Final Project

Brian Wiley

Final Project

AS.410.671.81.FA20 Gene Expression Data Analysis and Visualization

Summary

For a final project, you will be conducting an analysis pipeline in attempt to answer some questions about the data set being used. The data selected should include some sort of class structure with different levels (e.g. disease vs. normal, treated vs. non-treated, age<65 vs. age>=65, low dose vs. high dose vs. control etc.).

Each project will include analysis of a publicly available data set from some expression database (e.g., Stanford MicroArray Database, Gene Expression Omnibus, EMBL ArrayExpress).

Obtain dataset from expression database.

The dataset I chose is a study of various leukemic B-cell chronic lymphoproliferative disorders (B-CLPD). The accession id is GSE79196. This data set contains 189 samples with 159 belonging to a specific diagnosed B-CLPD of 1 of 9 different types and 30 samples belonging to B-CLPD not otherwise specified (B-CLPD NOS). The RNA samples were process on the Affymetrix U133 Plus2.0 array.

Below I read in the data matrix using `getGeo()` function so that I can obtain the pheno data. I will use the pheno data to use for reading in the raw CEL files with the `affy` package `justRMA()` function. This workflow was obtained from the Coursera class taught by Dr. Kasper Daniel Hansen at JHU for Bioconductor. The `justRMA()` function is a wrapper which reads in CEL files with optional phenodata and performs `rma()` on the batch of CEL files without need for actually saving the `AffyBatch`. It is equivalent to performing `rma()` on an `AffyBatch`.

Get matrix, pheno data, and determine groups with >= 20 samples per group.

```r
library(GEOquery)
library(affy)

my.gse <- "GSE79196"
## commenting out after download
#my.geo.matrix <- getGEO(my.gse, AnnotGPL = T, getGPL = F, destdir=".")
#my.geo.matrix <- my.geo.matrix[[1]]
my.geo.matrix <- getGEO(file=paste0(my.gse,"_series_matrix.txt.gz"), AnnotGPL = T, getGPL = F)

my.pdata <- as.data.frame(pData(my.geo.matrix), stringsAsFactors=FALSE)

colnames(my.pdata)
```

```
##  [1] "title"                    "geo_accession"
##  [3] "status"                  "submission_date"
##  [5] "last_update_date"        "type"
##  [7] "channel_count"           "source_name_ch1"
##  [9] "organism_ch1"            "characteristics_ch1"
## [11] "treatment_protocol_ch1"  "molecule_ch1"
## [13] "extract_protocol_ch1"    "label_ch1"
## [15] "label_protocol_ch1"      "taxid_ch1"
## [17] "hyb_protocol"            "scan_protocol"
## [19] "data_processing"         "platform_id"
## [21] "contact_name"            "contact_institute"
## [23] "contact_address"         "contact_city"
## [25] "contact_zip/postal_code" "contact_country"
## [27] "supplementary_file"      "data_row_count"
## [29] "tissue:ch1"
```

```
head(my.pdata[, c("title", "geo_accession", "source_name_ch1")])
```

```
##                title geo_accession                    source_name_ch1
## GSM2087693 FL P001    GSM2087693 B cell lymphocytes from FL patient
## GSM2087694 FL P002    GSM2087694 B cell lymphocytes from FL patient
## GSM2087695 FL P003    GSM2087695 B cell lymphocytes from FL patient
## GSM2087696 FL P004    GSM2087696 B cell lymphocytes from FL patient
## GSM2087697 FL P005    GSM2087697 B cell lymphocytes from FL patient
## GSM2087698 FL P006    GSM2087698 B cell lymphocytes from FL patient
```

```
my.pdata <- my.pdata[, c("title", "geo_accession", "source_name_ch1")]
row.names(my.pdata) = paste0(row.names(my.pdata), '.CEL.gz')
## write ALL of the samples to annotated df including groups with < 20 samples
write.table(my.pdata, file=paste0(my.gse,"_PhenoData_ALL.txt"), quote=F, sep = "\t")

## determine which groups > 20
groups <- gsub("GSM.* ", "", my.pdata$title)
groups <- gsub(" .*", "", groups)
big.groups <- names(table(groups)[table(groups)>20])
big.groups
```

```
## [1] "B-CLPD" "CLL"    "cMCL"   "nnMCL"  "SMZL"
```

```
## remove samples where total of group < 20
## write the keep indices to file
keep <- groups %in% big.groups
write.table(data.frame(as.numeric(keep), keep), "keep.txt", quote = F)
```

Writing GEO matrix to file so we can just compare the rma() on the raw CEL files

```
new.geo.matrix <- my.geo.matrix[, keep]
new.pdata <- as.data.frame(pData(new.geo.matrix), stringsAsFactors=FALSE)
newgroups <- gsub("GSM.* ", "", new.pdata$title)
newgroups <- gsub(" .*", "", newgroups)
## sanity all larger than 20 samples
all(table(newgroups)>20)
```

```
## [1] TRUE
```

```
## write selected samples pheno data to file
new.pdata <- new.pdata[, c("title", "geo_accession", "source_name_ch1")]
row.names(new.pdata) = paste0(row.names(new.pdata), '.CEL.gz')
## will use this text file when reading in CELs into AffyBatch for large groups only
write.table(new.pdata, file=paste0(my.gse,"_SelectPhenoData.txt"), quote=F, sep = "\t")
## Will use to confirm if rma() on AffyBatch is different than author's matrix
## this can be space delimited since it's just probes and intensities
#https://stackoverflow.com/questions/2478352/write-table-writes-unwanted-leading-empty-column-to-header
write.table(exprs(new.geo.matrix), file=paste0(my.gse,"_matrix.txt"), quote=F, col.names = NA)
```

Getting the raw CEL files with `getGEOSuppFiles()`. This will be commented out since I already downloaded this and is over 1GB in size.

```
# getOption('timeout') ## might need to change this depending on size
# options(timeout=1000)
# dir.create("geo_downloads")
# a <- getGEOSuppFiles(my.gse, makeDirectory=T, baseDir="./geo_downloads")
# ## untar
# untar(cel.path, exdir=paste0("geo_downloads/", my.gse, "/CEL"))
```

Read in the raw CEL files and perform Robust Multi-array Average (RMA)

We have to read in all the CEL files while performing the RMA as this is mostly likely what the authors did, maybe with a slightly modified RMA algorithm. Most of differences are minor but some are close to log2(intensity) = 1.

```
## we can start here when re-opening Rmarkdown if reading raw files
library(affy)

cel.path <- paste0("geo_downloads/", my.gse, "/CEL")
# get file names
my.cels <- list.files(paste0("geo_downloads/", my.gse, "/CEL"), pattern = "*.CEL")
## confirm CELS in correct order
## filenames have P followed by sample # in them
my.pdata <- read.table(paste0(my.gse,"_PhenoData_ALL.txt"), sep = "\t")
all(gsub("_P[0-9]*", "", my.cels) == paste0(my.pdata$geo_accession, '.CEL.gz'))
```

```
## [1] TRUE
```

```
## all correct order so read in with pheno data
justNorm.data <- justRMA(celfile.path=cel.path,
                         phenoData=paste0(my.gse,"_PhenoData_ALL.txt"),
                         compress=T)
```

```
## Warning: Mismatched phenoData and celfile names!
##
## Please note that the row.names of your phenoData object should be identical to what you get from lis
## Otherwise you are responsible for ensuring that the ordering of your phenoData object conforms to th
## If not, errors may result from using the phenoData for subsetting or creating linear models, etc.
## Warning: replacing previous import 'AnnotationDbi::tail' by 'utils::tail' when
## loading 'hgu133plus2cdf'

## Warning: replacing previous import 'AnnotationDbi::head' by 'utils::head' when
## loading 'hgu133plus2cdf'
```

```
## subset just the large groups
keep <- read.table("keep.txt", )[,2]
justNorm.data.keep <- justNorm.data[, keep]

## Confirm differences in first 3 kept samples for first 10 intensities
geo.matrix <- read.table(pipe(paste0("cut -f1,2,3,4 -d' ' ", my.gse,"_matrix.txt")), header = T)
sample.matrix <- as.matrix(geo.matrix[1:10,])
sample.raw <- exprs(justNorm.data.keep)[1:10, 1:3]
## what are differences?
abs(sample.raw-sample.matrix)
```

```
##           GSM2087702.CEL.gz GSM2087704.CEL.gz GSM2087707.CEL.gz
## 1007_s_at        0.38270354        0.57810089        0.45089969
## 1053_at          0.17821118        0.25244933        0.19798456
## 117_at           0.04131330        0.02625822        0.17243212
## 121_at           0.22368387        0.37470041        0.27604209
## 1255_g_at        0.51691512        0.61950463        0.56746828
## 1294_at          0.14616433        0.05995108        0.25498950
## 1316_at          0.84471049        0.71285683        0.84604880
## 1320_at          0.39592717        0.35188223        0.33011042
## 1405_i_at        0.32087017        0.23085906        0.43153158
## 1431_at          0.09406824        0.25943695        0.07672127
```

```
max(abs(sample.raw-sample.matrix))
```

```
## [1] 0.8460488
```

```
## write matrix of RMA on raw CEL files for later use
write.table(exprs(justNorm.data.keep), file = paste0(my.gse,"_justRMA_keep_matrix.txt"),
            sep="\t", quote = F)
```

```


## we can start here when re-opening Rmarkdown with matrix for large groups from RMA on raw data
library(affy)

my.gse <- "GSE79196"
mat <- as.matrix(read.table(paste0(my.gse,"_justRMA_keep_matrix.txt"),
                            sep = "\t", header = T))
annot <- AnnotatedDataFrame(read.table(paste0(my.gse,"_SelectPhenoData.txt"),
                                       sep = "\t", header = T))
b.cell.expr.set <- ExpressionSet(assayData = mat,
                                 phenoData = annot)
## diagnosis groups
groups <- gsub(" .*", "", pData(b.cell.expr.set)$title)
```

```
## add in the featureData

# https://www.biostars.org/p/254040/
# http://biolearnr.blogspot.com/2017/05/bfx-clinic-getting-up-to-date.html
# most of code below comes up link above from BFX clinic to map probes to genes
# BiocManager::install("hgu133plus2.db")
library(hgu133plus2.db)
library(dplyr)

db.annotation.all <- AnnotationDbi::select(
  x = hgu133plus2.db,
  keys = rownames(b.cell.expr.set),
  #columns = c("PROBEID", "ENSEMBL", "ENTREZID", "SYMBOL"),
  columns = c("PROBEID", "SYMBOL"),
  keytype = "PROBEID"
)
## are all probes in db and in order?
all(rownames(b.cell.expr.set) %in% db.annotation.all$PROBEID)
```

Reading in final matrix that we saved for samples of groups >= 20

```
## [1] TRUE
```

```
all(rownames(b.cell.expr.set) == unique(db.annotation.all$PROBEID))
```

```
## [1] TRUE
```

```
dup.ids <- db.annotation.all$PROBEID[duplicated(db.annotation.all$PROBEID)] %>%
   unique  %>%
   sort

## few examples, I have seen this before
db.annotation.all[ db.annotation.all$PROBEID == dup.ids[1], ]
```

```
##     PROBEID  SYMBOL
## 1 1007_s_at    DDR1
## 2 1007_s_at MIR4640
```

```
db.annotation.all[ db.annotation.all$PROBEID == dup.ids[40], ]
```

```
##           PROBEID       SYMBOL
## 1054 1553633_s_at       SLC9B1
## 1055 1553633_s_at LOC101929373
```

```
## this will concatenate all genes mapped to a probe
db.annot.mult.mapping <- db.annotation.all %>%
  group_by(PROBEID) %>%
  summarise(PROBEID=PROBEID,
            genes = paste0(SYMBOL, collapse = "|")) %>%
  slice(1)
```

```r
## finally assign featureData
featureData(b.cell.expr.set) <- AnnotatedDataFrame(db.annot.mult.mapping)
b.cell.expr.set
```

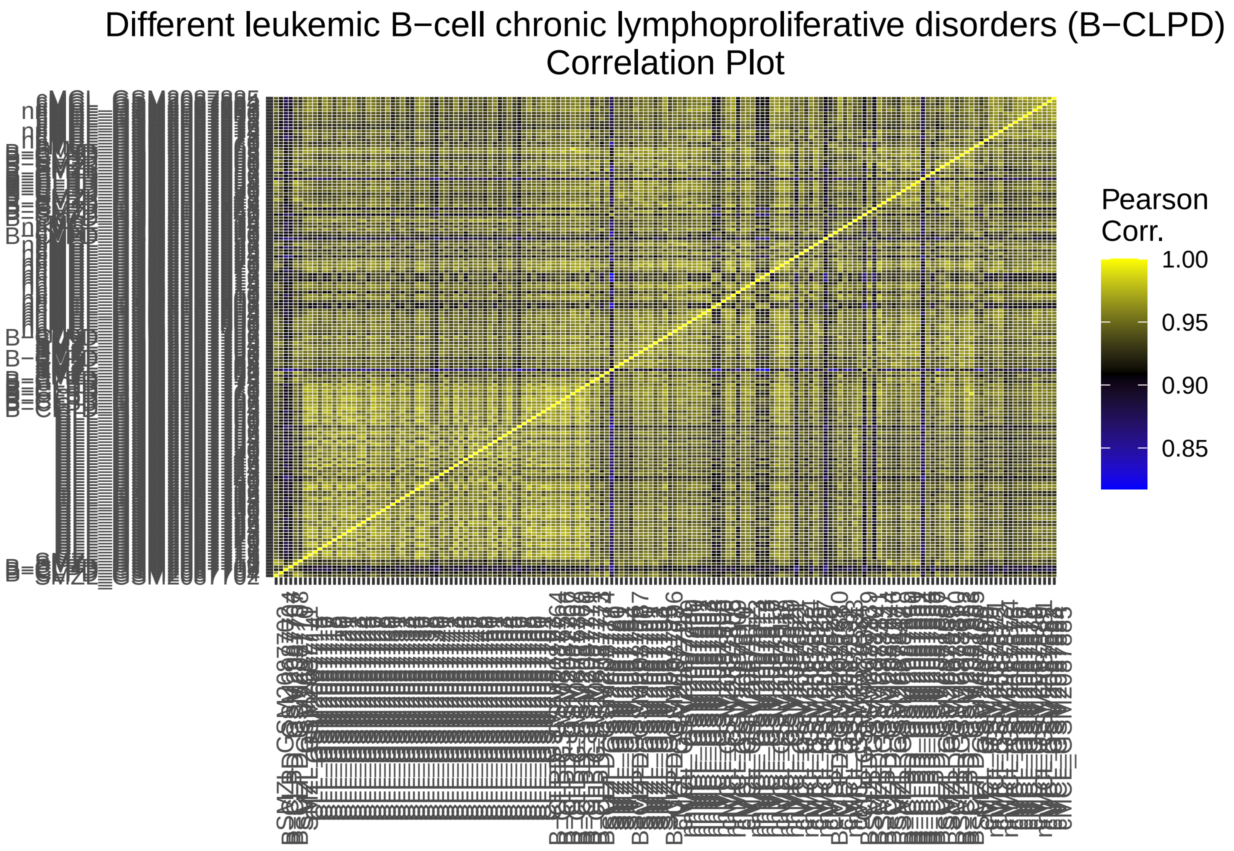
```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 54675 features, 161 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: GSM2087702.CEL.gz GSM2087704.CEL.gz ...
##     GSM2087885.CEL.gz (161 total)
##   varLabels: title geo_accession source_name_ch1
##   varMetadata: labelDescription
## featureData
##   featureNames: 1 2 ... 3 (54675 total)
##   fvarLabels: PROBEID genes
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation:
```

First test for outlier samples and provide visual proof. Remove these outliers.

First we will do correlation plot and heatmap to cluster the samples based on correlation. If this does not show any outliers, then we will do Coefficient of variation (CV) vs. Mean plot to visualize. Finally after determining outliers from CV vs. Mean plot we can visualize in heatmap again.

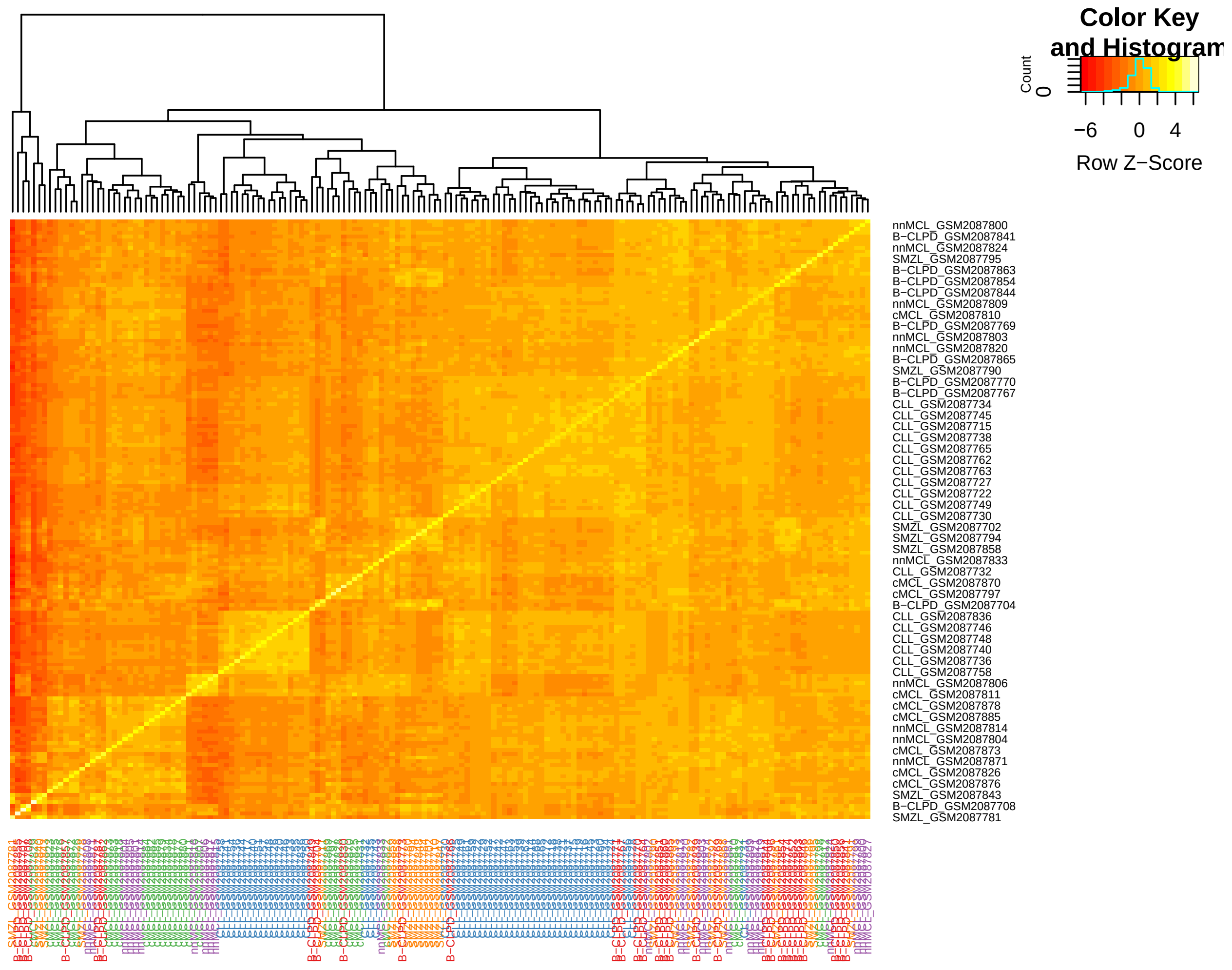
The initial correlation plot did show some blue but the range of correlation is not far apart so this cannot distinguish outliers completely.

```r
library(ggplot2)
library(reshape2)

## set dataframe for expressions
df <- exprs(b.cell.expr.set)
colnames(df) <- gsub("\\..*", "", colnames(df))
colnames(df) <- paste0(groups, "_", colnames(df))

## correlation
data.corr <- cor(df, use="pairwise.complete.obs", method="pearson")

melt.data.cor <- melt(data.corr)
mid <- (min(melt.data.cor$value)+max(melt.data.cor$value))/2

ggplot(melt.data.cor, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile(color="white") +
  scale_fill_gradient2(low="blue", mid="black", high="yellow",
                       midpoint=mid,
                       limit=c(min(melt.data.cor$value), 1),
                       name="Pearson\nCorr.") +
  theme(axis.text.x=element_text(angle=90),
        axis.title=element_blank(),
        plot.title=element_text(hjust=0.5)) +
  ggtitle("Different leukemic B-cell chronic lymphoproliferative disorders (B-CLPD)\nCorrelation Plot")
```



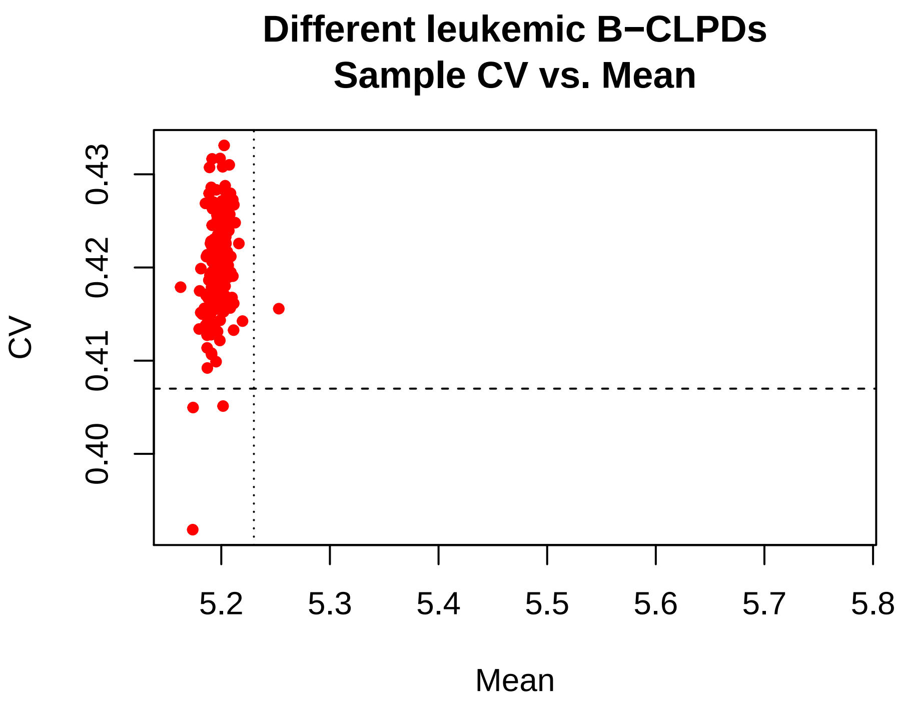
The heatmap of the correlation shows pretty normally distributed clustering and correlation.

```
## heatmap
library(gplots)
library(RColorBrewer)

cols <- RColorBrewer::brewer.pal(length(table(groups)), "Set1")
colors <- cols[as.factor(groups)]

# b l t r
pm <- par()$mar
par(mar=c(4.5, 4.1, 4.1, 2.1))

po <- par()$oma
par(oma=c(3.6,0,0,0))

new.lmat=rbind(c(0,3,4), c(0,1,2))
new.lhei=c(0.9,3.0)
new.lwid=c(0.5,4,1)

heatmap.2(abs(data.corr), trace="none", scale = "row",
          colCol = colors, key=T, dendrogram="column",
          labCol = gsub("_.*_", "_", colnames(df)),
          lmat = new.lmat, lhei = new.lhei, lwid = new.lwid)
```



The CV vs. Mean plot shows there are a few outliers.

```
library(qdapRegex)
means = apply(df, 2, function(x) mean(x, na.rm=T))
cvs = apply(df, 2, function(x) sd(x)/mean(x))

## plot
plot(means, cvs, xlab = "Mean", ylab = "CV", col="red", pch=20,
     xlim = c(min(means), max(means)*1.1))
title("Different leukemic B-CLPDs\nSample CV vs. Mean")
abline(h=0.407, lty=2)
abline(v=5.23, lty=3)
text(means, cvs, labels=ex_between(colnames(df),"_","_"),
     pos = 1, cex=0.6)
```



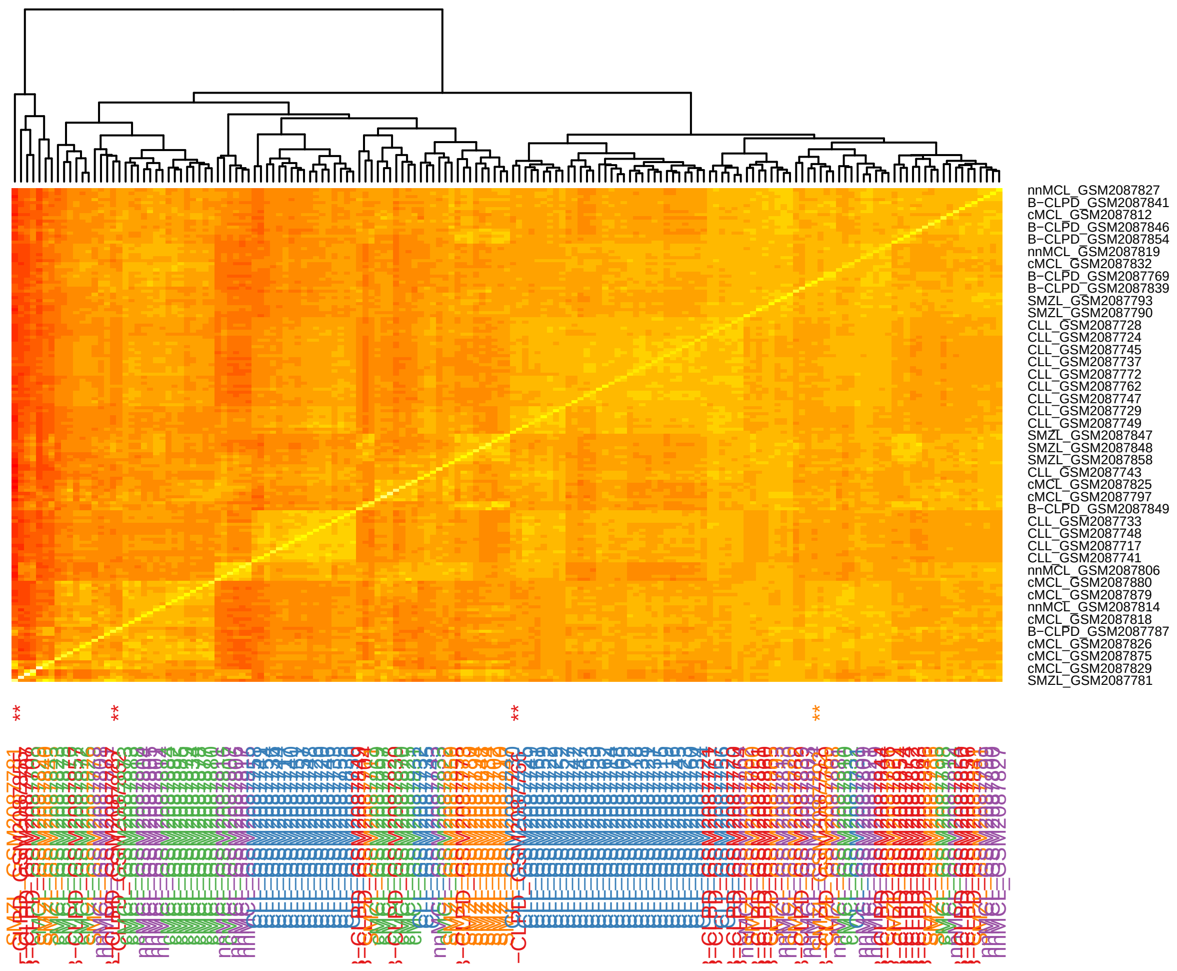
Show where the outliers samples are in heatmap.

```
outliers <- c(which(means>5.23), which(cvs<0.407))
## What are the outliers
outliers # 3 B-CLPD and 1 SMZL
```

```
##   SMZL_GSM2087792 B-CLPD_GSM2087766 B-CLPD_GSM2087855 B-CLPD_GSM2087862
##                77                59               134               141
```

```
#https://stackoverflow.com/questions/60798208/how-to-bold-a-group-of-labels-or-branches-in-heatmap-2-in
library(purrr)
make_bold_names <- function(mat, rc_fun, rc_names) {
  bold_names <- rc_fun(mat)
  ids <- rc_names %>% match(rc_fun(mat))
  ids %>%
    walk(
      function(i)
        bold_names[i] <<-
        bquote(bold(.(rc_fun(mat)[i]))) %>%
        as.expression()
    )
  bold_names
}

data.corr.copy <- data.corr
colnames(data.corr.copy)[outliers] <- paste(colnames(data.corr.copy)[outliers], "    **")
colnames(data.corr.copy)[-outliers] <- paste(colnames(data.corr.copy)[-outliers], "      ")
```

```r
par(mar=c(4.5, 4.1, 4.1, 2.1))
par(oma=c(6.1,0,0,0))
heatmap.2(abs(data.corr.copy), trace="none", scale = "row",
          colCol = colors, key=F, dendrogram="column",
          labCol = gsub("_.*_", "_", make_bold_names(data.corr.copy,colnames, outliers)),
          lmat = new.lmat, lhei = new.lhei, lwid = new.lwid, cexCol = 1)
```



Filter genes using Coefficient of variation (CV). According to Hackstadt and Hess, "When filtering by variance, we remove genes with low variance across arrays (ignoring treatment). The rationale is that expression for equally expressed genes (EEGs) should not differ greatly between treatment groups, hence leading to small overall variance." (Hackstadt and Hess, 2009)

```r
## remove outliers
df.rem.out <- df[, -outliers]
dim(df.rem.out)
```

```
## [1] 54675   157
```

```r
annot.rem.out <- annot[-outliers, ]
dim(annot.rem.out)
```

```
##    rowNames columnNames
##         157           3
```

```r
group.rem.out <- groups[-outliers]

## For non-specific filter exprs should be un-logged?
#https://math.usu.edu/jrstevens/stat5570/3.2.Filtering.pdf
df.unlogged <- 2^df.rem.out

cv.genes = apply(df.unlogged, 1, function(x) sd(x)/mean(x))
quant <- quantile(cv.genes, probs = c(0.1, 0.25, 0.5, 0.75, 0.9))
df.filter <- df.rem.out[cv.genes >= quant[1], ]
dim(df.filter)
```

```
## [1] 49207   157
```

Next, conduct some method of feature selection with a statistical test or other machine learning method. The type of test will depend upon how many factor levels are included in your data set. For example. two conditions would require a two-sample test, while greater than two conditions would require other tests.

I will first be performing ANOVA on the different samples for the 4 groups with diagnosis with which we will then adjust for multiplicity to confirm which genes are differentially expressed amongst groups, i.e. at least one group is different. Then I will filter from those, the genes which characterize or profile the sub-type. That is, only selecting genes in which the all the p-values are statically significant between 1 group vs. the rest, i.e. a 1 vs. All "complete" significance. Then we can use these features to perform both a cluster analysis as well as SVM on the B-CLPD NOS group to characterize those into their respective predicted sub group of B-CLPD and compare the accuracy of methods. These methods however cannot be compared to eachother on the B-CLPD NOS group because we don't have truth labels and these are different methods, unsupervised for kmeans vs. supervised with SVM.

First order groups and set aside B-CLPD NOS group.

```r
## first order the samples by group
groups.ord <- group.rem.out[order(group.rem.out)]
df.ord <- df.filter[, order(group.rem.out)]
dim(df.ord)
```

```
## [1] 49207   157
```

```r
colnames(df.ord)[1:5]
```

```
## [1] "B-CLPD_GSM2087704" "B-CLPD_GSM2087707" "B-CLPD_GSM2087708"
## [4] "B-CLPD_GSM2087764" "B-CLPD_GSM2087767"
```

```r
annot.ord <- annot.rem.out[order(group.rem.out), ]

## this is what we are trying to predict
B.CLPD <- grep("B-CLPD", groups.ord)
diagnosed.groups.df <- df.ord[, -B.CLPD]
annot.diagnosed.groups <- annot.ord[-B.CLPD, ]
dim(diagnosed.groups.df)
```

```
## [1] 49207   130
```

```r
dim(annot.diagnosed.groups)
```

```
##    rowNames columnNames
##         130           3
```

```r
diagnosed.groups <- groups.ord[-B.CLPD]
table(diagnosed.groups)
```

```
## diagnosed.groups
##   CLL  cMCL nnMCL  SMZL
##    54    30    24    22
```

Perform ANOVA on all genes and adjust for multiplicity filtering for significant genes using confidence level of 99% or .01 alpha.

```r
anova.all.genes <- function(x, groups) {
  d <- cbind(reshape2::melt(x), groups)
  a <- summary(aov(value~groups, d))
  pval <- a[[1]][1,5]
  return(pval)
}

anova.all.pval <- apply(diagnosed.groups.df, 1, anova.all.genes,
                        groups=diagnosed.groups)

# adjust p-values
p.adj <- p.adjust(anova.all.pval, method = "BH")
diagnosed.groups.keep.anova <- diagnosed.groups.df[p.adj<.01, ]
nrow(diagnosed.groups.keep.anova)
```

```
## [1] 20653
```

Now perform a novel function on each gene for defining the genes that separate one group from the rest with all p-values being significant. An example of a `pairwise.t.test()` with “melting modification” would be which is the output from `t.list()` inside the for loop:

```
$cMCL$pval
        pval         pval         pval 
0.0005676161 0.1170199554 0.0596549565 

$cMCL$fc
       fc        fc        fc 
-1.085794 -1.045249 -1.058075 


$CLL
$CLL$pval
        pval         pval         pval 
5.676161e-04 1.315473e-06 5.614498e-07 

$CLL$fc
       fc        fc        fc 
-1.085794 -1.134925 -1.148851 


$nnMCL
$nnMCL$pval
        pval         pval         pval 
1.315473e-06 1.170200e-01 6.702023e-01 

$nnMCL$fc
       fc        fc        fc 
-1.134925 -1.045249 -1.012271 


$SMZL
$SMZL$pval
        pval         pval         pval 
5.614498e-07 5.965496e-02 6.702023e-01 

$SMZL$fc
       fc        fc        fc 
-1.148851 -1.058075 -1.012271 
```

We see above that the only sub-type that have significant p-value < .01 against ALL other sub-types is CLL so this gene we keep for classification.

We will also use the `pairwise.fc()` function from the `MKmisc` package but need to confirm the pairwise order is same as `pairwise.t.test()`

```r
## test order of pairwise.fc() and pairwise.t.test() as they are different pacakges
library(MKmisc)

i = nrow(diagnosed.groups.keep.anova)
d <- diagnosed.groups.keep.anova[i, ]

pair.t <- pairwise.t.test(x=d, g=diagnosed.groups, p.adjust.method = "BH")
p <- pair.t$p.value
```

```
m <- melt(p)
m <- m[!is.na(m$value),]

fc <- pairwise.fc(x=d, g=diagnosed.groups)
fc
```

```
##   CLL vs cMCL  CLL vs nnMCL   CLL vs SMZL cMCL vs nnMCL  cMCL vs SMZL
##     -1.085794     -1.134925     -1.148851     -1.045249     -1.058075
## nnMCL vs SMZL
##     -1.012271
```

```
fc.df <- data.frame(do.call(rbind, strsplit(names(fc), " vs ")))

## columns diff order but all rows same so we are good
all(m[,c("Var2", "Var1")] == fc.df)
```

```
## [1] TRUE
```

Perform function in loop for all genes

```
probes <- rownames(diagnosed.groups.keep.anova)
## list to keep the significant profiling genes/probes
sig.genes <- list()
## list to keep the indices of those genes
sig.genes.idx <- c()
## list that will hold the final dataframe information
sig.genes.leuk.type <- list(type=c(), probe=c(), avg.p=c(), avg.fc=c())
## used to append arrays in sig.genes.leuk.type
count = 1

for (i in 1:nrow(diagnosed.groups.keep.anova)) {
  d <- diagnosed.groups.keep.anova[i, ]
  probe <- probes[i]
  pair.t <- pairwise.t.test(x=d, g=diagnosed.groups, p.adjust.method = "BH")


  ## We need to rearrange the p.value table so we can see
  ## by rows for each group the p-value against the 3 other groups
  ## i.e. 6 pvals correspond to 6 x 2 = 12 differences or 4 groups x 3 comparisons
  ## for each group against the other 3
  p <- pair.t$p.value
  m <- melt(p)
  m <- m[!is.na(m$value),]

  ## From help menu:
  ## The fold changes are returned in a slightly modified form if mod.fc = TRUE.
  ## Fold changes FC which are smaller than 1 are reported as to -1/FC.
  ## Also it unlogs the fold change in the function with the code
  ##    if (log) {
  ##  logFC <- ave(xj, ...) - ave(xi, ...)
  ##  FC <- base^logFC
  ##  }
  fc <- pairwise.fc(x=d, g=diagnosed.groups)
  # confirmd above loop same order as pairwise.t.test
  m$fc <- fc

  ## will be a 4x3 list for each group vs other 3, pretty nifty!
  t.list <- list()
  for (j in 1:nrow(m)) {
    for (k in 1:2) {
      key <- as.character(m[j,k])
      t.list[[key]]$pval <- c(t.list[[key]]$pval, pval=m[j,"value"])
      t.list[[key]]$fc <- c(t.list[[key]]$fc, fc=m[j,"fc"])
    }
  }

  ## only has completely different expression in one type verse the rest
  ## not interested in the gene expression is differentially expressed
  ## for example if gene is same in mMCL and CLL and same in nnMCL and SMZL
  ## but different between the two subsets that would be BH signficant but
  ## wouldn't help use in assigning leukemia type expression PROFILES
  significant <- names(which(lapply(t.list, function(x) all(x$pval < .01)) == T))
  bool <- length(significant) > 0
  if (bool) {
    sig.genes[[probe]] <- significant
    for (leuk in significant) {
      sig.genes.leuk.type$type[count]=leuk
      sig.genes.leuk.type$probe[count]=probe
      sig.genes.leuk.type$avg.p[count]=mean(t.list[[leuk]]$pval)
      sig.genes.leuk.type$avg.fc[count]=mean(t.list[[leuk]]$fc)
      count=count+1
    }
  }
  sig.genes.idx[i] <- bool
}
## total bool = T should be the length of nice significant gene list
sum(sig.genes.idx) == length(sig.genes)
```

```
## [1] TRUE
```

Provide the number of genes retained with the associated score (p-value, weight, test statistic, etc.) and threshold value that you used. Plot the scores of those genes retained in a histogram. The p-value retained for these genes were an average of the 3 p-values that were all < .01 in the function above.

```
cat(sum(sig.genes.idx), "significant profiling genes")
```

```
## 7589 significant profiling genes
```

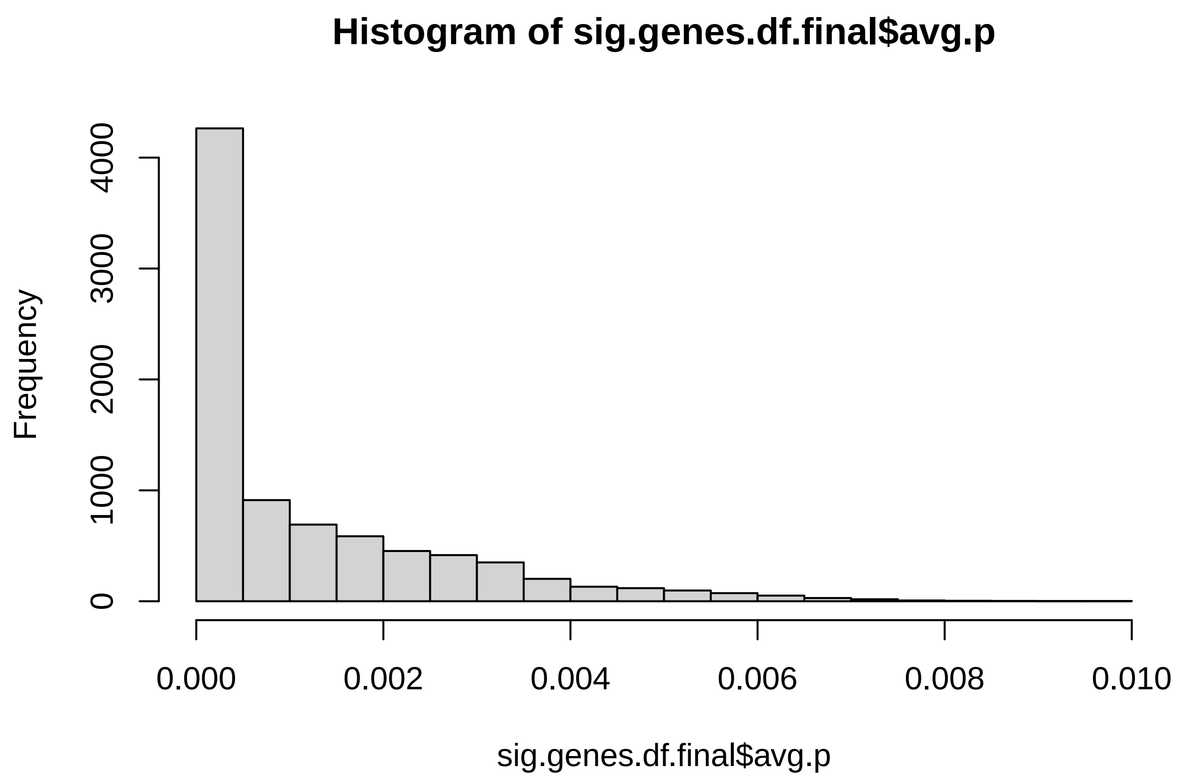
```r
## put in dataframe
sig.genes.df <- data.frame(sig.genes.leuk.type)
## some genes have significant 1 vs. ALL for multiple sub-types
## this may be longer that genes because the gene may be completely different
## for more than 1 type, i.e. 'gene6' has adjusted
## 3 pvals < .01 for cMCL verse the other 3 and also
## 3 pvals < .01 for nnMCL verse the other 3
dim(sig.genes.df)
```

```
## [1] 8409    4
```

```r
head(sig.genes.df)
```

```
##   type        probe        avg.p    avg.fc
## 1  CLL       117_at 8.261711e-04 -2.089694
## 2  CLL       121_at 1.474045e-04 -1.192581
## 3  CLL      1294_at 2.550691e-05 -1.605798
## 4 SMZL    1405_i_at 2.408498e-03 -3.137145
## 5  CLL      1438_at 3.446653e-03 -1.122142
## 6  CLL 1552256_a_at 1.345817e-03 -1.469403
```

```r
sig.genes.df.final <- sig.genes.df[order(sig.genes.df$avg.p),]

hist(sig.genes.df.final$avg.p)
```



Next, subset your data by the genes that you determined and use one of the clustering or dimensionality reduction methods discussed in class to visualize the samples in two-dimensional space (xy scatter plot, dendrogram, etc.).

First putting into expression set then plotting with PCA. We get pretty good results in the PCA plot.

```
## subset based on significant profiling scores
diagnosed.groups.keep.anova.pairT <- diagnosed.groups.keep.anova[sig.genes.idx, ]
dim(diagnosed.groups.keep.anova.pairT)
```

```
## [1] 7589  130
```

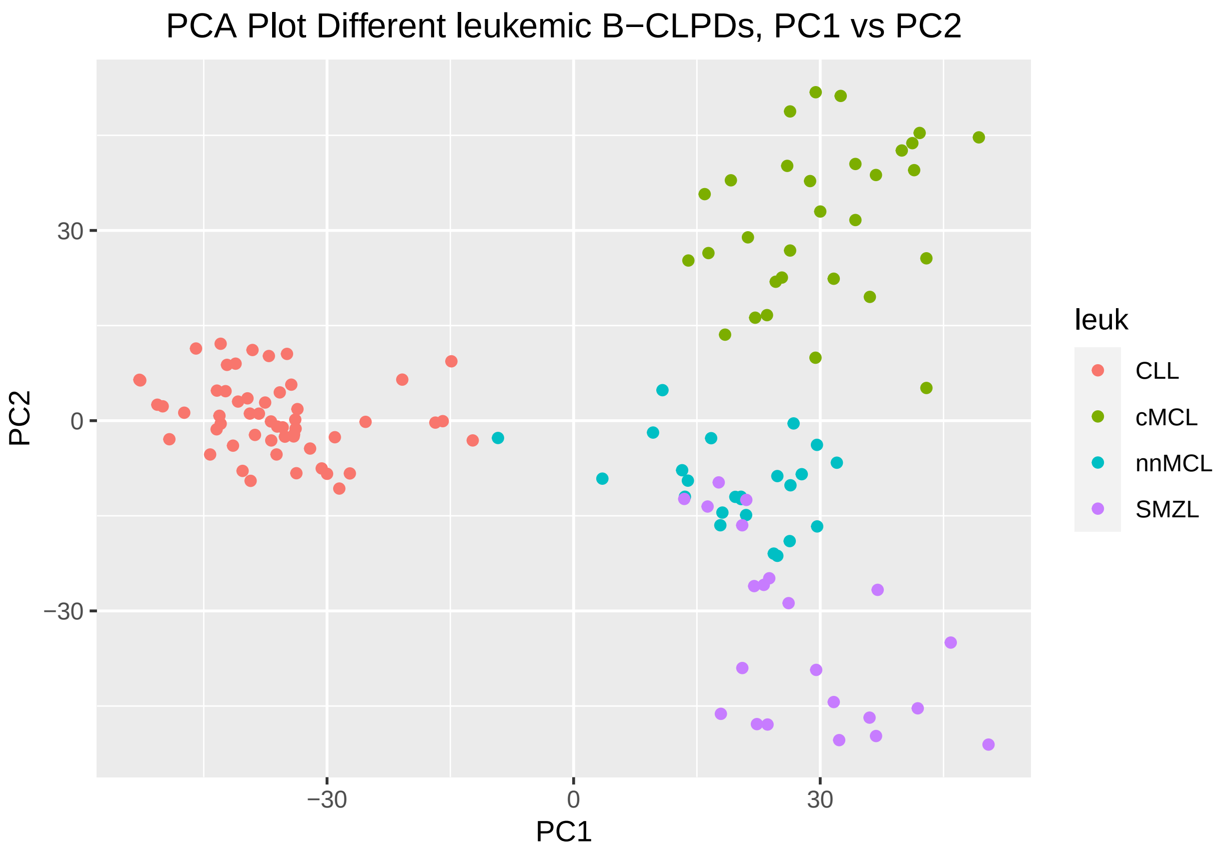
```
## put back into ExpressionSet with featureData
colnames(diagnosed.groups.keep.anova.pairT) <-
  paste0(gsub(".*_", "", colnames(diagnosed.groups.keep.anova.pairT)), ".CEL.gz")

significant.expset <- ExpressionSet(assayData=diagnosed.groups.keep.anova.pairT,
                                    phenoData=annot.diagnosed.groups)

db.annotation <- AnnotationDbi::select(
  x = hgu133plus2.db,
  keys = rownames(significant.expset),
  columns = c("PROBEID", "SYMBOL"),
  keytype = "PROBEID"
)
## same order?
all(rownames(significant.expset)==unique(db.annotation$PROBEID))
```

```
## [1] TRUE
```

```
dup.ids <- db.annotation$PROBEID[table(db.annotation$PROBEID) > 1] %>%
  unique %>%
  sort

db.annot.mult.mapping.significant <- db.annotation %>%
  group_by(PROBEID) %>%
  summarise(PROBEID=PROBEID,
            genes = paste0(SYMBOL, collapse = "|")) %>%
  dplyr::slice(1)

featureData(significant.expset) <- AnnotatedDataFrame(db.annot.mult.mapping)

## plot PCA
signficant.exprs <- exprs(significant.expset)
BH.pairT.pca <- prcomp(t(signficant.exprs))
pcomps <- data.frame(BH.pairT.pca$x[, 1:2])
dim(pcomps)
```

```
## [1] 130   2
```

```
pcomps$leuk <- diagnosed.groups

ggplot(pcomps, aes(PC1, PC2, color=leuk)) +
  geom_point() +
  ggtitle("PCA Plot Different leukemic B-CLPDs, PC1 vs PC2") +
  theme(plot.title = element_text(hjust = 0.5))
```



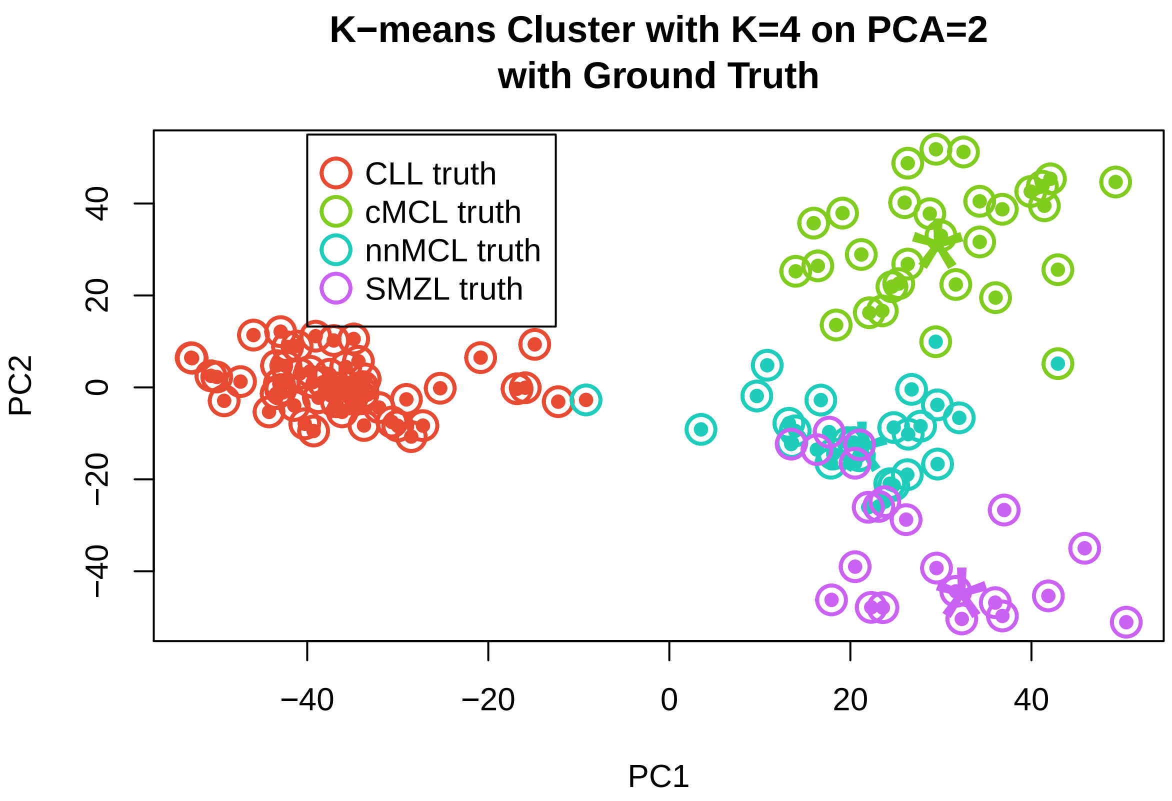
Using these linear projections of the original data (i.e. cluster centroids, latent variables, etc.), use a classification method to classify the samples into their respective classes.

First perform Kmeans clustering analysis on the first 2 principal components to see the accuracy and then try SVM splitting train and test for the diagnosed groups using these 2 principle components and finally training on all the diagnosed groups with all significant profiling genes to predict the B-CLPD NOS subtype.

We can see that with Kmeans clustering that it misclassifies 0 CLL samples, only a couple cMCL samples, but a lot of the SMZL and couple of the cMCL were classified incorrectly as nnMCL in teal color. This is to be expected and we did not get much separation between nnMCL and SMZL from the PCA plot above.

```
set.seed(1)
cl <- kmeans(pcomps[,1:2], centers=4, iter.max=20)

## map colors to the clusters from kmeans
#3 CLL red #e84a31
```

```
#1 cMCL green #7fcc1e
#2 nnMCL teal #1eccba
#4 SMZL purple #cb61f3
cols <- c("#7fcc1e", "#1eccba", "#e84a31", "#cb61f3")
cols.pred <- plyr::mapvalues(cl$cluster, from=c(1,2,3,4), to=cols)
# 3,1,2,4
cols.truth <- plyr::mapvalues(diagnosed.groups, from=c("CLL", "cMCL", "nnMCL", "SMZL"), to=cols[c(3, 1,


plot(pcomps[,1:2], col=cols.pred, cex=1, pch=16,
     main="K-means Cluster with K=4 on PCA=2\nwith Ground Truth")
points(cl$centers, col=cols, pch = '*' , cex=4.5)

points(pcomps, col = cols.truth, cex=2, lwd=2)
legend(-40, 55, legend=c("CLL truth", "cMCL truth", "nnMCL truth", "SMZL truth"),
       col=cols[c(3, 1, 2, 4)], pch=1, pt.cex=2, pt.lwd=2)
```



What is the accuracy from kmeans clustering? We get about 91% accuracy with PCA on the ~7,600 significant genes.

```
library(caret)

# 3,1,2,4
pred <- cl$cluster
levels(pred) <- unique(cl$cluster)
truth <- plyr::mapvalues(diagnosed.groups, from=c("CLL", "cMCL", "nnMCL", "SMZL"), to=c(3, 1, 2, 4))
```

```r
truth <- as.numeric(truth)
levels(truth) <- unique(truth)

confusionMatrix(factor(pred), factor(truth))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4
##          1 28  0  0  0
##          2  2 23  0  8
##          3  0  1 54  0
##          4  0  0  0 14
##
## Overall Statistics
##
##                Accuracy : 0.9154
##                  95% CI : (0.8536, 0.957)
##     No Information Rate : 0.4154
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8807
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity            0.9333   0.9583   1.0000   0.6364
## Specificity            1.0000   0.9057   0.9868   1.0000
## Pos Pred Value         1.0000   0.6970   0.9818   1.0000
## Neg Pred Value         0.9804   0.9897   1.0000   0.9310
## Prevalence             0.2308   0.1846   0.4154   0.1692
## Detection Rate         0.2154   0.1769   0.4154   0.1077
## Detection Prevalence   0.2154   0.2538   0.4231   0.1077
## Balanced Accuracy      0.9667   0.9320   0.9934   0.8182
```

Now trying SVM

We will train with a stratified 70/30 train/test split on the 2 PCA components for the 4 diagnosed groups. Here we get a 92% accuracy rate so slightly better than kmeans. The 3 misclassifications come from groups where prediction was 2=nnMCL but the truth was 4=SMZL which is the same story above for kmeans because the PCA does not have good separation on a few (around 1/3) of samples in each type.

```r
X.pca <- pcomps[,1:2]
## for simplicity we will assign same order to the classes from kmeans
X.pca$subtype <- as.factor(as.numeric(truth))

set.seed(1)
## get stratified index
## https://stackoverflow.com/questions/20776887/stratified-splitting-the-data
train.idx <- caret::createDataPartition(diagnosed.groups, p=0.7, list=F)
table(diagnosed.groups[train.idx])/table(diagnosed.groups)
```

```
##
##       CLL      cMCL     nnMCL      SMZL
## 0.7037037 0.7000000 0.7083333 0.7272727
```

```
train.pca <- X.pca[train.idx, ]
train.pca$subtype
```

```
##  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [77] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## Levels: 1 2 3 4
```

```
test.pca <- X.pca[-train.idx, -ncol(X.pca)]
y.test.pca <- X.pca$subtype[-train.idx]

svm.model.bcell.pca <- e1071::svm(subtype ~ ., data = train.pca, kernel = 'radial')
y.pred.pca <- predict(svm.model.bcell.pca, test.pca)

## accuracy
confusionMatrix(y.pred.pca, y.test.pca)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4
##          1  9  0  0  0
##          2  0  7  0  3
##          3  0  0 16  0
##          4  0  0  0  3
##
## Overall Statistics
##
##                Accuracy : 0.9211
##                  95% CI : (0.7862, 0.9834)
##     No Information Rate : 0.4211
##     P-Value [Acc > NIR] : 1.237e-10
##
##                   Kappa : 0.8881
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity            1.0000   1.0000   1.0000  0.50000
## Specificity            1.0000   0.9032   1.0000  1.00000
## Pos Pred Value         1.0000   0.7000   1.0000  1.00000
## Neg Pred Value         1.0000   1.0000   1.0000  0.91429
## Prevalence             0.2368   0.1842   0.4211  0.15789
## Detection Rate         0.2368   0.1842   0.4211  0.07895
## Detection Prevalence   0.2368   0.2632   0.4211  0.07895
## Balanced Accuracy      1.0000   0.9516   1.0000  0.75000
```

We will train with a stratified 70/30 train/test split on the 4 diagnosed groups. However we will use all ~7,600 features we obtained from ANOVA and pairwise t-tests.

We get 97% accuracy training with all ~7,600 significant genes. So this should be a good classifier for final classification.

```
library(e1071)

X <- data.frame(t(signficant.exprs))
## for simplicity we will assign same order to the classes from kmeans
#truth <- plyr::mapvalues(diagnosed.groups, from=c("CLL", "cMCL", "nnMCL", "SMZL"), to=c(3, 1, 2, 4))
X$subtype <- as.factor(as.numeric(truth))

## use same indices
train <- X[train.idx, ]
train$subtype
```

```
##  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [77] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## Levels: 1 2 3 4
```

```
test <- X[-train.idx, -ncol(X)]
y.test <- X$subtype[-train.idx]

set.seed(1)
svm.model.bcell <- e1071::svm(subtype ~ ., data = train, kernel = 'radial')
y.pred <- predict(svm.model.bcell, test)

## 97% accuracy
confusionMatrix(y.pred, y.test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4
##          1  9  0  0  1
##          2  0  7  0  0
##          3  0  0 16  0
##          4  0  0  0  5
##
## Overall Statistics
##
##                Accuracy : 0.9737
##                  95% CI : (0.8619, 0.9993)
##     No Information Rate : 0.4211
##     P-Value [Acc > NIR] : 2.826e-13
##
##                   Kappa : 0.9627
##
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity            1.0000   1.0000   1.0000   0.8333
## Specificity            0.9655   1.0000   1.0000   1.0000
## Pos Pred Value         0.9000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   0.9697
## Prevalence             0.2368   0.1842   0.4211   0.1579
## Detection Rate         0.2368   0.1842   0.4211   0.1316
## Detection Prevalence   0.2632   0.1842   0.4211   0.1316
## Balanced Accuracy      0.9828   1.0000   1.0000   0.9167
```

First see how SVM does on classifying B-CLPD NOS subtype on the 2 highest principal components as well as finally finishing with predicting B-CLPD NOS subtype using all ~7,600 genes. We need to perform PCA again with all 157 samples excluding outliers.

```
## filter the df of B-CLPD NOS and diagnosed groups for both of the ANOVA and pairwise T-tests
## already ordered by sample from before
all.groups.df <- df.ord[p.adj<.01, ]
all.groups.df2 <- all.groups.df[sig.genes.idx, ]
## sanity check
all(row.names(all.groups.df2)==row.names(diagnosed.groups.keep.anova.pairT))
```

```
## [1] TRUE
```

```
## plot PCA
all.groups.pca <- prcomp(t(all.groups.df2))
all.groups.pcomps <- data.frame(all.groups.pca$x[, 1:2])
all.groups.pcomps$leuk <- groups.ord

## train on all diagnosed groups, change to assign mappings with NOS types = 0
## unfortunately e1071 requires factors where we have to unfactor it back later
## same numbers as labels from before
all.groups.pcomps$leuk.number <- factor(as.numeric(plyr::mapvalues(all.groups.pcomps$leuk, from=c("B-CL

set.seed(1)
svm.model.bcell.pca.all.diagnosed <- e1071::svm(leuk.number ~ ., data = all.groups.pcomps[-B.CLPD, c(1,
## predict
BCDLP.NOS.predict <- predict(svm.model.bcell.pca.all.diagnosed, all.groups.pcomps[B.CLPD, c(1,2)])

#3 CLL red #e84a31
#1 cMCL green #7fcc1e
#2 nnMCL teal #1eccba
#4 SMZL purple #cb61f3
# what are predictions?
cat("Predictions:")
```

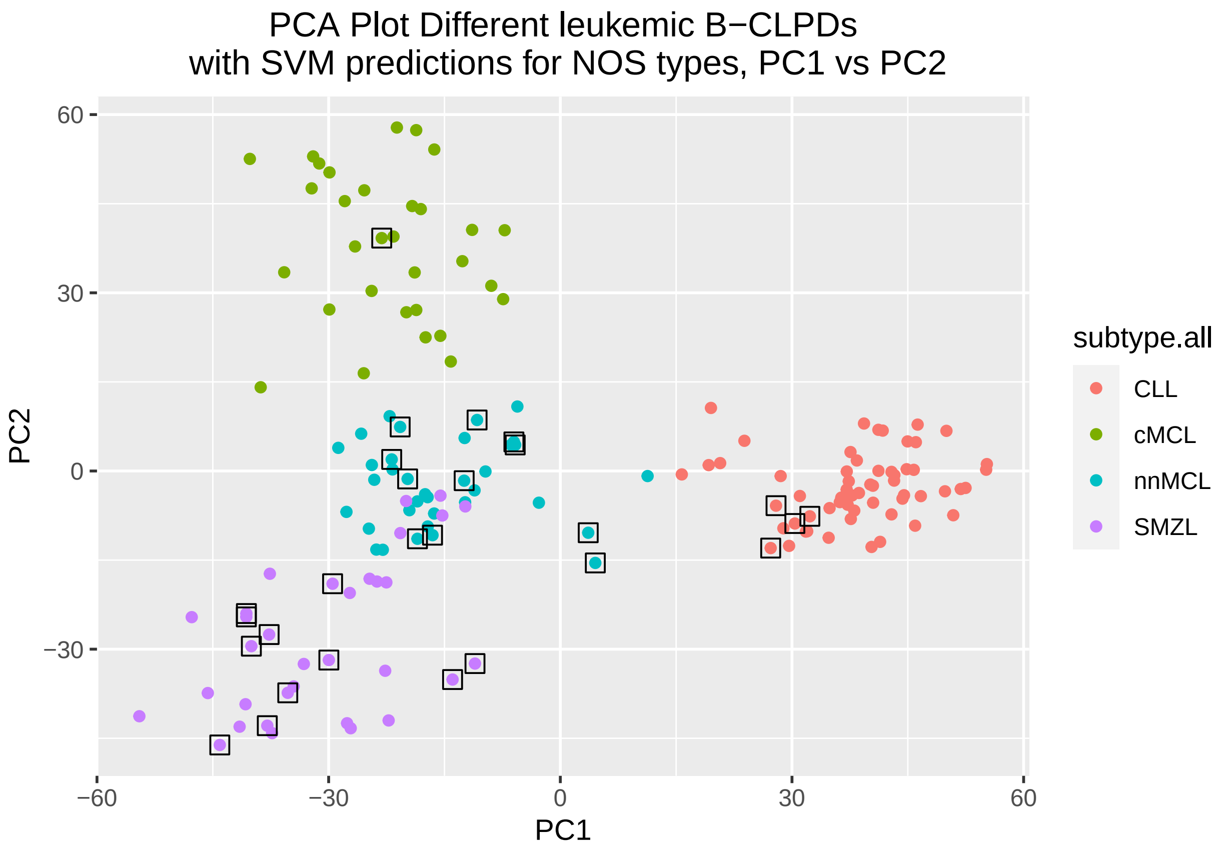
```
## Predictions:
```

```
table(c("CLL", "cMCL", "nnMCL", "SMZL")[order(c(3, 1, 2, 4))][as.numeric(as.character(BCDLP.NOS.predict
```

```
##
##   CLL  cMCL nnMCL  SMZL
##     4     1    11    11
```

Plot with predictions from SVM along with the known subtypes. The B-CLPD NOS are indicated in black squares around the predictions.

```
## need to map back
## e1071 and ggplot2 not super compatible
all.groups.pcomps$subtype.all <-
  c(as.numeric(as.character(BCDLP.NOS.predict)),
    as.numeric(as.character(all.groups.pcomps$leuk.number[-B.CLPD])))
all.groups.pcomps$subtype.all <- plyr::mapvalues(all.groups.pcomps$subtype.all, from=c(3, 1, 2, 4), to=

ggplot(all.groups.pcomps, aes(PC1, PC2, color=subtype.all)) +
  geom_point() +
  geom_point(data=all.groups.pcomps[B.CLPD,], aes(PC1, PC2), color="black", shape=0, size=3) +
  ggtitle("PCA Plot Different leukemic B-CLPDs\n with SVM predictions for NOS types, PC1 vs PC2") +
  theme(plot.title = element_text(hjust = 0.5))
```



Run SVM on all ~7,600 profiling genes.

```
## X is the entire train set of diagnosed groups of 130 samples
#3 CLL red #e84a31
#1 cMCL green #7fcc1e
#2 nnMCL teal #1eccba
#4 SMZL purple #cb61f3
dim(X)
```

```
## [1]  130 7590
```

```
X$subtype
```

```
##   [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##  [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4
## [112] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## Levels: 1 2 3 4
```

```
## test set is the 27 B-CLPD NOS samples
test <- data.frame(t(all.groups.df2[,B.CLPD]))
dim(test)
```

```
## [1]   27 7589
```

```
svm.model.bcell.all <- e1071::svm(subtype ~ ., data = X, kernel = 'radial')
B.CLPD.NOS.pred.all.features <- predict(svm.model.bcell.all, test)
## What are the predicted groups
c("CLL", "cMCL", "nnMCL", "SMZL")[order(c(3, 1, 2, 4))][as.numeric(as.character(B.CLPD.NOS.pred.all.fea
```

```
##  [1] "SMZL"  "SMZL"  "SMZL"  "CLL"   "CLL"   "SMZL"  "CLL"   "CLL"   "SMZL"
## [10] "SMZL"  "SMZL"  "nnMCL" "cMCL"  "CLL"   "nnMCL" "nnMCL" "SMZL"  "SMZL"
## [19] "SMZL"  "nnMCL" "SMZL"  "SMZL"  "SMZL"  "cMCL"  "nnMCL" "SMZL"  "nnMCL"
```

```
table(c("CLL", "cMCL", "nnMCL", "SMZL")[order(c(3, 1, 2, 4))][as.numeric(as.character(B.CLPD.NOS.pred.a
```

```
##
##   CLL  cMCL nnMCL  SMZL
##     5     2     6    14
```

This basically matches what the authors predicted. Most of their predictions fit the B-CLPD NOS into the SMZL subtype with the second most fitting into the CLL subtype. See Figure 1 in paper.

We can see that the predictions are slightly different (5 total differences) when training with all the features compared to just the top 2 principal components.

```
pred.pca <- c("CLL", "cMCL", "nnMCL", "SMZL")[order(c(3, 1, 2, 4))][as.numeric(as.character(BCDLP.NOS.p
pred.all.sig.genes <- c("CLL", "cMCL", "nnMCL", "SMZL")[order(c(3, 1, 2, 4))][as.numeric(as.character(B
table("pred.pca"=pred.pca, "pred.all.sig.genes"=pred.all.sig.genes)
```

```
##         pred.all.sig.genes
## pred.pca CLL cMCL nnMCL SMZL
##    CLL     4    0     0    0
##    cMCL    0    1     0    0
##    nnMCL   1    1     6    3
##    SMZL    0    0     0   11
```

Finally, using the top 5 discriminant genes (positive and negative direction) from your analysis, go to NCBI's DAVID and look up the gene information. Provide the gene name and functional information (associated pathways, GO terms, etc) for these 10 genes.

I am going to actually include the top discriminant genes positive and negative for all four groups CLL, cMCL, nnMCL, SMZL. So there will be 40 total.

```r
## put in dataframe
sig.genes.df <- data.frame(sig.genes.leuk.type)


db.annotation <- AnnotationDbi::select(
  x = hgu133plus2.db,
  keys = rownames(significant.expset),
  columns = c("PROBEID", "SYMBOL"),
  keytype = "PROBEID"
)
## same order?
all(rownames(significant.expset)==unique(db.annotation$PROBEID))
```

```
## [1] TRUE
```

```r
dup.ids <- db.annotation$PROBEID[table(db.annotation$PROBEID) > 1] %>%
  unique %>%
  sort

db.annot.mult.mapping.significant <- db.annotation %>%
  group_by(PROBEID) %>%
  summarise(PROBEID=PROBEID,
            genes = paste0(SYMBOL, collapse = "|")) %>%
  dplyr::slice(1)

featureData(significant.expset) <- AnnotatedDataFrame(db.annot.mult.mapping)

## assign gene names from hgu133plus2.db with the featureData from earlier
## again genes can be on multiple rows if they profile more than 1 subtype, but this is a small percent
sig.genes.df.with.names <- dplyr::left_join(x=sig.genes.df, y=featureData(significant.expset)@data, by=

## order by pval
sig.genes.df.with.names <- sig.genes.df.with.names[order(sig.genes.df.with.names$avg.p),]

## since there are duplicate gene probes in a single group
## i.e. cMCL has SOX11 twice keep the lowest pvalue row
## so we can obtain top 5 + and - for each group for 40 total
grouped.by.type <- sig.genes.df.with.names %>% group_by(type)
unique.sig.genes.df.with.names <- grouped.by.type[!duplicated(grouped.by.type[,c(1,5)]),]
```

```
## since we want to group of 5 for + and - we can create a direction for grouping
unique.sig.genes.df.with.names$direction <- ifelse(unique.sig.genes.df.with.names$avg.fc>0, "positive",

## some genes are NA so lose those
unique.sig.genes.df.with.names <- unique.sig.genes.df.with.names[!unique.sig.genes.df.with.names$genes=

## group and slice 5 for each subtype up and down
diff.exp.genes.dir <- unique.sig.genes.df.with.names %>%
  group_by(type, direction)

diff.exp.genes <- diff.exp.genes.dir %>%
  slice_min(avg.p, n=5)
diff.exp.genes
```

```
## # A tibble: 40 x 6
## # Groups:   type, direction [8]
##    type  probe           avg.p avg.fc genes      direction
##    <chr> <chr>           <dbl>  <dbl> <chr>      <chr>
##  1 CLL   229487_at    3.55e-36 -30.6  EBF1       negative
##  2 CLL   219049_at    3.52e-20 -31.4  CSGALNACT1 negative
##  3 CLL   203906_at    9.50e-20  -4.56 IQSEC1     negative
##  4 CLL   221004_s_at 2.27e-19  -4.95 ITM2C      negative
##  5 CLL   210448_s_at 6.68e-19  -4.75 P2RX5      negative
##  6 CLL   202709_at    2.10e-37  18.3  FMOD       positive
##  7 CLL   215285_s_at 3.07e-35   4.21 PHTF1      positive
##  8 CLL   1562587_at   2.15e-32  18.3  CLNK       positive
##  9 CLL   202922_at    2.22e-32   3.67 GCLC       positive
## 10 CLL   230551_at    3.85e-32  10.1  KSR2       positive
## # ... with 30 more rows
```

Just by looking at CLL and cMLC top 5 positively regulated genes we can see they match the Figure 2 in the paper by the authors for the subtype signatures. As well as some of the other positively regulated notables for these 2 subtypes.

```
diff.exp.genes[(diff.exp.genes$type=="CLL" | diff.exp.genes$type=="cMCL")
                & diff.exp.genes$direction=="positive", ]
```

```
## # A tibble: 10 x 6
## # Groups:   type, direction [2]
##    type  probe           avg.p avg.fc genes    direction
##    <chr> <chr>           <dbl>  <dbl> <chr>    <chr>
##  1 CLL   202709_at    2.10e-37  18.3  FMOD     positive
##  2 CLL   215285_s_at 3.07e-35   4.21 PHTF1    positive
##  3 CLL   1562587_at   2.15e-32  18.3  CLNK     positive
##  4 CLL   202922_at    2.22e-32   3.67 GCLC     positive
##  5 CLL   230551_at    3.85e-32  10.1  KSR2     positive
##  6 cMCL  204915_s_at 3.18e-55  12.0  SOX11    positive
##  7 cMCL  228266_s_at 3.13e-45   3.68 HDGFL3   positive
##  8 cMCL  230441_at    3.08e-38   2.31 PLEKHG4B positive
##  9 cMCL  230424_at    2.97e-29   1.96 NREP     positive
## 10 cMCL  222101_s_at 3.10e-27   2.89 DCHS1    positive
## other notable genes in CLL and cMCL that are up-regulated in each
diff.exp.genes.notables <- diff.exp.genes.dir[(diff.exp.genes.dir$type=="CLL" |
                                                  diff.exp.genes.dir$type=="cMCL") &
                                                   diff.exp.genes.dir$avg.fc>0,]
notables <- diff.exp.genes.notables %>% slice_min(avg.p, n=15)
notables[c(6:10, 21:29),]
```

```
## # A tibble: 14 x 6
## # Groups:   type, direction [2]
##    type  probe          avg.p avg.fc genes   direction
##    <chr> <chr>          <dbl>  <dbl> <chr>   <chr>
##  1 CLL   202421_at   4.40e-31  9.19  IGSF3   positive
##  2 CLL   221558_s_at 1.01e-29 38.6   LEF1    positive
##  3 CLL   204135_at   1.29e-28  8.73  FILIP1L positive
##  4 CLL   215100_at   2.91e-28 10.4   ADTRP   positive
##  5 CLL   234362_s_at 1.81e-26 14.1   CTLA4   positive
##  6 cMCL  202551_s_at 2.66e-25  2.58  CRIM1   positive
##  7 cMCL  210830_s_at 6.14e-25  8.21  PON2    positive
##  8 cMCL  202806_at   4.96e-23  2.99  DBN1    positive
##  9 cMCL  215807_s_at 4.12e-22  0.747 PLXNB1  positive
## 10 cMCL  223627_at   1.98e-21  0.927 MEX3B   positive
## 11 cMCL  209570_s_at 5.60e-20  1.32  NSG1    positive
## 12 cMCL  91816_f_at  1.97e-19  2.16  MEX3D   positive
## 13 cMCL  222640_at   5.06e-19  0.863 DNMT3A  positive
## 14 cMCL  203240_at   5.47e-19  4.05  FCGBP   positive
```

Use the `RDAVIDWebService` package to annotate the differentially expressed profiling genes.

First get Go IDs and associated Go terms and make dataframe. Then in addition will make separate dataframes including gene names for the following other annotations:

1. KEGG_PATHWAY 2. REACTOME_PATHWAY 3. MINT 4. INTACT 5. UCSC_TFBS

```
library(RDAVIDWebService)
david <- DAVIDWebService$new(email="bwiley4@jh.edu", url="https://david.ncifcrf.gov/webservice/services
result <- addList(david, diff.exp.genes$probe,
                  idType="AFFYMETRIX_3PRIME_IVT_ID",
                  listName="Top5All", listType="Gene")

## see 39 of the 40 probe ids were annotated
david
```

```
## DAVIDWebService object to access DAVID's website.
## User email:  bwiley4@jh.edu
## Available Gene List/s:
##      Name Using
## 1 Top5All     *
## Available Specie/s:
##                Name Using
## 1 Homo sapiens(39)     *
## Available Background List/s:
##            Name Using
## 1 Homo sapiens     *
## get GO and Paths/interactions, need to set at least 1 annotation
## that has 100% of the IDs so there are rows for each annotation = number of IDs
## that is pretty silly
setAnnotationCategories(david, c("GOTERM_CC_DIRECT",
                                 "KEGG_PATHWAY", "REACTOME_PATHWAY",
                                 "MINT", "INTACT", "UCSC_TFBS", "ENSEMBL_GENE_ID"))

## get annotations selected, returns sparse matrix for your probes for each annotation
## need novel way to convert below
annotTable <- RDAVIDWebService::getFunctionalAnnotationTable(david)
memberships <- RDAVIDWebService::membership(annotTable)
cat("Gene Names\n")
```

```
## Gene Names
```

```
annotTable@Genes ## gene names
```

```
##              ID
## 1     236226_at
## 2     224733_at
## 3     204511_at
## 4     203906_at
## 5   207480_s_at
## 6     208717_at
## 7     203075_at
## 8   204915_s_at
## 10    235020_at
## 11    235779_at
## 12  214366_s_at
## 13  219340_s_at
## 14    219049_at
## 15    236984_at
## 16 1553102_a_at
## 17   1562587_at
## 18  222101_s_at
## 19    219646_at
## 20    229487_at
## 21    202709_at
## 22    202922_at
## 23  228266_s_at
## 24  221004_s_at
## 25    230551_at
## 26    203072_at
## 27    230424_at
## 28    230441_at
## 29    204004_at
## 30  207000_s_at
## 31  210448_s_at
## 32  215285_s_at
## 33  201204_s_at
## 34  201825_s_at
## 35    226627_at
## 36  210423_s_at
## 37    218272_at
## 38    220118_at
## 39  218312_s_at
## 92    238009_at
## 91    207336_at
##                                                                    Name
## 1                                  B and T lymphocyte associated(BTLA)
## 2            CKLF like MARVEL transmembrane domain containing 3(CMTM3)
## 3              FERM, ARH/RhoGEF and pleckstrin domain protein 2(FARP2)
## 4                                   IQ motif and Sec7 domain 1(IQSEC1)
## 5                                               Meis homeobox 2(MEIS2)
## 6                   OXA1L, mitochondrial inner membrane protein(OXA1L)
## 7                                          SMAD family member 2(SMAD2)
## 8                                                    SRY-box 11(SOX11)
## 10                TATA-box binding protein associated factor 4b(TAF4B)
## 11                                  ZNF790 antisense RNA 1(ZNF790-AS1)
## 12                                  arachidonate 5-lipoxygenase(ALOX5)
## 13                             ceroid-lipofuscinosis, neuronal 8(CLN8)
## 14 chondroitin sulfate N-acetylgalactosaminyltransferase 1(CSGALNACT1)
## 15                         chromosome 4 open reading frame 26(C4orf26)
## 16                            coiled-coil domain containing 69(CCDC69)
## 17                  cytokine dependent hematopoietic cell linker(CLNK)
## 18                                  dachsous cadherin-related 1(DCHS1)
## 19                    differentially expressed in FDCP 8 homolog(DEF8)
## 20                                         early B-cell factor 1(EBF1)
## 21                                                  fibromodulin(FMOD)
## 22                   glutamate-cysteine ligase catalytic subunit(GCLC)
## 23          hepatoma-derived growth factor, related protein 3(HDGFRP3)
## 24                                 integral membrane protein 2C(ITM2C)
## 25                                    kinase suppressor of ras 2(KSR2)
## 26                                                    myosin IE(MYO1E)
## 27                         neuronal regeneration related protein(NREP)
## 28      pleckstrin homology and RhoGEF domain containing G4B(PLEKHG4B)
## 29                                   pro-apoptotic WT1 regulator(PAWR)
## 30               protein phosphatase 3 catalytic subunit gamma(PPP3CC)
## 31                                    purinergic receptor P2X 5(P2RX5)
## 32                  putative homeodomain transcription factor 1(PHTF1)
## 33                                   ribosome binding protein 1(RRBP1)
## 34                       saccharopine dehydrogenase (putative)(SCCPDH)
## 35                                                     septin 8(SEPT8)
## 36                          solute carrier family 11 member 1(SLC11A1)
## 37                           tetratricopeptide repeat domain 38(TTC38)
## 38                    zinc finger and BTB domain containing 32(ZBTB32)
## 39                  zinc finger and SCAN domain containing 18(ZSCAN18)
## 92                                                     SRY-box 5(SOX5)
## 91                                                     SRY-box 5(SOX5)
##         Species
## 1  Homo sapiens
## 2  Homo sapiens
## 3  Homo sapiens
## 4  Homo sapiens
## 5  Homo sapiens
## 6  Homo sapiens
## 7  Homo sapiens
## 8  Homo sapiens
## 10 Homo sapiens
## 11 Homo sapiens
## 12 Homo sapiens
## 13 Homo sapiens
## 14 Homo sapiens
## 15 Homo sapiens
## 16 Homo sapiens
## 17 Homo sapiens
## 18 Homo sapiens
## 19 Homo sapiens
## 20 Homo sapiens
## 21 Homo sapiens
## 22 Homo sapiens
## 23 Homo sapiens
## 24 Homo sapiens
## 25 Homo sapiens
## 26 Homo sapiens
## 27 Homo sapiens
## 28 Homo sapiens
## 29 Homo sapiens
## 30 Homo sapiens
## 31 Homo sapiens
## 32 Homo sapiens
## 33 Homo sapiens
## 34 Homo sapiens
## 35 Homo sapiens
## 36 Homo sapiens
## 37 Homo sapiens
## 38 Homo sapiens
## 39 Homo sapiens
## 92 Homo sapiens
## 91 Homo sapiens
```

```
nrow(annotTable@Genes) ## 40 since I have 4 groups, you should have 10
```

```
## [1] 40
```

```
## confirm all are length of query
nrow(memberships$KEGG_PATHWAY) ## this is 40
```

```
## [1] 40
```

```
all(lapply(memberships, function(x) nrow(x))==nrow(diff.exp.genes)) ## now all are 40 in length good, 1
```

```
## [1] TRUE
```

```
## add full genes names to your results
diff.exp.genes.with.names <- left_join(diff.exp.genes, annotTable@Genes[,1:2], by=c("probe"="ID"))

## BELOW to add column for go ids and columns for go terms
```

```r
## GO comes with a dictionary mapping of ids to terms, pretty nice
go.matrix <- as.data.frame(memberships$GOTERM_CC_DIRECT)
go.dict <- annotTable@Dictionary$GOTERM_CC_DIRECT
## above is not real dict, lists are real dicts in R
go.dict.real.dict <- setNames(split(go.dict[,2], seq(nrow(go.dict))), go.dict[,1])
## Example 'endoplasmic reticulum'
go.dict.real.dict$`GO:0005783`
```

```
## [1] "endoplasmic reticulum"
```

```r
## create adjacency lists from sparse go matrix for ids and use dict for terms
go.ids.per.probe <- apply(go.matrix, 1, function(x) colnames(go.matrix)[which(x==TRUE)])
names(go.ids.per.probe) <- annotTable@Genes[,"ID"]
go.terms.per.probe <- lapply(go.ids.per.probe, function(x) sapply(x, function(y) go.dict.real.dict[[y]]
names(go.terms.per.probe) <- annotTable@Genes[,"ID"]

## make GO ID dataframe and join
## set the collapse to any delimter you want with your annotations
## NCBI David uses comma separated
## This is really a matrix need to convert
## https://stackoverflow.com/questions/4227223/convert-a-list-to-a-data-frame
goid.list <- lapply(go.ids.per.probe, function(x) paste0(x, collapse = ", "))
goid.df <- data.frame(matrix(unlist(goid.list), nrow=length(goid.list), byrow = T), stringsAsFactors = F
colnames(goid.df) <- "goIDs"
rownames(goid.df) <- annotTable@Genes[,"ID"]
goid.df$ID <- annotTable@Genes[,"ID"]
## add go terms to differential expression dataframe
diff.exp.genes.with.names.goids <- left_join(diff.exp.genes.with.names,
                                             goid.df, by=c("probe"="ID"))


# same for terms
goTerm.list <- lapply(go.terms.per.probe, function(x) paste0(x, collapse = ", "))
goTerm.df <- data.frame(matrix(unlist(goTerm.list), nrow=length(goTerm.list), byrow = T),
                        stringsAsFactors = FALSE)
colnames(goTerm.df) <- "goTerms"
rownames(goTerm.df) <- annotTable@Genes[,"ID"]
goTerm.df$ID <- annotTable@Genes[,"ID"]
## add go terms to differential expression dataframe
diff.exp.genes.with.names.goidsAndTerms <- left_join(diff.exp.genes.with.names.goids,
                                                     goTerm.df, by=c("probe"="ID"))


cat(sum(diff.exp.genes.with.names.goidsAndTerms$goIDs != ""), "entries for", "Go IDs")
```

```
## 37 entries for Go IDs
```

```r
cat(sum(diff.exp.genes.with.names.goidsAndTerms$goTerms != ""), "entries for", "Go Terms")
```

```
## 37 entries for Go Terms
```

```r
## so for me two of the probes matched (SOX5) so 37 is really 36

diff.exp.genes.with.names.goidsAndTerms
```

```
## # A tibble: 40 x 9
## # Groups:   type, direction [8]
##    type  probe     avg.p avg.fc genes  direction Name      goIDs     goTerms
##    <chr> <chr>     <dbl>  <dbl> <chr>  <chr>     <chr>     <chr>     <chr>
##  1 CLL   22948~ 3.55e-36 -30.6  EBF1   negative  early B-~ GO:00056~ nucleus
##  2 CLL   21904~ 3.52e-20 -31.4  CSGAL~ negative  chondroi~ GO:00001~ Golgi memb~
##  3 CLL   20390~ 9.50e-20  -4.56 IQSEC1 negative  IQ motif~ GO:00057~ cytoplasm,~
##  4 CLL   22100~ 2.27e-19  -4.95 ITM2C  negative  integral~ GO:00058~ plasma mem~
##  5 CLL   21044~ 6.68e-19  -4.75 P2RX5  negative  purinerg~ GO:00058~ plasma mem~
##  6 CLL   20270~ 2.10e-37  18.3  FMOD   positive  fibromod~ GO:00056~ extracellu~
##  7 CLL   21528~ 3.07e-35   4.21 PHTF1  positive  putative~ GO:00160~ integral c~
##  8 CLL   15625~ 2.15e-32  18.3  CLNK   positive  cytokine~ GO:00056~ intracellu~
##  9 CLL   20292~ 2.22e-32   3.67 GCLC   positive  glutamat~ GO:00160~ integral c~
## 10 CLL   23055~ 3.85e-32  10.1  KSR2   positive  kinase s~ GO:00057~ cytoplasm,~
## # ... with 30 more rows
```

```r
## write go annotation to file
write.table(diff.exp.genes.with.names.goidsAndTerms, file = "GoIDs_and_Terms.txt",
            row.names = F, sep = "\t", quote = F)
```

Repeat for 5 other annotations. See zip folder will all the annotation files.

```r
## repeat for other annotations but make individual dataframes
## i.e. the original columns of diff.exp.genes and new column for
## each of "KEGG_PATHWAY", "REACTOME_PATHWAY", "MINT", "INTACT", "UCSC_TFBS"
memberships <- RDAVIDWebService::membership(annotTable)
names(memberships)
```

```
## [1] "ENSEMBL_GENE_ID"  "GOTERM_CC_DIRECT" "INTACT"           "KEGG_PATHWAY"
## [5] "MINT"             "REACTOME_PATHWAY" "UCSC_TFBS"
```

```r
# make annotation dataframe function
make.annot.df <- function(annotation) {
  matrix <- as.data.frame(memberships[[annotation]])
  annot.per.probe <- apply(matrix, 1, function(x) colnames(matrix)[which(x==TRUE)])
  names(annot.per.probe) <- annotTable@Genes[,"ID"]

  annot.list <- lapply(annot.per.probe, function(x) paste0(x, collapse = ", "))
  annot.df <- data.frame(matrix(unlist(annot.list), nrow=length(annot.list), byrow = T),
                         stringsAsFactors = FALSE)
  colnames(annot.df) <- annotation
  rownames(annot.df) <- annotTable@Genes[,"ID"]
  annot.df$ID <- annotTable@Genes[,"ID"]
  ## add add annotation to differential expression dataframe with full gene names
  diff.exp.genes.with.names.annot <- left_join(diff.exp.genes.with.names,
                                               annot.df, by=c("probe"="ID"))
  cat(sum(diff.exp.genes.with.names.annot[annotation] != ""), "entries for", annotation)
  ## write to file
  write.table(diff.exp.genes.with.names.annot, file = paste0(annotation,".txt"),
              row.names = F, sep = "\t", quote = F)

  diff.exp.genes.with.names.annot

}

## KEGG
kegg.df <- make.annot.df("KEGG_PATHWAY")  ## SOX5 empty so the 14 matches results DAVID.pdf
```

```
## 14 entries for KEGG_PATHWAY
```

```
head(kegg.df)
```

```
## # A tibble: 6 x 8
## # Groups:   type, direction [2]
##   type  probe      avg.p avg.fc genes  direction Name               KEGG_PATHWAY
##   <chr> <chr>      <dbl>  <dbl> <chr>  <chr>     <chr>              <chr>
## 1 CLL   229487~ 3.55e-36 -30.6  EBF1   negative  early B-cell fac~ ""
## 2 CLL   219049~ 3.52e-20 -31.4  CSGAL~ negative  chondroitin sulf~ "hsa01100, h~
## 3 CLL   203906~ 9.50e-20  -4.56 IQSEC1 negative  IQ motif and Sec~ "hsa04144"
## 4 CLL   221004~ 2.27e-19  -4.95 ITM2C  negative  integral membran~ ""
## 5 CLL   210448~ 6.68e-19  -4.75 P2RX5  negative  purinergic recep~ "hsa04020, h~
## 6 CLL   202709~ 2.10e-37  18.3  FMOD   positive  fibromodulin(FMO~ ""
```

```
## REACTOME
reactome <- make.annot.df("REACTOME_PATHWAY") ## SOX5 empty empty so the 15 matches results DAVID.pdf
```

```
## 15 entries for REACTOME_PATHWAY
```

```
head(reactome)
```

```
## # A tibble: 6 x 8
## # Groups:   type, direction [2]
##   type  probe     avg.p avg.fc genes  direction Name       REACTOME_PATHWAY
##   <chr> <chr>     <dbl>  <dbl> <chr>  <chr>     <chr>      <chr>
## 1 CLL   22948~ 3.55e-36 -30.6  EBF1   negative  early B-c~ "R-HSA-381340"
## 2 CLL   21904~ 3.52e-20 -31.4  CSGAL~ negative  chondroit~ "R-HSA-2022870"
## 3 CLL   20390~ 9.50e-20  -4.56 IQSEC1 negative  IQ motif ~ ""
## 4 CLL   22100~ 2.27e-19  -4.95 ITM2C  negative  integral ~ ""
## 5 CLL   21044~ 6.68e-19  -4.75 P2RX5  negative  purinergi~ "R-HSA-139853, R-HSA~
## 6 CLL   20270~ 2.10e-37  18.3  FMOD   positive  fibromodu~ "R-HSA-2022854, R-HS~
```

```
## IntAct, lot of entries here!
intact <- make.annot.df("INTACT") ## again SOX5 twice so really 31 as in DAVID.pdf
```

```
## 32 entries for INTACT
```

```
head(intact)
```

```
## # A tibble: 6 x 8
## # Groups:   type, direction [2]
##   type  probe     avg.p avg.fc genes  direction Name            INTACT
##   <chr> <chr>     <dbl>  <dbl> <chr>  <chr>     <chr>           <chr>
## 1 CLL   22948~ 3.55e-36 -30.6  EBF1   negative  early B-cell f~ "11281:POU clas~
## 2 CLL   21904~ 3.52e-20 -31.4  CSGAL~ negative  chondroitin su~ ""
## 3 CLL   20390~ 9.50e-20  -4.56 IQSEC1 negative  IQ motif and S~ "54101:receptor~
## 4 CLL   22100~ 2.27e-19  -4.95 ITM2C  negative  integral membr~ "4544:melatonin~
## 5 CLL   21044~ 6.68e-19  -4.75 P2RX5  negative  purinergic rec~ "729991:BLOC-1 ~
## 6 CLL   20270~ 2.10e-37  18.3  FMOD   positive  fibromodulin(F~ "10910:SGT1 hom~
```

```
## MINT
mint <- make.annot.df("MINT") ## SOX5 twice so 16 as in DAVID.pdf
```

```
## 17 entries for MINT
```

```
head(mint)
```

```
## # A tibble: 6 x 8
## # Groups:   type, direction [2]
##   type  probe      avg.p avg.fc genes   direction Name                      MINT
##   <chr> <chr>      <dbl>  <dbl> <chr>   <chr>     <chr>                     <chr>
## 1 CLL   229487~ 3.55e-36 -30.6  EBF1    negative  early B-cell factor 1(E~ ""
## 2 CLL   219049~ 3.52e-20 -31.4  CSGALN~ negative  chondroitin sulfate N-a~ ""
## 3 CLL   203906~ 9.50e-20  -4.56 IQSEC1  negative  IQ motif and Sec7 domai~ ""
## 4 CLL   221004~ 2.27e-19  -4.95 ITM2C   negative  integral membrane prote~ ""
## 5 CLL   210448~ 6.68e-19  -4.75 P2RX5   negative  purinergic receptor P2X~ ""
## 6 CLL   202709~ 2.10e-37  18.3  FMOD    positive  fibromodulin(FMOD)        ""
```

```
## UCSC_TFBS
ucsc.tfbs <- make.annot.df("UCSC_TFBS") ## SOX5 twice so 38 as in DAVID.pdf
```

```
## 39 entries for UCSC_TFBS
```

```
head(ucsc.tfbs)
```

```
## # A tibble: 6 x 8
## # Groups:   type, direction [2]
##   type  probe     avg.p avg.fc genes  direction Name          UCSC_TFBS
##   <chr> <chr>     <dbl>  <dbl> <chr>  <chr>     <chr>         <chr>
## 1 CLL   22948~ 3.55e-36 -30.6  EBF1   negative  early B-cell~ AML1, AREB6, ARP1~
## 2 CLL   21904~ 3.52e-20 -31.4  CSGAL~ negative  chondroitin ~ AML1, AREB6, ARP1~
## 3 CLL   20390~ 9.50e-20  -4.56 IQSEC1 negative  IQ motif and~ AML1, AREB6, ARP1~
## 4 CLL   22100~ 2.27e-19  -4.95 ITM2C  negative  integral mem~ AREB6, EVI1, GCNF~
## 5 CLL   21044~ 6.68e-19  -4.75 P2RX5  negative  purinergic r~ AML1, AREB6, ARP1~
## 6 CLL   20270~ 2.10e-37  18.3  FMOD   positive  fibromodulin~ AML1, CDPCR3HD, E~
```