Noah L. Schrick - 1492657

Lab 4. Introduction to gene expression data Part 2: Differential Expression. We will use the major depressive disorder (MDD) RNA-Seq data and some basic statistics to identify genes that are differentially expressed between phenotype groups (*e.g.*, between disease and healthy control).

A. Prepare Data. Load the RNA-Seq and clinical data from lab 3. Repeat last week's preprocessing shown below: quantile normalization and log2 transformation. **1.** How many genes are in the data?

```
# load gene expression data
load("sense.filtered.cpm.Rdata")  # remember setwd

# load phenotype (mdd/hc) data
subject.attrs <- read.csv("Demographic_symptom.csv",
                            stringsAsFactors = FALSE)
library(dplyr)
# grab intersecting X (subject ids) and Diag (Diagnosis) from columns
phenos.df <- subject.attrs %>%
             filter(X %in% colnames(sense.filtered.cpm)) %>%
             dplyr::select(X, Diag)
mddPheno <- as.factor(phenos.df$Diag)

# Normalized and transform
library(preprocessCore)
mddExprData_quantile <- normalize.quantiles(sense.filtered.cpm)
mddExprData_quantileLog2 <- log2(mddExprData_quantile)
# attach phenotype names and gene names to data
colnames(mddExprData_quantileLog2) <- mddPheno
rownames(mddExprData_quantileLog2) <- rownames(sense.filtered.cpm)
```

1. length(rownames(sense.filtered.cpm))
8923

B. Filter noise genes. There are many genes in the data that are noise or irrelevant to the disease study at hand. Furthermore, these noise genes waste computational time during analysis. The coefficient of variation (CoV) of a gene (call it gene x) is the ratio `cv(x)=sd(x)/abs(mean(x))`, where sd is the standard deviation of the gene expression. A small CoV may mean the experimental effect size is large compared to the measurement uncertainty. The code below uses CoV to filter the genes. Note CoV does not use phenotype information, so we don't have to worry about biasing the differential expression analysis. **2.** Add the following code. What does it do? Specifically, what is "apply" doing (use ?apply)? How many genes are there now?

```
# coefficient of variation filter sd(x)/abs(mean(x))
```

```
CoV_values <- apply(mddExprData_quantileLog2,1,
                      function(x) {sd(x)/abs(mean(x))})
# smaller threshold, the higher the experimental effect relative to the
# measurement precision
sum(CoV_values<.045)
# there is one gene that has 0 variation -- remove
sd_values <- apply(mddExprData_quantileLog2,1, function(x) {sd(x)})
rownames(mddExprData_quantileLog2)[sd_values==0]
# filter the data matrix
GxS.covfilter <- mddExprData_quantileLog2[CoV_values<.045 & sd_values>0,]
dim(GxS.covfilter)
```

2. The "apply" portion loops through the mdd expression data rows that have undergone quantile normalization, and for each element, it computes the standard deviation and the absolute value of the mean, divides the first by the latter, and adds it to our CoV_values.

The remaining parts perform "clean up" work, where it removes values with 0 variation (by using apply to compute standard deviation for each value in the original data), and also only selects data with CoV values less than 0.045, but greater than 0.

There are now 5587 genes.

C. Using the t-test to calculate differential expression. The following equation is the two-sample t-test with unequal variances for gene x with a vector of expression levels for groups 1 and 2 (subscripts). The numerator is the difference of the mean expression of gene x between groups 1 and 2. And the denominator involves the pooled deviation from the means of each group, with sizes n and m.

$$t=\frac{\bar{x}_1-\bar{x}_2}{\sqrt{\frac{s_1^2}{n}+\frac{s_2^2}{m}}}$$

Before we compute the t-test, we need to create a factor variable for our two phenotype groups. A factor is a special R data type: it is a categorical variable with discrete states (levels) that do not necessarily have a meaningful order. **3**. Add the following code. What are the levels and how many are there?

```
# convert phenotype
```

```
pheno.factor <- as.factor(colnames(GxS.covfilter))
pheno.factor
str(pheno.factor)
levels(pheno.factor)
```

3. There are 2 levels: HC, and MDD.

There are a few interfaces for calculating the t-test in R, which we now try on the second gene in the data (myrow=2 of the data):

```
myrow <- 2  # first pick a gene row index to test
mygene<-rownames(GxS.covfilter)[myrow]
mygene
```

4. Run the following implementations of the t-test for this gene. Show the output and discuss how implementations a and b differ (use `?t.test`). Discuss the evidence for whether this gene is differentially expressed.

```
# a. traditional R interface
mdd <- GxS.covfilter[myrow,pheno.factor=="MDD"]
hc <- GxS.covfilter[myrow,pheno.factor=="HC"]
t.result <- t.test(mdd,hc)
t.result

# b. formula interface ~ saves a step
t.result <- t.test(GxS.covfilter[myrow,] ~ pheno.factor)
t.result
t.result$p.value
t.result$statistic
```

4.

a:

```
data:  mdd and hc
t = 1.4367, df = 154.9, p-value = 0.1528
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-0.01003427  0.06355762
sample estimates:
mean of x mean of y
5.590087  5.563325
```

b:

```
data:  GxS.covfilter[myrow, ] by pheno.factor
t = -1.4367, df = 154.9, p-value = 0.1528
alternative hypothesis: true difference in means between group HC and
        group MDD is not equal to 0
95 percent confidence interval:
 -0.06355762  0.01003427
sample estimates:
mean in group HC mean in group MDD
5.563325         5.590087
```

Differences:

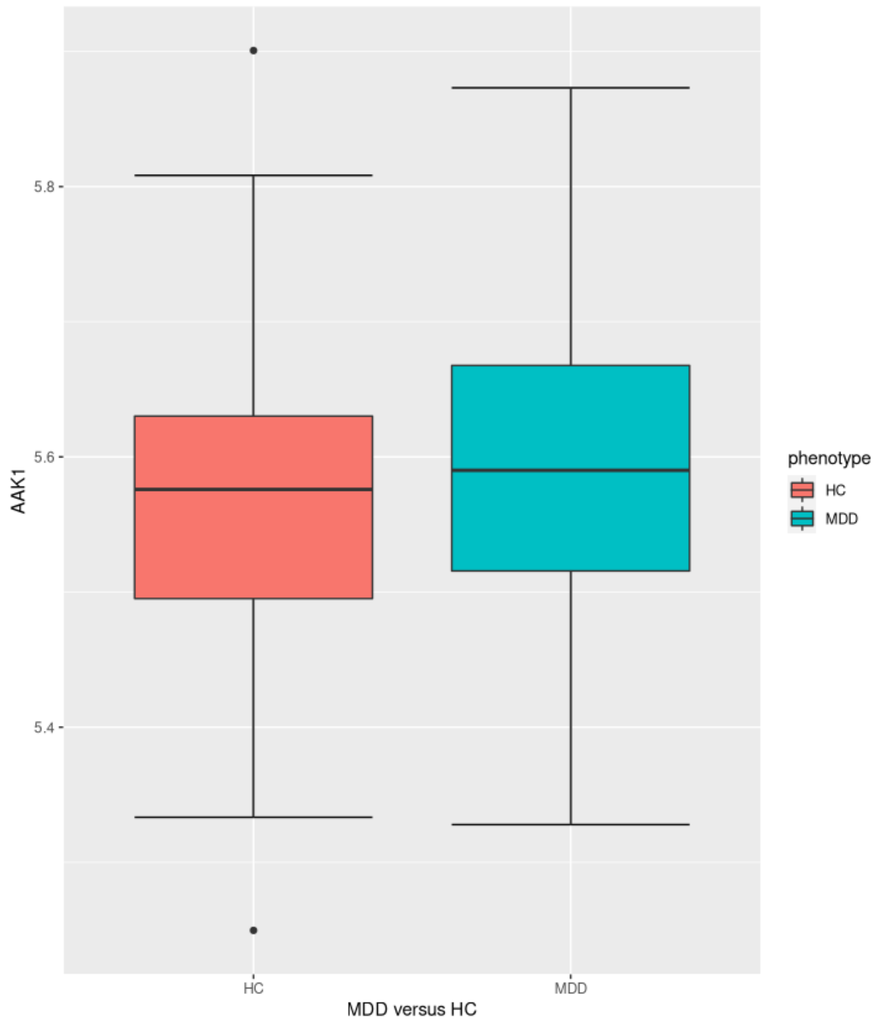
"a" has a positive t value, while b is negative. 95% CI is also flipped between a and b, as are the means. All of this is due to group HC being first in "b", while group MDD is first in "a". For "a", mdd and hc were deliberately set, and in part "b", the t-test was run by feeding in the pheno.factor. The p-value obtained was 0.15, and the t-test returned a value of 1.436705 with the alternative hypothesis. It can be concluded that the genes are differentially expressed.

5. Plot the data. What gene name was analyzed (which gene is myrow=2)? Would you say this gene is differentially expressed? Why/why not?

```
# plot the data
library(ggplot2)
# create data frame for gene
mygene.data.df <-
data.frame(gene=GxS.covfilter[myrow,],phenotype=pheno.factor)
# boxplot
p <- ggplot(mygene.data.df, aes(x=phenotype, y=gene, fill=phenotype)) +
stat_boxplot(geom ='errorbar') + geom_boxplot()
```

5.

```
> rownames(GxS.covfilter)[2]
[1] "AAK1"
```



Yes, this gene is differentially expressed. Comparing the central tendency of MDD and HC, and examining the Q3 for MDD, it is evident that there is a significant value difference between the two, and that there is differential expression.

D. Calculate the t-test for all filtered genes. We need to repeat the t-test for all genes in the data, so we want to choose the fastest and cleanest way of the above implementations. I argue the formula interface is the simplest. In addition we want to create a results function for a given gene that we can use to automate the analysis of all genes in a for loop. Use the following function in your script to do a t-test for a gene with row index i.

```
# Put it all together into a function to run in loop.
# First write a function that computes t-test for one gene.
# i is the data row for the gene
ttest_fn <- function(i){
      mygene <- rownames(GxS.covfilter)[i]
      t.result <- t.test(GxS.covfilter[i,] ~ pheno.factor)
      tstat <- t.result$statistic
      pval <- t.result$p.value
      # return vector of three things for each gene
      c(mygene, tstat, pval)
}
```

Test the function on the second gene using the following command. **6.** Show the output and explain what the values represent.

```
ttest_fn(2)
```

6.

```
> ttest_fn(2)
                                          t
 "AAK1" "-1.43670544207136" "0.152818373849126"
```

The results are in the format: “gene name”, “t-value”, “p-value”
In this case, we can compare the results to those seen in Question 4, and confirm that the results are the same.

The following code uses a for loop to apply the user-defined `ttest_fn` function above to all genes. Then we sort the results by p-value **7.** Show the top 10 gene results based on smallest p-values. You could use `sapply` instead of a for loop.

```
# initialize an empty matrix to store the results
ttest_allgene.mat <- matrix(0,nrow=nrow(GxS.covfilter), ncol=3)
# run analysis on all gene rows
for (i in 1:nrow(GxS.covfilter)){
  ttest_allgene.mat[i,] <- ttest_fn(i)
}
# convert matrix to data frame and colnames
ttest_allgene.df <- data.frame(ttest_allgene.mat)
colnames(ttest_allgene.df) <- c("gene ", "t.stat", "p.val")

library(dplyr)          # sort based on p-value
```

```
ttest_allgene.sorted <- ttest_allgene.df                  %>%
                        mutate_at("p.val", as.character) %>%
                        mutate_at("p.val", as.numeric)   %>%
                        arrange(p.val) # sort
ttest_allgene.sorted[1:10,] # look at top 10
```

7.

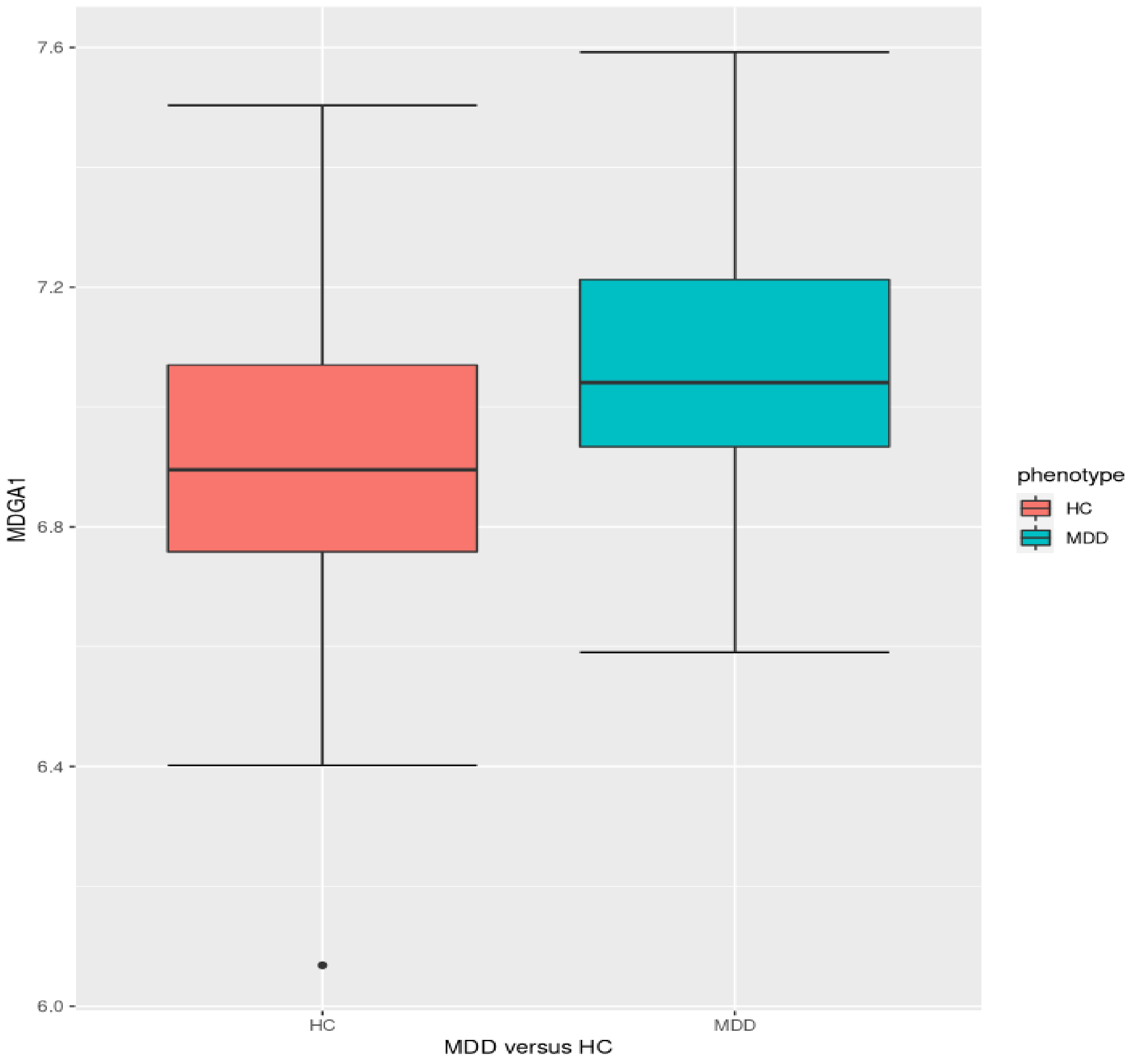
```
> ttest_allgene.sorted[1:10,] # look at top 10
      gene            t.stat        p.val
1    MDGA1 -4.60305377574858 8.769336e-06
2  ZDHHC20 -4.14665384310082 5.560038e-05
3     NPFF  3.92088360163192 1.322952e-04
4  ARFGAP1 -3.84850546153888 1.745521e-04
5  FAM138A  3.81609056502266 1.977449e-04
6  IRF2BPL -3.80484711944538 2.037987e-04
7      UBD -3.77495951345364 2.295039e-04
8  BCL2L12 -3.70907581977241 2.899766e-04
9    KANTR  3.69624774873673 3.048811e-04
10     CBL   -3.694547522781 3.056099e-04
```

8. Add a boxplot for the top gene using the code from C5. First use the following code to specify the top gene for plotting:

```
# find data row index of top gene name
myrow <- which(ttest_allgene.df$gene=="ENTER TOP GENE NAME HERE")
mygene<-rownames(GxS.covfilter)[myrow]
```

8.

myrow <- which(ttest_allgene.df$gene==ttest_allgene.sorted[1,1])



E. Interpretation of top genes. Look up the top gene in NCBI or Google. **9.** What if any is the biological evidence for the relevance of the top gene to major depressive disorder?

https://www.ncbi.nlm.nih.gov/gene/

9.
From NCBI: “MDGAs regulate the formation of neuroligin-neurexin trans-synaptic bridges by sterically blocking access of neurexins to neuroligins.” Numerous related articles in PubMed, as well as the summary of the gene on NCBI, state that this gene is associated with bipolar disorder and schizophrenia.

Use the following code to print the top-200 gene list.

```
top_cutoff <- 200
top_genes <- as.character(ttest_allgene.sorted[1:top_cutoff,1])
write.table(top_genes, sep="\t", file="", quote=F, row.names=F, col.names=F)
```

Paste the list into the following MSigDB (molecular signatures database) gene set enrichment analysis (GSEA) tool. You will need to provide an email address to register. Go to Investigate Gene Sets, paste your list of gene names into the Input Box, check Reactome, and click Compute Overlaps. **10.** What is the top enriched geneset/pathway and what genes from the top list are in it (there is a “Save to: Text” link)?

10.
The top enriched geneset/pathway is a post-translational protein modification:
http://www.gsea-msigdb.org/gsea/msigdb/human/geneset/REACTOME_POST_TRANSLATIONAL_PROTEIN_MODIFICATION.html

There are 23 genes that overlap, and they are:

NUP88
DDX58
TUSC3
ST3GAL3
PSMD1
NFE2L2
RBBP5
MEN1

NR1H2
BTBD1
KLHL22
CDC20
UBD
ARFGAP1
TADA3
SUMO3
MITF
VDAC3
ARFGAP3
COG2
PIGH
MDGA1
TNC

http://www.broadinstitute.org/gsea/msigdb/annotate.jsp

Optional 1. Run the code below. Which subject might be an outlier? Discuss whether or not we should remove this subject. There is no right or wrong answer.

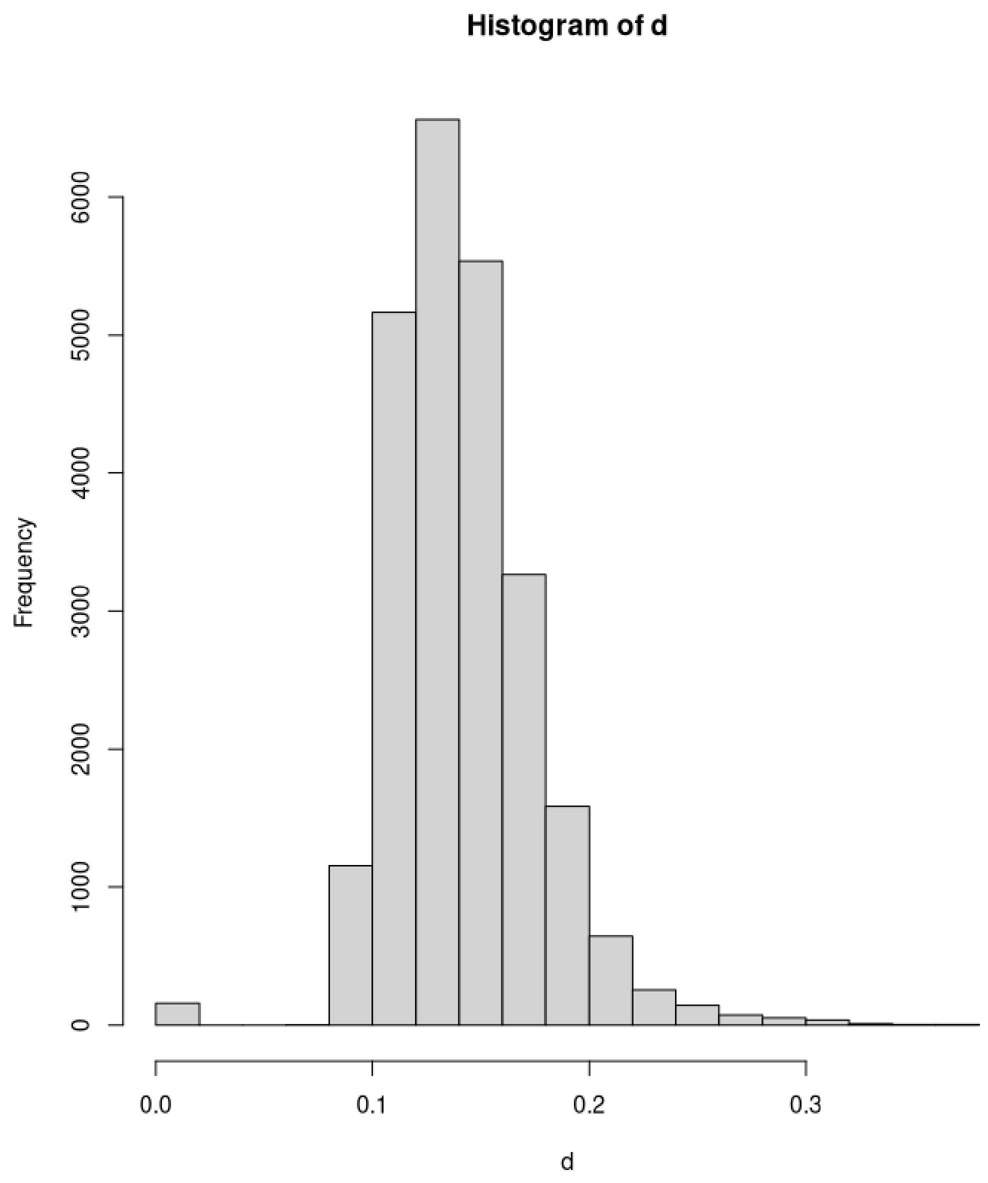
```
mddCorr<-cor(GxS.covfilter)  # distance based on correlation
d <- sqrt(1-mddCorr)
dim(d)
rownames(d)
mddTree = hclust(as.dist(d))
mddTree$labels <- phenos.df$X
plot(mddTree)
```

Opt 1.
Using the “1.5x” above or below rule, we can compute the following in R:

1.5x above third quartile:
which(d>1.5*quantile(d)[4])

1.5x below first quartile:
which(d<1.5*quantile(d)[2])

Histogram:



Using the 1.5x rule, and even from just examining the histogram, there are a large number of genes that could be considered outliers. The “main” outlier however is evident from the tree, and its distance is identifiable by max(d): AI270.

If only a single outlier would be removed, then it is arguably better to not remove it. Assuming AI270 is removed but no others would be removed, gene AA365 would then become a noticeable outlier, and the same question could be posed to this subject. Likewise, if AA365 would be removed, then AD665 would become a noticeable outlier.

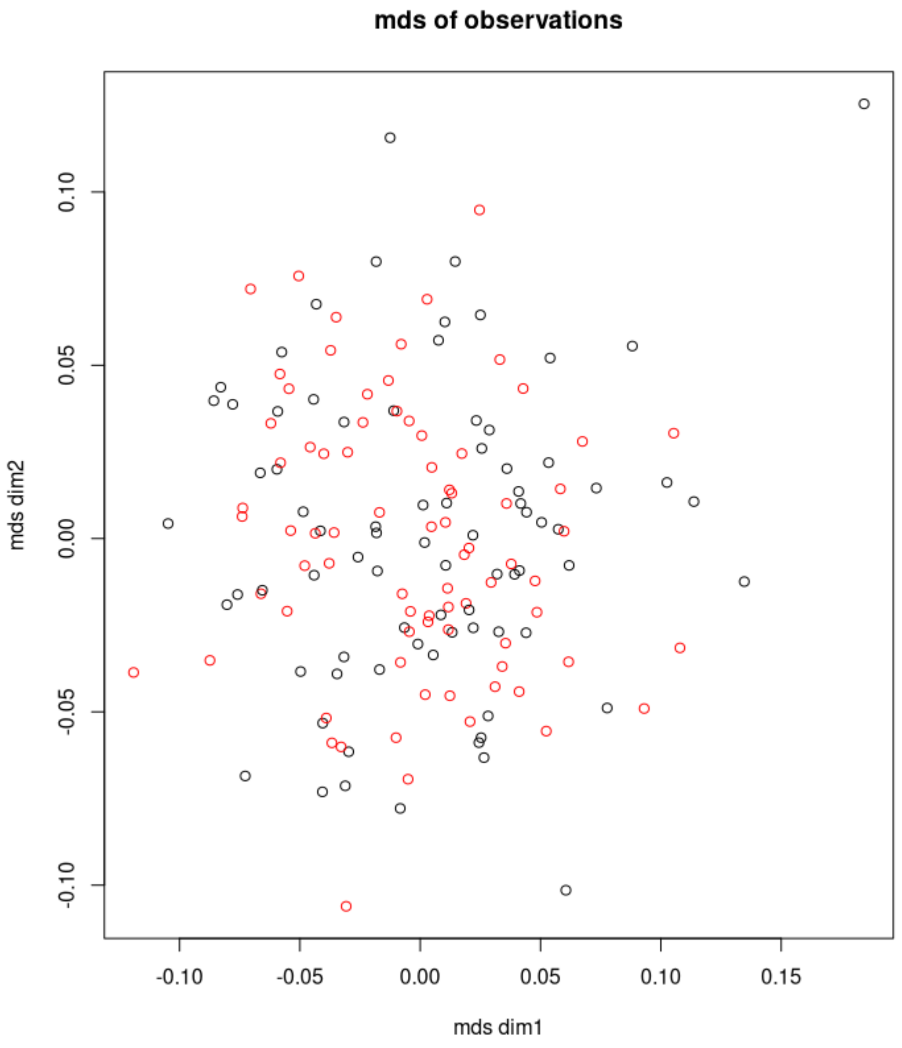
Either the data should be filtered entirely, or it should be left as-is. Manually removing individual subjects raises the question of how it ensures data integrity, especially since it would get more subjective as more data is cut. Establishing a filtering heuristic to remove multiple outliers is the best option in this case.

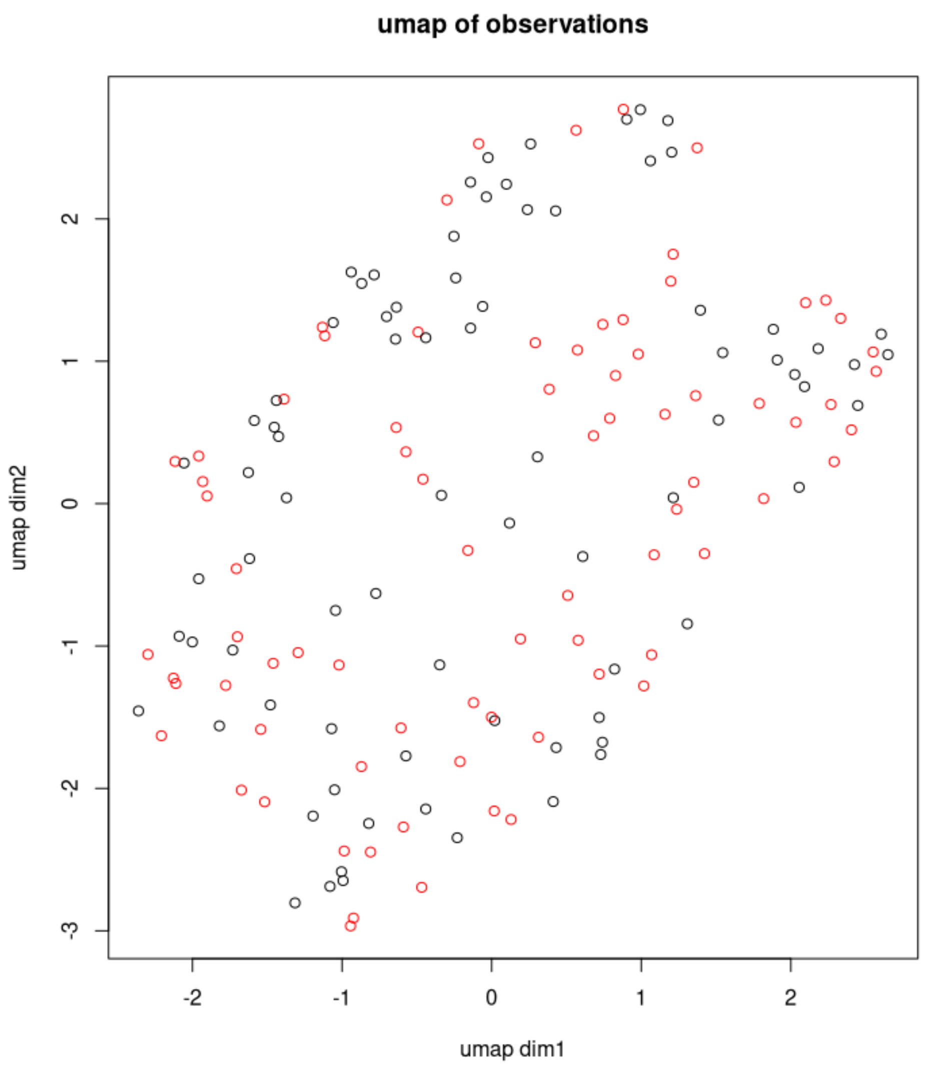
Optional 2. Compare MDS and UMAP clustering of observations. One thing to comment on is how they treat the potential outlier from optional 1.

```
obs_mds = cmdscale(d, k=2)
#add colors for MDD/HC
colors = rep("black",nrow(SxG.df))
colors[startsWith(rownames(SxG.df),"MDD")] <- "red"
plot(obs_mds, col=colors,
     main="mds of observations", xlab="mds dim1", ylab="mds dim2")

library(umap)
SxG.df <- data.frame(t(GxS.covfilter))
# change umap config parameters
custom.config = umap.defaults
custom.config$random_state = 123
custom.config$n_epochs = 500

obs_umap = umap(SxG.df, config=custom.config)
#add colors for MDD/HC
colors = rep("black",nrow(top_gene_data))
colors[startsWith(rownames(top_gene_data),"MDD")] <- "red"
plot(obs_umap$layout, col=colors,
     main="umap of observations", xlab="umap dim1", ylab="umap dim2")
```



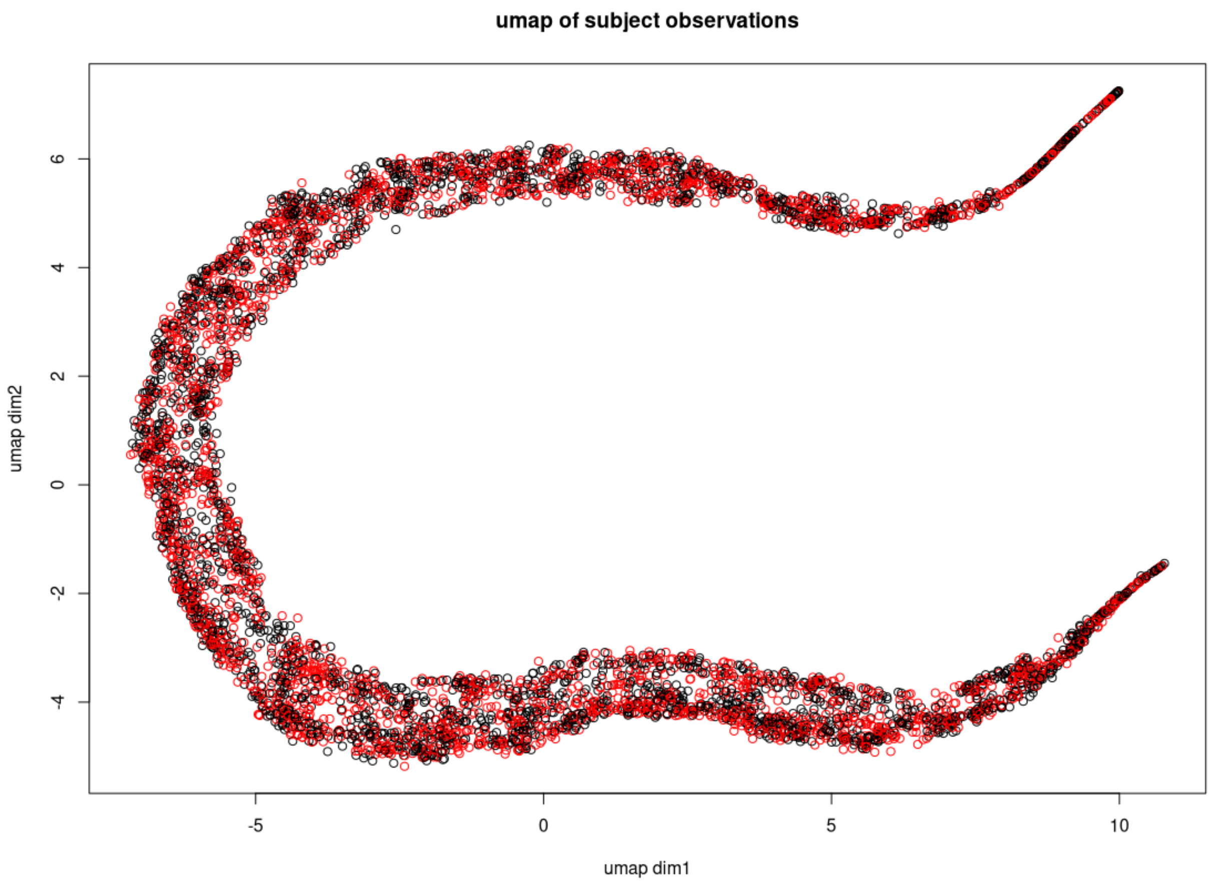


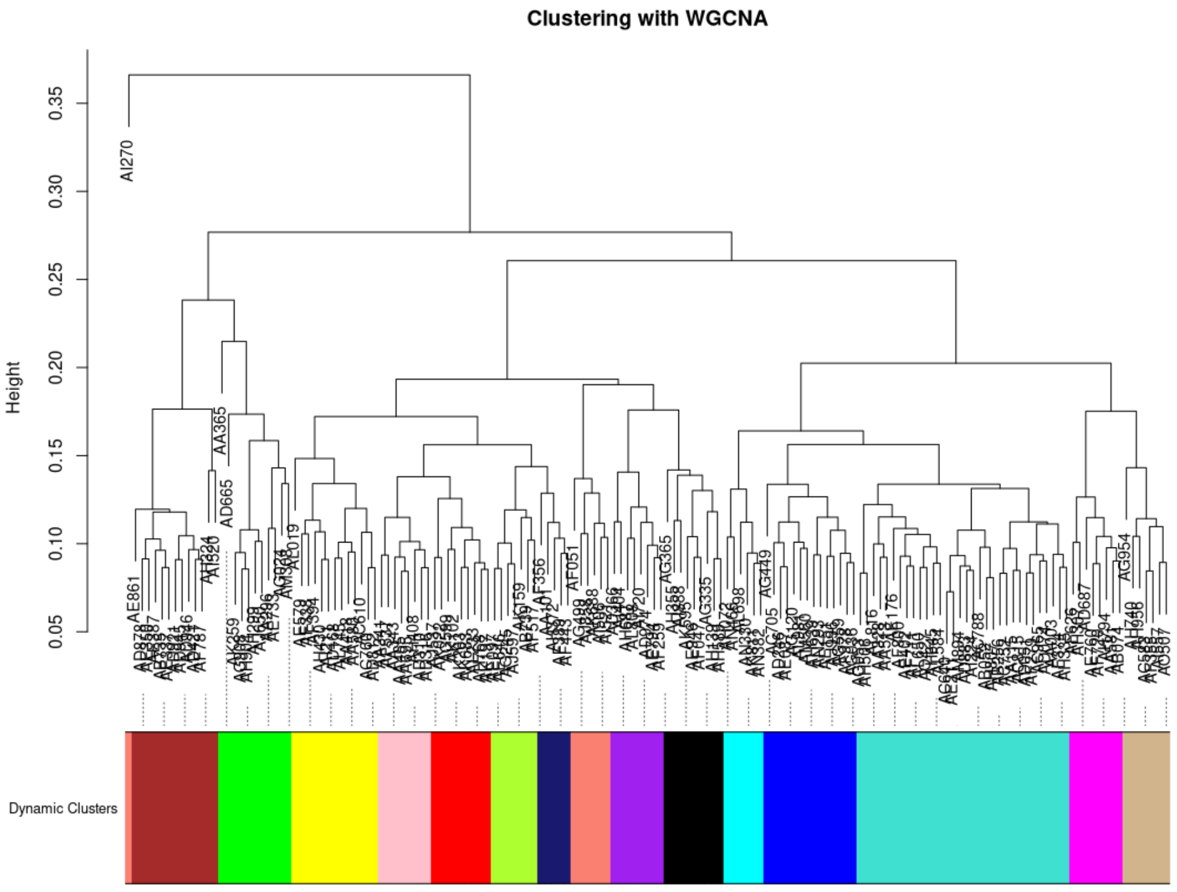
Comparing the UMAP and MDS clustering, there are noticeable visual differences. With MDS, most subjects (for the most part) are centrally located. The largest outlier identified in Optional 1, AI270, is very distant in comparison to the central location of most others. It is also fairly easy to spot the other potential outliers, such as the ones at roughly (0.00, 0.13), (0.14, -0.02), and (-0.03, -0.11). The mixing of MDD is and HC is fairly even, and apart from a few exceptions, it would not be easy to draw clusters based off MDS.

UMAP, on the other hand, has the subjects much more spread out. Rather than most being centrally located with a few outliers, it appears as though computing the average pairwise distance between all in the UMAP transformation would be much greater than that for MDS. It is much harder to identify the outliers since they are well incorporated. Though it would still be difficult, it is a bit easier to spot areas that can be clustered together. Since the results are more spread out, there is less overlap, and subjects can therefore be clustered together a bit easier.

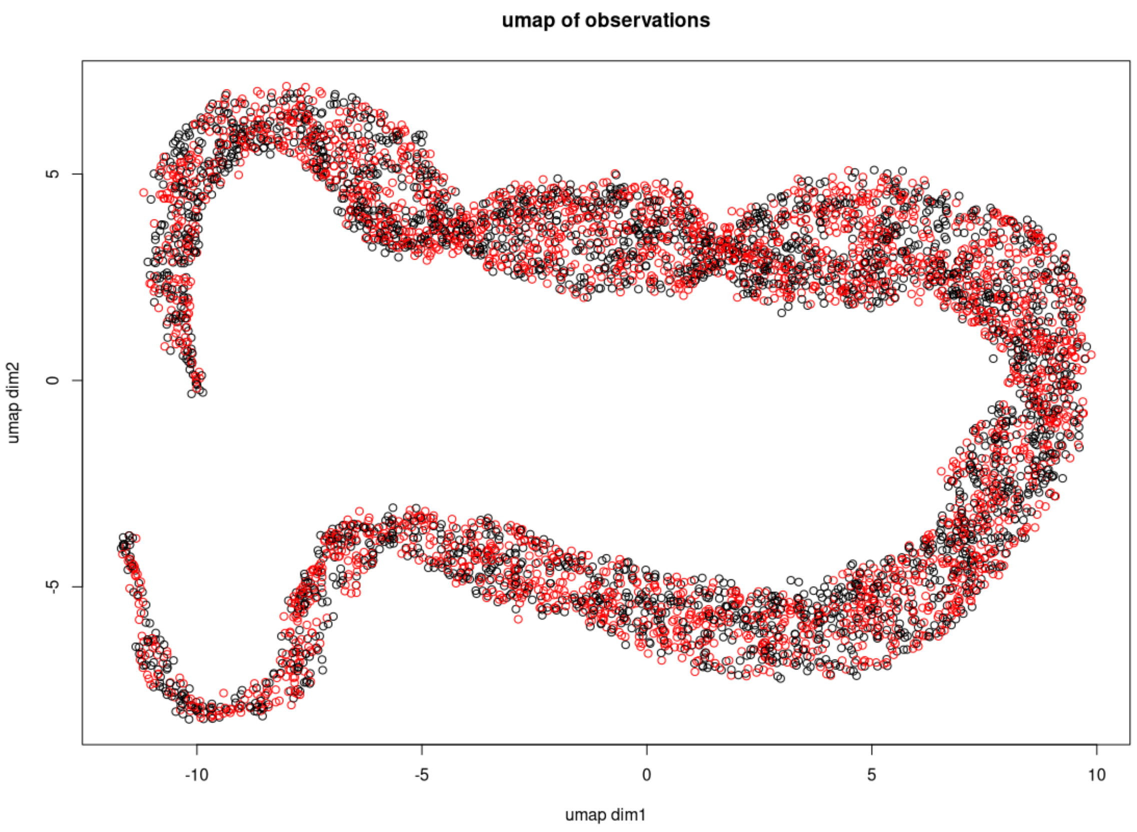
`Optional 3. You could try to cluster the genes (as opposed to subjects) using WGCNA and UMAP. UMAP will probably require tweaking parameters.`

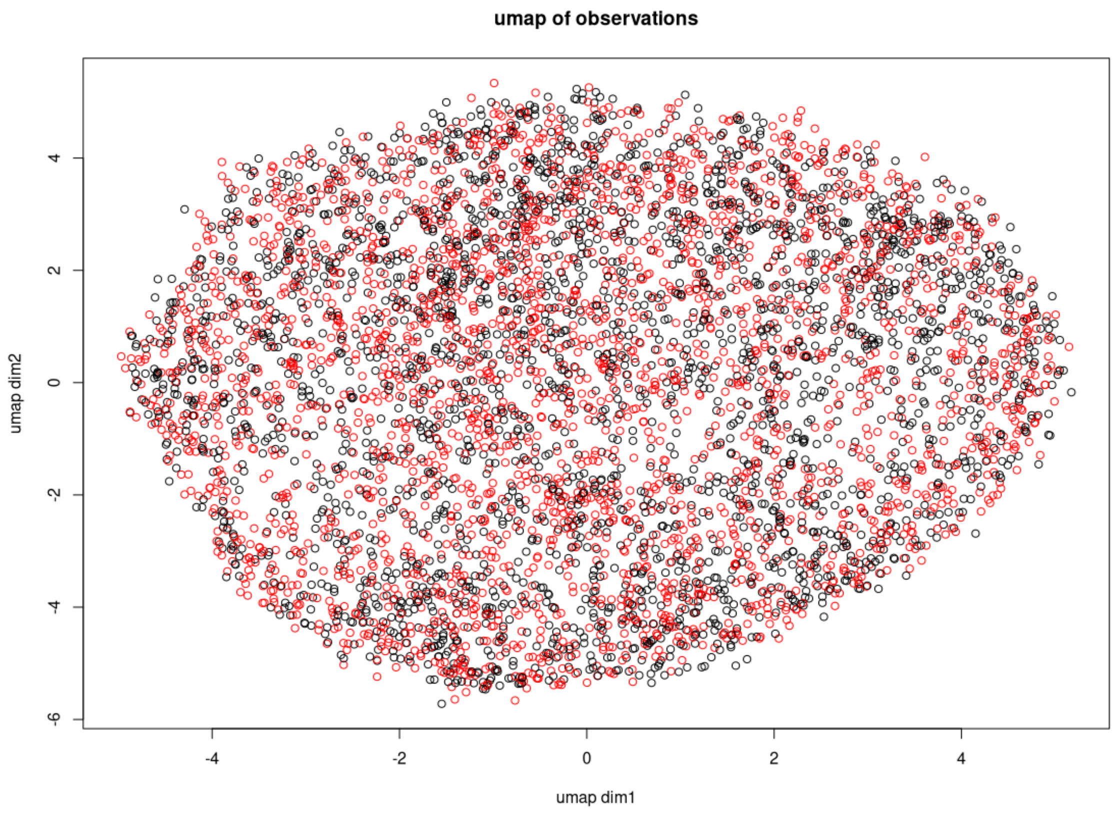
Mostly Default UMAP:

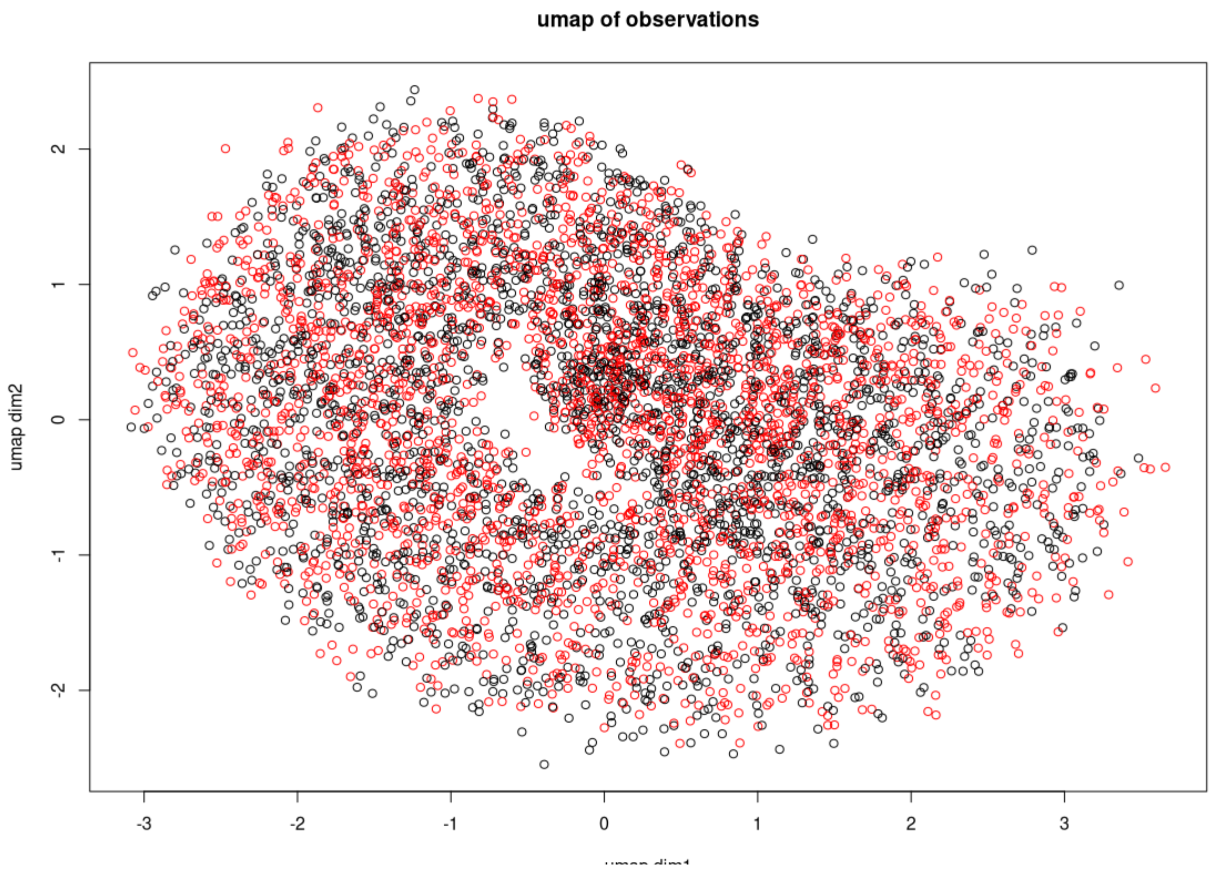


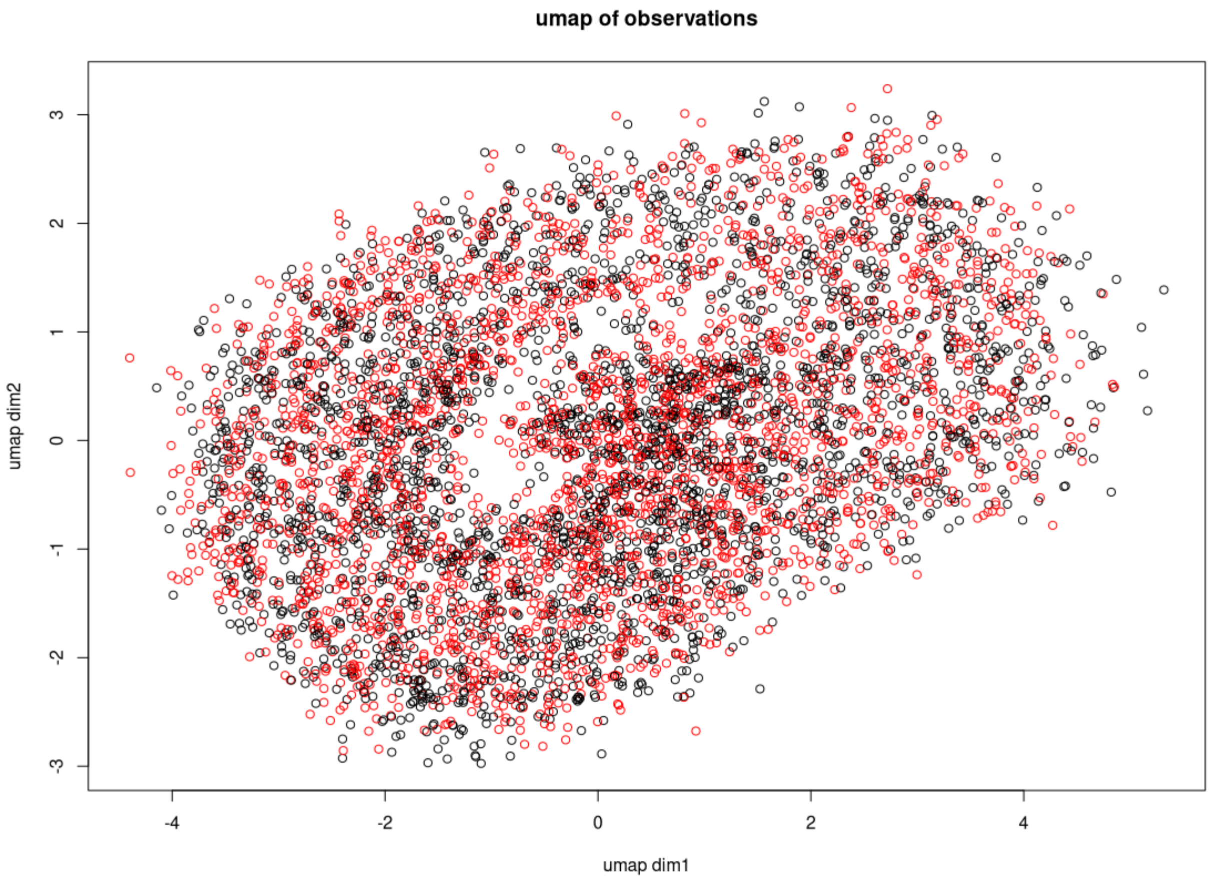


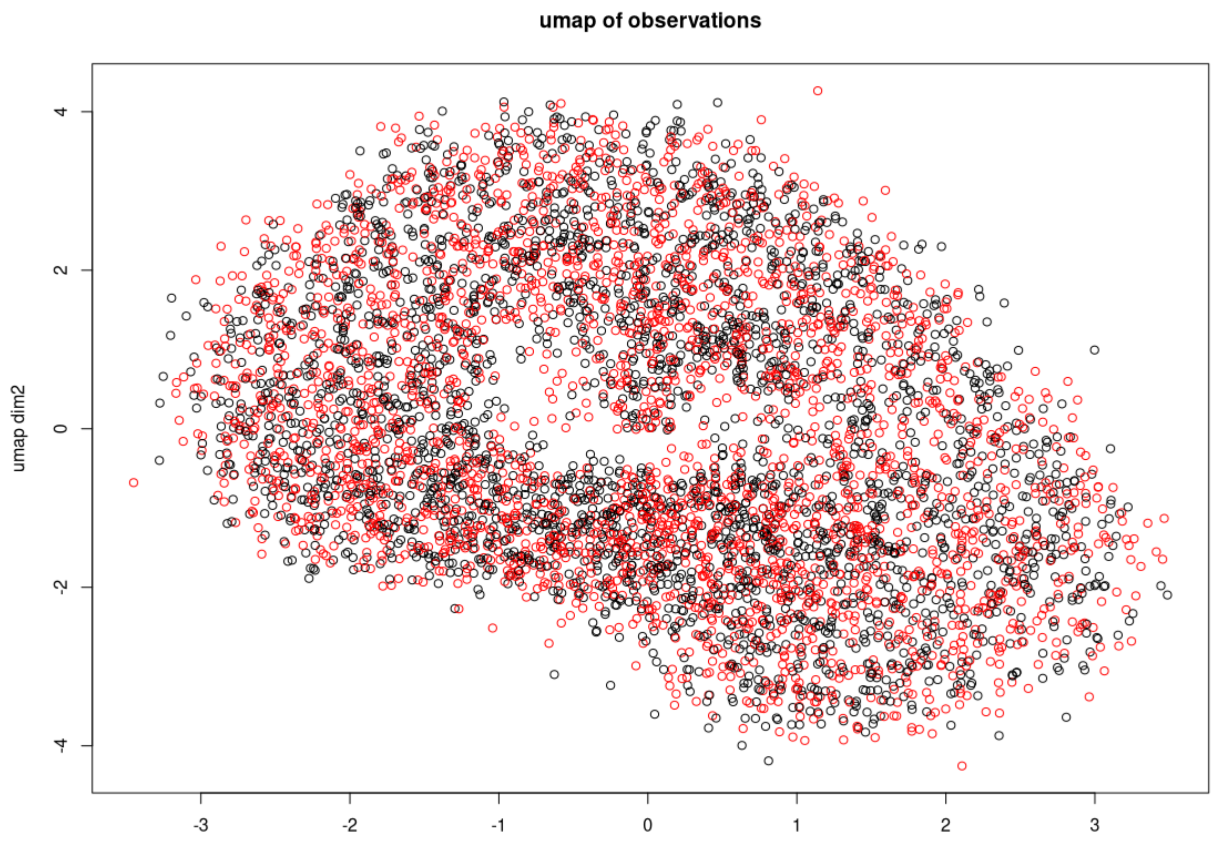
The UMAP defaults yielded results that were not particularly helpful. A number of config settings were altered, ranging from changing the metrics, to changing spread and min_dist, changing a and b, etc. In the end, too much time had been spent trying to get something promising. I have posted a few of the intermediate results below, and posted the "best" in my opinion at the end.

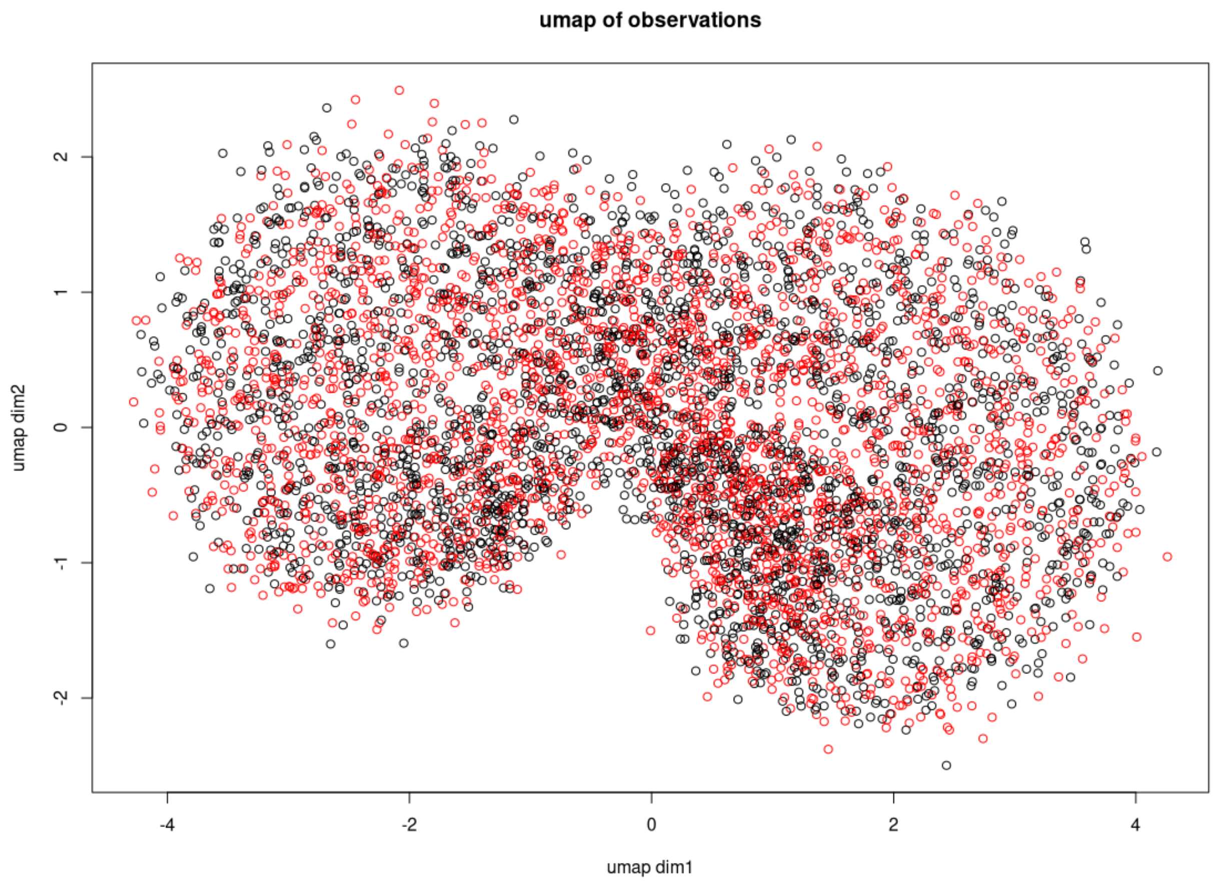

umap of observations
umap dim2
umap dim1
5
0
-5
-10
-5
0
5
10

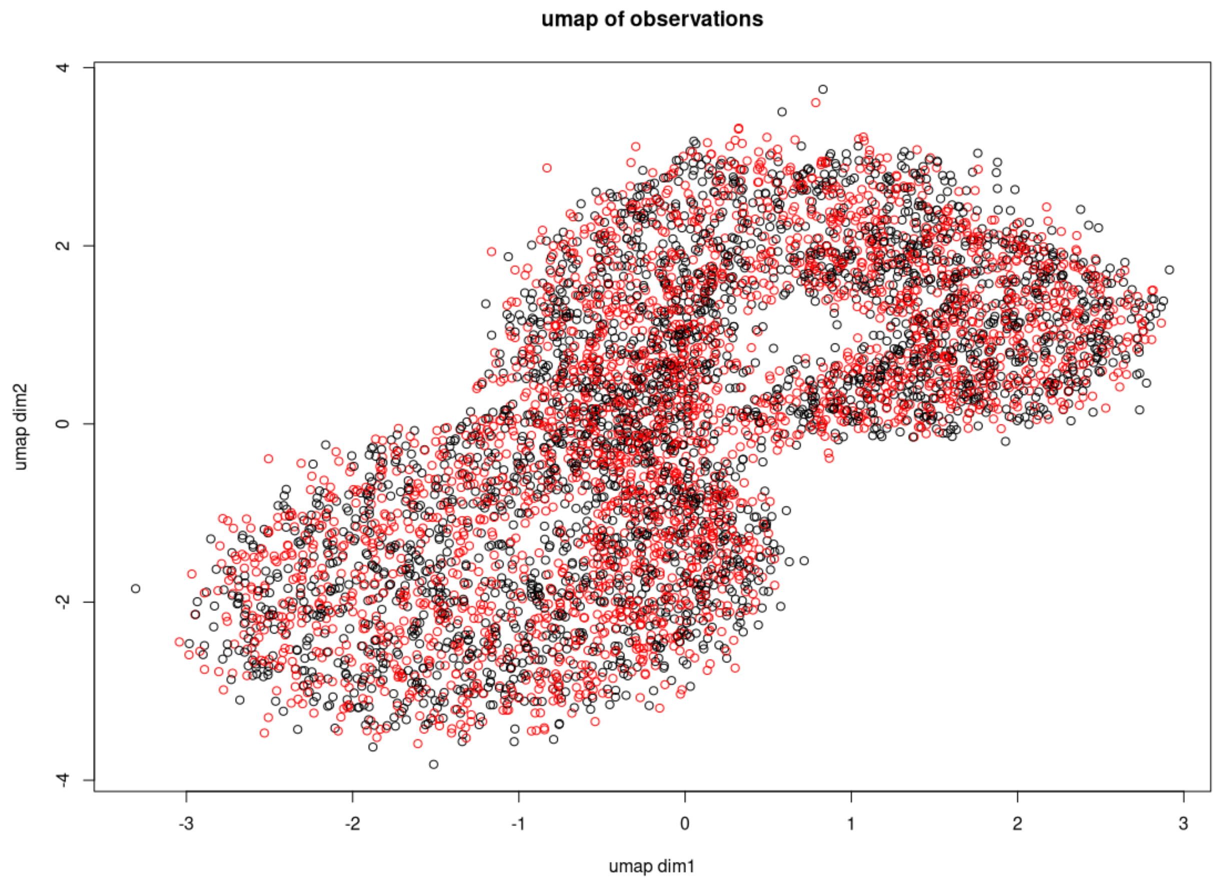

umap of observations
umap dim2
umap dim1
4
2
0
-2
-4
-6
-4
-2
0
2
4

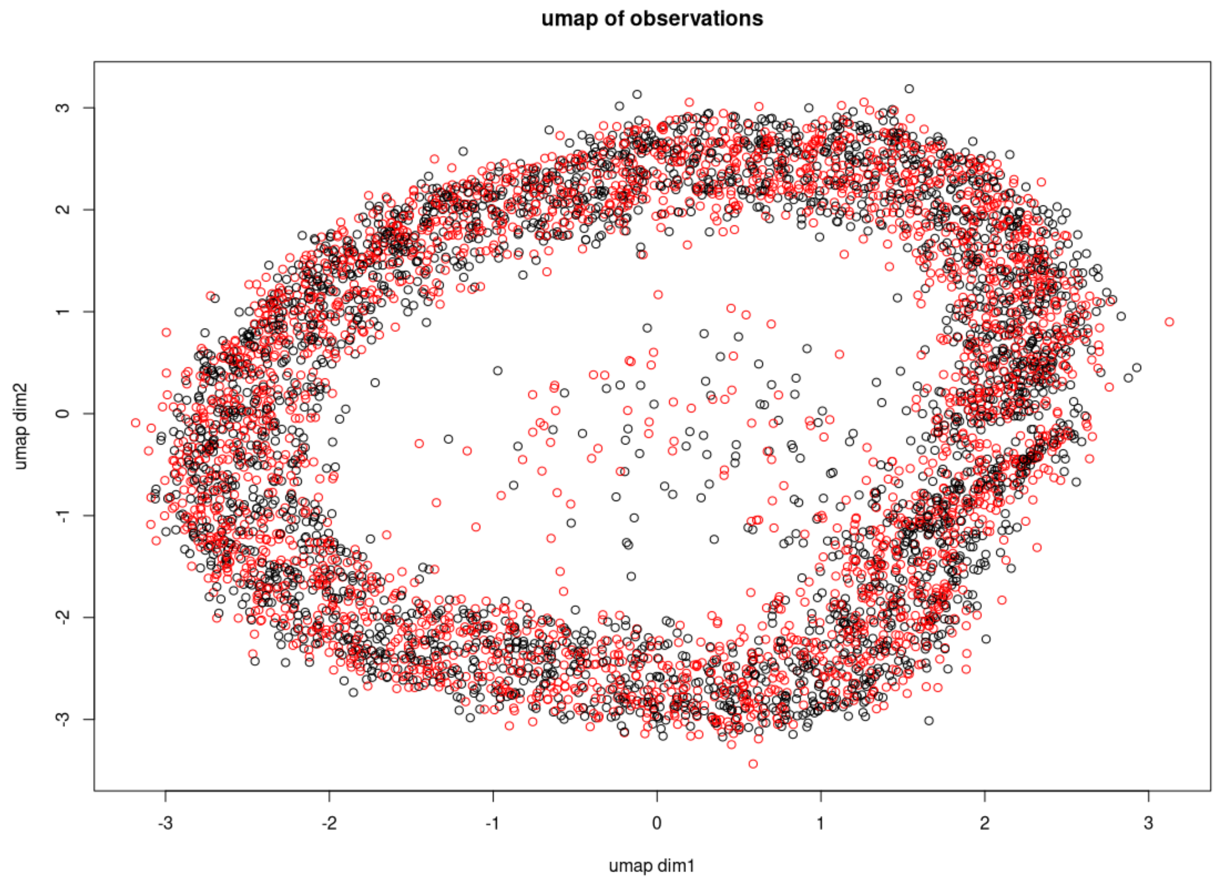

umap of observations
umap dim2
umap dim1
-3
-2
-1
0
1
2
3

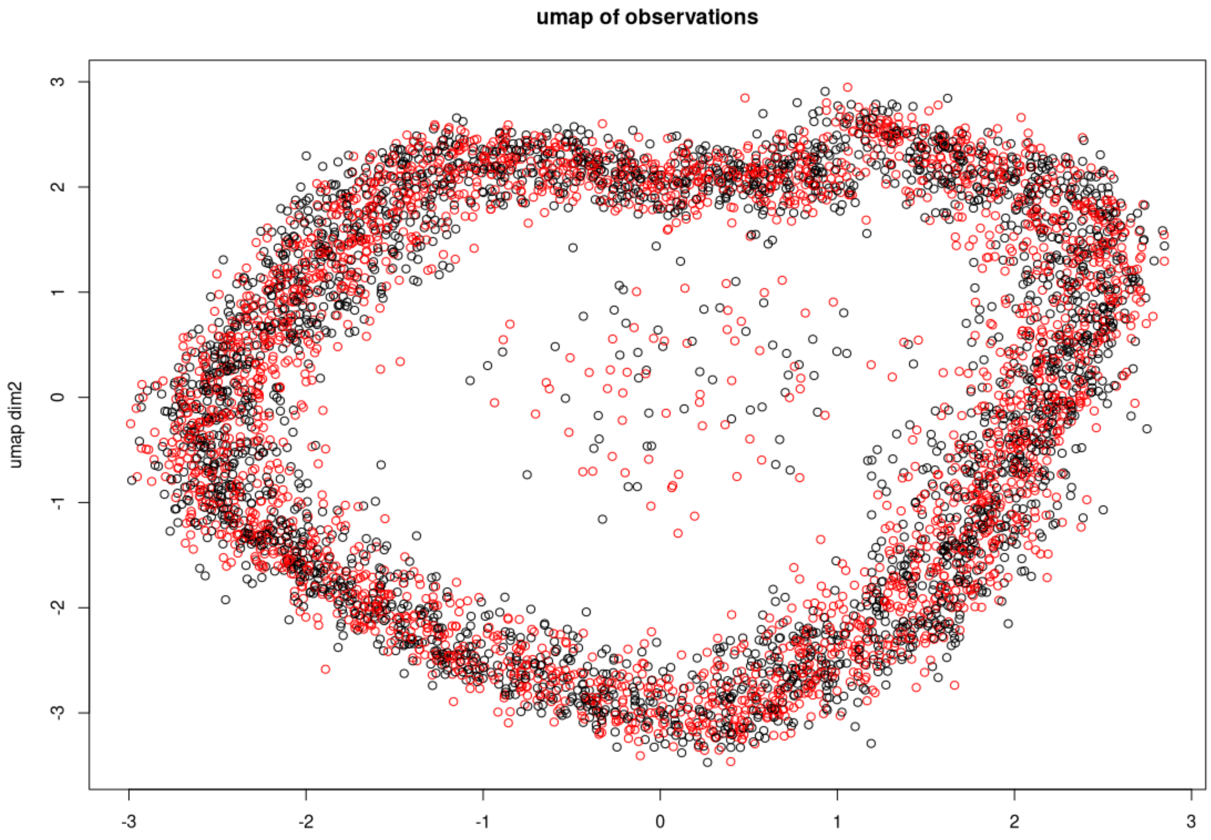

umap of observations
umap dim2
umap dim1
-4
-2
0
2
4
-3
-2
-1
0
1
2
3


umap of observations
umap dim2
4
2
0
-2
-4
-3
-2
-1
0
1
2
3

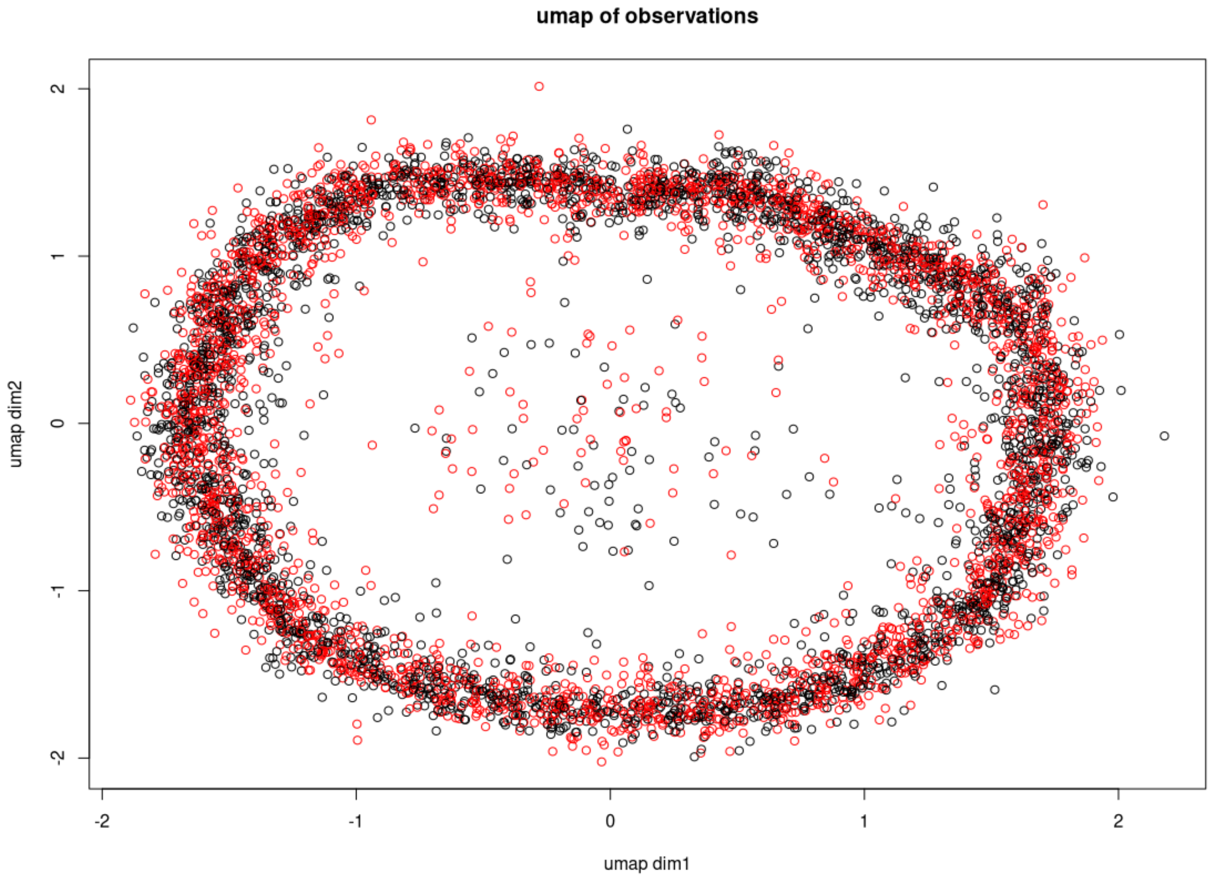

umap of observations
umap dim2
umap dim1
2
1
0
-1
-2
-4
-2
0
2
4


umap of observations
umap dim1
umap dim2


umap of observations
umap dim1
umap dim2



umap of observations
umap dim2
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3



umap of observations
umap dim2
umap dim1
2
1
0
-1
-2
-2
-1
0
1
2

Final config:

```
custom.config = umap.defaults
custom.config$random_state = 123
custom.config$n_epochs = 50
custom.config$n_neighbors=30
custom.config$metric = "pearson"
custom.config$input = "dist"
custom.config$a = 9.5
custom.config$b = 0.5
```

More work can be done to optimize this, but this was quickly becoming a time-consuming rabbit-hole so I decided to call it how it was.