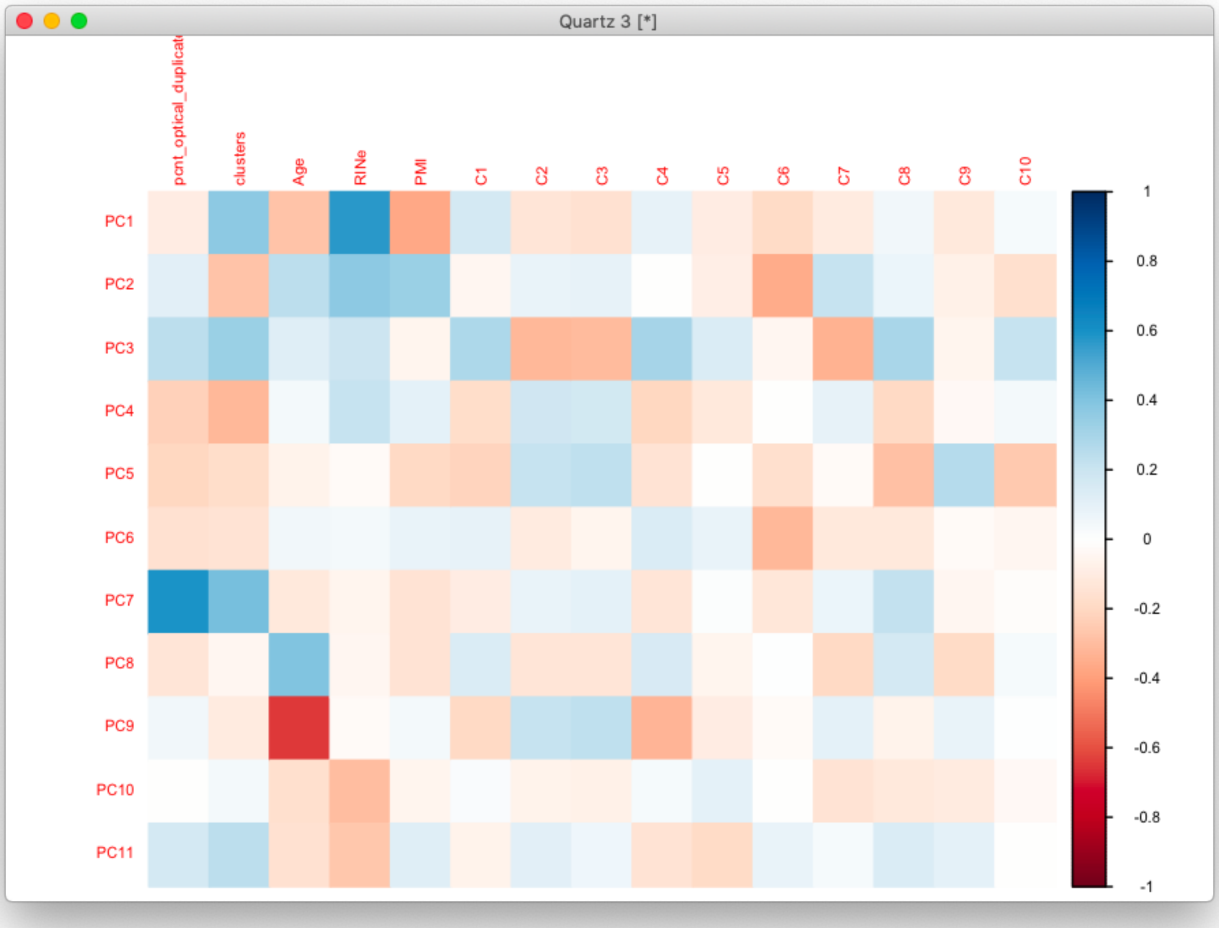
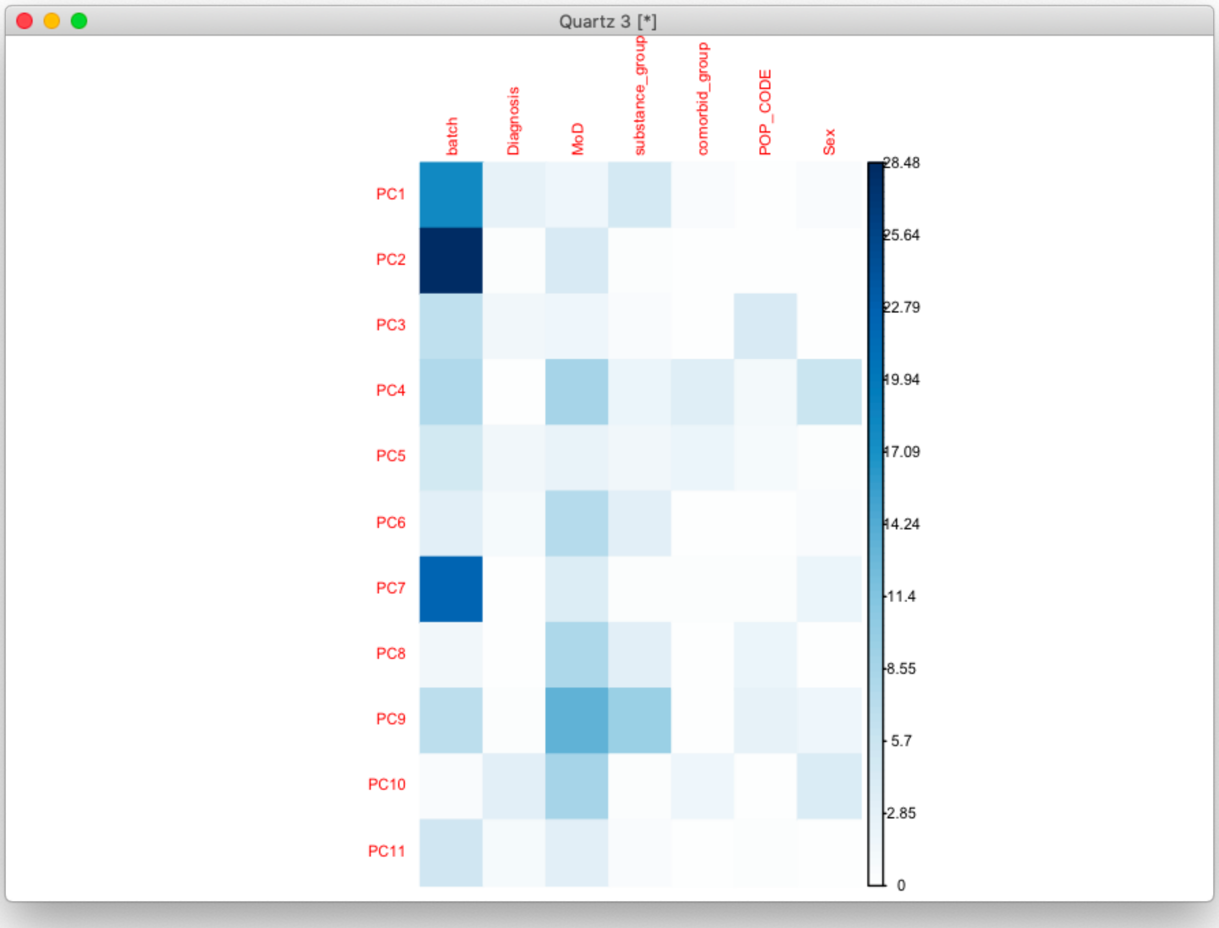


```
categ_vars = c('batch', 'Diagnosis', 'MoD', 'substance_group',
               'comorbid_group', 'POP_CODE', 'Sex')
categ_corrs = matrix(nrow=length(categ_vars), ncol=length(pc_vars),
                     dimnames=list(categ_vars, pc_vars))
categ_pvals = categ_corrs
for (x in categ_vars) {
    for (y in pc_vars) {
        res = kruskal.test(mydata[, y], data[, x])
        categ_corrs[x, y] = res$statistic
        categ_pvals[x, y] = res$p.value
    }
}
corrplot(t(categ_corrs), method='color', tl.cex=.5, cl.cex=.5, is.corr=F)
```



So, ideally here we'd keep only the PCs related to Diagnosis. But none of them are (even at nominal p < .05... closest is PC10 at .07). So the other option is to remove anything that captures the nuisance variables, say, at p<.05.

```
r$> which(categ_pvals < .05, arr.ind = T)
                row col
batch             1   1
batch             1   2
POP_CODE          6   3
batch             1   4
MoD               3   4
Sex               7   4
batch             1   7
MoD               3   8
MoD               3   9
substance_group   4   9
MoD               3  10
Sex               7  10
r$> which(categ_pvals < .01, arr.ind = T)
                row col
batch             1   1
batch             1   2
batch             1   7
```

```
MoD                3   9
substance_group    4   9
```

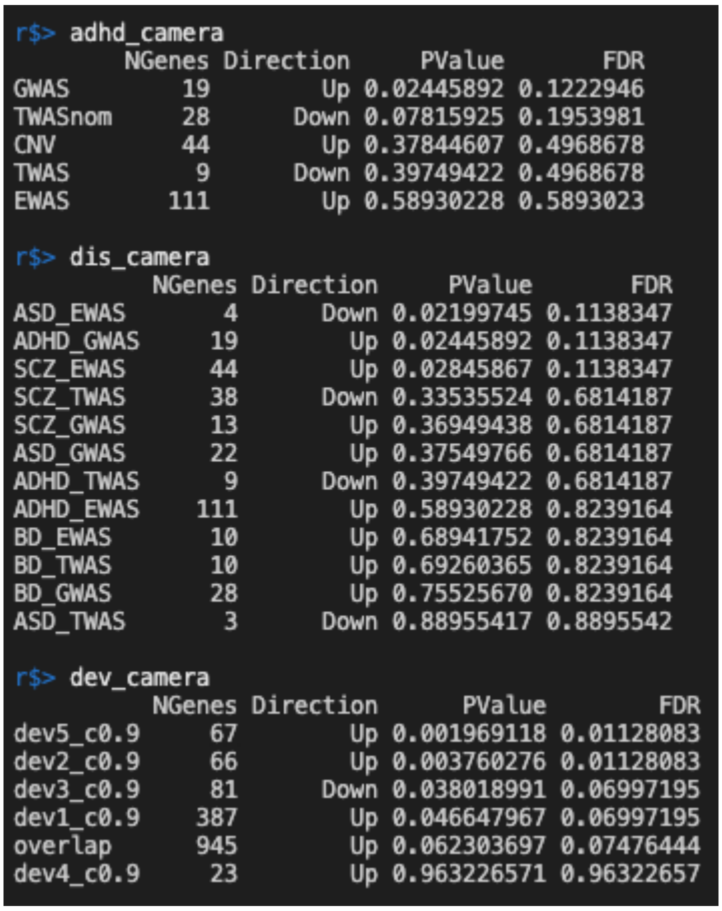
And just for the record, here's the same result using the numeric variables:

```
r$> which(num_pvals < .05, arr.ind = T)
                        row col
clusters                  2   1
Age                       3   1
RINe                      4   1
PMI                       5   1
clusters                  2   2
RINe                      4   2
PMI                       5   2
C6                       11   2
clusters                  2   3
C1                        6   3
C2                        7   3
C3                        8   3
C4                        9   3
C7                       12   3
C8                       13   3
clusters                  2   4
C8                       13   5
C6                       11   6
pcnt_optical_duplicates   1   7
clusters                  2   7
Age                       3   8
Age                       3   9
C4                        9   9
RINe                      4  10
RINe                      4  11

r$> which(num_pvals < .01, arr.ind = T)
                        row col
clusters                  2   1
RINe                      4   1
PMI                       5   1
RINe                      4   2
C6                       11   2
pcnt_optical_duplicates   1   7
clusters                  2   7
Age                       3   8
Age                       3   9
```

So, let's remove 1, 2, 7, 8 and 9. Interesting that we didn't really have anything related to the population PCs at p < .01 (only C6). We actually can't even do .05, because it would remove all 11 components.

Alright then. How do the results look?

```
dis_camera = get_enrich_order2( res, disorders)
dev_camera = get_enrich_order2( res, genes_unique )
```



```
r$> adhd_camera
        NGenes Direction     PValue       FDR
GWAS        19        Up 0.02445892 0.1222946
TWASnom     28      Down 0.07815925 0.1953981
CNV         44        Up 0.37844607 0.4968678
TWAS         9      Down 0.39749422 0.4968678
EWAS       111        Up 0.58930228 0.5893023

r$> dis_camera
          NGenes Direction     PValue       FDR
ASD_EWAS       4      Down 0.02199745 0.1138347
ADHD_GWAS     19        Up 0.02445892 0.1138347
SCZ_EWAS      44        Up 0.02845867 0.1138347
SCZ_TWAS      38      Down 0.33535524 0.6814187
SCZ_GWAS      13        Up 0.36949438 0.6814187
ASD_GWAS      22        Up 0.37549766 0.6814187
ADHD_TWAS      9      Down 0.39749422 0.6814187
ADHD_EWAS    111        Up 0.58930228 0.8239164
BD_EWAS       10        Up 0.68941752 0.8239164
BD_TWAS       10        Up 0.69260365 0.8239164
BD_GWAS       28        Up 0.75525670 0.8239164
ASD_TWAS       3      Down 0.88955417 0.8895542

r$> dev_camera
          NGenes Direction      PValue        FDR
dev5_c0.9     67        Up 0.001969118 0.01128083
dev2_c0.9     66        Up 0.003760276 0.01128083
dev3_c0.9     81      Down 0.038018991 0.06997195
dev1_c0.9    387        Up 0.046647967 0.06997195
overlap      945        Up 0.062303697 0.07476444
dev4_c0.9     23        Up 0.963226571 0.96322657
```

```
r$> rownames(c5_camera)[c5_camera$FDR < .05]
 [1] "GO_INTRINSIC_COMPONENT_OF_POSTSYNAPTIC_MEMBRANE"
 [2] "GO_INTRINSIC_COMPONENT_OF_SYNAPTIC_MEMBRANE"
 [3] "GO_NEUROTRANSMITTER_RECEPTOR_ACTIVITY"
 [4] "GO_CYTOSOLIC_RIBOSOME"
 [5] "GO_INTRINSIC_COMPONENT_OF_POSTSYNAPTIC_SPECIALIZATION_MEMBRANE"
 [6] "GO_SYNAPSE_ASSEMBLY"
 [7] "GO_IGG_BINDING"
 [8] "GO_GABA_ERGIC_SYNAPSE"
 [9] "GO_POSTSYNAPTIC_MEMBRANE"
[10] "GO_CYTOSOLIC_SMALL_RIBOSOMAL_SUBUNIT"
[11] "GO_G_PROTEIN_COUPLED_AMINE_RECEPTOR_ACTIVITY"
[12] "GO_REGULATION_OF_SYNAPTIC_PLASTICITY"
[13] "GO_IMMUNOGLOBULIN_BINDING"
[14] "GO_INTRINSIC_COMPONENT_OF_POSTSYNAPTIC_DENSITY_MEMBRANE"
[15] "GO_SYNAPTIC_MEMBRANE"
[16] "GO_NEURON_SPINE"
[17] "GO_SEROTONIN_RECEPTOR_SIGNALING_PATHWAY"
[18] "GO_REGULATION_OF_POSTSYNAPTIC_MEMBRANE_POTENTIAL"
[19]
"GO_SIGNAL_TRANSDUCTION_INVOLVED_IN_CELLULAR_RESPONSE_TO_AMMONIUM_ION"
[20] "GO_COTRANSLATIONAL_PROTEIN_TARGETING_TO_MEMBRANE"
[21] "GO_SYNAPTIC_SIGNALING"
[22] "GO_POSTSYNAPTIC_SPECIALIZATION_MEMBRANE"
```

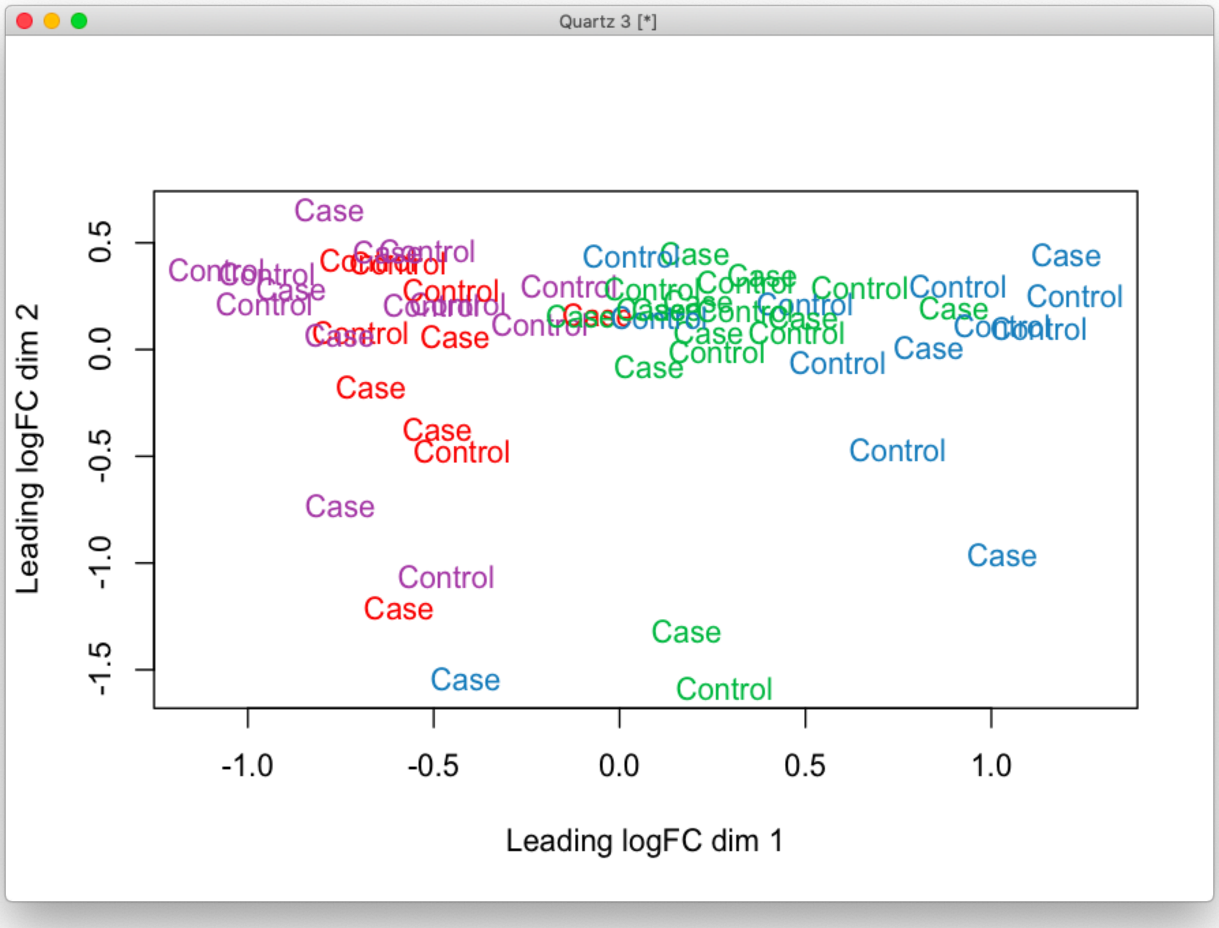
```
 [23] "GO_REGULATION_OF_SYNAPSE_ASSEMBLY"
 [24] "GO_REGULATION_OF_SYNAPSE_STRUCTURE_OR_ACTIVITY"
 [25] "GO_G_PROTEIN_COUPLED_NEUROTRANSMITTER_RECEPTOR_ACTIVITY"
```

This is looking promising.

2020-10-06 06:02:15

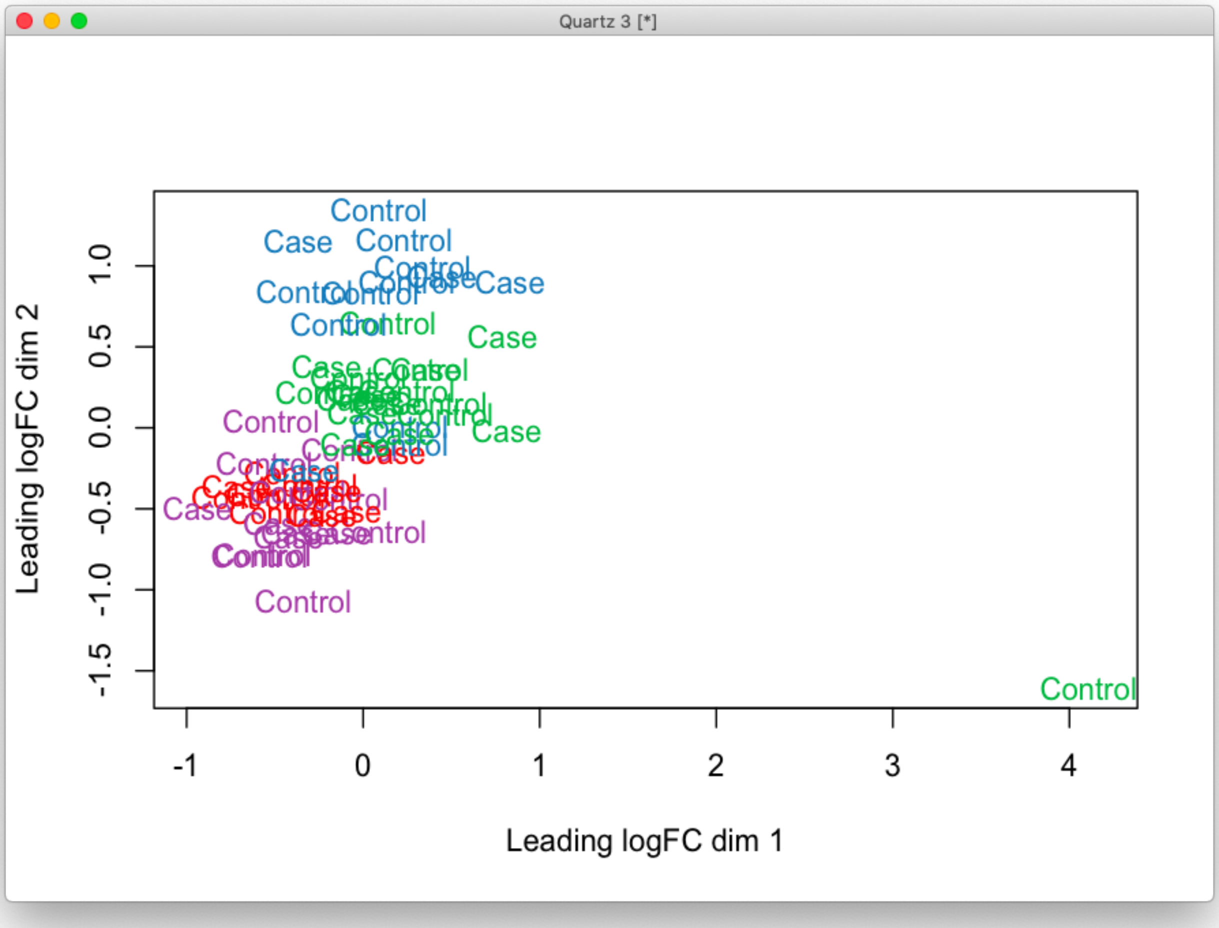
Before I get started on Caudate, let's make sure we don't have any more outliers here.

```
lcpm <- cpm(genes, log=TRUE)
library(RColorBrewer)
col.group <- data$batch
levels(col.group) <-  brewer.pal(nlevels(col.group), "Set1")
col.group <- as.character(col.group)
plotMDS(lcpm, labels=data$Diagnosis, col=col.group)
```



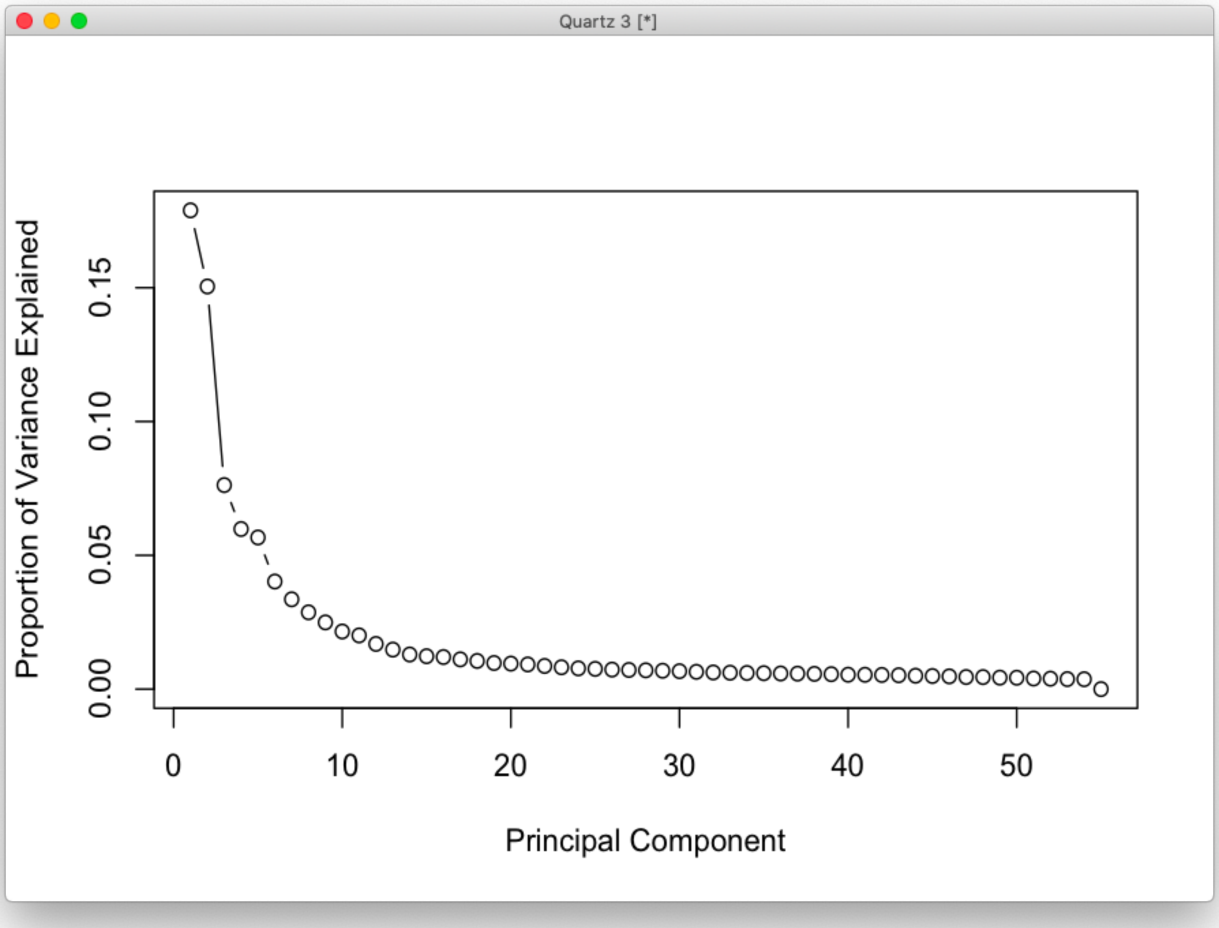
Colors are batches, and there isn't really an outlier. At least, not like when compared to this:

```
                       G_list2$chromosome_name != 'MT')
geneCounts = geneCounts[imautosome, ]
G_list2 = G_list2[imautosome, ]
library(edgeR)
isexpr <- filterByExpr(geneCounts, group=data$Diagnosis)
genes = DGEList( geneCounts[isexpr,], genes=G_list2[isexpr,] )
genes = calcNormFactors( genes)

lcpm <- cpm(genes, log=TRUE)
library(RColorBrewer)
col.group <- data$batch
levels(col.group) <-  brewer.pal(nlevels(col.group), "Set1")
col.group <- as.character(col.group)
plotMDS(lcpm, labels=data$Diagnosis, col=col.group)
```



And just for sanity, let's plot our PCA results (note that these are still results on the clean set pecause I never calculated PCA on this set with the outlier).

```
std_dev <- lcpm.pca$sdev
pr_var <- std_dev^2
prop_varex <- pr_var/sum(pr_var)
plot(prop_varex, xlab = "Principal Component",
             ylab = "Proportion of Variance Explained",
             type = "b")
```

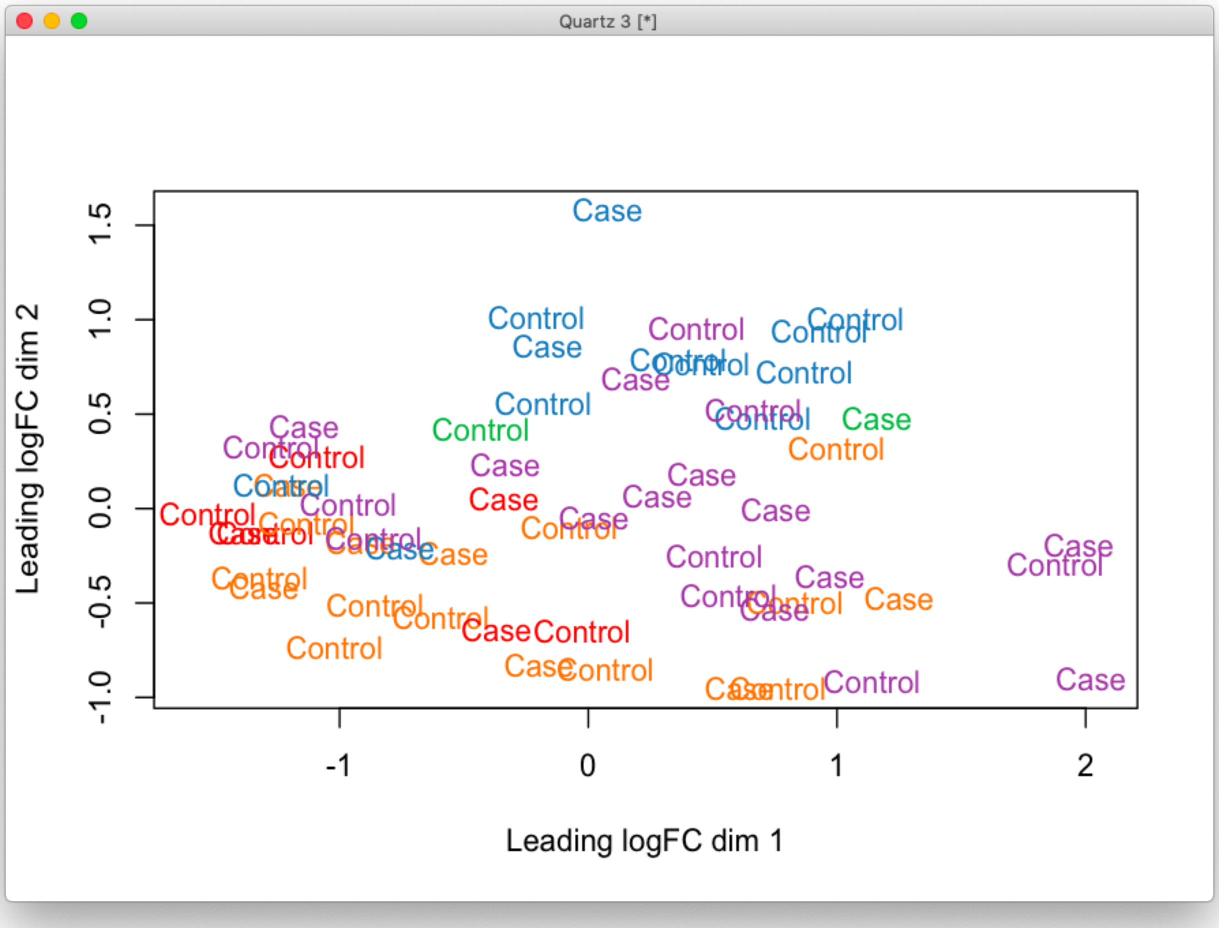


So, with 11 PCs we are at about the asymptote, which is fine. Just remember that we are not using just those 11 in the analysis! We just did that to restrict the data where we look for noise!

Caudate

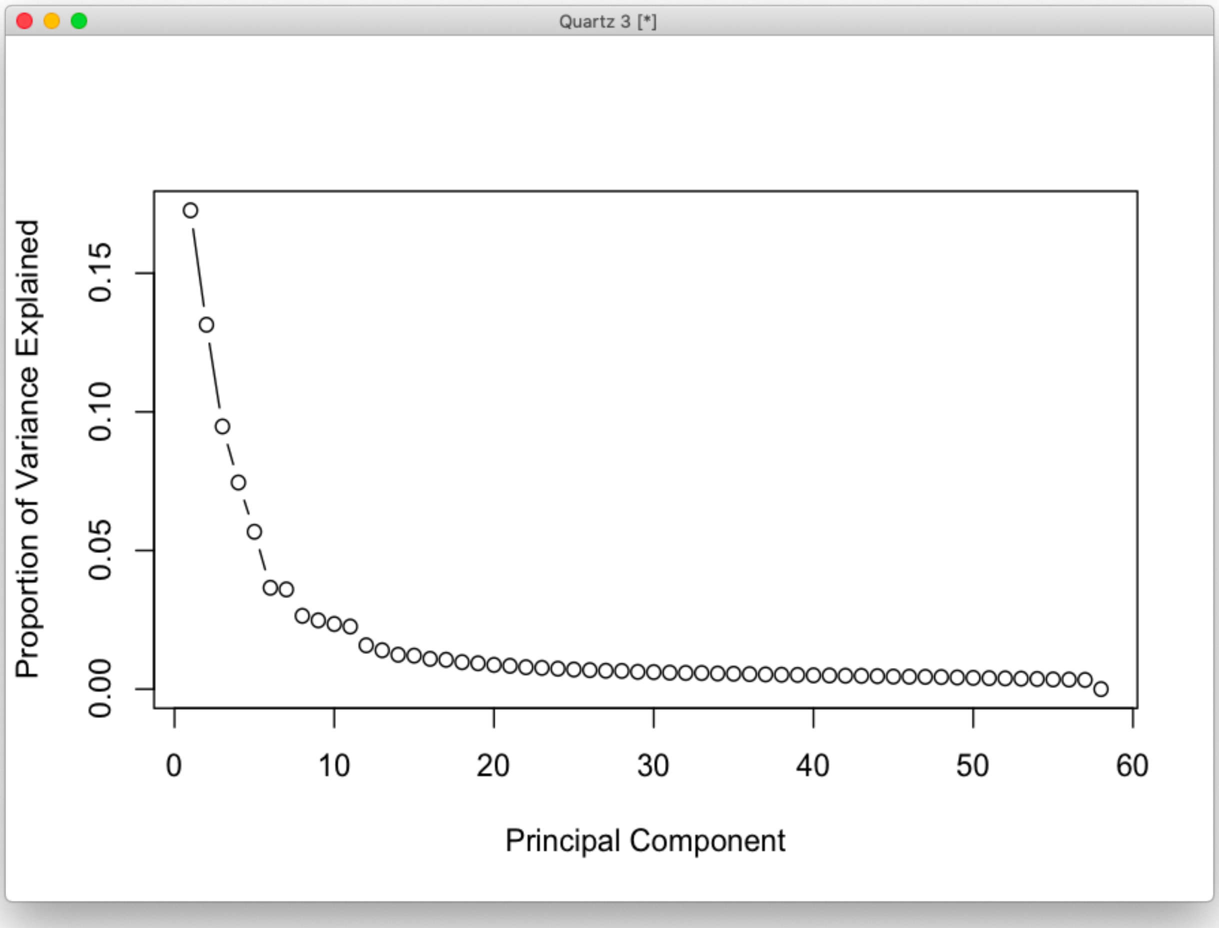
Let's run the same analysis, but now for Caudate:

```
myregion = 'Caudate'
data = readRDS('~/data/rnaseq_derek/complete_rawCountData_05132020.rds')
rownames(data) = data$submitted_name  # just to ensure compatibility later
data = data[data$Region==myregion, ]
more = readRDS('~/data/rnaseq_derek/data_from_philip_POP_and_PCs.rds')
more = more[!duplicated(more$hbcc_brain_id),]
data = merge(data, more[, c('hbcc_brain_id', 'comorbid', 'comorbid_group',
                            'substance', 'substance_group')],
             by='hbcc_brain_id', all.x=T, all.y=F)
grex_vars = colnames(data)[grepl(colnames(data), pattern='^ENS')]
count_matrix = t(data[, grex_vars])
data = data[, !grepl(colnames(data), pattern='^ENS')]
id_num = sapply(grex_vars, function(x) strsplit(x=x, split='\\.')[[1]][1])
```



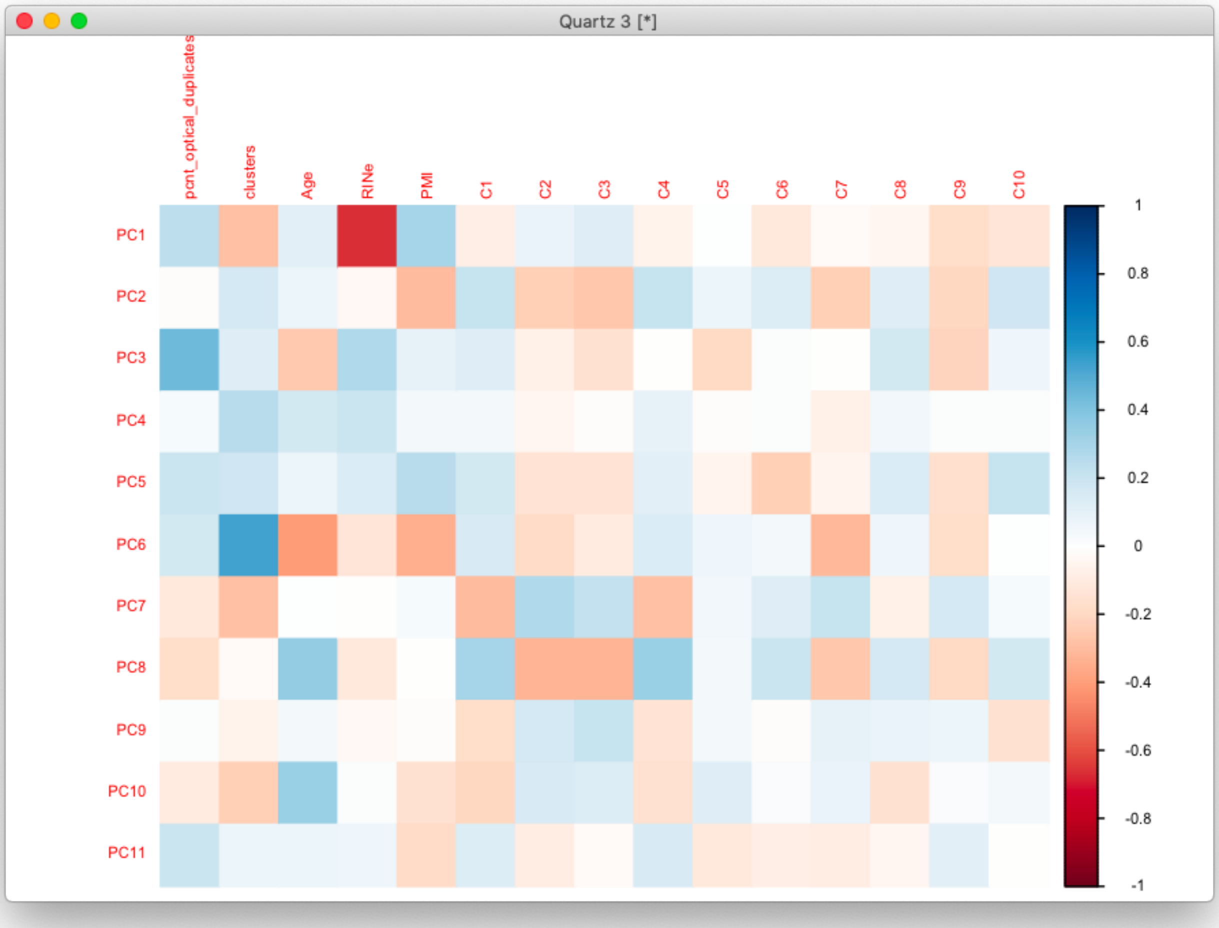
No outliers here, so we can just proceed with the PCA analysis.

```
set.seed(42)
lcpm.pca <- prcomp(t(lcpm), scale=TRUE)
library(nFactors)
eigs <- lcpm.pca$sdev^2
nS = nScree(x=eigs)
keep_me = 1:nS$Components$nkaiser
mydata = data.frame(lcpm.pca$x[, keep_me])

std_dev <- lcpm.pca$sdev
pr_var <- std_dev^2
prop_varex <- pr_var/sum(pr_var)
plot(prop_varex, xlab = "Principal Component",
             ylab = "Proportion of Variance Explained",
             type = "b")
```

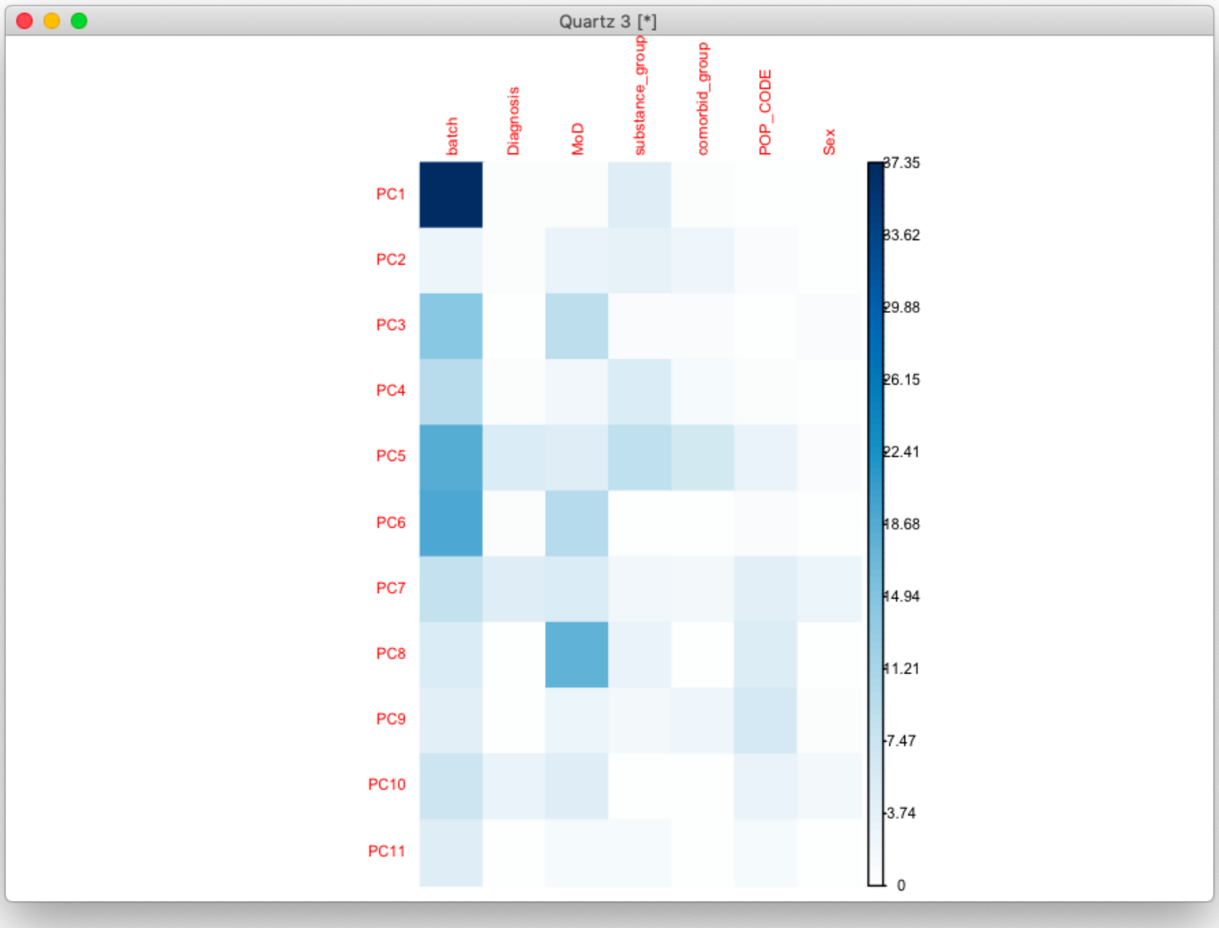


Kaiser selects 11 as well, which makes sense in the variance explained plot too.

```
num_vars = c('pcnt_optical_duplicates', 'clusters', 'Age', 'RINe', 'PMI',
             'C1', 'C2', 'C3', 'C4', 'C5', 'C6', 'C7', 'C8', 'C9', 'C10')
pc_vars = colnames(mydata)
num_corrs = matrix(nrow=length(num_vars), ncol=length(pc_vars),
                   dimnames=list(num_vars, pc_vars))
num_pvals = num_corrs
for (x in num_vars) {
    for (y in pc_vars) {
        res = cor.test(data[, x], mydata[, y])
        num_corrs[x, y] = res$estimate
        num_pvals[x, y] = res$p.value
    }
}

library(corrplot)
corrplot(t(num_corrs), method='color', tl.cex=.5, cl.cex=.5)
```



```
categ_vars = c('batch', 'Diagnosis', 'MoD', 'substance_group',
               'comorbid_group', 'POP_CODE', 'Sex')
categ_corrs = matrix(nrow=length(categ_vars), ncol=length(pc_vars),
                     dimnames=list(categ_vars, pc_vars))
categ_pvals = categ_corrs
for (x in categ_vars) {
    for (y in pc_vars) {
        res = kruskal.test(mydata[, y], data[, x])
        categ_corrs[x, y] = res$statistic
        categ_pvals[x, y] = res$p.value
    }
}
corrplot(t(categ_corrs), method='color', tl.cex=.5, cl.cex=.5, is.corr=F)
```

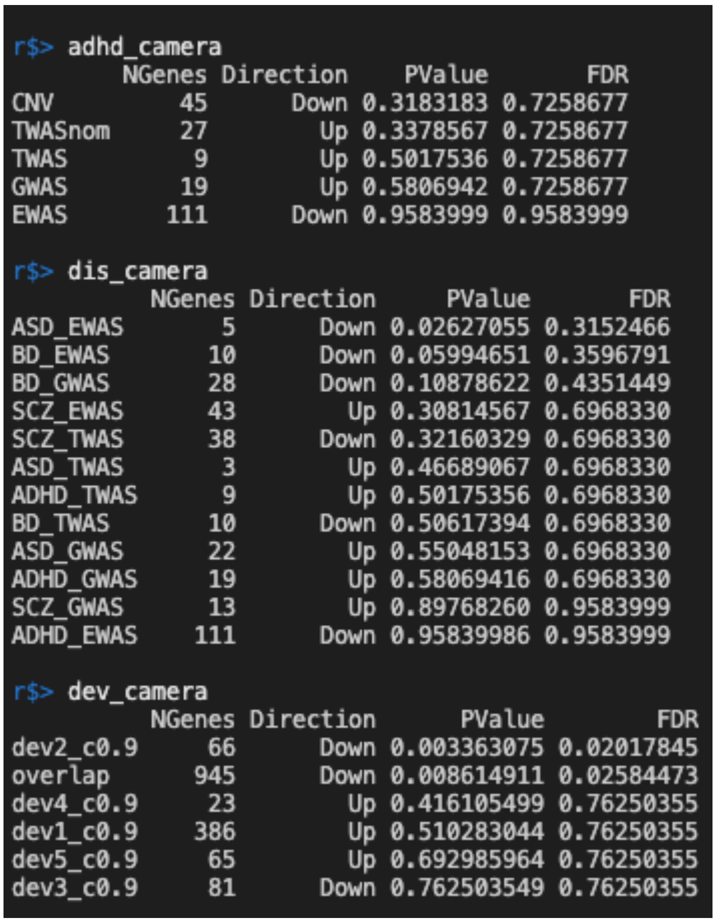


```
r$> which(num_pvals < .01, arr.ind = T)
                         row col
RINe                       4   1
pcnt_optical_duplicates    1   3
clusters                   2   6
Age                        3   6
PMI                        5   6
Age                        3   8

r$> which(categ_pvals < .01, arr.ind = T)
      row col
batch   1   1
batch   1   3
batch   1   5
batch   1   6
MoD     3   8
```

So, for Caudate we will include PCs 1, 3, 5, 6, and 8.

```
data2 = cbind(data, mydata)
form = ~ Diagnosis + PC1 + PC3 + PC5 + PC6 + PC8
design = model.matrix( form, data2)
```

```
vobj = voom( genes, design, plot=FALSE)
fit <- lmFit(vobj, design)
fit2 <- eBayes( fit )
res = topTable(fit2, coef='DiagnosisControl', number=Inf)

adhd_camera = get_enrich_order2( res, t2 )
c5_camera = get_enrich_order2( res, c5_all)
dis_camera = get_enrich_order2( res, disorders)
dev_camera = get_enrich_order2( res, genes_unique )
```



```
r$> adhd_camera
        NGenes Direction    PValue       FDR
CNV         45      Down 0.3183183 0.7258677
TWASnom     27        Up 0.3378567 0.7258677
TWAS         9        Up 0.5017536 0.7258677
GWAS        19        Up 0.5806942 0.7258677
EWAS       111      Down 0.9583999 0.9583999

r$> dis_camera
          NGenes Direction     PValue       FDR
ASD_EWAS       5      Down 0.02627055 0.3152466
BD_EWAS       10      Down 0.05994651 0.3596791
BD_GWAS       28      Down 0.10878622 0.4351449
SCZ_EWAS      43        Up 0.30814567 0.6968330
SCZ_TWAS      38      Down 0.32160329 0.6968330
ASD_TWAS       3        Up 0.46689067 0.6968330
ADHD_TWAS      9        Up 0.50175356 0.6968330
BD_TWAS       10      Down 0.50617394 0.6968330
ASD_GWAS      22        Up 0.55048153 0.6968330
ADHD_GWAS     19        Up 0.58069416 0.6968330
SCZ_GWAS      13        Up 0.89768260 0.9583999
ADHD_EWAS    111      Down 0.95839986 0.9583999

r$> dev_camera
          NGenes Direction      PValue        FDR
dev2_c0.9     66      Down 0.003363075 0.02017845
overlap      945      Down 0.008614911 0.02584473
dev4_c0.9     23        Up 0.416105499 0.76250355
dev1_c0.9    386        Up 0.510283044 0.76250355
dev5_c0.9     65        Up 0.692985964 0.76250355
dev3_c0.9     81      Down 0.762503549 0.76250355
```

```
 [1] "GO_CILIUM_MOVEMENT"
 [2] "GO_AXONEME_ASSEMBLY"
 [3] "GO_INTRACILIARY_TRANSPORT"
 [4] "GO_CILIUM_ORGANIZATION"
 [5] "GO_CILIARY_PLASM"
 [6] "GO_CILIARY_TIP"
 [7] "GO_INTRACILIARY_TRANSPORT_PARTICLE"
 [8] "GO_CILIUM_OR_FLAGELLUM_DEPENDENT_CELL_MOTILITY"
 [9] "GO_INTRACILIARY_TRANSPORT_INVOLVED_IN_CILIUM_ASSEMBLY"
[10] "GO_MICROTUBULE_BUNDLE_FORMATION"
[11] "GO_CILIUM"
[12] "GO_CILIARY_BASAL_BODY"
[13] "GO_SPERM_MOTILITY"
[14] "GO_MOTILE_CILIUM"
[15] "GO_PROTEIN_TRANSPORT_ALONG_MICROTUBULE"
[16] "GO_CILIUM_MOVEMENT_INVOLVED_IN_CELL_MOTILITY"
```

```
[17] "GO_AXONEMAL_DYNEIN_COMPLEX_ASSEMBLY"
[18] "GO_BITTER_TASTE_RECEPTOR_ACTIVITY"
[19] "GO_INTRACILIARY_TRANSPORT_PARTICLE_B"
[20] "GO_CILIARY_BASAL_BODY_PLASMA_MEMBRANE_DOCKING"
[21] "GO_EXTRACELLULAR_TRANSPORT"
[22] "GO_RESPONSE_TO_TYPE_I_INTERFERON"
[23] "GO_INNER_DYNEIN_ARM_ASSEMBLY"
[24] "GO_REGULATION_OF_CILIUM_MOVEMENT"
[25]
"GO_DETECTION_OF_CHEMICAL_STIMULUS_INVOLVED_IN_SENSORY_PERCEPTION_OF_TASTE
"
[26] "GO_CENTRIOLE"
```

Mostly sperm stuff... not sure what to make of these results. 19 of those are also significant at q < .01.

Adding more genes

I was a bit worried that our conversion to genes with HUGO IDs was eliminating about 30K markers. Let's see if converting using a different database would do better:

```
myregion = 'Caudate'
data = readRDS('~/data/rnaseq_derek/complete_rawCountData_05132020.rds')
rownames(data) = data$submitted_name  # just to ensure compatibility later
data = data[data$Region==myregion, ]
more = readRDS('~/data/rnaseq_derek/data_from_philip_POP_and_PCs.rds')
more = more[!duplicated(more$hbcc_brain_id),]
data = merge(data, more[, c('hbcc_brain_id', 'comorbid', 'comorbid_group',
                            'substance', 'substance_group')],
             by='hbcc_brain_id', all.x=T, all.y=F)
grex_vars = colnames(data)[grepl(colnames(data), pattern='^ENS')]
count_matrix = t(data[, grex_vars])
data = data[, !grepl(colnames(data), pattern='^ENS')]
id_num = sapply(grex_vars, function(x) strsplit(x=x, split='\\.')[[1]][1])
rownames(count_matrix) = id_num
dups = duplicated(id_num)
id_num = id_num[!dups]
count_matrix = count_matrix[!dups, ]

library(biomaRt)
mart <- useDataset("hsapiens_gene_ensembl", useMart("ensembl"))
G_list0 <- getBM(filters= "ensembl_gene_id", attributes=
c("ensembl_gene_id",
                 "hgnc_symbol", "chromosome_name"),values=id_num,mart=
mart)
G_list <- G_list0[!is.na(G_list0$hgnc_symbol),]
G_list = G_list[G_list$hgnc_symbol!='',]
G_list <- G_list[!duplicated(G_list$ensembl_gene_id),]
imnamed = rownames(count_matrix) %in% G_list$ensembl_gene_id
count_matrix_bm = count_matrix[imnamed, ]
```