Homework Assignment 3

Jasmine Kwok and Alyssa Keehan

November 22, 2020

```
library(knitr)
# set global chunk options: images will be 7x5 inches
knitr::opts_chunk$set(fig.width=7, fig.height=5)
options(digits = 4)


## indents are for indenting r code as formatted text
## They may need to be adjusted depending on your OS
# if your output looks odd, increase or decrease indent
indent1 = '    '
indent2 = '        '
indent3 = '      '
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ROCR)
library(tree)
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
library(maptree)
```

```
## Loading required package: cluster
```

```
## Loading required package: rpart
```

```
library(class)
library(lattice)
library(ggridges)
library(superheat)
```

```
drug_use <- read_csv('drug.csv',
                     col_names = c('ID','Age','Gender','Education','Country','Ethnicity',
       'Choc','Coke','Crack','Ecstasy','Heroin','Ketamine',
       'Legalh','LSD','Meth','Mushrooms','Nicotine','Semer','VSA'))
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character(),
##   ID = col_double(),
##   Age = col_double(),
##   Gender = col_double(),
##   Education = col_double(),
##   Country = col_double(),
##   Ethnicity = col_double(),
##   Nscore = col_double(),
##   Escore = col_double(),
##   Oscore = col_double(),
##   Ascore = col_double(),
##   Cscore = col_double(),
##   Impulsive = col_double(),
##   SS = col_double()
## )
## i Use `spec()` for the full column specifications.
```

1. Logistic regression for drug use prediction

```
drug_use <- drug_use %>% mutate_at(as.ordered, .vars=vars(Alcohol:VSA))
drug_use <- drug_use %>%
mutate(Gender = factor(Gender, labels=c("Male", "Female"))) %>%
mutate(Ethnicity = factor(Ethnicity, labels=c("Black", "Asian", "White",
"Mixed:White/Black", "Other",
"Mixed:White/Asian",
"Mixed:Black/Asian"))) %>%
mutate(Country = factor(Country, labels=c("Australia", "Canada", "New Zealand",
"Other", "Ireland", "UK", "USA")))
head(drug_use)
```

```
## # A tibble: 6 x 32
##      ID     Age Gender Education Country Ethnicity Nscore Escore  Oscore Ascore
##   <dbl>   <dbl> <fct>      <dbl> <fct>   <fct>      <dbl>  <dbl>   <dbl>  <dbl>
## 1     1  0.498  Female   -0.0592 USA     Mixed:Wh~  0.313 -0.575 -0.583  -0.917
## 2     2 -0.0785 Male      1.98   USA     White     -0.678  1.94   1.44    0.761
## 3     3  0.498  Male     -0.0592 USA     White     -0.467  0.805 -0.847  -1.62
## 4     4 -0.952  Female    1.16   USA     White     -0.149 -0.806 -0.0193  0.590
```

```
## 5     5  0.498  Female    1.98   USA     White      0.735 -1.63  -0.452  -0.302
## 6     6  2.59   Female   -1.23   UK      White     -0.678 -0.300 -1.56    2.04
## # ... with 22 more variables: Cscore <dbl>, Impulsive <dbl>, SS <dbl>,
## #   Alcohol <ord>, Amphet <ord>, Amyl <ord>, Benzos <ord>, Caff <ord>,
## #   Cannabis <ord>, Choc <ord>, Coke <ord>, Crack <ord>, Ecstasy <ord>,
## #   Heroin <ord>, Ketamine <ord>, Legalh <ord>, LSD <ord>, Meth <ord>,
## #   Mushrooms <ord>, Nicotine <ord>, Semer <ord>, VSA <ord>
```

(a). Define a new factor response variable recent_cannabis_use which is "Yes" if a person has used cannabis within a year, and "No" otherwise. This can be done by checking if the Cannabis variable is greater than or equal to CL3. Hint: use mutate with the ifelse command. When creating the new factor set levels argument to levels=c("No", "Yes") (in that order).

```
drug_use <- drug_use %>% mutate(recent_cannabis_use =
                                  factor(ifelse(Cannabis >= "CL3", "Yes", "No"),
                                         levels=c("No", "Yes")))

class(drug_use$recent_cannabis_use)
```

```
## [1] "factor"
```

```
levels(drug_use$recent_cannabis_use)
```

```
## [1] "No"  "Yes"
```

(b). We will create a new tibble that includes a subset of the original variables. We will focus on all variables between age and SS as well as the new factor related to recent cannabis use. Create drug_use_subset.

```
drug_use_subset <- drug_use %>% select(Age:SS, recent_cannabis_use)
```

Split drug_use_subset into a training data set and a test data set called drug_use_train and drug_use_test.The training data should include 1500 randomly sampled observation and the test data should include the remaining observations in drug_use_subset. Verify that the data sets are of the right size by printing dim(drug_use_train) and dim(drug_use_test).

```
# randomly sample to split data into training set and test set
set.seed(1)
train.indices = sample(1:nrow(drug_use_subset), 1500)
drug_use_train <- drug_use_subset[train.indices,]
drug_use_test <- drug_use_subset[-train.indices,]

# verify data sets are the right size
dim(drug_use_train) #1500   13
```

```
## [1] 1500   13
```

```
dim(drug_use_test) #385  13
```

```
## [1] 385  13
```

(c) Fit a logistic regression to model recent_cannabis_use as a function of all other predictors in drug_use_train. Fit this regression using the training data only. Display the results by calling the summary function on the logistic regression object.

```
glm.fit <- glm(recent_cannabis_use ~ ., data=drug_use_train, family=binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = recent_cannabis_use ~ ., family = binomial, data = drug_use_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9072  -0.5971   0.1416   0.5426   2.6600
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   0.94949    0.64574   1.470 0.141457
## Age                          -0.84406    0.09328  -9.049  < 2e-16 ***
## GenderFemale                 -0.55929    0.15715  -3.559 0.000372 ***
## Education                    -0.33389    0.07962  -4.193 2.75e-05 ***
## CountryCanada                13.10904  627.22755   0.021 0.983325
## CountryNew Zealand           -1.16844    0.31848  -3.669 0.000244 ***
## CountryOther                 -0.05676    0.46772  -0.121 0.903412
## CountryIreland               -0.28763    0.67573  -0.426 0.670354
## CountryUK                    -0.43371    0.37043  -1.171 0.241674
## CountryUSA                   -1.75636    0.19262  -9.118  < 2e-16 ***
## EthnicityAsian               -0.67025    0.96037  -0.698 0.485230
## EthnicityWhite                0.74053    0.63843   1.160 0.246081
## EthnicityMixed:White/Black   -0.04713    1.09013  -0.043 0.965515
## EthnicityOther                1.07889    0.76823   1.404 0.160206
## EthnicityMixed:White/Asian    0.72525    1.01565   0.714 0.475178
## EthnicityMixed:Black/Asian   14.27149  766.28165   0.019 0.985141
## Nscore                       -0.10143    0.09034  -1.123 0.261551
## Escore                       -0.13375    0.09559  -1.399 0.161742
## Oscore                        0.71000    0.09137   7.770 7.83e-15 ***
## Ascore                        0.03058    0.08232   0.372 0.710251
## Cscore                       -0.35855    0.09132  -3.926 8.63e-05 ***
## Impulsive                    -0.09043    0.10093  -0.896 0.370290
## SS                            0.58068    0.10836   5.359 8.39e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2077.2  on 1499  degrees of freedom
## Residual deviance: 1202.1  on 1477  degrees of freedom
## AIC: 1248.1
##
## Number of Fisher Scoring iterations: 14
```

2. Decision Tree Models of drug use

```
tree_parameters = tree.control(nobs=nrow(drug_use_train), minsize=10, mindev=1e-3)
```

a)Use 10-fold CV to select the tree which minimizes the cross-validation misclassification rate. Use the function cv.tree, and set the argument FUN=prune.misclass. Note: you do not need to use a do.chunk function since the tree package will do cross validation for you. Find the size of the tree which minimizes the cross validation error. If multiple trees have the same minimum cross validated misclassification rate, set best_size to the smallest tree size with that minimum rate.

```
#number of folds
nfold = 10
set.seed(1)
folds = seq.int(nrow(drug_use_train)) %>% ## sequential obs ids
cut(breaks = nfold, labels=FALSE) %>% ## sequential fold ids
sample ## random fold ids

# fit the model on training set
tree.drug_use = tree(recent_cannabis_use~., data = drug_use_train, control = tree_parameters)

set.seed(3)

#K-fold Cross validation
cv = cv.tree(tree.drug_use, rand=folds, FUN = prune.misclass, K=10)
cv
```

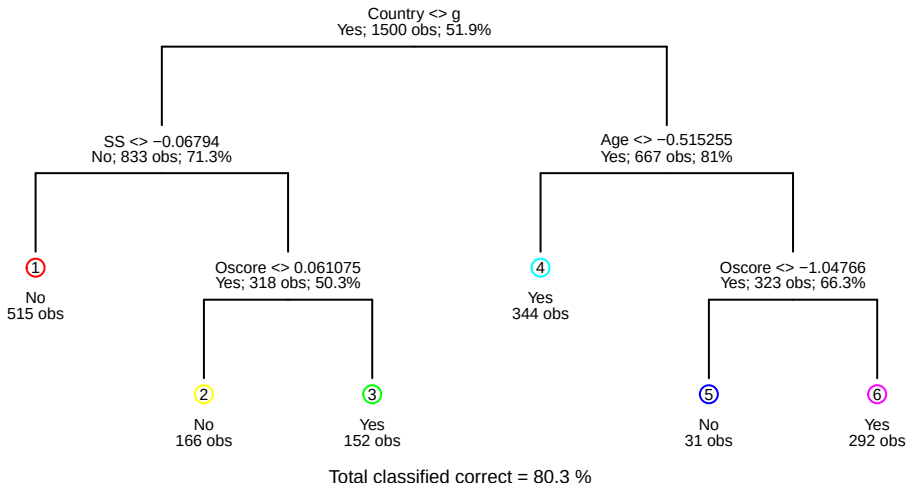
```
## $size
##  [1] 127  81  69  67  57  51  45  42  38  32  29  21  13  10   8   6   4   2   1
##
## $dev
##  [1] 319 319 319 319 319 319 319 319 319 319 319 319 319 319 319 333 370 721
##
## $k
##  [1]        -Inf   0.0000000   0.3333333   0.5000000   1.0000000   1.3333333
##  [7]   1.5000000   1.6666667   2.0000000   2.5000000   2.6666667   2.7500000
## [13]   3.0000000   3.6666667   4.5000000   5.5000000  10.5000000  25.0000000
## [19] 355.0000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
# best size
best.cv=cv$size[max(which(cv$dev==min(cv$dev)))]
best.cv #6
```

```
## [1] 6
```

(b). Prune the tree to the size found in the previous part and plot the tree using the draw.tree function from the maptree package. Set nodeinfo=TRUE. Which variable is split first in this decision tree?

```
# Prune the tree to best size
pruned.drug_use = prune.misclass(tree.drug_use, best = best.cv)

# Plot pruned tree
draw.tree(pruned.drug_use, nodeinfo = TRUE, cex = 0.5)
title("Classification Tree for Drug Use Built on Training Set")
```



The variable that is split first in the decision tree is Country.

(c). Compute and print the confusion matrix for the test data using the function table(truth, predictions) where truth and predictions are the true classes and the predicted classes from the tree model respectively. Note: when generated the predicted classes for the test data, set type="class" in the predict function. Calculate the true positive rate (TPR) and false positive rate (FPR) for the confusion matrix. Show how you arrived at your answer.

```
set.seed(1)

# predict on test set
pred.drug_use = predict(pruned.drug_use, drug_use_test, type = "class")

# Obtain confusion matrix
error = table(pred.drug_use,drug_use_test$recent_cannabis_use)
error
```

```
##
## pred.drug_use  No Yes
```

```
##           No  125  41
##           Yes  40 179
```

```
# True positive rate
TPR = error[2,2]/sum(error[c(1,2),2])
TPR #0.8136
```

```
## [1] 0.8136364
```

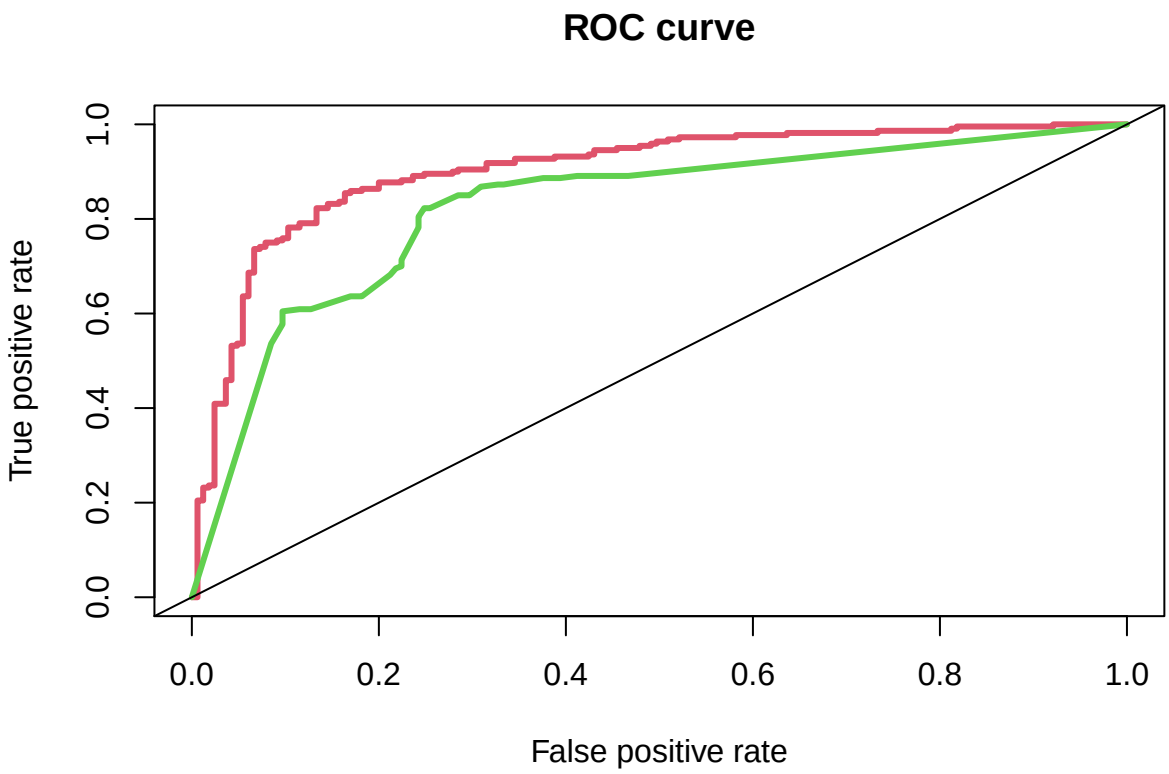
```
# False positive rate
FPR = error[2,1]/sum(error[c(1,2),1])
FPR #0.2424
```

```
## [1] 0.2424242
```

3. Model Comparison

(a). Plot the ROC curves for both the logistic regression fit and the decision tree on the same plot. Use drug_use_test to compute the ROC curves for both the logistic regression model and the best pruned tree model.

```
# Specify type="reponse" to get estimated probabilities
prob_glm = predict(glm.fit, drug_use_test ,type = "response")

#changing the elements of response y from class to vector
pred.drug_use1 = predict(pruned.drug_use, drug_use_test, type = "vector")
#pred.drug_use1 - gives two columns of No and Yes

# first argument is the prob.training, second is true.labels
pred_glm = prediction(prob_glm,drug_use_test$recent_cannabis_use)
pred_tree = prediction(pred.drug_use1[,2],drug_use_test$recent_cannabis_use)

roc1 = performance(pred_glm, measure="tpr", x.measure="fpr")
roc2 = performance(pred_tree, measure="tpr", x.measure="fpr")
plot(roc1, col=2, lwd=3, main="ROC curve")
plot(roc2, col=3, lwd=3, main="ROC curve", add = TRUE)
abline(0,1)
```



(b). Compute the AUC for both models and print them. Which model has larger AUC?

```
# calculate AUC
auc_glm = performance(pred_glm,"auc")@y.values
auc_tree = performance(pred_tree,"auc")@y.values
c(auc_glm,auc_tree)
```

```
## [[1]]
## [1] 0.902562
##
## [[2]]
## [1] 0.8235399
```

The model using logistic regression has a larger AUC value of 0.9026 in comparison to the pruned tree model which has a lower AUC value of 0.8571.

4. Clustering and dimension reduction for gene expression data

```
leukemia_data <- read_csv("leukemia_data.csv")
```

a) The class of the first column of leukemia_data, Type, is set to character by default. Convert the Type column to a factor using the mutate function. Use the table command to print the number of patients with each leukemia subtype. Which leukemia subtype occurs the least in this data?

```
# mutate Type from character to a factor
leukemia_data <-  leukemia_data %>% mutate(Type = factor(Type))
class(leukemia_data$Type)
```

```
## [1] "factor"
```

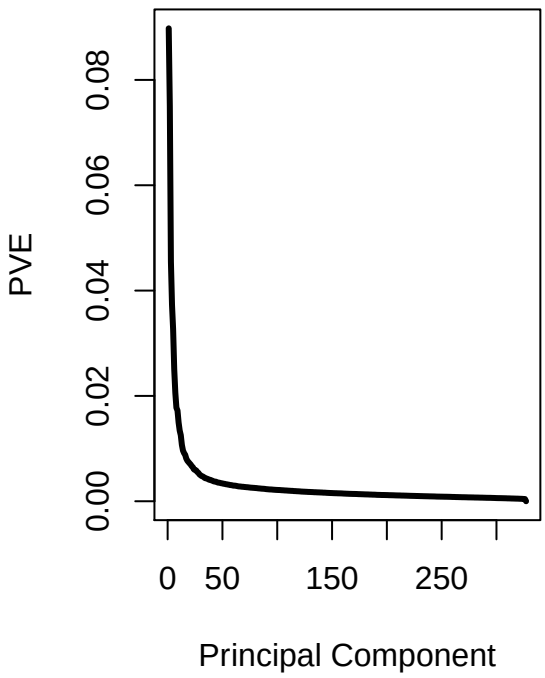
```
# number of patients
#nrow(leukemia_data) #327

#print the number of patients
table(leukemia_data$Type)
```

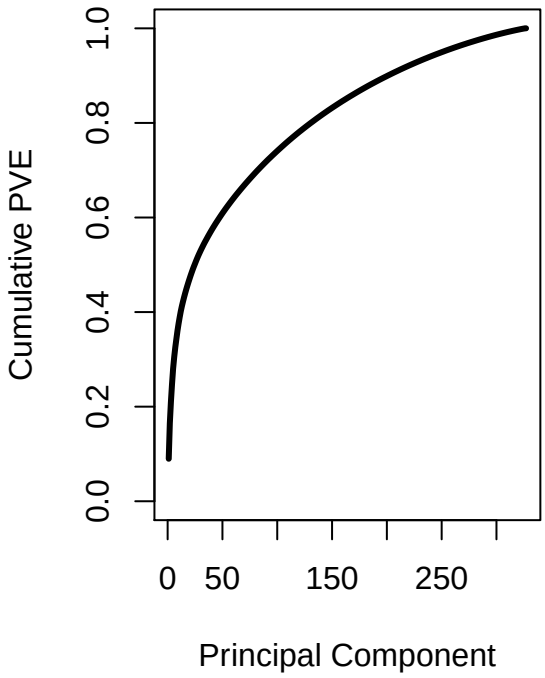
```
##
##    BCR-ABL   E2A-PBX1 Hyperdip50        MLL     OTHERS      T-ALL   TEL-AML1
##         15         27         64         20         79         43         79
```

```
#table(leukemia_data$Type) - sum of total patients 327
```

b) Run PCA on the leukemia data using prcomp function with scale=TRUE and center=TRUE (this scales each gene to have mean 0 and variance 1). Make sure you exclude the Type column when you run the PCA function (we are only interested in reducing the dimension of the gene expression values and PCA doesn't 3 work with categorical data anyway). Plot the proportion of variance explained by each principal component (PVE) and the cumulative PVE side-by-side.

```
leukemia_pca <- prcomp(leukemia_data[,-1], scale =TRUE, center = TRUE)
sdev <- leukemia_pca$sdev
pve <- sdev^2/sum(sdev^2)
cumulative_pve <- cumsum(pve)
## This will put the next two plots side by side
par(mfrow=c(1, 2))

## Plot proportion of variance explained
plot(pve, type="l", lwd=3, xlab="Principal Component",
     ylab="PVE")
plot(cumulative_pve, type="l", lwd=3, xlab="Principal Component", ylab="Cumulative PVE", ylim=c(0,1))
```





c) Use the results of PCA to project the data into the first two principal component dimensions. prcomp returns this dimension reduced data in the first columns of x. Plot the data as a scatter plot using plot function with col=plot_colors. This will color the points according to the leukemia subtype. Add the leukemia type labels to the plot using text with labels argument set to the leukemia type and the col to plot_colors (it may help legibility to make the points on the plot very small by setting cex to a small number). Which group is most clearly separated from the others along the PC1 axis? Which genes have the highest absolute loadings for PC1 (the genes that have the largest weights in the weighted average used to create the new variable PC1)? You can find these by taking the absolute values of the first principal component loadings and sorting them. Print the first 6 genes in this sorted vector using the head function.

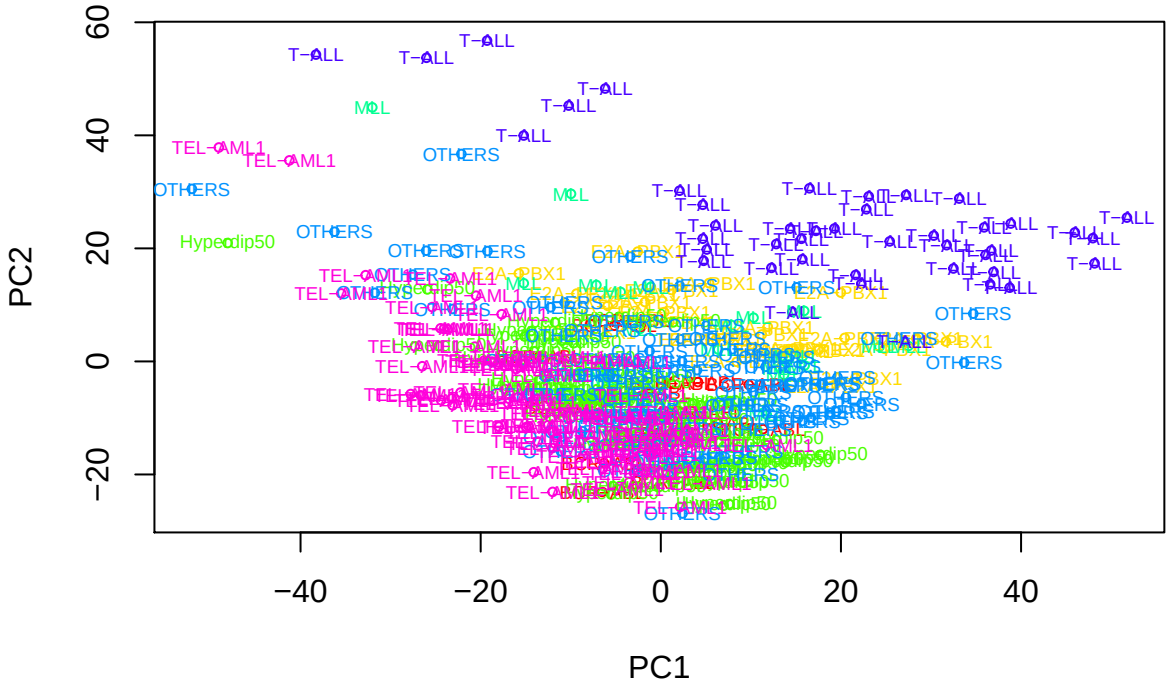
```
# extract x for PC1 and PC2
head(leukemia_pca$x[,1:2])
```

```
##             PC1        PC2
## [1,] -10.414898  -8.107584
## [2,]  -1.377304  -5.386586
## [3,]  -3.720294   7.290351
## [4,]   1.159456  -3.953322
## [5,]  -5.177178   6.313023
## [6,]  11.346689 -11.979690
```

```
# absolute value of PC1 and sort
head(sort(abs(leukemia_pca$rotation[,1]),decreasing = TRUE))
```

```
##     SEMA3F       CCT2       LDHB      COX6C     SNRPD2       ELK3
## 0.04517148 0.04323818 0.04231619 0.04183480 0.04179822 0.04155821
```

```
#SEMA3F    CCT2    LDHB    COX6C  SNRPD2    ELK3

# defining plot_colors
rainbow_colors <- rainbow(7)
plot_colors <- rainbow_colors[leukemia_data$Type]

#plot data as a scattered plot
plot(leukemia_pca$x[,1:2], col=plot_colors, cex=0.6)
text(leukemia_pca$x[,1:2], labels=leukemia_data$Type,cex=0.6, col = plot_colors)
```



Along the PC1 axis the group T-ALL is the most clearly separated type. The SRSF8 gene has the highest absolute loading value of 0.04517 for PC1.

f) Use the filter command to create a new tibble leukemia_subset by subsetting to include only rows for which Type is either T-ALL, TEL-AML1, or Hyperdip50. Compute a euclidean distance matrix between the subjects using the dist function and then run hierarchical clustering using complete linkage. Plot two dendrograms based on the hierarchical clustering result. In the first plot, force 3 leukemia types to be the labels of terminal nodes, color the branches and labels to have 3 groups and rotate the dendrogram counter-clockwise to have all the terminal nodes on the right. In the second plot, do all the same things except that this time color all the branches and labels to have 5 groups. Please make sure library dendextend is installed. Hint: as.dendrogram, set_labels, color_branches, color_labels and plot(..., horiz = TRUE) may be useful.

```
# create new subset which only includes rows with Type T-ALL, TEL-AML1, or Hyperdip50
#leukemia_subset <- leukemia_data %>% filter(Type==c("T-ALL", "TEL-AML1", "Hyperdip50"))
```

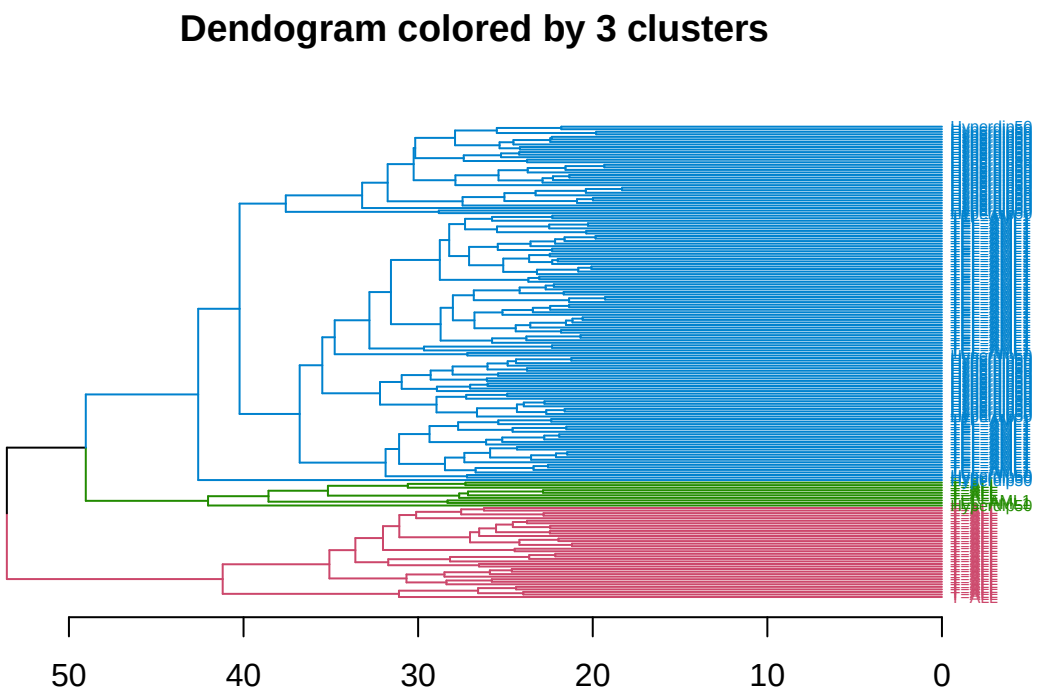
```
leukemia_subset <- leukemia_data %>% filter(Type=="T-ALL" | Type=="TEL-AML1"|Type=="Hyperdip50")
leukemia_subset
```

```
## # A tibble: 186 x 3,142
##    Type  FCGRT FCGRT_1 `31444_s_at` TMSB10  PGK1 EIF3K `31503_at` HDLBP TXNIP
##    <fct> <dbl>   <dbl>        <dbl>  <dbl> <dbl> <dbl>      <dbl> <dbl> <dbl>
##  1 Hype~  8.99    9.57         10.5   10.4  7.21  9.32       8.10  8.44  9.30
##  2 Hype~  9.32    9.80         10.3   10.5  7.76  9.59       8.44  8.58  9.85
##  3 Hype~  7.87    8.70         10.5   11.2  7.47  9.45       7.91  8.83 10.5
##  4 Hype~  8.20    9.09         10.6   10.6  7.74  9.35       8.45  8.43  9.63
##  5 Hype~  9.20    9.50         10.1   10.5  8.30  9.98       8.92  8.56  9.59
##  6 Hype~  9.07    8.95         10.6   10.8  8.42  9.47       8.44  7.88  9.83
##  7 Hype~  8.49    8.91         10.4   10.4  7.65  9.15       8.34  8.59  9.34
##  8 Hype~  8.62    9.31         10.2   10.7  8.13  9.68       8.67  8.04 10.5
##  9 Hype~  9.48    8.97         10.7   10.8  8.33  9.56       8.54  8.01  9.81
## 10 Hype~  8.62    9.28         10.6   10.7  8.46  9.56       8.82  8.82  9.60
## # ... with 176 more rows, and 3,132 more variables: `31510_s_at` <dbl>,
## #   `31519_f_at` <dbl>, `31522_f_at` <dbl>, HIST1H2BE <dbl>,
## #   `31524_f_at` <dbl>, `31526_f_at` <dbl>, HIST1H2BM <dbl>, RTN4 <dbl>,
## #   GUSBP11 <dbl>, `31600_s_at` <dbl>, VDAC1 <dbl>, NDUFS7 <dbl>, SRP72 <dbl>,
## #   `31673_s_at` <dbl>, TOP1P2 <dbl>, GLUD1 <dbl>, GPR35 <dbl>, HSBP1 <dbl>,
## #   BTF3 <dbl>, DDX11 <dbl>, MARF1 <dbl>, MT4 <dbl>, `31993_f_at` <dbl>,
## #   `32004_s_at` <dbl>, `32007_at` <dbl>, `32408_s_at` <dbl>, RPL15 <dbl>,
## #   ATP6V0E2 <dbl>, SUMO4 <dbl>, TSPO2 <dbl>, OR2B6 <dbl>, S1PR4 <dbl>,
## #   TXNL4A <dbl>, GNA13 <dbl>, AIF1 <dbl>, GM2A <dbl>, HNRNPC <dbl>,
## #   TUBB4B <dbl>, TUBB4B_1 <dbl>, TUG1 <dbl>, `33689_s_at` <dbl>, LONRF1 <dbl>,
## #   IGSF9B <dbl>, VIM <dbl>, `34093_at` <dbl>, `34099_f_at` <dbl>, IGHM <dbl>,
## #   HIST1H2AL <dbl>, SLC6A7 <dbl>, DNTT <dbl>, AKAP17A <dbl>, EFNA3 <dbl>,
## #   FLT3 <dbl>, YWHAZ <dbl>, `34647_at` <dbl>, SSR1 <dbl>, SSR1_1 <dbl>,
## #   COMT <dbl>, `HLA-J` <dbl>, ZNF254 <dbl>, ZNF273 <dbl>, DCUN1D4 <dbl>,
## #   GPRIN2 <dbl>, `35566_f_at` <dbl>, ZNF253 <dbl>, HIST1H2BL <dbl>, LY9 <dbl>,
## #   HIST1H2BN <dbl>, ERG <dbl>, CTCF <dbl>, IRF7 <dbl>, HDGF <dbl>,
## #   PMS2P1 <dbl>, HSP90AA1 <dbl>, `32317_s_at` <dbl>, `HLA-E` <dbl>,
## #   YWHAB <dbl>, IDH2 <dbl>, ALDOA <dbl>, PNMT <dbl>, PKM <dbl>, MRE11 <dbl>,
## #   `32872_at` <dbl>, TCF3 <dbl>, `32877_i_at` <dbl>, `32878_f_at` <dbl>,
## #   PTPRE <dbl>, `32921_at` <dbl>, RIF1 <dbl>, FCMR <dbl>, RIPOR2 <dbl>,
## #   GAB1 <dbl>, `32980_f_at` <dbl>, `33458_r_at` <dbl>, `33499_s_at` <dbl>,
## #   `33500_i_at` <dbl>, `33501_r_at` <dbl>, HBD <dbl>, P2RX1 <dbl>, PNN <dbl>,
## #   ...
```

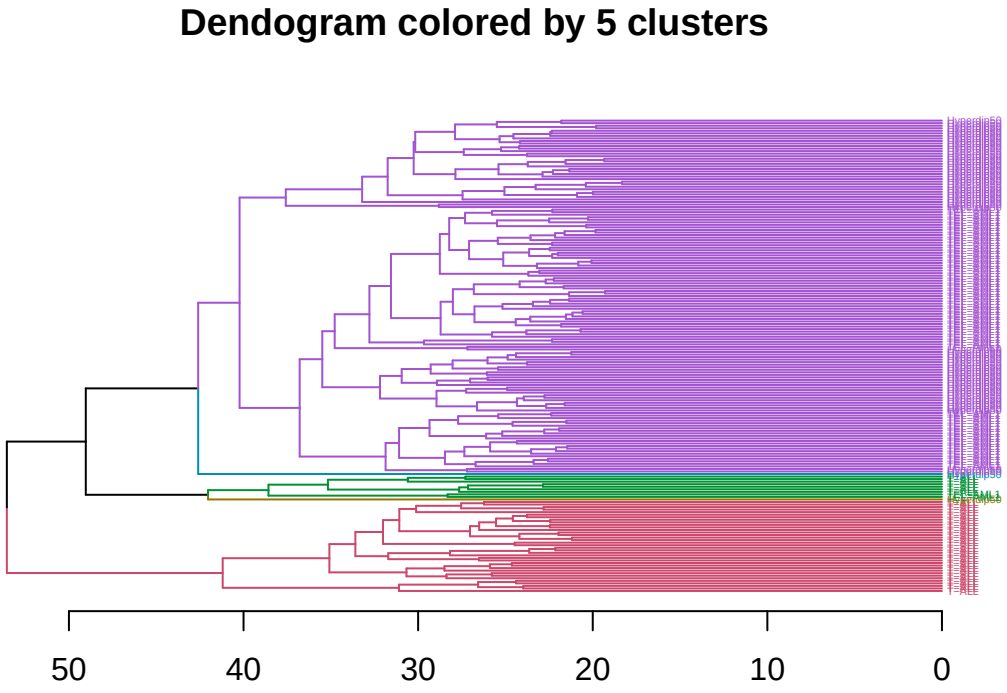
```
# compute euclidean distance
leuk_dist <- dist(leukemia_subset[,-1], method = "euclidean")

set.seed(1)
#hierarchical clustering using complete linkage
leuk.hclust = hclust(leuk_dist)
```

```
# install.packages("dendextend")
library(dendextend)
## dendrogram: branches colored by 3 groups
dend1 = as.dendrogram(leuk.hclust)
# color branches and labels by 3 clusters
dend1 = color_branches(dend1, k=3)
```

```
dend1 = color_labels(dend1, k=3)
# change label size
dend1 = set(dend1, "labels_cex", 0.5)
dend1 = set_labels(dend1, labels=leukemia_subset$Type[order.dendrogram(dend1)])
# Plot dendogram
# rotate the dendrogram counter-clockwise to have all the terminal nodes on the right
plot(dend1, horiz=T, main='Dendogram colored by 3 clusters')
```




```
## dendrogram: branches colored by 5 groups
dend2 = as.dendrogram(leuk.hclust)
# color branches and labels by 5 clusters
dend2 = color_branches(dend2, k=5)
dend2 = color_labels(dend2, k=5)
# change label size
dend2 = set(dend2, "labels_cex", 0.3)
dend2 = set_labels(dend2, labels=leukemia_subset$Type[order.dendrogram(dend2)])
# Plot dendogram
# rotate the dendrogram counter-clockwise to have all the terminal nodes on the right
plot(dend2, horiz=T, main='Dendogram colored by 5 clusters')
```


Dendogram colored by 5 clusters
50
40
30
20
10
0