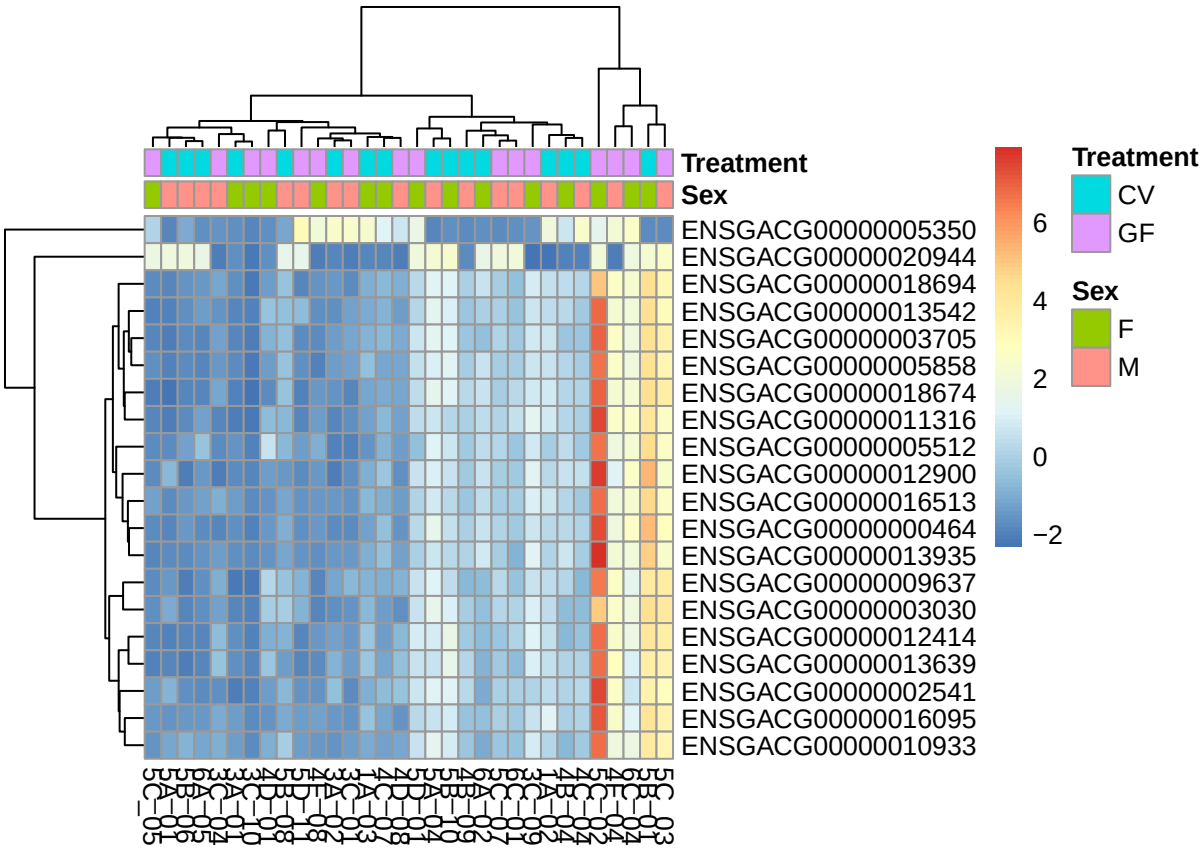

```
# MA plot without shrink
res_ma_noshr = results(dds, name="Treatment_GF_vs_CV")
plotMA(res_ma_noshr,ylim = c(-5, 5))
```




```
# histogram of pvalues
hist(res$pvalue[res$baseMean > 1], breaks = 0:20/20,
     col = "grey50", border = "white")
```




```
# isolate top varying genes
topVarGenes = head(order(rowVars(assay(rld)), decreasing = TRUE), 20)
mat = assay(rld)[topVarGenes,]

# mean normalize
mat = mat - rowMeans(mat)

# dataframe of each effect variable
anno = as.data.frame(colData(rld))[,c("Sex","Treatment")]

# plot heatmap
pheatmap(mat, annotation_col = anno)
```



1677 genes with p value less than 0.1

48 genes with padj less than 0.1

How are the results different when Population is excluded from the design? When Population is excluded from the design, we find fewer significant results. When population is included, we find 74 of genes to be significantly differentially expressed, as opposed to the 48 of genes found when population is excluded. This is unsurprising when taking the 'Colored by Population' plots into consideration. The dimensionally-reduced plots appear to cluster best based by population, meaning that population likely explains a large quantity of the difference between expression profiles. Removing population as an effect variable tells the program not to take this variable into account while performing hypothesis testing.