Homework 3 - 131/231

Due on Friday March 1, 2019 at 11:59 pm

For this homework you will need use the following packages.

```
library(tidyverse)
library(ROCR)
library(tree)
library(maptree)
library(class)
library(lattice)
library(ggridges)
library(superheat)
library(kableExtra)
```

Analyzing drug use

The first half of this homework involves the analysis of drug use. The data set includes a total of 1885 observations on 32 variables. A detailed description of the data set can be found here. For each observation, 12 attributes are known:

- ID: number of record in original database. Used for reference only.
- Age: Age of the participant
- Gender: Gender of the participant (M/F)
- Education: Level of education of the participant
- Country: Country of current residence of the participant
- Ethnicity: Ethnicity of the participant

Many of the covariates have been transformed: some ordinal or categorical variables have been given numeric codes. Part of this problem will involve appropriately re-transforming these variables. The data also contains the following personality measurements:

- Nscore: NEO- FFI- R Neuroticism (Ranging from 12 to 60)
- Escore: NEO- FFI- R Extraversion (Ranging from 16 to 59)
- Oscore: NEO- FFI- R Openness (Ranging from 24 to 60)
- Ascore: NEO- FFI- R Agreeableness (Ranging from 12 to 60)
- Cscore: NEO- FFI- R Conscientiousness (Ranging from 17 to 59)
- Impulsive: Impulsiveness measured by BIS- 11
- SS: Sensation Seeking measured by ImpSS

Finally, participants were questioned concerning their use of 18 legal and illegal drugs (alcohol, amphetamines, amyl nitrite, benzodiazepine, cannabis, chocolate, cocaine, caffeine, crack, ecstasy, heroin, ketamine, legal highs, LSD, methadone, mushrooms, nicotine and volatile substance abuse) and one fictitious drug (Semeron) which was introduced to identify over-claimers. All of the drugs use the class system of CL0-CL6: CL0 = "Never Used", CL1 = "Used over a decade ago", CL2 = "Used in last decade", CL3 = "Used in last year", CL4 = "Used in last month", CL5 = "Used in last week", CL6 = "Used in last day".

```
drug_use <- read_csv('drug.csv',
                     col_names = c('ID','Age','Gender','Education','Country','Ethnicity',
                                   'Nscore','Escore','Oscore','Ascore','Cscore','Impulsive',
                                  'SS','Alcohol','Amphet','Amyl','Benzos','Caff','Cannabis',
                                  'Choc','Coke','Crack','Ecstasy','Heroin','Ketamine',
                                 'Legalh','LSD','Meth','Mushrooms','Nicotine','Semer','VSA'))
```

1. Logistic regression for drug use prediction

This problem has 3 parts for 131 students and 4 parts for 231 students. As mentioned, the data uses some strange encodings for variables. For instance, you may notice that the gender variable has type `double`. Here the value -0.48246 means male and 0.48246 means female. Age was recorded at a set of categories but rescaled to a mean 0 numeric variable (we will leave that variable as is). Similarly education is a scaled numeric quantity (we will also leave this variable as is). We will however, start by transforming gender, ethnicity, and country to factors, and the drug response variables as ordered factors:

```
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")))
```

(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")))
```

(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` with the command:

```
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)`.

```
train_index <- sample(nrow(drug_use_subset), 1500)

drug_use_train <- drug_use_subset[train_index,]
drug_use_test <- drug_use_subset[-train_index, ]

dim(drug_use_train)
```

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

```
dim(drug_use_test)
```

```
## [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.

```
logistic.cannabis =glm(recent_cannabis_use~., data=drug_use_train, family =binomial)

summary(logistic.cannabis)
```

```
##
## Call:
## glm(formula = recent_cannabis_use ~ ., family = binomial, data = drug_use_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8889  -0.5879   0.1328   0.5367   2.6734
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  1.23845    0.70023   1.769  0.07696 .
## Age                         -0.86935    0.09314  -9.334  < 2e-16 ***
## GenderFemale                -0.65676    0.15628  -4.203 2.64e-05 ***
## Education                   -0.37107    0.08028  -4.622 3.80e-06 ***
## CountryCanada               12.99813  611.76995   0.021  0.98305
## CountryNew Zealand          -1.11297    0.33330  -3.339  0.00084 ***
## CountryOther                -0.35186    0.48931  -0.719  0.47209
## CountryIreland              -0.51711    0.68027  -0.760  0.44716
## CountryUK                   -0.82927    0.37800  -2.194  0.02825 *
## CountryUSA                  -1.94555    0.19741  -9.855  < 2e-16 ***
## EthnicityAsian              -1.10916    0.99084  -1.119  0.26296
## EthnicityWhite               0.70795    0.69231   1.023  0.30650
## EthnicityMixed:White/Black   0.08910    1.01587   0.088  0.93011
## EthnicityOther               0.54577    0.81405   0.670  0.50258
## EthnicityMixed:White/Asian  -0.20192    1.09063  -0.185  0.85312
## EthnicityMixed:Black/Asian  14.15495  763.55655   0.019  0.98521
## Nscore                      -0.14137    0.09080  -1.557  0.11950
## Escore                      -0.13023    0.09499  -1.371  0.17037
## Oscore                       0.62539    0.09156   6.831 8.45e-12 ***
## Ascore                       0.16045    0.08258   1.943  0.05202 .
## Cscore                      -0.37264    0.08968  -4.155 3.25e-05 ***
## Impulsive                   -0.08234    0.10158  -0.811  0.41759
## SS                           0.63795    0.10989   5.805 6.42e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 2076.4  on 1499  degrees of freedom
## Residual deviance: 1188.0  on 1477  degrees of freedom
## AIC: 1234
##
## Number of Fisher Scoring iterations: 14
```

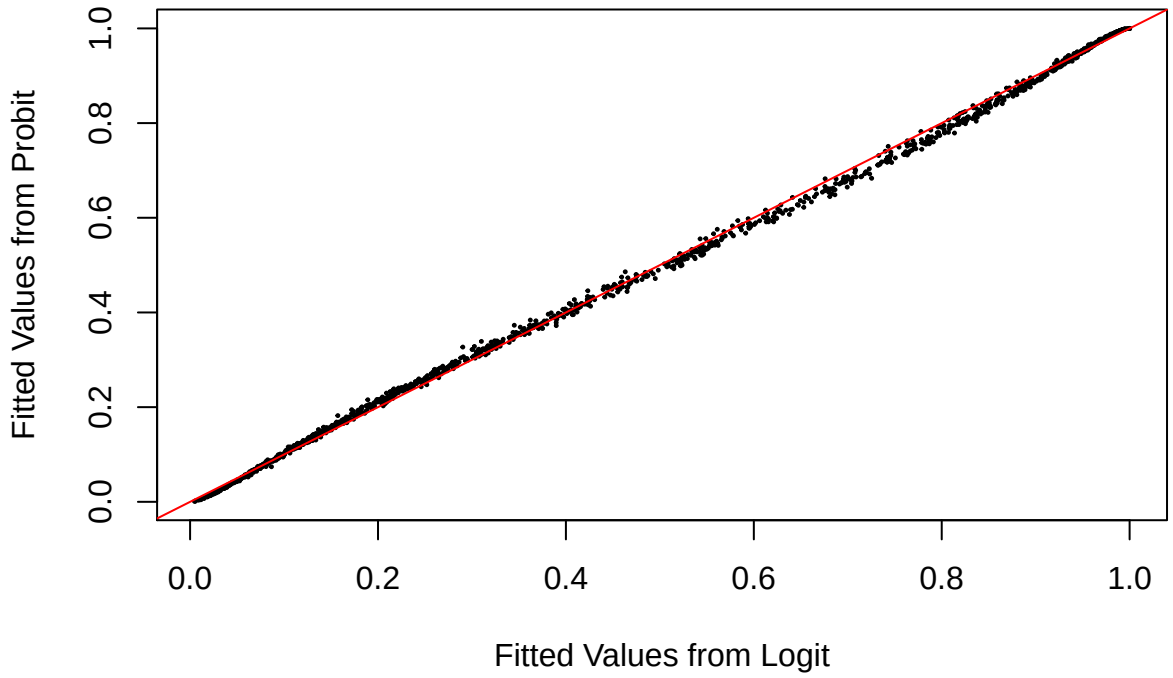
(d). (**231** only). Generalized linear models for binary data involve a link function, g which relates a linear function of the predictors to a function of the probability p: $g(p) = \beta_0 + \beta_1 X_1 + ... \beta_p X_p$. g is a function which maps $p \in [0, 1]$ to $\mathbb{R}$. Logistic regression is based on the *logit* link function, $g(p) = log(p/(1-p))$. In class we mentioned another link function, called the probit: $g(p) = \Phi^{-1}(p)$ where Φ is the cumulative density function of the normal distribution. Another often used link function is the "c-log-log" link: $g(p) = log(-log(1-p))$.

Plot the fitted values for logistic regression fit of the training data on the x-axis and the fitted values for the probit regression on y-axis. In the plot command (assuming you use the base plotting package, not ggplot) set `pch=19` and `cex=0.2` (this makes the points smaller and more legible). Include the line y=x with the command `abline(a=0, b=1, col="red")`. Make another identical plot, this time replacing the y-axis with the predicted values from a cloglog

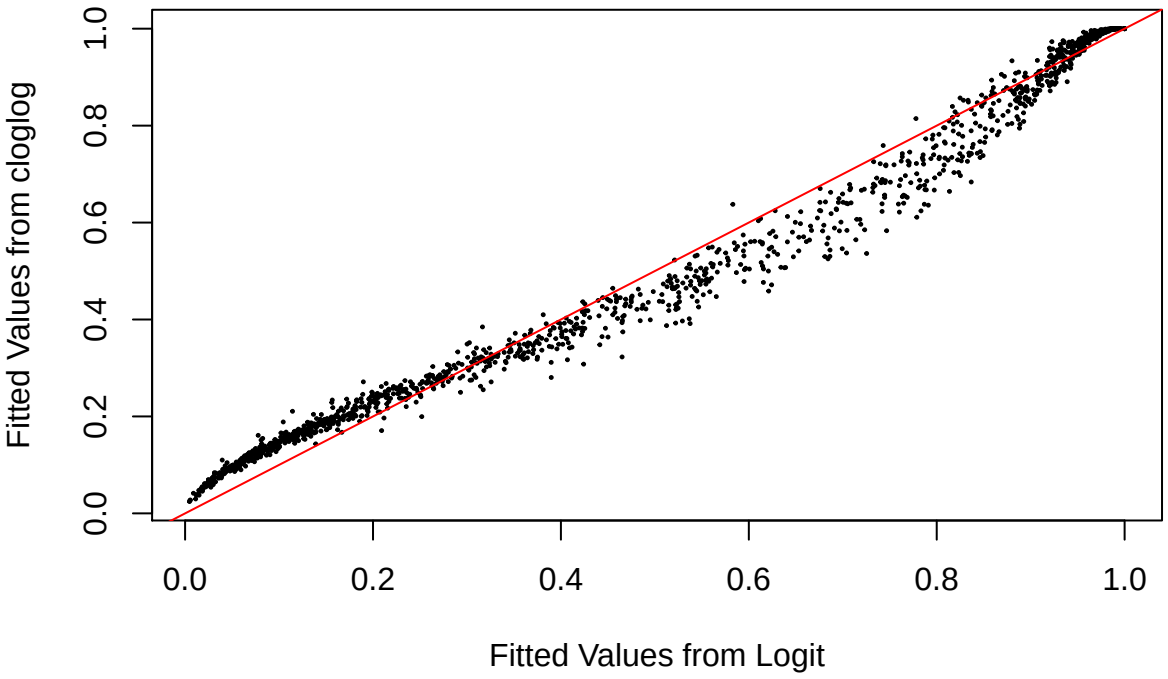
```
probit.cannabis = glm(recent_cannabis_use~., data=drug_use_train, family = binomial(link = "probit"))
cloglog.cannabis = glm(recent_cannabis_use~., data=drug_use_train, family = binomial(link = "cloglog"))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
plot(logistic.cannabis[["fitted.values"]], probit.cannabis[["fitted.values"]], pch = 19, cex = 0.2, xlab
abline(a=0, b=1, col="red")
```




```
plot(logistic.cannabis[["fitted.values"]], cloglog.cannabis[["fitted.values"]], pch = 19, cex = 0.2, xla
abline(a=0, b=1, col="red")
```



Comment on the differences between the estimated probabilities in each plot. Things you should comment on include: 1) which link function (probit or cloglog) leads to predictions that are most similar to the logistic regression predictions? 2) for what range of probabilities are the probit and cloglog predictions values more or less extreme than the logit values? 3) Does either probit or cloglog regression seem to estimate systematically smaller or larger probabilities than the logistic regression for a certain range of probabilities?

The probit function leads to predictions that are more similar to the logistic predictions because the fitted values most closely follow the line y = x. The cloglog fitted vales are more extreme than the logit function for probablities of roughly 0.3 and below and they are less extreme than the logit function for probabilities of about 0.5 to 0.9. As a result, the cloglog seems to systematically estimate smaller or larger probabilities over these certain probability ranges. The probit function tends to have slightly higher values than the logit function for the range of probabilities from 0.3 until close to 0.5 and tends to estimate lower values than the logit function for the range from 0.6 until close to 0.9. These differences between the probit and logit are much less pronounced than in the cloglog vs logit comparison. Due to this minimal difference it feels a little strange to call these differences between probit and logit "systematic" but maybe this description is still fair given the consistently higher or lower points.

Hint: in logistic regression we set `family=binomial(link="logit"")`. To fit probit and cloglog regressions change the value of the `link` argument appropriately.

2. Decision tree models of drug use

This problem has 3 parts for all students.

Construct a decision tree to predict `recent_cannabis_use` using all other predictors in `drug_use_train`. Set the value of the argument `control = tree_parameters` where `tree_parameters` are:

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

This sets the smallest number of allowed observations in each leaf node to 10 and requires a deviance of at least 1e-3 to split a node.

```
cannabis.tree = tree(recent_cannabis_use~.,data = drug_use_train, control = tree_parameters)
summary(cannabis.tree)
```

```
##
## Classification tree:
## tree(formula = recent_cannabis_use ~ ., data = drug_use_train,
##     control = tree_parameters)
## Number of terminal nodes:  124
## Residual mean deviance:  0.446 = 613.7 / 1376
## Misclassification error rate: 0.1007 = 151 / 1500
```

(a). Use 10-fold CV to select the a 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.

```
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

cannabis.tree.cv = cv.tree(cannabis.tree, FUN = prune.misclass, K = 10, rand = folds)

best.cv = min(cannabis.tree.cv$size[which(cannabis.tree.cv$dev==min(cannabis.tree.cv$dev, na.rm = TRUE))]

cannabis.tree.cv$size
```

```
##  [1] 124  79  76  74  68  54  50  45  32  25  20  16  12  10   9   8   7
## [18]   4   2   1
```

```
cannabis.tree.cv$dev
```

```
##  [1] 313 313 313 313 313 313 313 313 313 313 313 313 313 305 306 306 316
## [18] 321 357 716
```

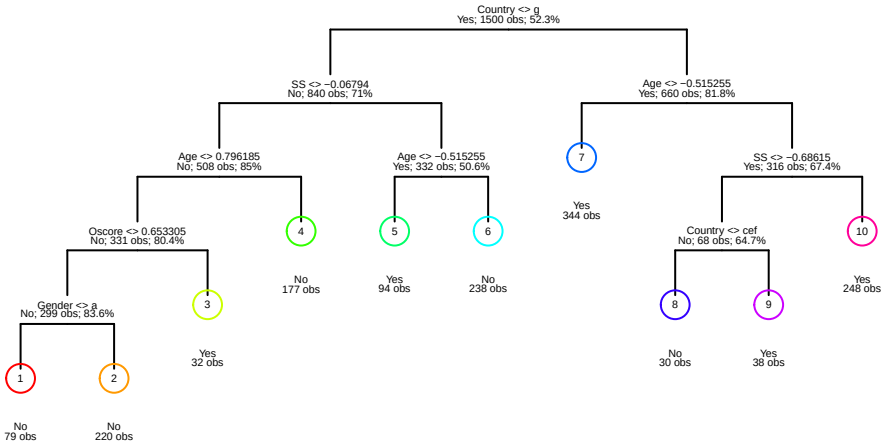
```
best.cv
```

```
## [1] 10
```

The tree that minimizes the cross validation error has 10 nodes.

(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?

```
pruned.cannabis.tree = prune.tree(tree = cannabis.tree, best = best.cv)

draw.tree(tree = pruned.cannabis.tree, size = 2, cex = .3, nodeinfo=TRUE)
```



(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.

```
pruned.predict.test = predict(pruned.cannabis.tree, drug_use_test,
                              type = "class")
testtable = table(pruned.predict.test, drug_use_test$recent_cannabis_use)

testtable
```

```
##
## pruned.predict.test  No Yes
##                  No  137  46
##                  Yes  33 169
```

```
tpr = testtable[2,2]/(testtable[2,2] + testtable[1,2])
fpr = testtable[2,1]/(testtable[2,1] + testtable[1,1])

tpr
```

```
## [1] 0.7860465
```

```
fpr
```

```
## [1] 0.1941176
```

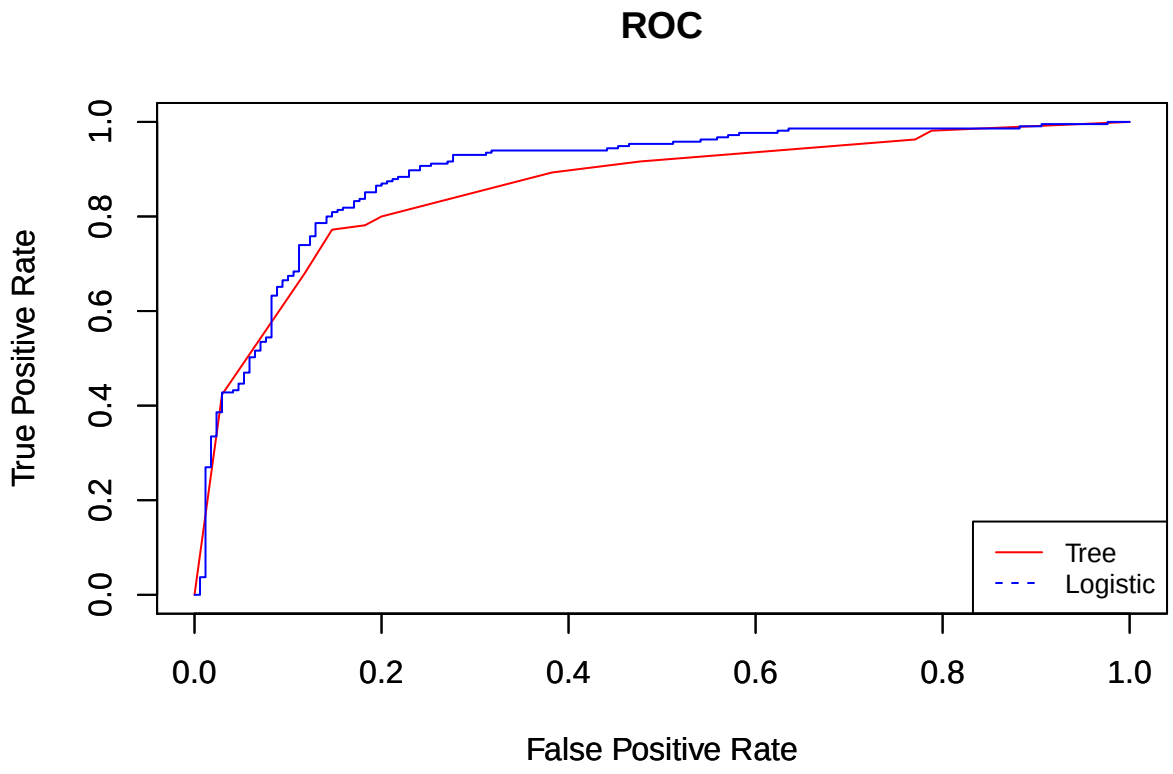
The true positive rate is 0.7860465 and the false positive rate is 0.1941176. TPR = TP/(TP + FN) and FPR = FP/(FP + TN)

3. Model Comparison

This problem has 2 parts for all students.

(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.

```
pruned.predict.test = predict(pruned.cannabis.tree, drug_use_test,  type = "vector")
pred.tree <- prediction(pruned.predict.test[,2], drug_use_test$recent_cannabis_use)
perf.tree <- performance(pred.tree, "tpr", "fpr")

logistic.probs.test = predict(logistic.cannabis, drug_use_test, type="response")
pred.logistic = prediction(logistic.probs.test, drug_use_test$recent_cannabis_use)
perf.logistic <- performance(pred.logistic, "tpr", "fpr")

perf.tree@x.name = perf.logistic@x.name= "False Positive Rate"
perf.tree@y.name = perf.logistic@y.name= "True Positive Rate"

# plot the two ROC curves on the same plot
plot(perf.tree, type = "l", col = "red")
par(new=TRUE)
plot(perf.logistic, type="l", col = "blue")

legend("bottomright", legend=c("Tree", "Logistic"), col=c("red", "blue"), lty=1:2, cex=0.8)
title("ROC", cex = 0.8)
```



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

```
# Calculate AUC
auc.tree = performance(pred.tree, "auc")@y.values
auc.logistic = performance(pred.logistic, "auc")@y.values

auc.tree
```

```
## [[1]]
## [1] 0.8603557
```

Table 1: Number of Patients with Each Sub-Type of Leukemia

Var1	Freq
BCR-ABL	15
E2A-PBX1	27
Hyperdip50	64
MLL	20
OTHERS	79
T-ALL	43
TEL-AML1	79

```
auc.logistic
```

```
## [[1]]
## [1] 0.8925581
```

The AUC for the tree is 0.8603557 which is smaller than the AUC from the logistic model which is 0.8925581.

4. Clustering and dimension reduction for gene expression data

This problem involves the analysis of gene expression data from 327 subjects from Yeoh *et al* (2002). The data set includes abundance levels for 3141 genes and a class label indicating one of 7 leukemia subtypes the patient was diagnosed with. The paper describing their analysis of this data can be found here. Read in the csv data in `leukemia_data.csv`. It is posted on Piazza in the resources tab with the homework:

```
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?

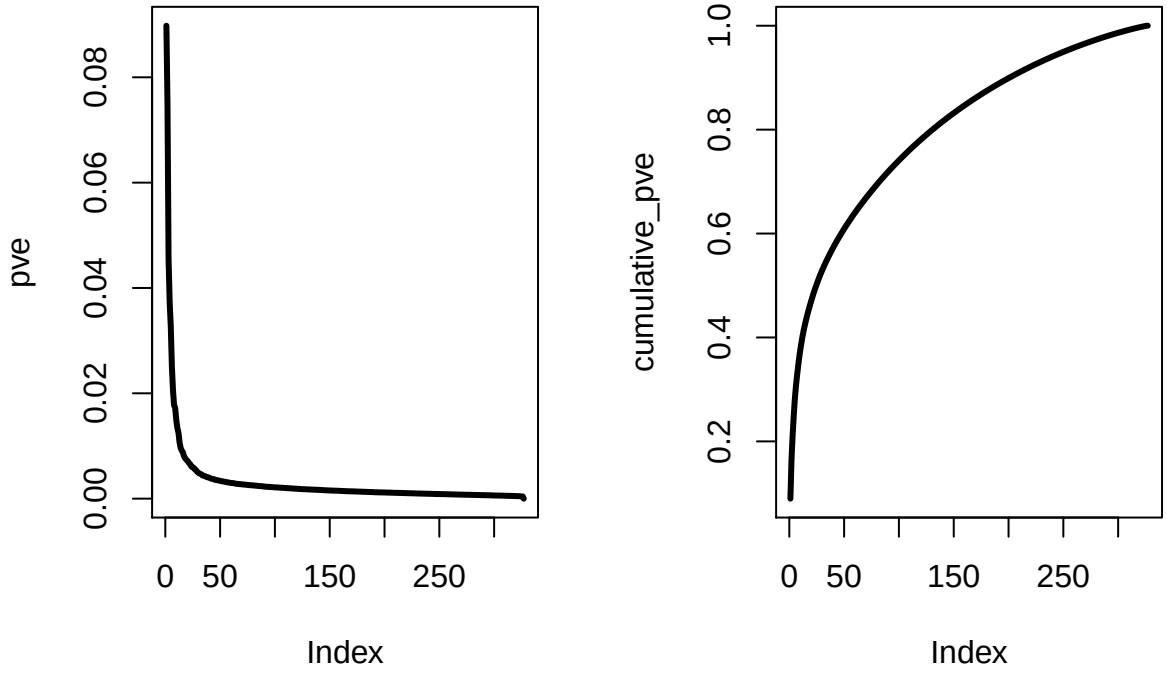
```
leukemia_data_factor <- leukemia_data %>%
  mutate(Type = factor(Type))

type_table = table(leukemia_data_factor$Type)
```

```
kable(type_table, "latex", booktabs = T,
      caption = "Number of Patients with Each Sub-Type of Leukemia", digits = 4) %>%
      kable_styling(bootstrap_options = "striped", full_width = F, position = "center")
```

The BCR-ABL sub-type of leukemia has the fewest patients in this data set at 15 individuals.

(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 work with categorical data anyway). Plot the proportion of variance explained by each principal component (PVE) and the cumulative PVE side-by-side.

```
# Standardize the variables by subtracting mean and divided by standard deviation
scale.leukemia = scale(leukemia_data_factor[,-c(1)], center=TRUE, scale=TRUE)

pr.out = prcomp(scale.leukemia, scale =TRUE, center=TRUE)

pr.var = pr.out$sdev^2 # variance is the square of the standard deviation of the PCA output

pve <- pr.var/sum(pr.var)
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)
plot(cumulative_pve, type="l", lwd=3)
```

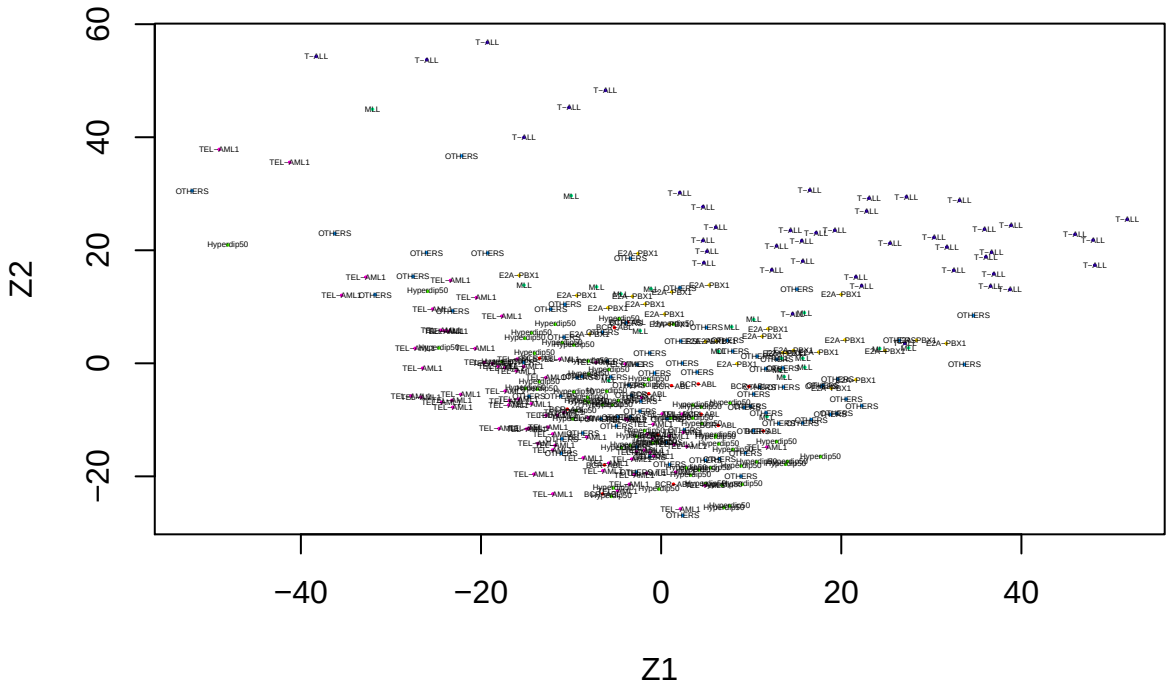


(c). Use the results of PCA to project the data into the first two principal component dimensions. `prcomp` returns this dimension reduced data in the first columns of `x`. Plot the data as a scatter plot using `plot` function with `col=plot_colors` where `plot_colors` is defined

```
rainbow_colors <- rainbow(7)
plot_colors <- rainbow_colors[leukemia_data_factor$Type]
```

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.

```
par(mfrow=c(1, 1))

PC1 = pr.out$x[,1]
PC2 = pr.out$x[,2]
plot(PC1, PC2, col = plot_colors, pch =19, xlab ="Z1",ylab="Z2", cex = 0.1)
text(pr.out$x[,c(1,2)], labels=(leukemia_data_factor$Type), cex = 0.25)
```



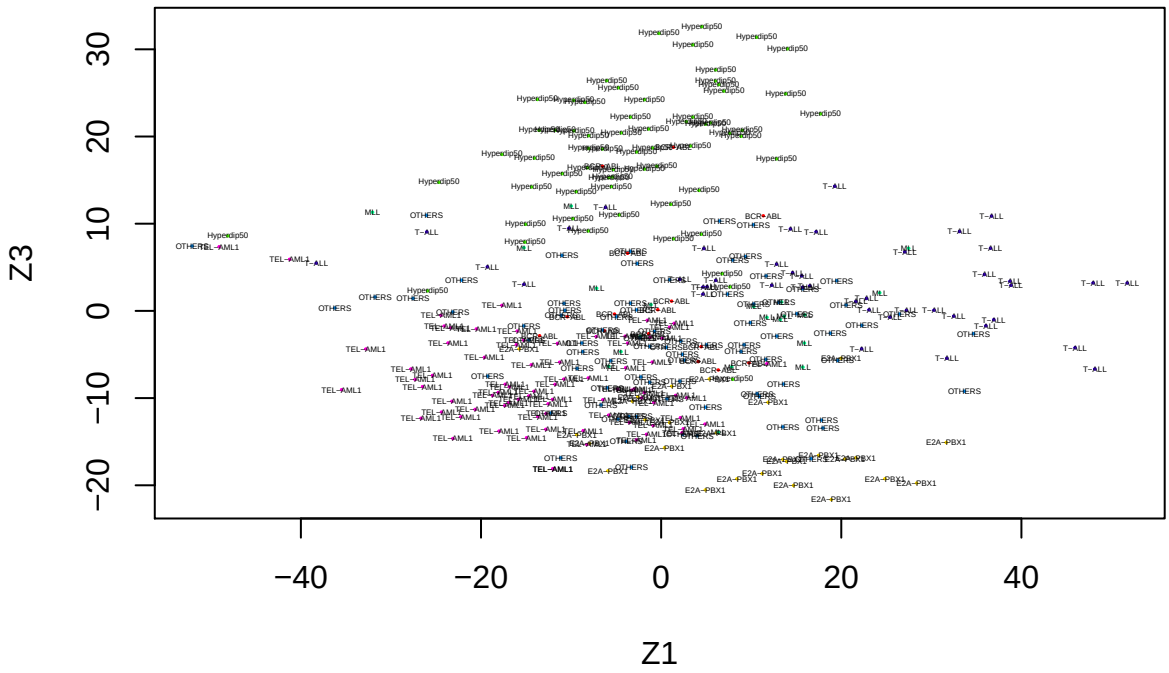

```
abs_pc1 = order(abs(PC1))
head(leukemia_data_factor$Type[abs_pc1], 6)
```

```
## [1] TEL-AML1   Hyperdip50 BCR-ABL    E2A-PBX1   Hyperdip50 OTHERS
## Levels: BCR-ABL E2A-PBX1 Hyperdip50 MLL OTHERS T-ALL TEL-AML1
```

T-ALL seems to be most clearly separated from the other groups on the PC1 axis based on the figure. The top 6 genes that seem to have the highest absolute loadings for PC1 are: TEL-AML1, Hyperdip50, BCR-ABL, E2A-PBX1, Hyperdip50, and OTHERS.

(d). (**231 Only**) PCA orders the principal components according to the amount of total variation in the data that they explain. This does not mean, however, that the principal components are sorted in terms of how useful they are at capturing variation between the leukemia groups. For example, if gene expression varied significantly with age and gender (independent of leukemia status), the first principal components could reflect genetic variation due to age and gender, but not to leukemia. The first scatter plot shows that the second PC is not a good discriminator of leukemia type. See if the 3rd PC is better at discriminating between leukemia types by plotting the data projected onto the *first* and *third* principal components (not the second).

```
par(mfrow=c(1, 1))

PC3 = pr.out$x[,3]
plot(PC1, PC3, col = plot_colors, pch =19, xlab ="Z1",ylab="Z3", cex = 0.1)
text(pr.out$x[,c(1,3)], labels=(leukemia_data_factor$Type), cex = 0.25)
```



This second scatter with the first and third principal components does a better job of separating the different genes as compared to the first scatter plot with the PC1 and PC2. In this second scatter, the separation of T-ALL has slightly worsened than in the previous scatter. However, T-ALL is still fairly well separated in the second scatter. Additionally there is better separation of E2A-PBX1, Hyperdip50, and TEL-AML1 genes in this second scatter. As a result, PC3 seems to be better at discriminating leukemia types than PC2.

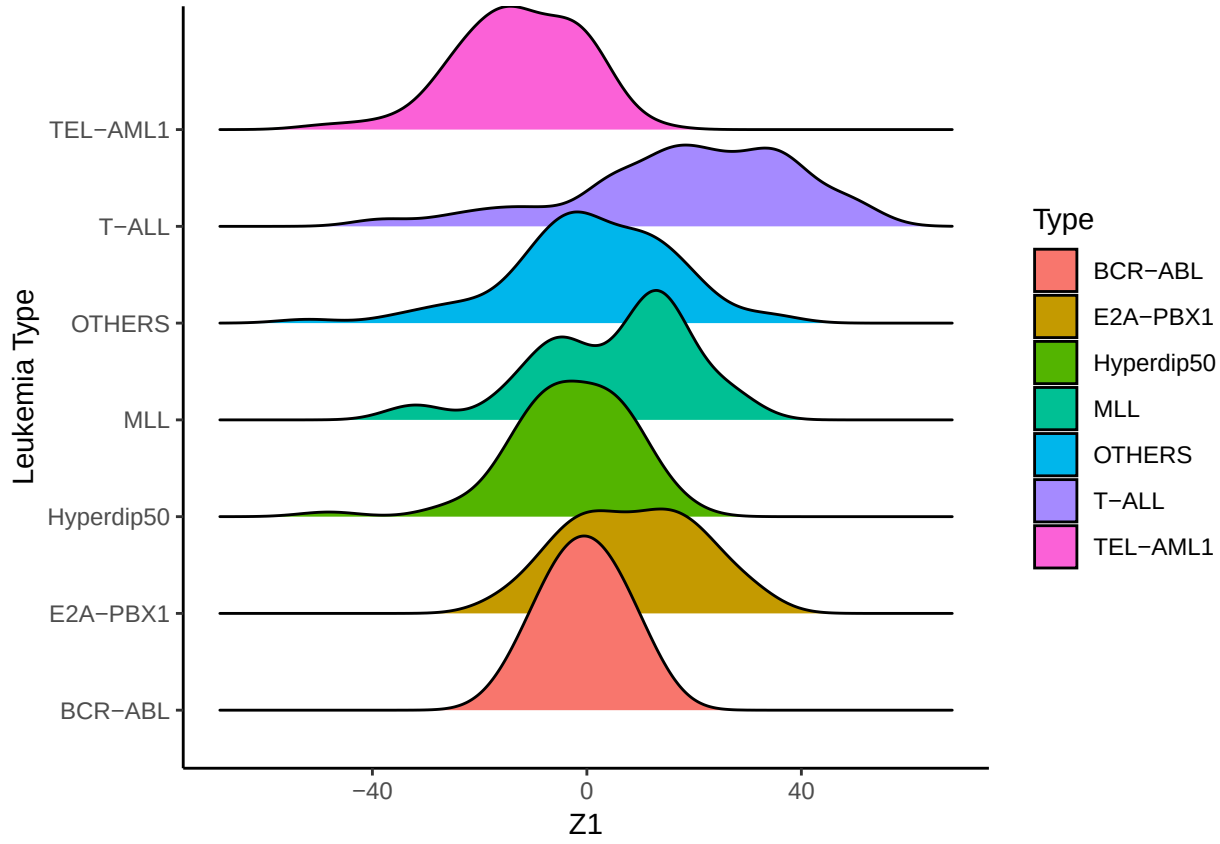
(e.) (**231 Only**) For this part we will be using the `ggridges` library. Create a new tibble where the first column (call it `z1`) is the projection of the data onto the first principal component and the second column is the leukemia subtype (`Type`). Use `ggplot` with `geom_density_ridges` to create multiple stacked density plots of the projected gene expression data. Set the ggplot aesthetics to `aes(x = z1, y = Type, fill = Type)`. Make another identical plot, except replace `z1` with `z3`, the projection of the data onto the third principal component. Identify two leukemia subtypes that are nearly indistinguishable when the gene expression data is projected onto the first PC direction, but easily distinguishable when projected onto the third PC direction.

```
pca_df = data.frame("z1" = PC1, "Type" = leukemia_data_factor$Type)
pca_tibble = as.tibble(pca_df)

plot.new()
ggridge_plot_z1 = ggplot(pca_tibble, aes(x=z1, y = Type, fill = Type)) +
  geom_density_ridges() +
  title("Projection of Leukemia Type Data on PC1 from PCA") +
  labs(x = "Z1", y = "Leukemia Type") +
  theme_classic()
```



```
ggridge_plot_z1
```

```
## Picking joint bandwidth of 5.46
```

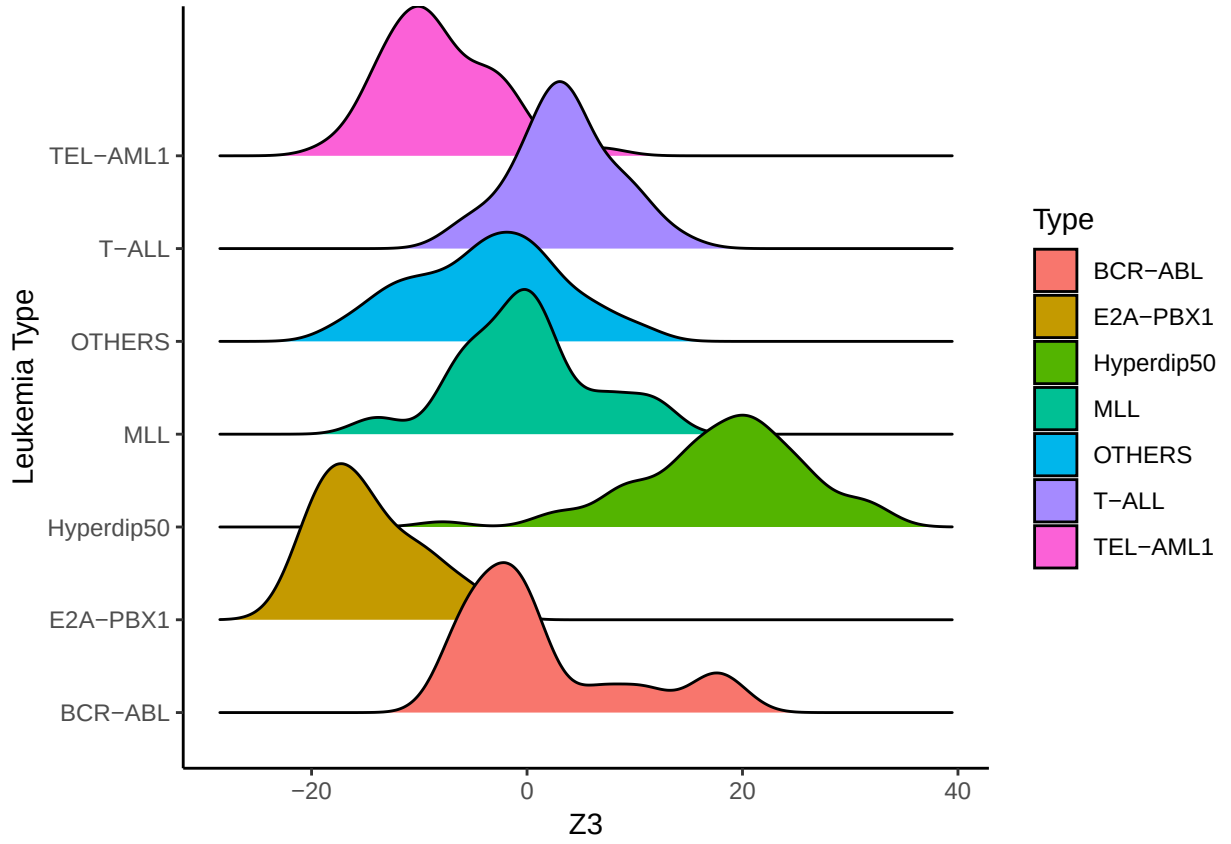



```
plot.new()
pca_df_z3 = data.frame("z3" = PC3, "Type" = leukemia_data_factor$Type)
pca_tibble_z3 = as.tibble(pca_df_z3)

ggridge_plot_z3 = ggplot(pca_tibble_z3, aes(x=z3, y = Type, fill = Type)) +
  geom_density_ridges() +
  title("Projection of Leukemia Type Data on PC3 from PCA") +
  labs(x = "Z3", y = "Leukemia Type") +
  theme_classic()
```

Projection of Leukemia Type Data on PC3 from PCA

```
ggridge_plot_z3
```

```
## Picking joint bandwidth of 2.29
```



E2A-PBX1 and the OTHERS genes are fairly indistinguishable when projected onto the first principal component, but when the data is projected on the third principal component, these two gene categories are fairly distinct from each other. Hyperdip50 and BCR-ABL are also fairly similar when projected on the first principal component, and while there some distinguishing attributes for the densities of these two curves when they're projected on the third principal component, it isn't clear that this projection on the third component is sufficient to distinguish these two gene groups.

(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.

```
library(dendextend)
```

```
##
## ---------------------
## Welcome to dendextend version 1.9.0
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
```

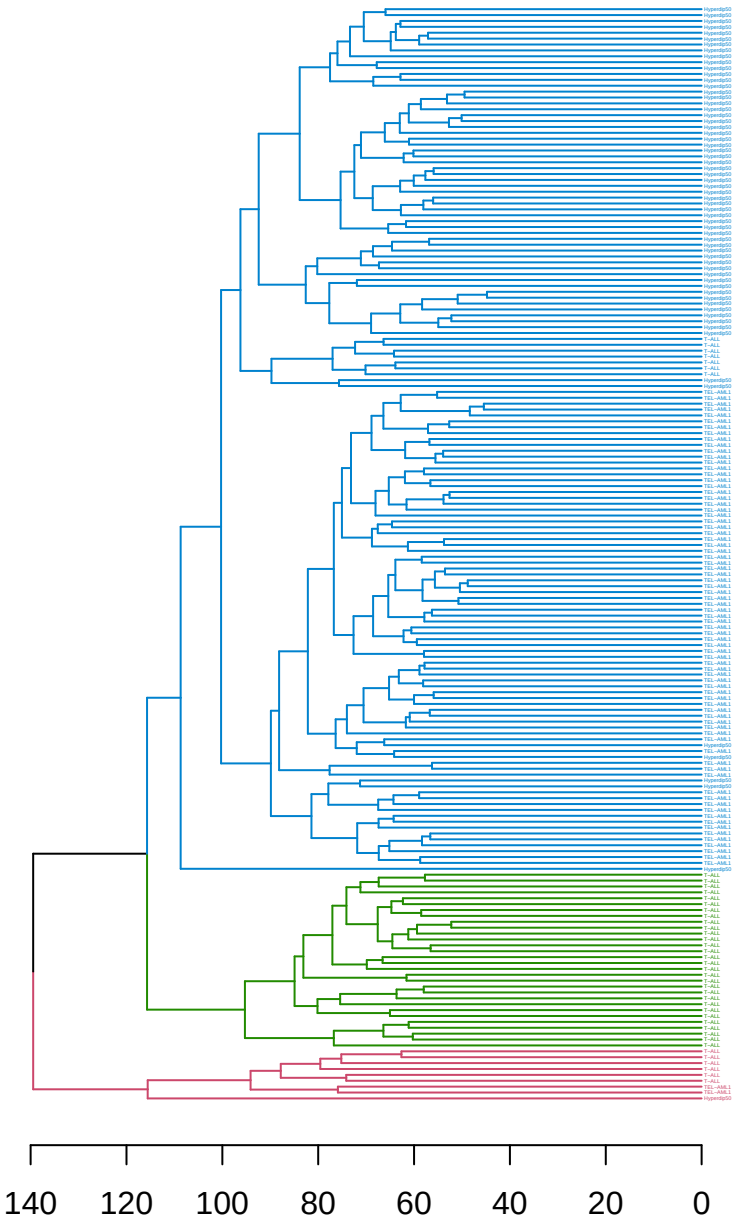
```
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ----------------------
```

```
##
## Attaching package: 'dendextend'
```

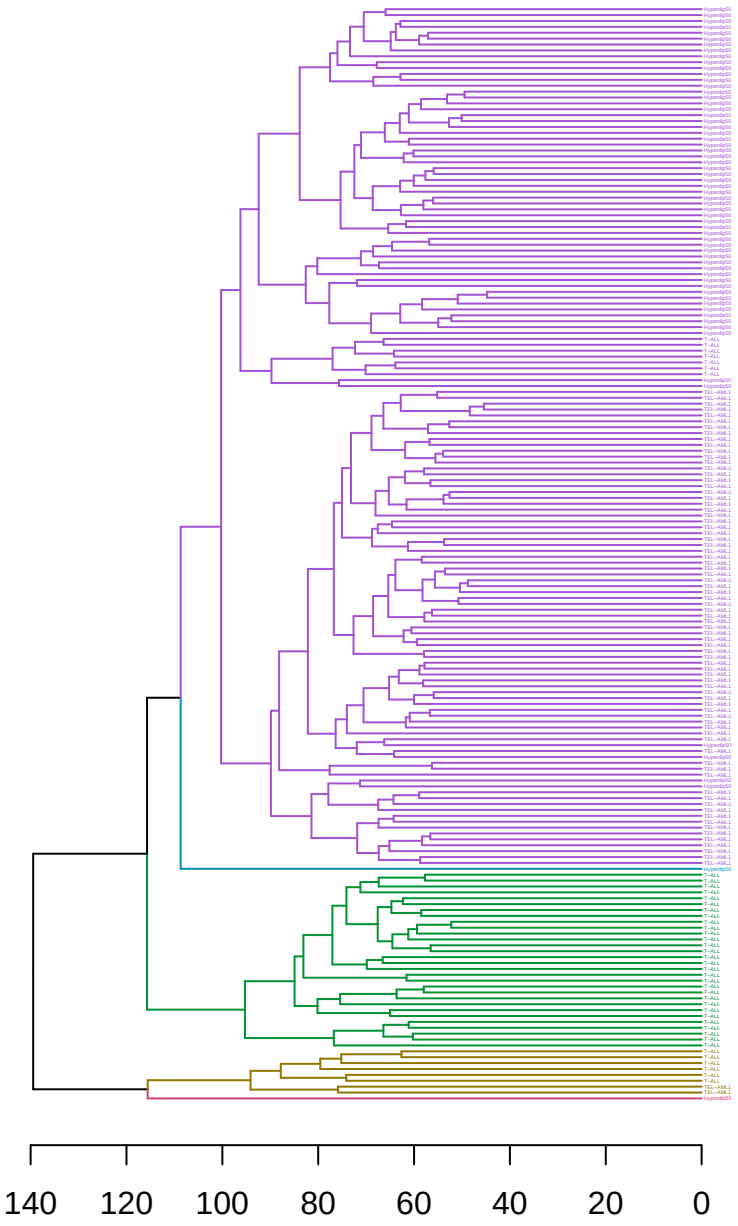
```
## The following object is masked from 'package:rpart':
##
##     prune
```

```
## The following object is masked from 'package:stats':
##
##     cutree
```

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

# do we need to re-scale the subset before computing the distance matrix?
dis  <- dist(scale(leukemia_subset[,-c(1)], center=TRUE, scale=TRUE), method = "euclidean")

set.seed(1)
leuk_hc = hclust(dis, method = "complete")

leukemia_dend3 = leuk_hc %>%
  as.dendrogram() %>%
  color_branches(k = 3) %>%
  color_labels(k = 3)

leukemia_dend3 = set_labels(leukemia_dend3, labels=leukemia_subset$Type[order.dendrogram(leukemia_dend3)
leukemia_dend3= set(leukemia_dend3, "labels_cex", 0.15)
plot(leukemia_dend3, horiz = T)
```



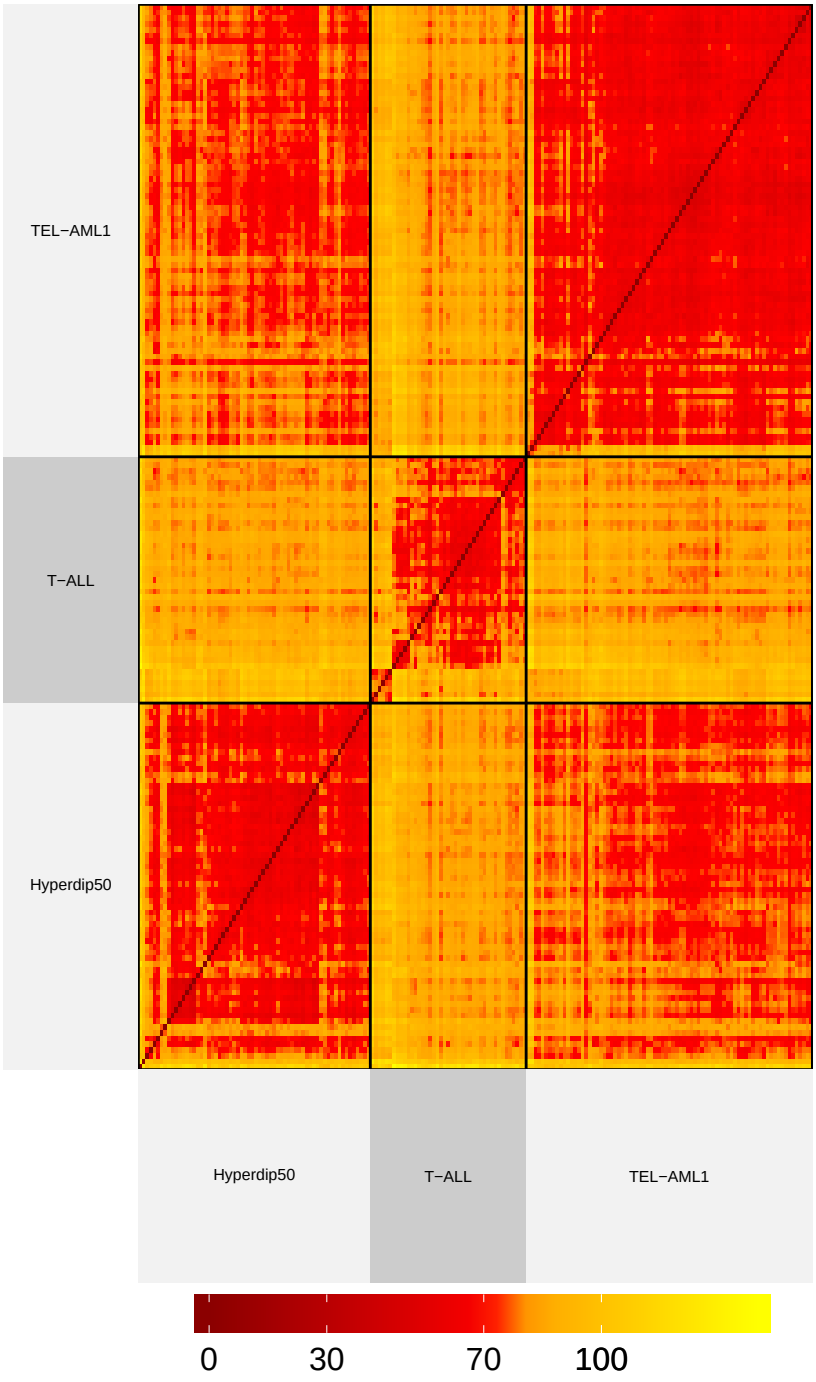

```
leukemia_dend5 = leuk_hc %>%
  as.dendrogram() %>%
  color_branches(k = 5) %>%
  color_labels(k = 5)

leukemia_dend5 = set_labels(leukemia_dend5, labels=leukemia_subset$Type[order.dendrogram(leukemia_dend5)
leukemia_dend5 = set(leukemia_dend5, "labels_cex", 0.15)
plot(leukemia_dend5, horiz = T)
```



(g). (**231 only**). Use `superheat` to plot the distance matrix from the part above. Order the rows and columns by the hierarchical clustering you obtained in the previous part. You should see a matrix with a *block diagonal* structure. The labels (corresponding to leukemia types) will not be available to read on the plot. Print them out by looking at `leukemia_subset$Type` ordered by clustering order. Based on this plot which two leukemia types (of the three in the subset) seem more similar to one another? Hint: use `heat.pal = c("dark red", "red", "orange", "yellow"))` for colorbar specification in `superheat`.

```
types = paste(leukemia_subset$Type[order.dendrogram(leukemia_dend3)])
hc.order = leuk_hc$order

superheat(as.matrix(dis)[hc.order, hc.order],
          membership.rows = types,
          membership.cols = types,
          left.label.text.size = 2,
```

```
        bottom.label.text.size = 2,
        heat.pal = c("dark red", "red", "orange", "yellow"))
```



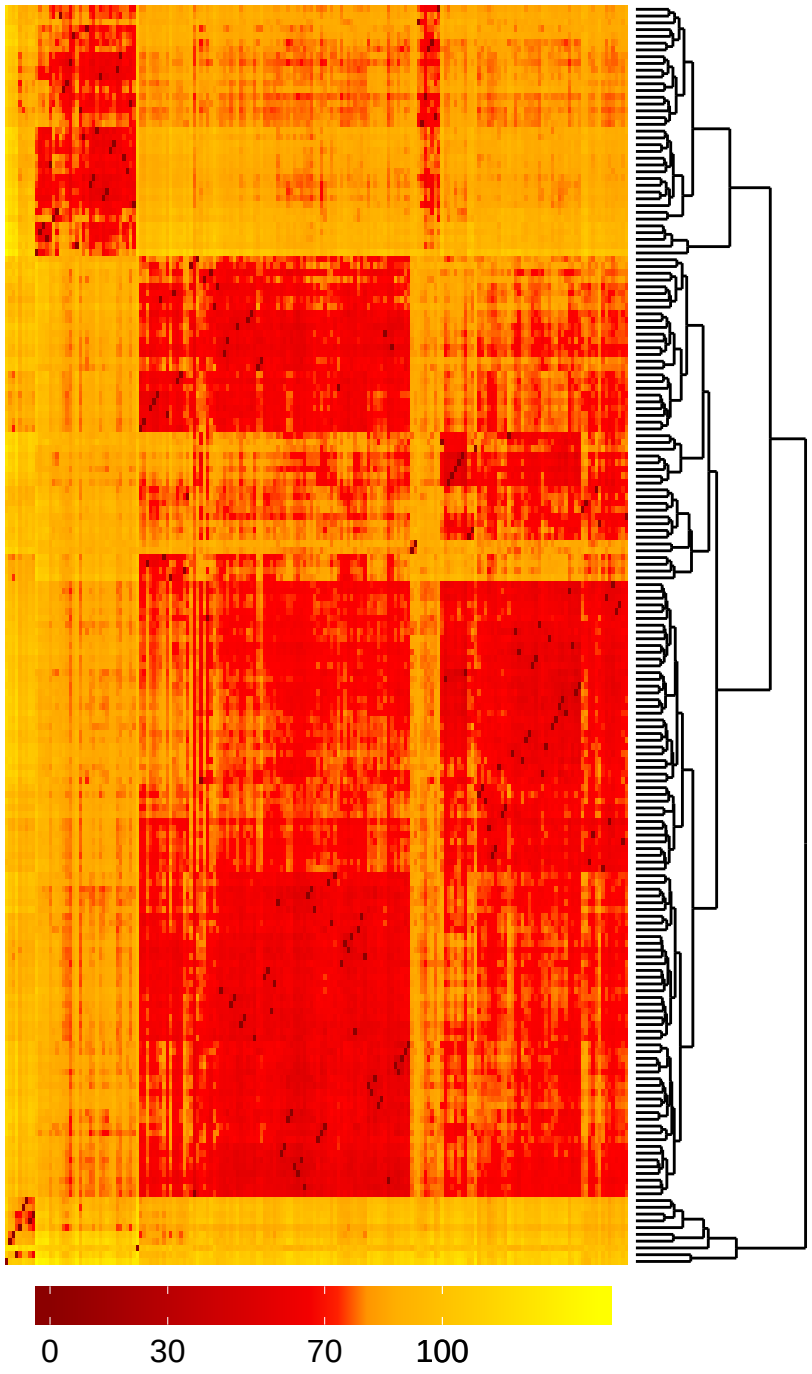
Based on the heatmap, T-ALL and Hyperdip50 seem to have more in common with one another than other subsets of genes. However, T-ALL and TEL-AML1 seem to have only slightly less in common than the T-ALL and Hyperdip50 pairing of genes based on visual inspection.

(h). (**231 only**). You can also use `superheat` to generate a hierachical clustering dendrogram or a kmeans clustering. First, use `leukemia_subset` to run hierachical clustering and draw the dendrogram. Second, use

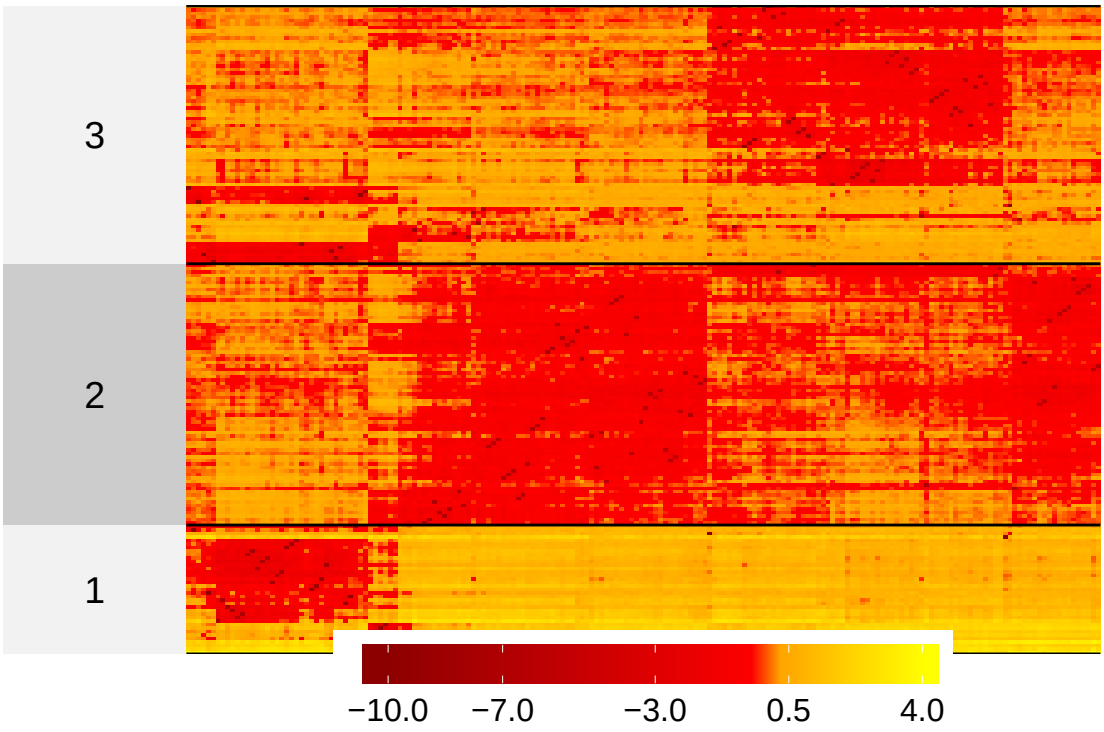
the same dataset to run kmeans clustering with three the optimal number of clusters, and order the genes (columns) based on hierarchical clustering.

Hint: arguments `row.dendrogram`, `clustering.method`, `n.cluster.rows` and `pretty.order.cols` may be useful, please read the argument descriptions before you attempt the problem. The package manual can be found here: https://cran.r-project.org/web/packages/superheat/superheat.pdf

```
hc.heatmap = superheat(as.matrix(dis)[hc.order, hc.order],
          clustering.method = "hierarchical",
          row.dendrogram = T,
          #pretty.order.cols = T,
          # bottom.label = "variable",
          left.label.text.size = 2,
          bottom.label.text.size = 2,
          heat.pal = c("dark red", "red", "orange", "yellow"))
```




```
set.seed(1)
kmeans.heatmap = superheat(as.matrix(dis)[hc.order, hc.order],
          scale = T,
          n.clusters.rows = 3,
          pretty.order.cols = T, # logical specifying to order the columns based on hierarchical cluste
          left.label = "cluster",
          heat.pal = c("dark red", "red", "orange", "yellow")
          )
```


3
2
1
−10.0
−7.0
−3.0
0.5
4.0