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)
```

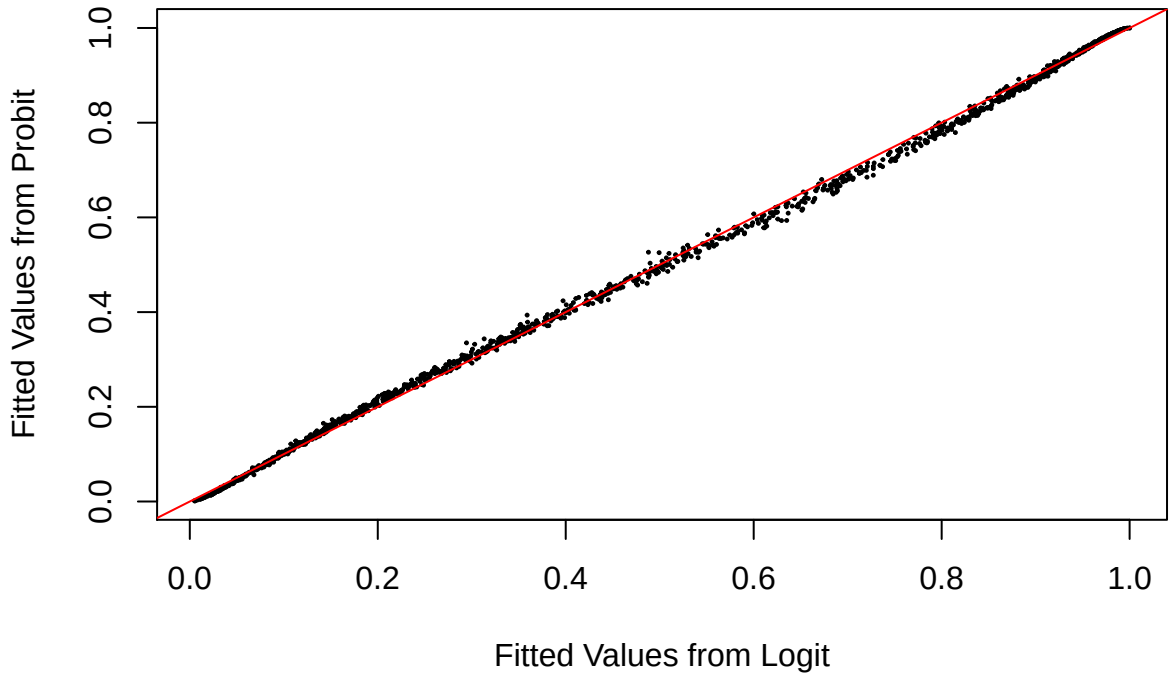
```
##
## Call:
## glm(formula = recent_cannabis_use ~ ., family = binomial, data = drug_use_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9123  -0.6105   0.1467   0.5296   2.6379
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  0.85544    0.71037   1.204  0.22850
## Age                         -0.89797    0.09270  -9.686  < 2e-16 ***
## GenderFemale                -0.73649    0.15472  -4.760 1.93e-06 ***
## Education                   -0.32574    0.07982  -4.081 4.48e-05 ***
## CountryCanada               -1.12260    1.37742  -0.815  0.41507
## CountryNew Zealand          -1.09249    0.32518  -3.360  0.00078 ***
## CountryOther                -0.44176    0.46132  -0.958  0.33826
## CountryIreland              -0.56745    0.70818  -0.801  0.42297
## CountryUK                   -0.73832    0.37091  -1.991  0.04653 *
## CountryUSA                  -1.84994    0.19127  -9.672  < 2e-16 ***
## EthnicityAsian              -1.33750    1.06717  -1.253  0.21009
## EthnicityWhite               1.02801    0.70289   1.463  0.14359
## EthnicityMixed:White/Black   0.53092    1.03652   0.512  0.60850
## EthnicityOther               0.78603    0.82416   0.954  0.34022
## EthnicityMixed:White/Asian   0.44469    1.05080   0.423  0.67215
## EthnicityMixed:Black/Asian  11.64681  376.71489   0.031  0.97534
## Nscore                      -0.10265    0.09039  -1.136  0.25608
## Escore                      -0.23144    0.09682  -2.390  0.01683 *
## Oscore                       0.66097    0.09102   7.262 3.81e-13 ***
## Ascore                       0.07862    0.08257   0.952  0.34105
## Cscore                      -0.24226    0.09322  -2.599  0.00935 **
## Impulsive                   -0.04257    0.10068  -0.423  0.67240
## SS                           0.57335    0.11029   5.198 2.01e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 2076.7  on 1499  degrees of freedom
## Residual deviance: 1204.9  on 1477  degrees of freedom
## AIC: 1250.9
##
## Number of Fisher Scoring iterations: 12
```

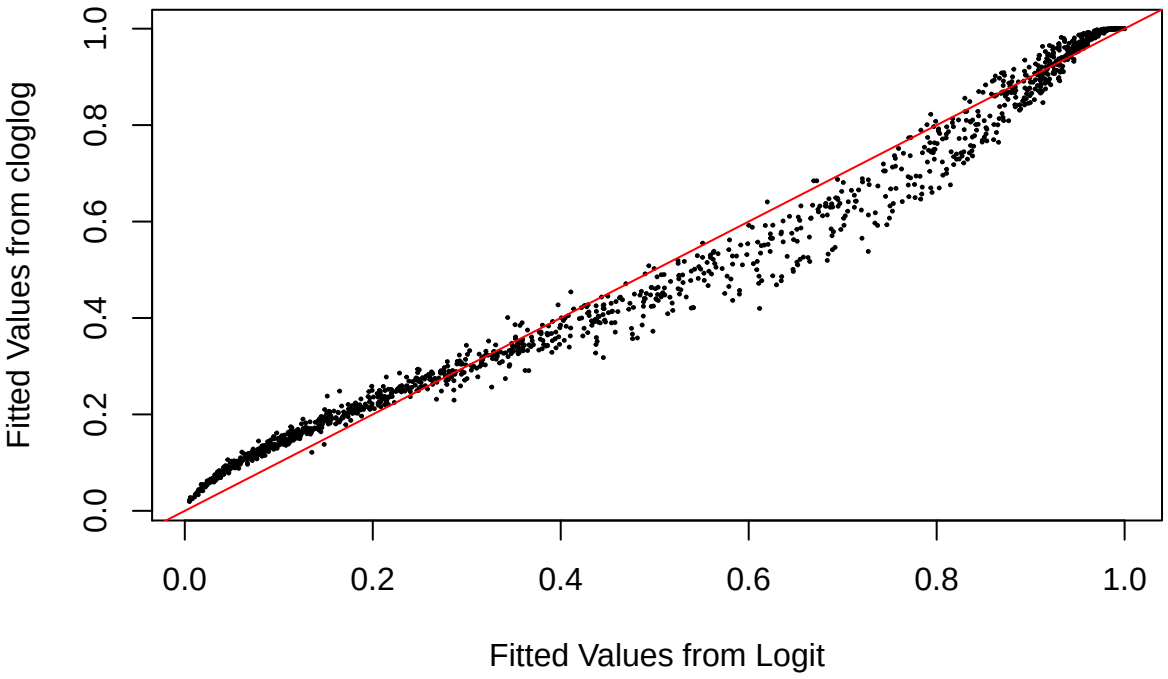
(d). (**231** only). Generalized linear models for binary data involve a link function, g which relates a linear function of the predictors to a function of the probability p: $g(p) = \beta_0 + \beta_1 X_1 + ... \beta_p X_p$. g is a function which maps $p \in [0, 1]$ to $\mathbb{R}$. Logistic regression 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"))

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:  132
## Residual mean deviance:  0.4419 = 604.5 / 1368
## Misclassification error rate: 0.1013 = 152 / 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] 132  84  82  79  75  68  62  56  48  42  38  23  20  15   7   6   4
## [18]   2   1
```

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

```
##  [1] 307 307 307 307 307 307 307 307 307 307 307 307 307 307 307 309 323
## [18] 367 718
```

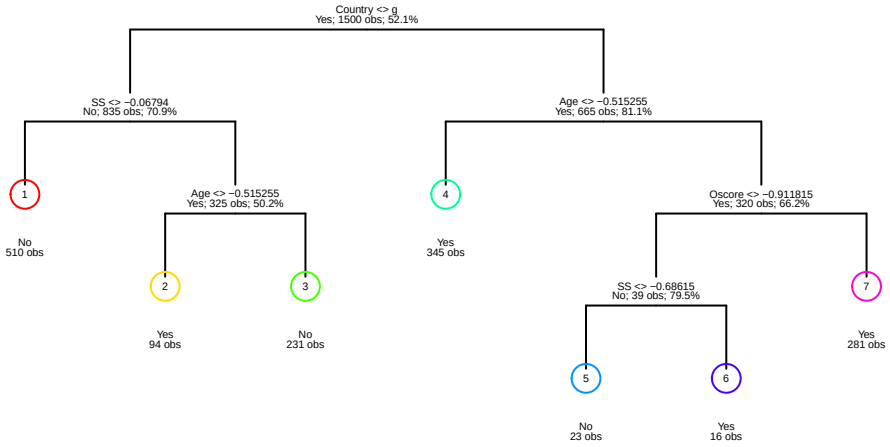
```
best.cv
```

```
## [1] 7
```

The tree that minimizes the cross validation error has 7 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  139  50
##                  Yes  29 167
```

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

tpr
```

```
## [1] 0.7695853
```

```
fpr
```

```
## [1] 0.172619
```

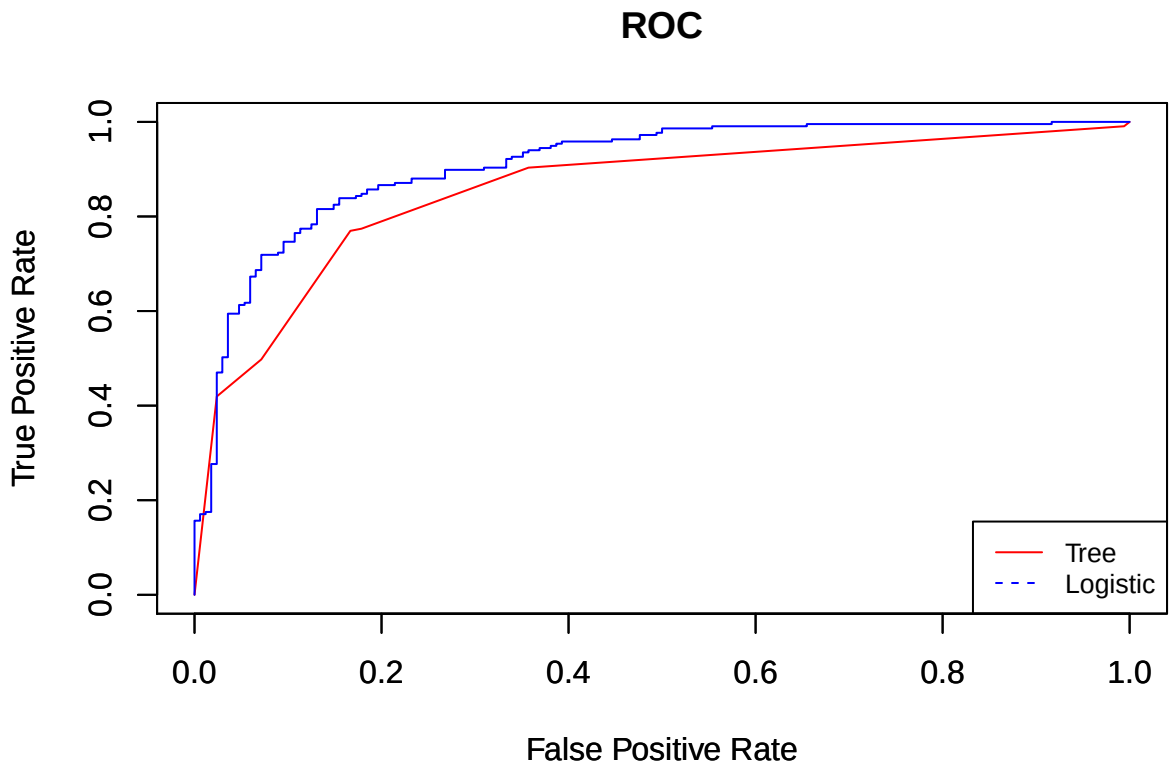
The true positive rate is 0.7695853 and the false positive rate is 0.172619. TPR = TP/(TP + FN) and FPR = FP/(FP + TN)

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] 0.855209
```

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] 0.9096719
```

The AUC for the tree is 0.855209 which is smaller than the AUC from the logistic model which is 0.9096719.

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?

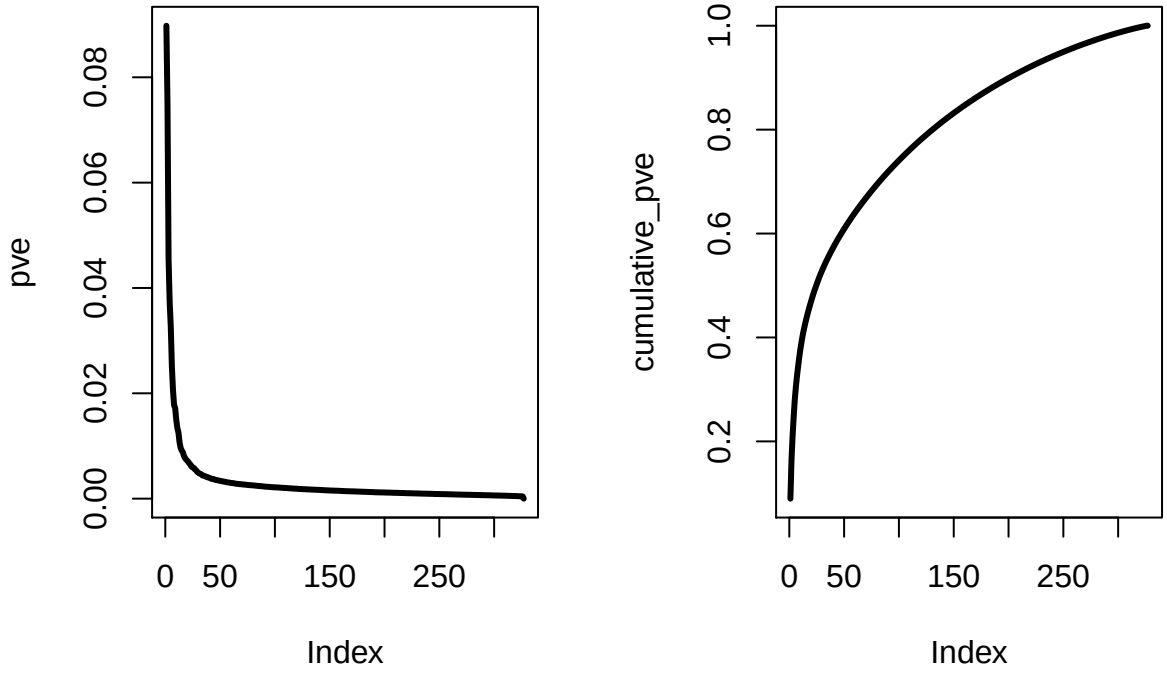
```
leukemia_data_factor <- leukemia_data %>%
  mutate(Type = factor(Type))

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)
```

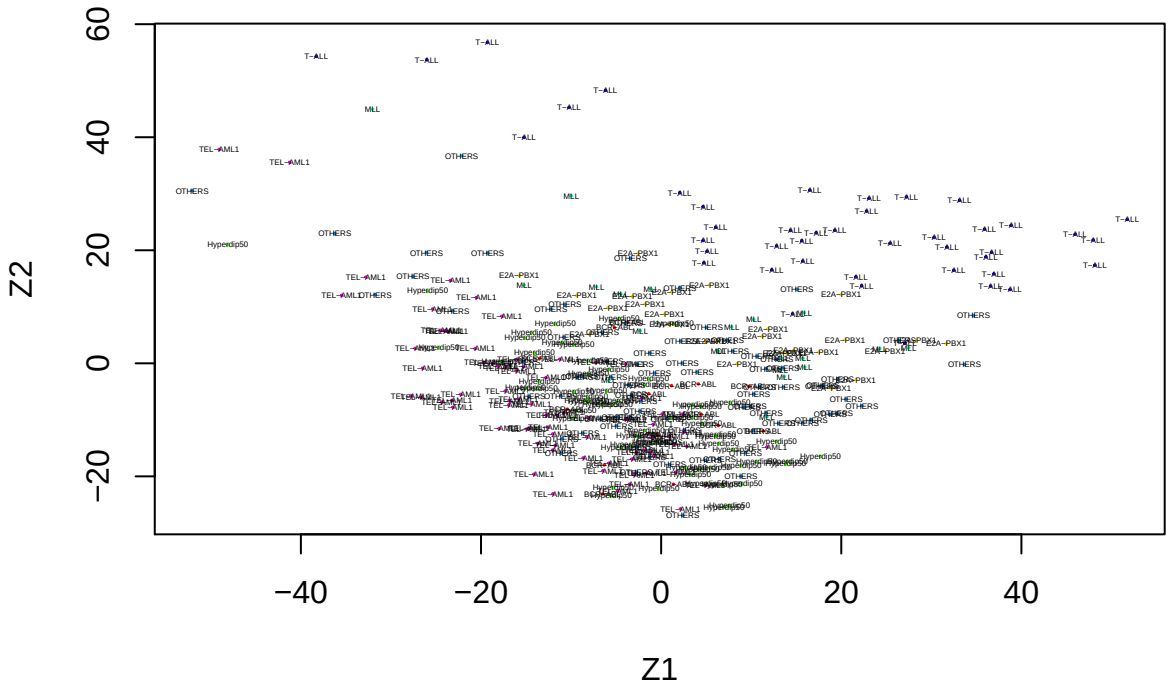


(c). Use the results of PCA to project the data into the first two principal component dimensions. `prcomp` returns this dimension reduced data in the first columns of `x`. Plot the data as a scatter plot using `plot` function with `col=plot_colors` where `plot_colors` is defined

```
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)
```



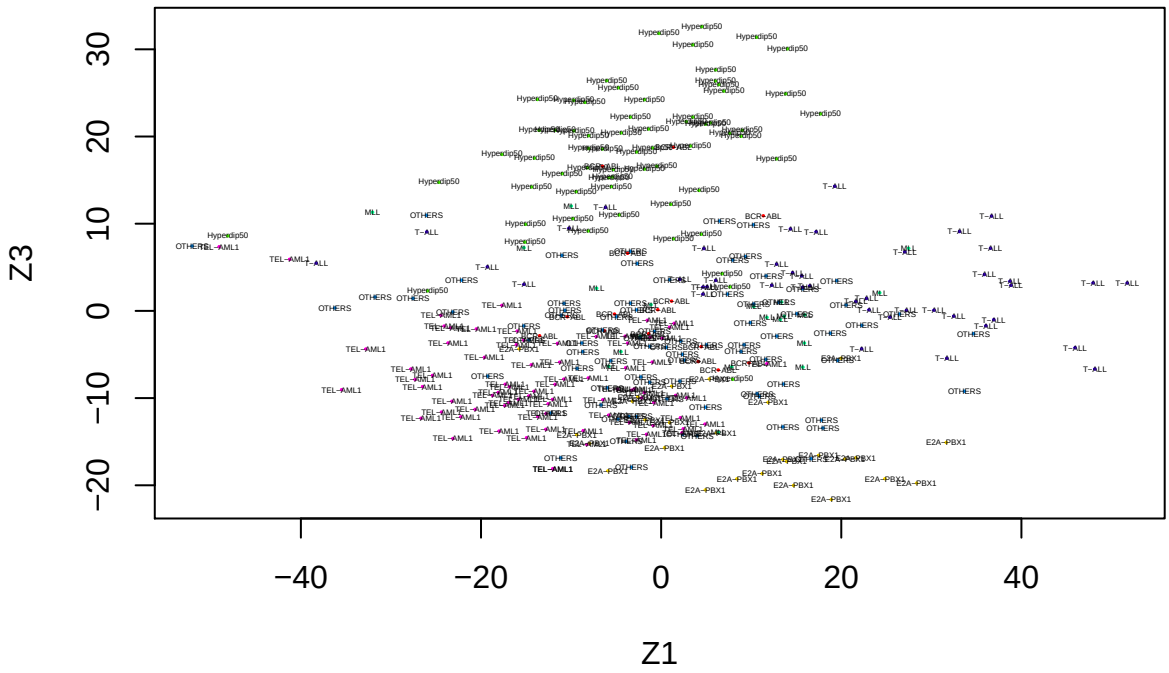

```
abs_pc1 = order(abs(PC1))
head(leukemia_data_factor$Type[abs_pc1], 6)
```

```
## [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.

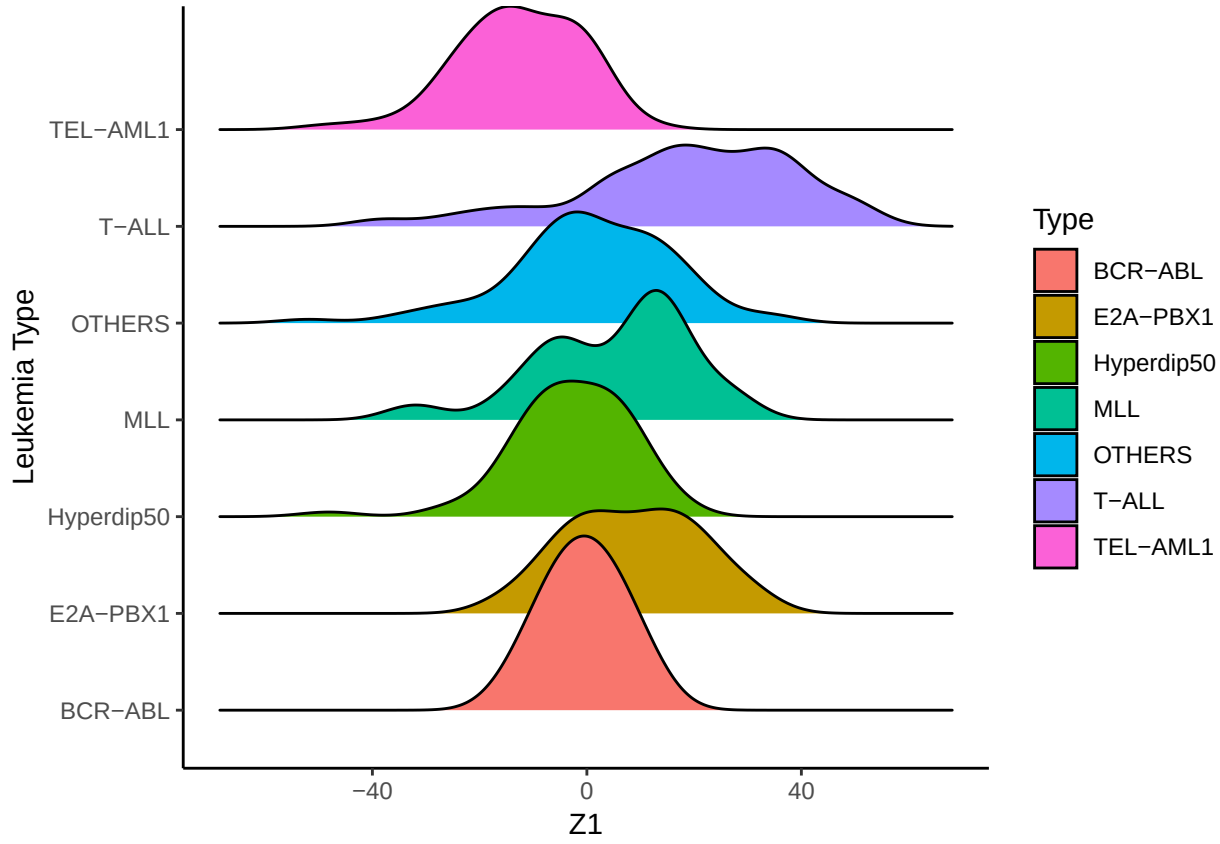
(e.) (**231 Only**) For this part we will be using the `ggridges` library. Create a new tibble where the first column (call it `z1`) is the projection of the data onto the first principal component and the second column is the leukemia subtype (`Type`). Use `ggplot` with `geom_density_ridges` to create 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
```

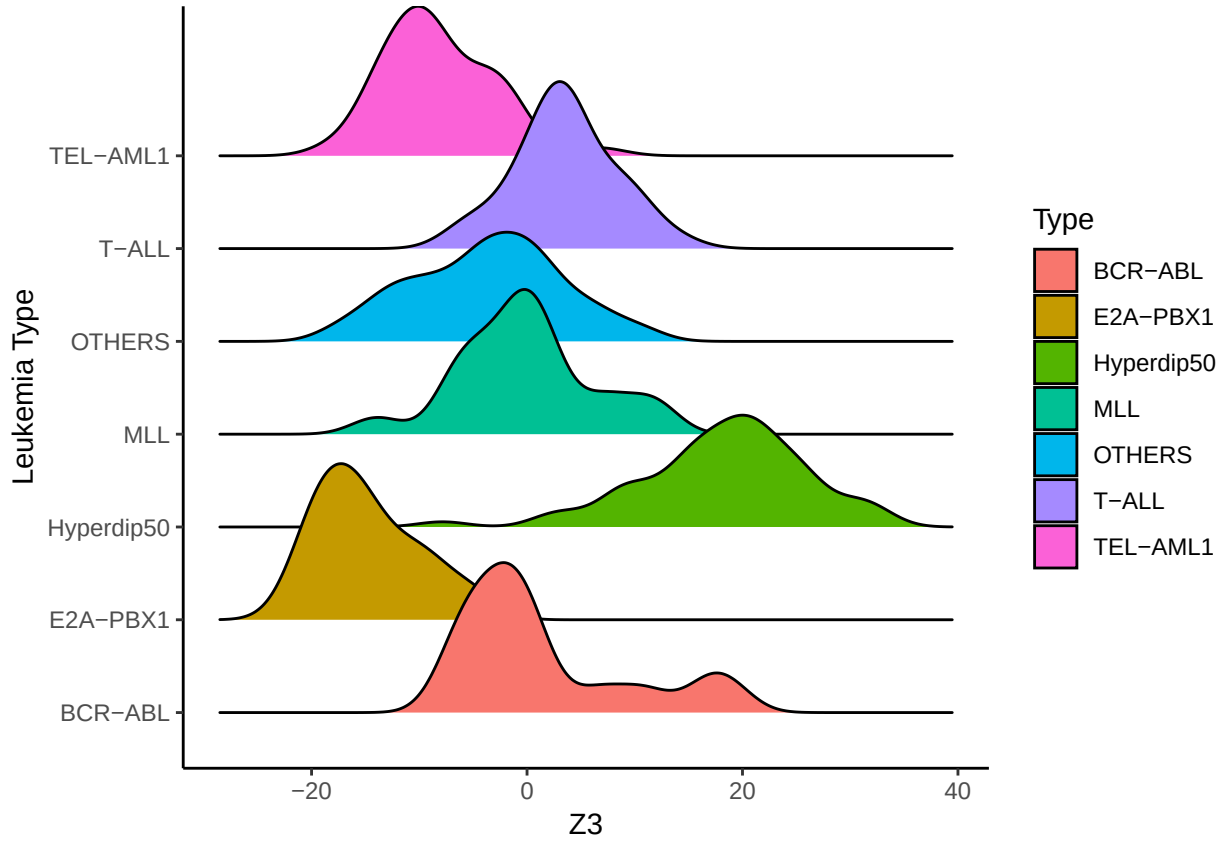



```
plot.new()
pca_df_z3 = data.frame("z3" = PC3, "Type" = leukemia_data_factor$Type)
pca_tibble_z3 = as.tibble(pca_df_z3)

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/
##
```

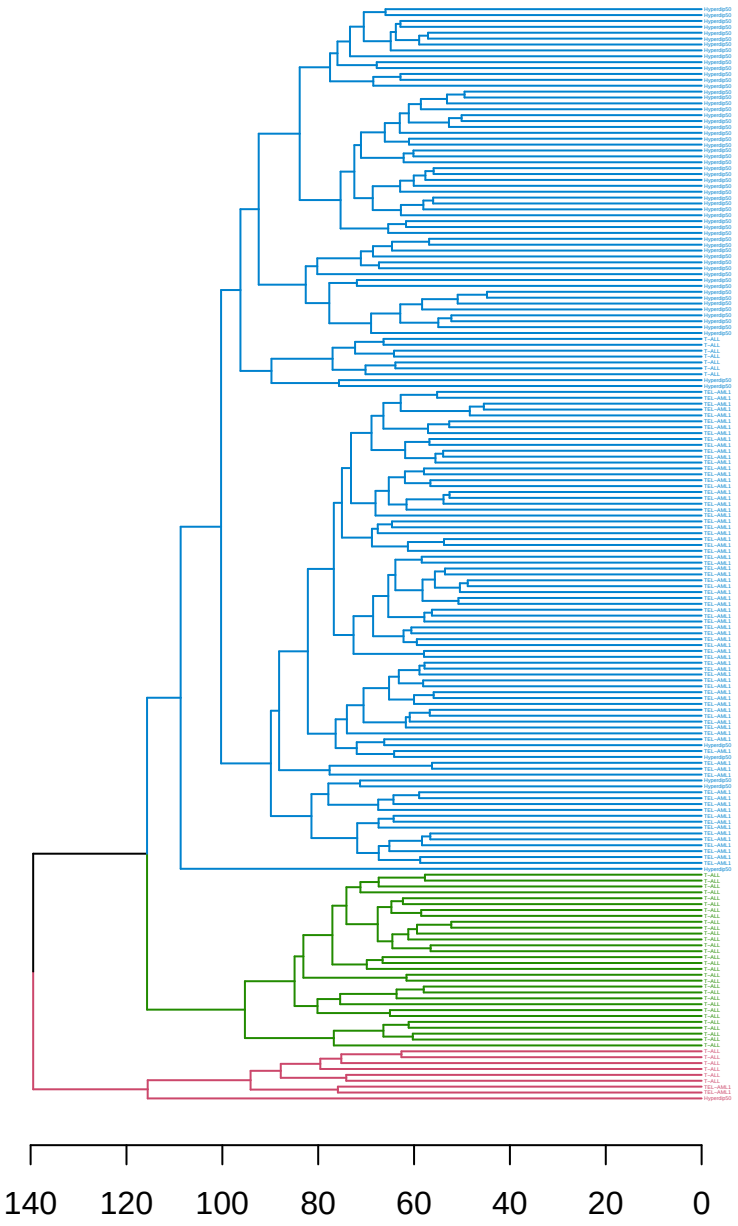
```
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ----------------------
```

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

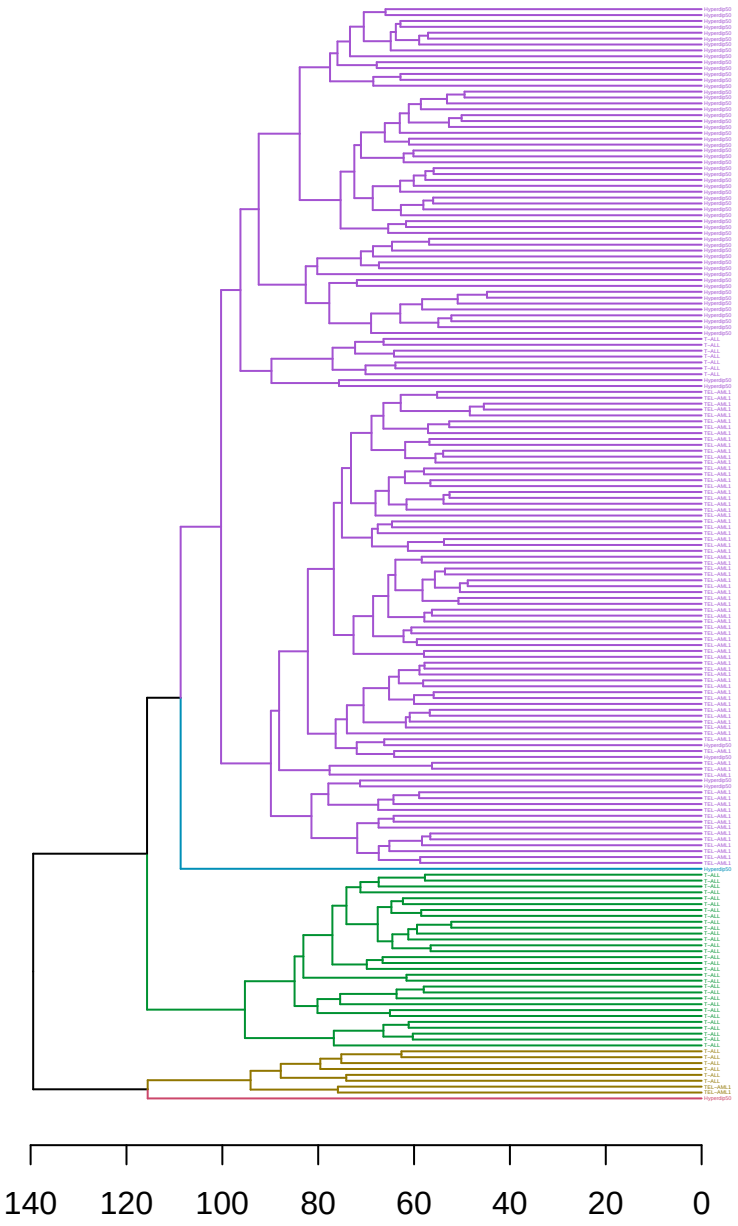
```
## The following object is masked from 'package:rpart':
##
##     prune
```

```
## 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)
```



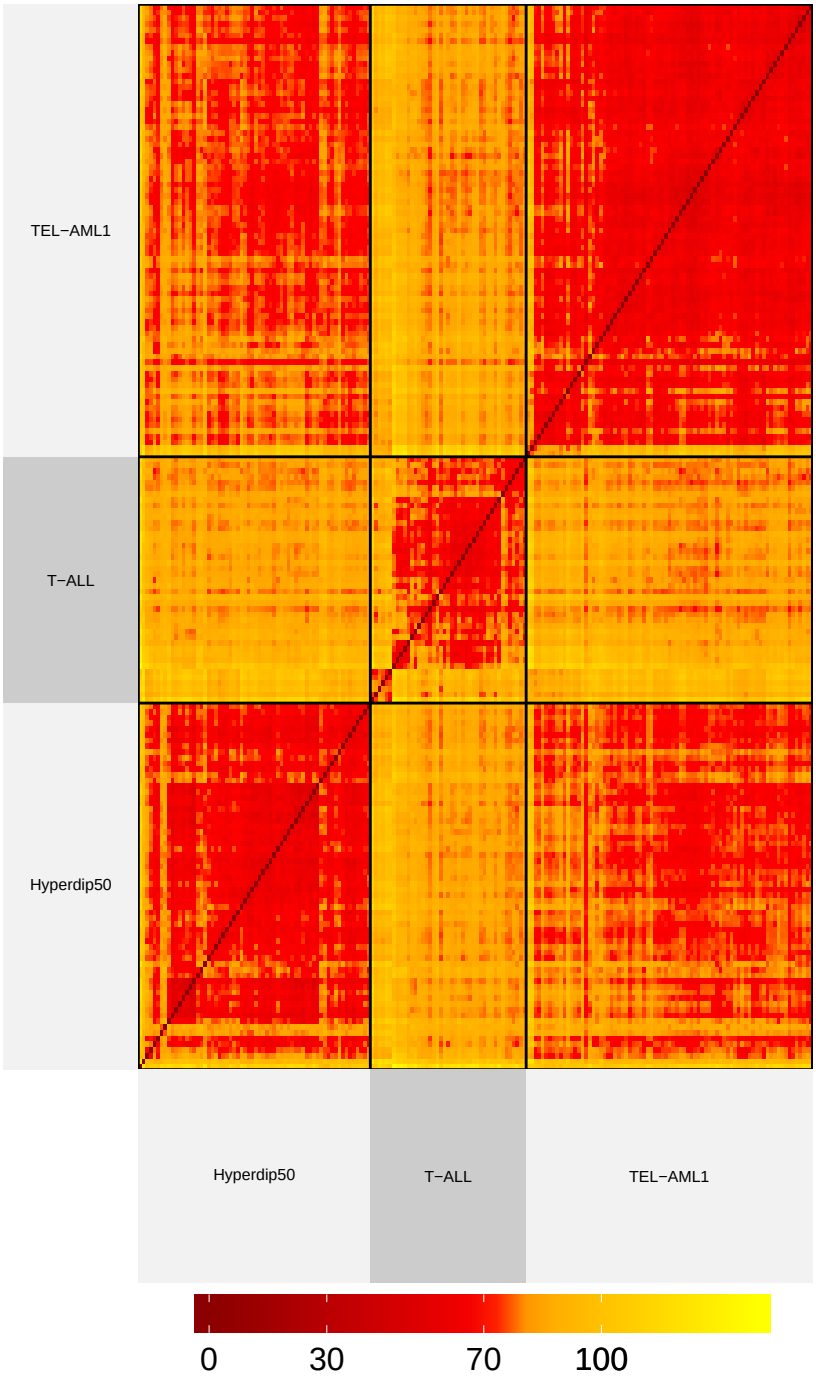

```
leukemia_dend5 = leuk_hc %>%
  as.dendrogram() %>%
  color_branches(k = 5) %>%
  color_labels(k = 5)

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"))
```



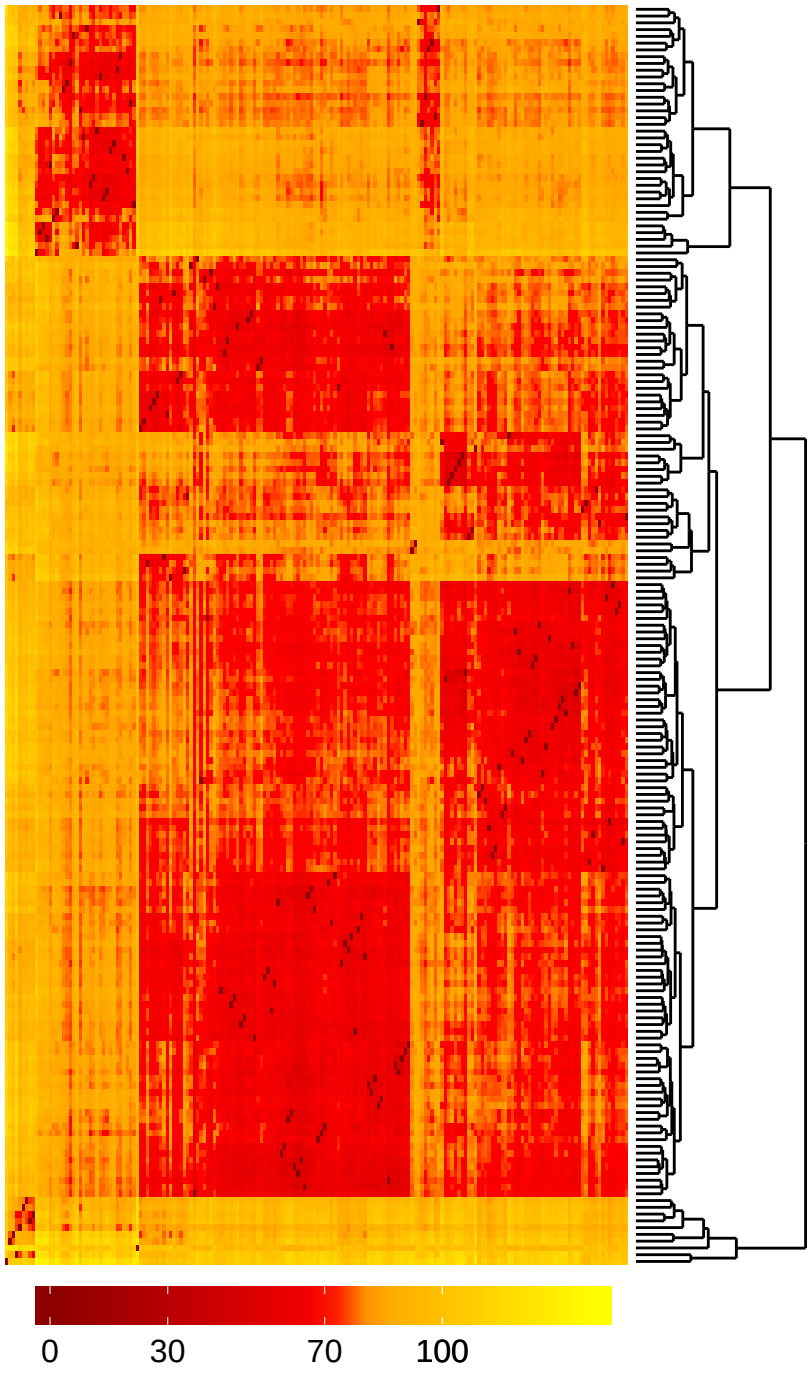
Based on the heatmap, T-ALL and Hyperdip50 seem to have more in common with one another than other subsets of genes. However, T-ALL and TEL-AML1 seem to have only slightly less in common than the T-ALL and Hyperdip50 pairing of genes based on visual inspection.

(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

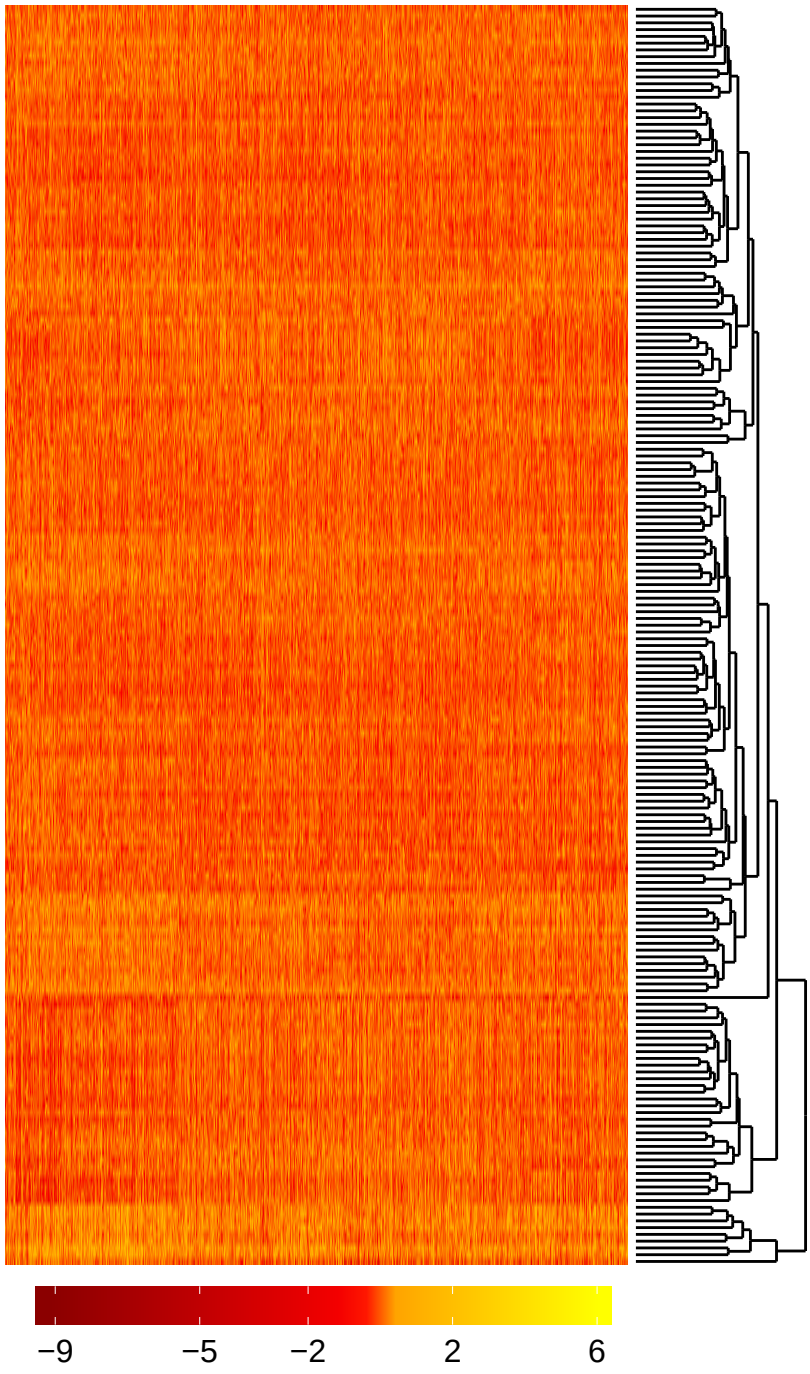
the same dataset to run kmeans clustering with three the optimal number of clusters, and order the genes (columns) based on hierarchical clustering.

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




```
set.seed(1)
hc2.heatmap = superheat(leukemia_subset[,-1],
          scale = T,
          clustering.method = "hierarchical",
          row.dendrogram = T,
          heat.pal = c("dark red", "red", "orange", "yellow"),
          grid.hline.col = "white",
          grid.vline.col = "white")
```




```
hc2.heatmap$membership.cols
```

```
##     [1]   1   2   3   4   5   6   7   8   9  10  11  12  13
##    [14]  14  15  16  17  18  19  20  21  22  23  24  25  26
##    [27]  27  28  29  30  31  32  33  34  35  36  37  38  39
##    [40]  40  41  42  43  44  45  46  47  48  49  50  51  52
##    [53]  53  54  55  56  57  58  59  60  61  62  63  64  65
##    [66]  66  67  68  69  70  71  72  73  74  75  76  77  78
##    [79]  79  80  81  82  83  84  85  86  87  88  89  90  91
##    [92]  92  93  94  95  96  97  98  99 100 101 102 103 104
```

```
##  [105] 105 106 107 108 109 110 111 112 113 114 115 116 117
##  [118] 118 119 120 121 122 123 124 125 126 127 128 129 130
##  [131] 131 132 133 134 135 136 137 138 139 140 141 142 143
##  [144] 144 145 146 147 148 149 150 151 152 153 154 155 156
##  [157] 157 158 159 160 161 162 163 164 165 166 167 168 169
##  [170] 170 171 172 173 174 175 176 177 178 179 180 181 182
##  [183] 183 184 185 186 187 188 189 190 191 192 193 194 195
##  [196] 196 197 198 199 200 201 202 203 204 205 206 207 208
##  [209] 209 210 211 212 213 214 215 216 217 218 219 220 221
##  [222] 222 223 224 225 226 227 228 229 230 231 232 233 234
##  [235] 235 236 237 238 239 240 241 242 243 244 245 246 247
##  [248] 248 249 250 251 252 253 254 255 256 257 258 259 260
##  [261] 261 262 263 264 265 266 267 268 269 270 271 272 273
##  [274] 274 275 276 277 278 279 280 281 282 283 284 285 286
##  [287] 287 288 289 290 291 292 293 294 295 296 297 298 299
##  [300] 300 301 302 303 304 305 306 307 308 309 310 311 312
##  [313] 313 314 315 316 317 318 319 320 321 322 323 324 325
##  [326] 326 327 328 329 330 331 332 333 334 335 336 337 338
##  [339] 339 340 341 342 343 344 345 346 347 348 349 350 351
##  [352] 352 353 354 355 356 357 358 359 360 361 362 363 364
##  [365] 365 366 367 368 369 370 371 372 373 374 375 376 377
##  [378] 378 379 380 381 382 383 384 385 386 387 388 389 390
##  [391] 391 392 393 394 395 396 397 398 399 400 401 402 403
##  [404] 404 405 406 407 408 409 410 411 412 413 414 415 416
##  [417] 417 418 419 420 421 422 423 424 425 426 427 428 429
##  [430] 430 431 432 433 434 435 436 437 438 439 440 441 442
##  [443] 443 444 445 446 447 448 449 450 451 452 453 454 455
##  [456] 456 457 458 459 460 461 462 463 464 465 466 467 468
##  [469] 469 470 471 472 473 474 475 476 477 478 479 480 481
##  [482] 482 483 484 485 486 487 488 489 490 491 492 493 494
##  [495] 495 496 497 498 499 500 501 502 503 504 505 506 507
##  [508] 508 509 510 511 512 513 514 515 516 517 518 519 520
##  [521] 521 522 523 524 525 526 527 528 529 530 531 532 533
##  [534] 534 535 536 537 538 539 540 541 542 543 544 545 546
##  [547] 547 548 549 550 551 552 553 554 555 556 557 558 559
##  [560] 560 561 562 563 564 565 566 567 568 569 570 571 572
##  [573] 573 574 575 576 577 578 579 580 581 582 583 584 585
##  [586] 586 587 588 589 590 591 592 593 594 595 596 597 598
##  [599] 599 600 601 602 603 604 605 606 607 608 609 610 611
##  [612] 612 613 614 615 616 617 618 619 620 621 622 623 624
##  [625] 625 626 627 628 629 630 631 632 633 634 635 636 637
##  [638] 638 639 640 641 642 643 644 645 646 647 648 649 650
##  [651] 651 652 653 654 655 656 657 658 659 660 661 662 663
##  [664] 664 665 666 667 668 669 670 671 672 673 674 675 676
##  [677] 677 678 679 680 681 682 683 684 685 686 687 688 689
##  [690] 690 691 692 693 694 695 696 697 698 699 700 701 702
##  [703] 703 704 705 706 707 708 709 710 711 712 713 714 715
##  [716] 716 717 718 719 720 721 722 723 724 725 726 727 728
##  [729] 729 730 731 732 733 734 735 736 737 738 739 740 741
##  [742] 742 743 744 745 746 747 748 749 750 751 752 753 754
##  [755] 755 756 757 758 759 760 761 762 763 764 765 766 767
##  [768] 768 769 770 771 772 773 774 775 776 777 778 779 780
##  [781] 781 782 783 784 785 786 787 788 789 790 791 792 793
##  [794] 794 795 796 797 798 799 800 801 802 803 804 805 806
```

```
##  [807]  807  808  809  810  811  812  813  814  815  816  817  818  819
##  [820]  820  821  822  823  824  825  826  827  828  829  830  831  832
##  [833]  833  834  835  836  837  838  839  840  841  842  843  844  845
##  [846]  846  847  848  849  850  851  852  853  854  855  856  857  858
##  [859]  859  860  861  862  863  864  865  866  867  868  869  870  871
##  [872]  872  873  874  875  876  877  878  879  880  881  882  883  884
##  [885]  885  886  887  888  889  890  891  892  893  894  895  896  897
##  [898]  898  899  900  901  902  903  904  905  906  907  908  909  910
##  [911]  911  912  913  914  915  916  917  918  919  920  921  922  923
##  [924]  924  925  926  927  928  929  930  931  932  933  934  935  936
##  [937]  937  938  939  940  941  942  943  944  945  946  947  948  949
##  [950]  950  951  952  953  954  955  956  957  958  959  960  961  962
##  [963]  963  964  965  966  967  968  969  970  971  972  973  974  975
##  [976]  976  977  978  979  980  981  982  983  984  985  986  987  988
##  [989]  989  990  991  992  993  994  995  996  997  998  999 1000 1001
## [1002] 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014
## [1015] 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027
## [1028] 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040
## [1041] 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053
## [1054] 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 1066
## [1067] 1067 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079
## [1080] 1080 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090 1091 1092
## [1093] 1093 1094 1095 1096 1097 1098 1099 1100 1101 1102 1103 1104 1105
## [1106] 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115 1116 1117 1118
## [1119] 1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131
## [1132] 1132 1133 1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144
## [1145] 1145 1146 1147 1148 1149 1150 1151 1152 1153 1154 1155 1156 1157
## [1158] 1158 1159 1160 1161 1162 1163 1164 1165 1166 1167 1168 1169 1170
## [1171] 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181 1182 1183
## [1184] 1184 1185 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196
## [1197] 1197 1198 1199 1200 1201 1202 1203 1204 1205 1206 1207 1208 1209
## [1210] 1210 1211 1212 1213 1214 1215 1216 1217 1218 1219 1220 1221 1222
## [1223] 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235
## [1236] 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247 1248
## [1249] 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261
## [1262] 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274
## [1275] 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287
## [1288] 1288 1289 1290 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300
## [1301] 1301 1302 1303 1304 1305 1306 1307 1308 1309 1310 1311 1312 1313
## [1314] 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326
## [1327] 1327 1328 1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339
## [1340] 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350 1351 1352
## [1353] 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363 1364 1365
## [1366] 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378
## [1379] 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391
## [1392] 1392 1393 1394 1395 1396 1397 1398 1399 1400 1401 1402 1403 1404
## [1405] 1405 1406 1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417
## [1418] 1418 1419 1420 1421 1422 1423 1424 1425 1426 1427 1428 1429 1430
## [1431] 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440 1441 1442 1443
## [1444] 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455 1456
## [1457] 1457 1458 1459 1460 1461 1462 1463 1464 1465 1466 1467 1468 1469
## [1470] 1470 1471 1472 1473 1474 1475 1476 1477 1478 1479 1480 1481 1482
## [1483] 1483 1484 1485 1486 1487 1488 1489 1490 1491 1492 1493 1494 1495
## [1496] 1496 1497 1498 1499 1500 1501 1502 1503 1504 1505 1506 1507 1508
```

```
## [1509] 1509 1510 1511 1512 1513 1514 1515 1516 1517 1518 1519 1520 1521
## [1522] 1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534
## [1535] 1535 1536 1537 1538 1539 1540 1541 1542 1543 1544 1545 1546 1547
## [1548] 1548 1549 1550 1551 1552 1553 1554 1555 1556 1557 1558 1559 1560
## [1561] 1561 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573
## [1574] 1574 1575 1576 1577 1578 1579 1580 1581 1582 1583 1584 1585 1586
## [1587] 1587 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599
## [1600] 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1610 1611 1612
## [1613] 1613 1614 1615 1616 1617 1618 1619 1620 1621 1622 1623 1624 1625
## [1626] 1626 1627 1628 1629 1630 1631 1632 1633 1634 1635 1636 1637 1638
## [1639] 1639 1640 1641 1642 1643 1644 1645 1646 1647 1648 1649 1650 1651
## [1652] 1652 1653 1654 1655 1656 1657 1658 1659 1660 1661 1662 1663 1664
## [1665] 1665 1666 1667 1668 1669 1670 1671 1672 1673 1674 1675 1676 1677
## [1678] 1678 1679 1680 1681 1682 1683 1684 1685 1686 1687 1688 1689 1690
## [1691] 1691 1692 1693 1694 1695 1696 1697 1698 1699 1700 1701 1702 1703
## [1704] 1704 1705 1706 1707 1708 1709 1710 1711 1712 1713 1714 1715 1716
## [1717] 1717 1718 1719 1720 1721 1722 1723 1724 1725 1726 1727 1728 1729
## [1730] 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739 1740 1741 1742
## [1743] 1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753 1754 1755
## [1756] 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1766 1767 1768
## [1769] 1769 1770 1771 1772 1773 1774 1775 1776 1777 1778 1779 1780 1781
## [1782] 1782 1783 1784 1785 1786 1787 1788 1789 1790 1791 1792 1793 1794
## [1795] 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804 1805 1806 1807
## [1808] 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819 1820
## [1821] 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833
## [1834] 1834 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844 1845 1846
## [1847] 1847 1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859
## [1860] 1860 1861 1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872
## [1873] 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885
## [1886] 1886 1887 1888 1889 1890 1891 1892 1893 1894 1895 1896 1897 1898
## [1899] 1899 1900 1901 1902 1903 1904 1905 1906 1907 1908 1909 1910 1911
## [1912] 1912 1913 1914 1915 1916 1917 1918 1919 1920 1921 1922 1923 1924
## [1925] 1925 1926 1927 1928 1929 1930 1931 1932 1933 1934 1935 1936 1937
## [1938] 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950
## [1951] 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963
## [1964] 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976
## [1977] 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989
## [1990] 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002
## [2003] 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015
## [2016] 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026 2027 2028
## [2029] 2029 2030 2031 2032 2033 2034 2035 2036 2037 2038 2039 2040 2041
## [2042] 2042 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052 2053 2054
## [2055] 2055 2056 2057 2058 2059 2060 2061 2062 2063 2064 2065 2066 2067
## [2068] 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080
## [2081] 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091 2092 2093
## [2094] 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2105 2106
## [2107] 2107 2108 2109 2110 2111 2112 2113 2114 2115 2116 2117 2118 2119
## [2120] 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132
## [2133] 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144 2145
## [2146] 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158
## [2159] 2159 2160 2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171
## [2172] 2172 2173 2174 2175 2176 2177 2178 2179 2180 2181 2182 2183 2184
## [2185] 2185 2186 2187 2188 2189 2190 2191 2192 2193 2194 2195 2196 2197
## [2198] 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2210
```

```
## [2211] 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223
## [2224] 2224 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236
## [2237] 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249
## [2250] 2250 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260 2261 2262
## [2263] 2263 2264 2265 2266 2267 2268 2269 2270 2271 2272 2273 2274 2275
## [2276] 2276 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286 2287 2288
## [2289] 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299 2300 2301
## [2302] 2302 2303 2304 2305 2306 2307 2308 2309 2310 2311 2312 2313 2314
## [2315] 2315 2316 2317 2318 2319 2320 2321 2322 2323 2324 2325 2326 2327
## [2328] 2328 2329 2330 2331 2332 2333 2334 2335 2336 2337 2338 2339 2340
## [2341] 2341 2342 2343 2344 2345 2346 2347 2348 2349 2350 2351 2352 2353
## [2354] 2354 2355 2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366
## [2367] 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379
## [2380] 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392
## [2393] 2393 2394 2395 2396 2397 2398 2399 2400 2401 2402 2403 2404 2405
## [2406] 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 2416 2417 2418
## [2419] 2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 2430 2431
## [2432] 2432 2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 2444
## [2445] 2445 2446 2447 2448 2449 2450 2451 2452 2453 2454 2455 2456 2457
## [2458] 2458 2459 2460 2461 2462 2463 2464 2465 2466 2467 2468 2469 2470
## [2471] 2471 2472 2473 2474 2475 2476 2477 2478 2479 2480 2481 2482 2483
## [2484] 2484 2485 2486 2487 2488 2489 2490 2491 2492 2493 2494 2495 2496
## [2497] 2497 2498 2499 2500 2501 2502 2503 2504 2505 2506 2507 2508 2509
## [2510] 2510 2511 2512 2513 2514 2515 2516 2517 2518 2519 2520 2521 2522
## [2523] 2523 2524 2525 2526 2527 2528 2529 2530 2531 2532 2533 2534 2535
## [2536] 2536 2537 2538 2539 2540 2541 2542 2543 2544 2545 2546 2547 2548
## [2549] 2549 2550 2551 2552 2553 2554 2555 2556 2557 2558 2559 2560 2561
## [2562] 2562 2563 2564 2565 2566 2567 2568 2569 2570 2571 2572 2573 2574
## [2575] 2575 2576 2577 2578 2579 2580 2581 2582 2583 2584 2585 2586 2587
## [2588] 2588 2589 2590 2591 2592 2593 2594 2595 2596 2597 2598 2599 2600
## [2601] 2601 2602 2603 2604 2605 2606 2607 2608 2609 2610 2611 2612 2613
## [2614] 2614 2615 2616 2617 2618 2619 2620 2621 2622 2623 2624 2625 2626
## [2627] 2627 2628 2629 2630 2631 2632 2633 2634 2635 2636 2637 2638 2639
## [2640] 2640 2641 2642 2643 2644 2645 2646 2647 2648 2649 2650 2651 2652
## [2653] 2653 2654 2655 2656 2657 2658 2659 2660 2661 2662 2663 2664 2665
## [2666] 2666 2667 2668 2669 2670 2671 2672 2673 2674 2675 2676 2677 2678
## [2679] 2679 2680 2681 2682 2683 2684 2685 2686 2687 2688 2689 2690 2691
## [2692] 2692 2693 2694 2695 2696 2697 2698 2699 2700 2701 2702 2703 2704
## [2705] 2705 2706 2707 2708 2709 2710 2711 2712 2713 2714 2715 2716 2717
## [2718] 2718 2719 2720 2721 2722 2723 2724 2725 2726 2727 2728 2729 2730
## [2731] 2731 2732 2733 2734 2735 2736 2737 2738 2739 2740 2741 2742 2743
## [2744] 2744 2745 2746 2747 2748 2749 2750 2751 2752 2753 2754 2755 2756
## [2757] 2757 2758 2759 2760 2761 2762 2763 2764 2765 2766 2767 2768 2769
## [2770] 2770 2771 2772 2773 2774 2775 2776 2777 2778 2779 2780 2781 2782
## [2783] 2783 2784 2785 2786 2787 2788 2789 2790 2791 2792 2793 2794 2795
## [2796] 2796 2797 2798 2799 2800 2801 2802 2803 2804 2805 2806 2807 2808
## [2809] 2809 2810 2811 2812 2813 2814 2815 2816 2817 2818 2819 2820 2821
## [2822] 2822 2823 2824 2825 2826 2827 2828 2829 2830 2831 2832 2833 2834
## [2835] 2835 2836 2837 2838 2839 2840 2841 2842 2843 2844 2845 2846 2847
## [2848] 2848 2849 2850 2851 2852 2853 2854 2855 2856 2857 2858 2859 2860
## [2861] 2861 2862 2863 2864 2865 2866 2867 2868 2869 2870 2871 2872 2873
## [2874] 2874 2875 2876 2877 2878 2879 2880 2881 2882 2883 2884 2885 2886
## [2887] 2887 2888 2889 2890 2891 2892 2893 2894 2895 2896 2897 2898 2899
## [2900] 2900 2901 2902 2903 2904 2905 2906 2907 2908 2909 2910 2911 2912
```

```
## [2913] 2913 2914 2915 2916 2917 2918 2919 2920 2921 2922 2923 2924 2925
## [2926] 2926 2927 2928 2929 2930 2931 2932 2933 2934 2935 2936 2937 2938
## [2939] 2939 2940 2941 2942 2943 2944 2945 2946 2947 2948 2949 2950 2951
## [2952] 2952 2953 2954 2955 2956 2957 2958 2959 2960 2961 2962 2963 2964
## [2965] 2965 2966 2967 2968 2969 2970 2971 2972 2973 2974 2975 2976 2977
## [2978] 2978 2979 2980 2981 2982 2983 2984 2985 2986 2987 2988 2989 2990
## [2991] 2991 2992 2993 2994 2995 2996 2997 2998 2999 3000 3001 3002 3003
## [3004] 3004 3005 3006 3007 3008 3009 3010 3011 3012 3013 3014 3015 3016
## [3017] 3017 3018 3019 3020 3021 3022 3023 3024 3025 3026 3027 3028 3029
## [3030] 3030 3031 3032 3033 3034 3035 3036 3037 3038 3039 3040 3041 3042
## [3043] 3043 3044 3045 3046 3047 3048 3049 3050 3051 3052 3053 3054 3055
## [3056] 3056 3057 3058 3059 3060 3061 3062 3063 3064 3065 3066 3067 3068
## [3069] 3069 3070 3071 3072 3073 3074 3075 3076 3077 3078 3079 3080 3081
## [3082] 3082 3083 3084 3085 3086 3087 3088 3089 3090 3091 3092 3093 3094
## [3095] 3095 3096 3097 3098 3099 3100 3101 3102 3103 3104 3105 3106 3107
## [3108] 3108 3109 3110 3111 3112 3113 3114 3115 3116 3117 3118 3119 3120
## [3121] 3121 3122 3123 3124 3125 3126 3127 3128 3129 3130 3131 3132 3133
## [3134] 3134 3135 3136 3137 3138 3139 3140 3141
```

```r
# what if you calculate kmeans first and then you set membership rows based on the kmeans and the membe

set.seed(1)
kmeans.heatmap = superheat(as.matrix(dis)[hc.order, hc.order],
          scale = T,
          n.clusters.rows = 3,
          #membership.cols = hc.heatmap$membership.cols,
          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")
          )
```