Assignment 3*

PSTAT 231

Villaseñor-Derbez J.C. | 8749749

1 Set up

```r
# Load packages
suppressPackageStartupMessages({
  library(startR)
  library(here)
  library(magrittr)
  library(tree)
  library(maptree)
  library(ROCR)
  library(dendextend)
  library(superheat)
  library(ggridges)
  library(tidyverse)
})
```

```r
# Some housekeeping
update_geom_defaults("point", list(fill = "steelblue",
                                   color = "black",
                                   shape = 21,
                                   size = 2))

update_geom_defaults("line", list(color = "black",
                                  size = 1))

update_geom_defaults("density_ridges", list(fill = "steelblue",
                                            color = "black",
                                            size = 1,
                                            alpha = 0.5))

# Set global theme
theme_set(startR::ggtheme_plot())
```

```r
# Load the data
drug_use <- read_csv(here("data", "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"),
                     col_types = cols())
```

*Code available on GitHUb at: https://github.com/jcvdav/PSTAT231/tree/master/docs/assig3

2 Logistic regression for drug use

2.1 Feature engineering

```
# Create ordered factors for alcohol trhoug VSA
drug_use <- drug_use %>%
  mutate_at(as.ordered, .vars=vars(Alcohol:VSA))

# Create orederd factor for gender, ethnicity and country
drug_use <- drug_use %>%
  mutate(Gender = factor(Gender,
                         labels = c("Male",
                                    "Female")),
         Ethnicity = factor(Ethnicity,
                            labels = c("Black",
                                       "Asian",
                                       "White",
                                       "Mixed:White/Black",
                                       "Other",
                                       "Mixed:White/Asian",
                                       "Mixed:Black/Asian")),
         Country = factor(Country,
                          labels = c("Australia",
                                     "Canada",
                                     "New Zealand",
                                     "Other",
                                     "Ireland",
                                     "UK",
                                     "USA")))
```

2.2 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 = ifelse(Cannabis %in% c("CL0", "CL1", "CL2", "CL3"), "Yes", "No"),
         recent_cannabis_use = factor(recent_cannabis_use,
                                      labels = c("No", "Yes")))
```

2.3 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)`.

```
# set seed
set.seed(42)

# Get rows for training set
train_rows <- sample(x = 1:nrow(drug_use_subset),
                     size = 1500,
                     replace = FALSE)

# Create training set
drug_use_train <- drug_use_subset[train_rows, ]
# Create testing set
drug_use_test <- drug_use_subset[-train_rows, ]

# Check dimensions
dim(drug_use_train)
```

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

```
dim(drug_use_test)
```

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

2.4 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.

```
cannabis_model <- glm(recent_cannabis_use ~ .,
                      data = drug_use_train,
                      family = binomial(link = "logit"))

stargazer::stargazer(cannabis_model,
                     single.row = T,
                     header = F,
                     title = "Logistic regression modelling recent cannabis use as a function of all oth
```

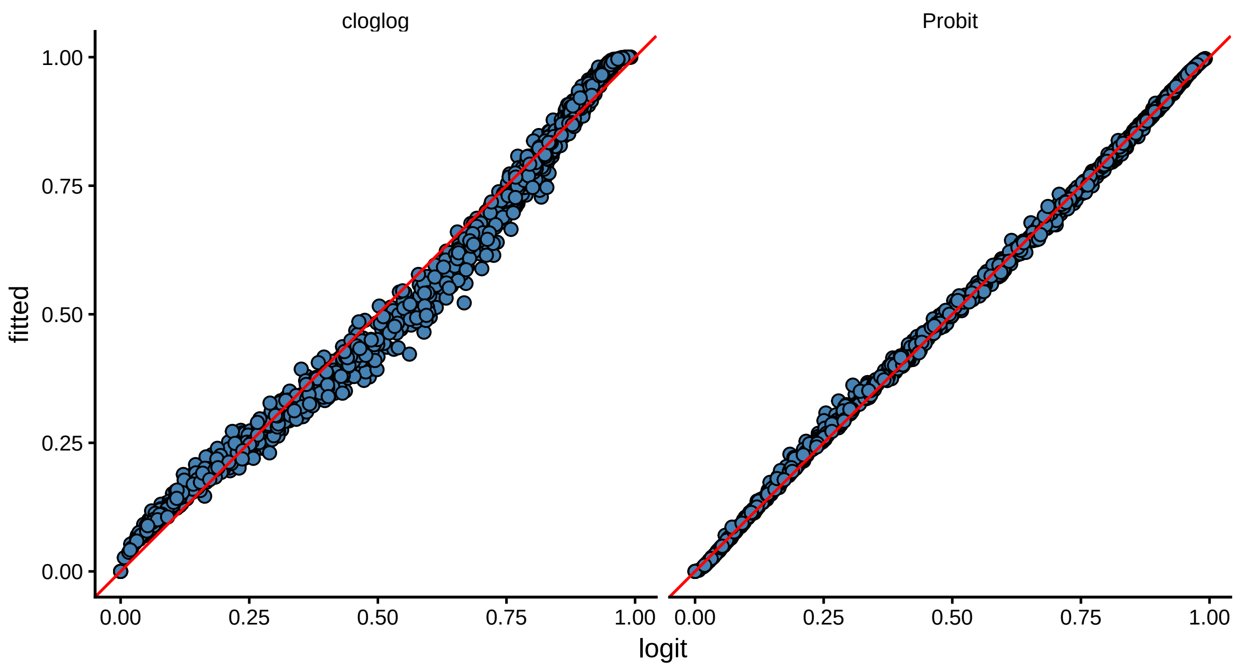
2.5 Probit and c-log-log link functions

```
cannabis_model_probit <- glm(recent_cannabis_use ~ .,
                             data = drug_use_train,
                             family = binomial(link = "probit"))

cannabis_model_cloglog <- glm(recent_cannabis_use ~ .,
                              data = drug_use_train,
                              family = binomial(link = "cloglog"))
```

Table 1: Logistic regression modelling recent cannabis use as a function of all other predictors in the training dataset. Numbers in parentheses are standard errors of the estimates.

	Dependent variable:
	recent_cannabis_use
Age	0.636^{***} (0.089)
GenderFemale	0.623^{***} (0.150)
Education	0.441^{***} (0.080)
CountryCanada	-14.220 (608.645)
CountryNew Zealand	0.669^{**} (0.273)
CountryOther	0.357 (0.379)
CountryIreland	0.414 (0.670)
CountryUK	0.562^{*} (0.322)
CountryUSA	1.767^{***} (0.170)
EthnicityAsian	-0.007 (1.032)
EthnicityWhite	-0.933 (0.685)
EthnicityMixed:White/Black	-0.024 (1.080)
EthnicityOther	-1.245 (0.779)
EthnicityMixed:White/Asian	-2.087^{**} (1.060)
EthnicityMixed:Black/Asian	-15.549 (969.459)
Nscore	0.147^{*} (0.085)
Escore	0.061 (0.087)
Oscore	-0.514^{***} (0.085)
Ascore	-0.098 (0.076)
Cscore	0.166^{*} (0.086)
Impulsive	-0.085 (0.099)
SS	-0.390^{***} (0.107)
Constant	-0.002 (0.688)
Observations	1,500
Log Likelihood	-641.279
Akaike Inf. Crit.	1,328.557
Note:	*p<0.1; **p<0.05; ***p<0.01

```r
tibble(logit = cannabis_model$fitted.values,
       Probit = cannabis_model_probit$fitted.values,
       cloglog = cannabis_model_cloglog$fitted.values) %>%
  gather(model, fitted, -logit) %>%
  ggplot(aes(x = logit, y = fitted)) +
  geom_point() +
  facet_wrap(~model) +
  geom_abline(intercept = 0,
              slope = 1,
              color = "red") +
  coord_equal()
```



The probit regression produced fitted values that are more similar to the logistic regression using a logit-link function. For probabilities between 0.25 and 0.75, the c-log-log function produces lower fitted values (below the red line). The c-log-log link function has more variation, showing data that are more spread around the red line.

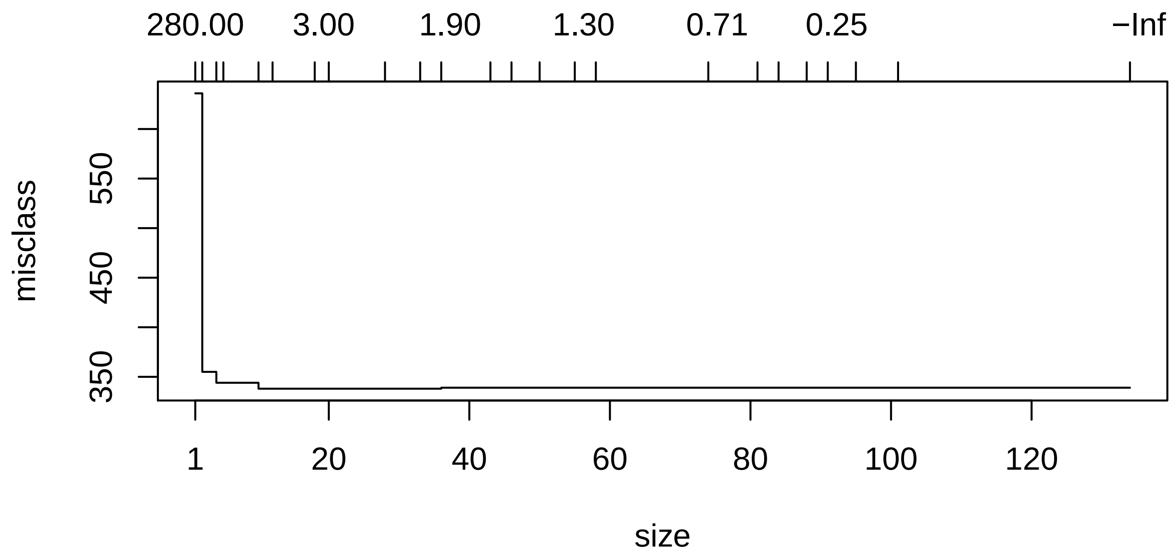
3 Decision Tree for drug use

3.1 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)
```

3.2 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`. 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.

```
set.seed(43)
# Grow decision tree
drug_tree <- tree(recent_cannabis_use ~ .,
                  data = drug_use_train,
                  control = tree_parameters)

# Cross-validate tree
cv_drug_tree <- cv.tree(object = drug_tree,
                        method = "misclass",
                        K = 10)

plot(cv_drug_tree)
```



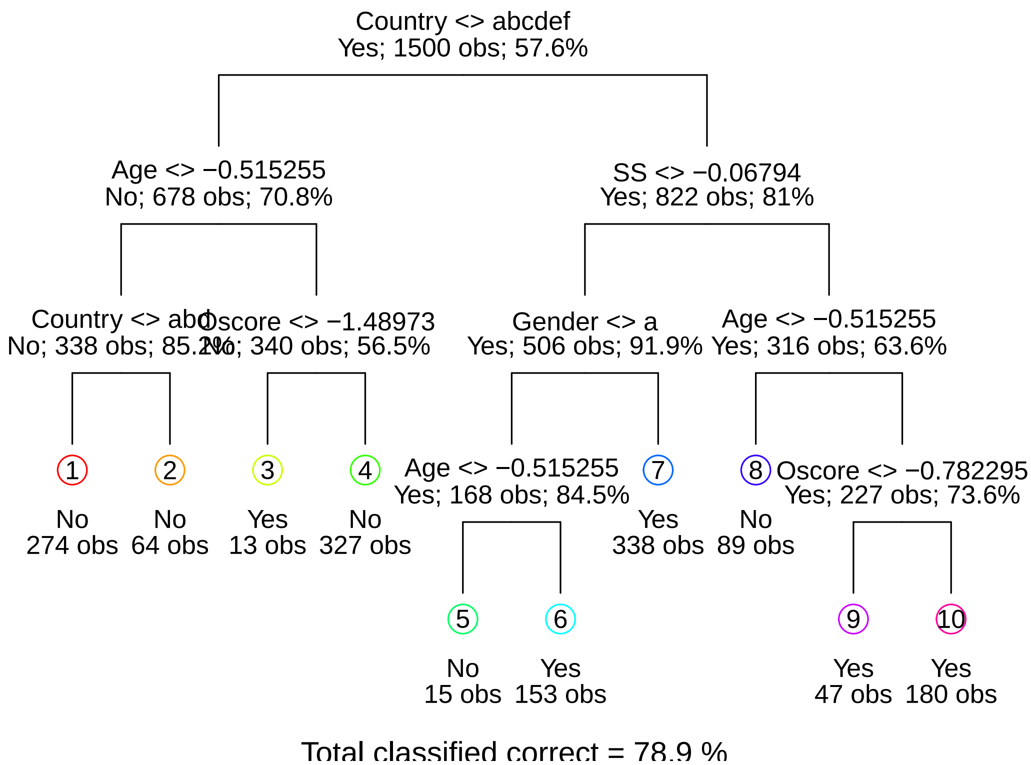

```
# create a tidy version of the diagnostics
cv_drug_tidy <- tibble(size = cv_drug_tree$size,
                       misclass = cv_drug_tree$dev)

# Find the smallest tree with the smallest misclassification error
cv_drug_tidy %>%
  filter(misclass <= min(misclass)) %>%
  filter(size == min(size))
```

```
## # A tibble: 1 x 2
##    size misclass
##   <int>    <dbl>
## 1    10     338.
```

3.3 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?

The first variable to split the data is `Country`.

```
drug_tree_pruned <- prune.tree(tree = drug_tree, best = 10)

draw.tree(drug_tree_pruned, nodeinfo = TRUE)
```



3.4 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. Calculate the true positive rate (TPR) and false positive rate (FPR) for the confusion matrix. Show how you arrived at your answer.

```r
truth <- drug_use_test$recent_cannabis_use
predictions <- predict(object = drug_tree_pruned, newdata = drug_use_test, type = "class")

table(truth, predictions)
```

```
##       predictions
## truth  No Yes
##   No  125  27
##   Yes  70 163
```

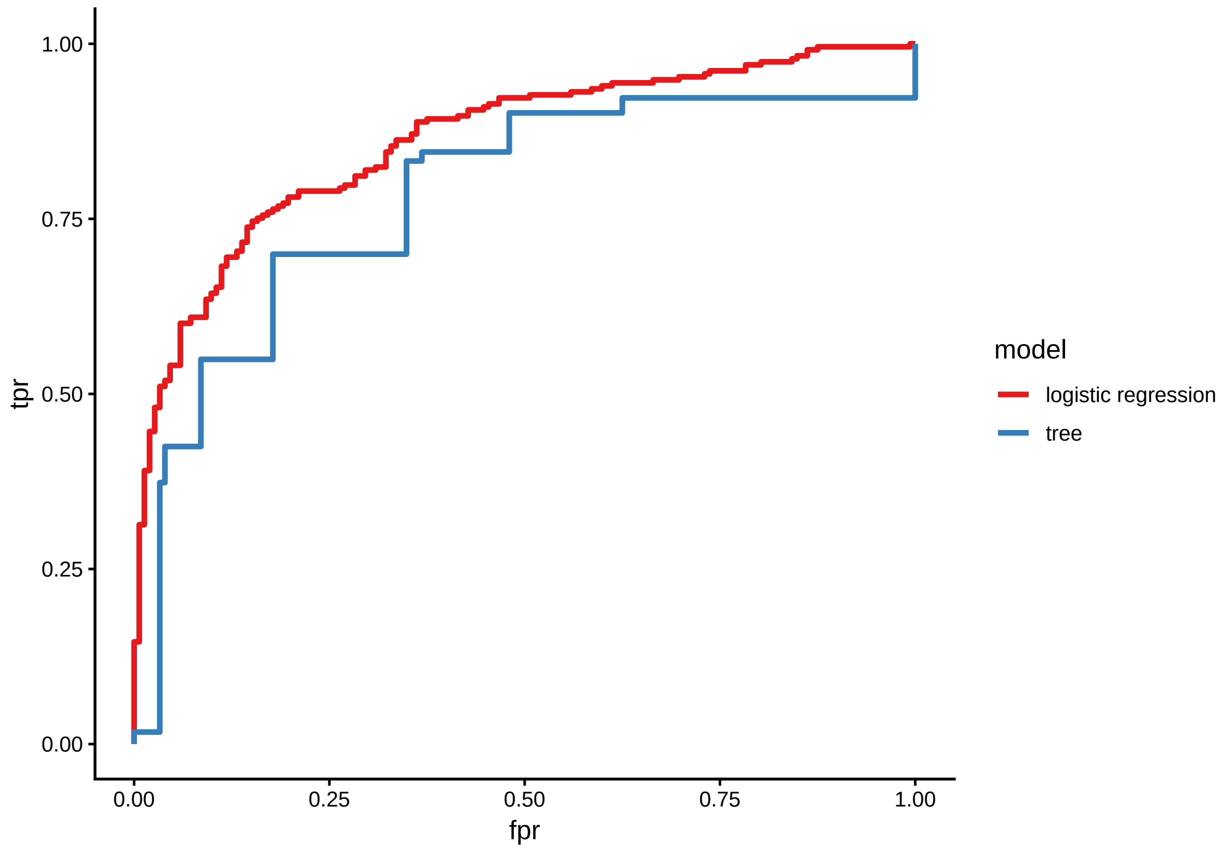
- TPR is $27/(125 + 27) = 0.177$

- FPR is $163/(70 + 163) = 0.699$

4 Model Comparison

4.1 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.

```
# Logistic ROC
log_pred <- prediction(predict(cannabis_model,
                               newdata = drug_use_test,
                               type = "response"), truth)
log_perf <- performance(log_pred, "tpr", "fpr")

# Tree ROC
tree_pred <- prediction(predict(object = drug_tree_pruned,
                                newdata = drug_use_test)[, 2],
                        truth)
tree_perf <- performance(tree_pred, "tpr", "fpr")

# Plot it
tibble(fpr = log_perf@x.values[[1]],
       tpr = log_perf@y.values[[1]],
       model = "logistic regression") %>%
  rbind(tibble(fpr = tree_perf@x.values[[1]],
               tpr = tree_perf@y.values[[1]],
               model = "tree")) %>%
  ggplot(aes(x = fpr, y = tpr, color = model)) +
  geom_step(size = 1) +
  scale_color_brewer(palette = "Set1")
```



4.2 Compute the AUC for both models and print them. Which model has larger AUC?

```
performance(log_pred, "auc")@y.values[[1]]
```

```
## [1] 0.8611927
```

```
performance(tree_pred, "auc")@y.values[[1]]
```

```
## [1] 0.8268014
```

The AUC for logistic is $AUC_{logistic} = 0.86$ and the AUC for the tree is $AUC_{tree} = 0.82$.

5 Clustering and dimension reduction for gene expression data

5.1 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 <- read.csv(here("data", "leukemia_data.csv"),
                          stringsAsFactors = F) %>%
  mutate(Type = factor(Type))
```

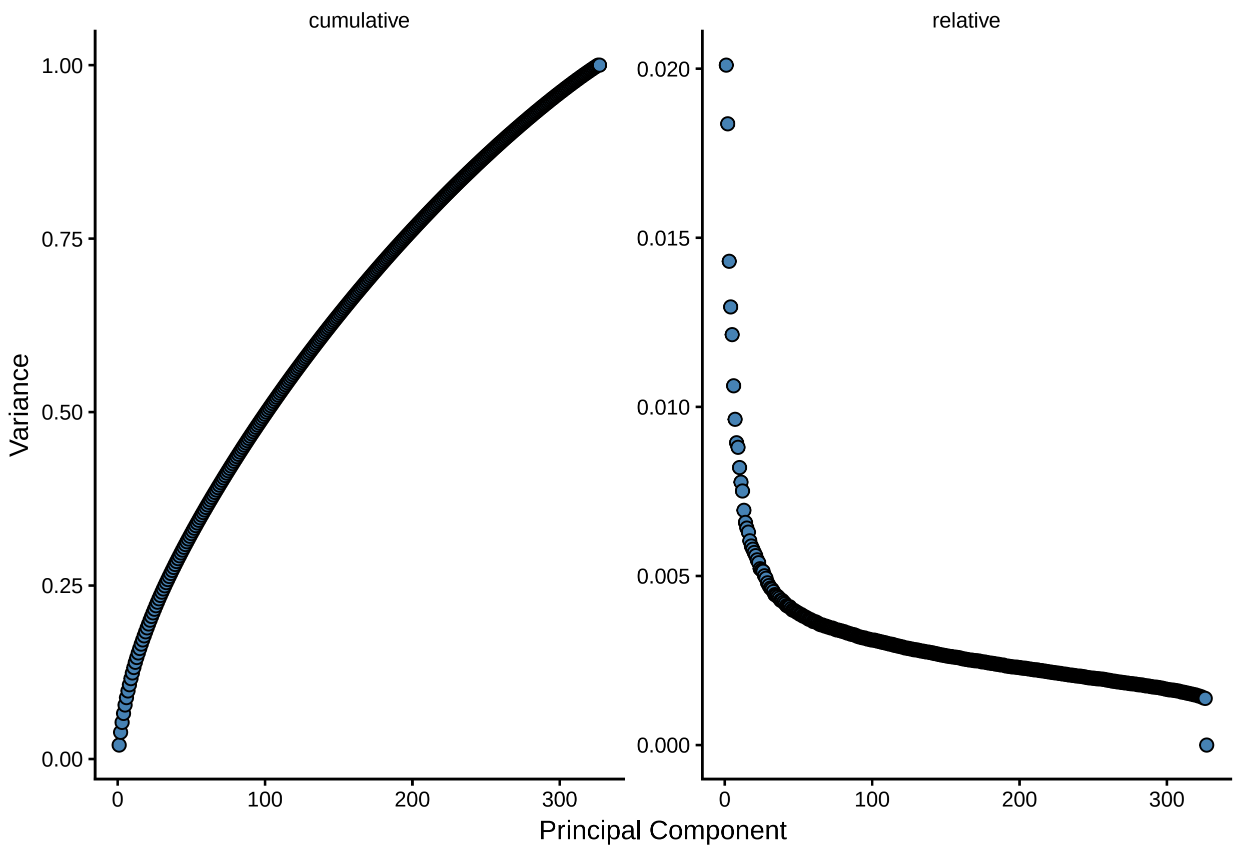
```r
table(leukemia_data$Type) %>%
  as_tibble() %>%
  arrange(n) %>%
  knitr::kable(booktabs = T,
               col.names = c("Type", "Frequency"))
```

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

The least common leukemia subtype is `BCR-ABL`, with 15 cases.

5.2 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.

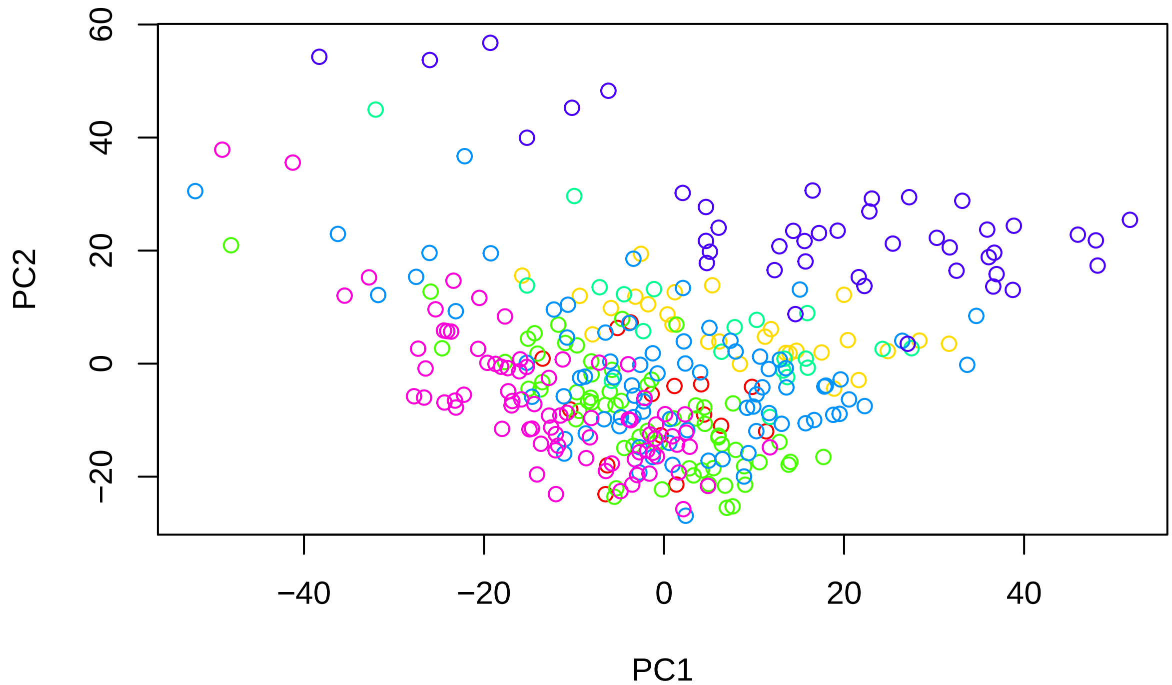
```r
leuk_pca <- leukemia_data %>%
  select(-Type) %>%
  prcomp(scale = T, center = T)

tibble(component = 1:length(leuk_pca$sdev),
       variance = leuk_pca$sdev) %>%
  mutate(relative = variance / sum(variance),
         cumulative = cumsum(relative)) %>%
  select(-variance) %>%
  gather(variance, value, -component) %>%
  ggplot(aes(x = component, y = value)) +
  geom_point() +
  facet_wrap(~variance, scales = "free_y") +
  labs(x = "Principal Component", y = "Variance")
```



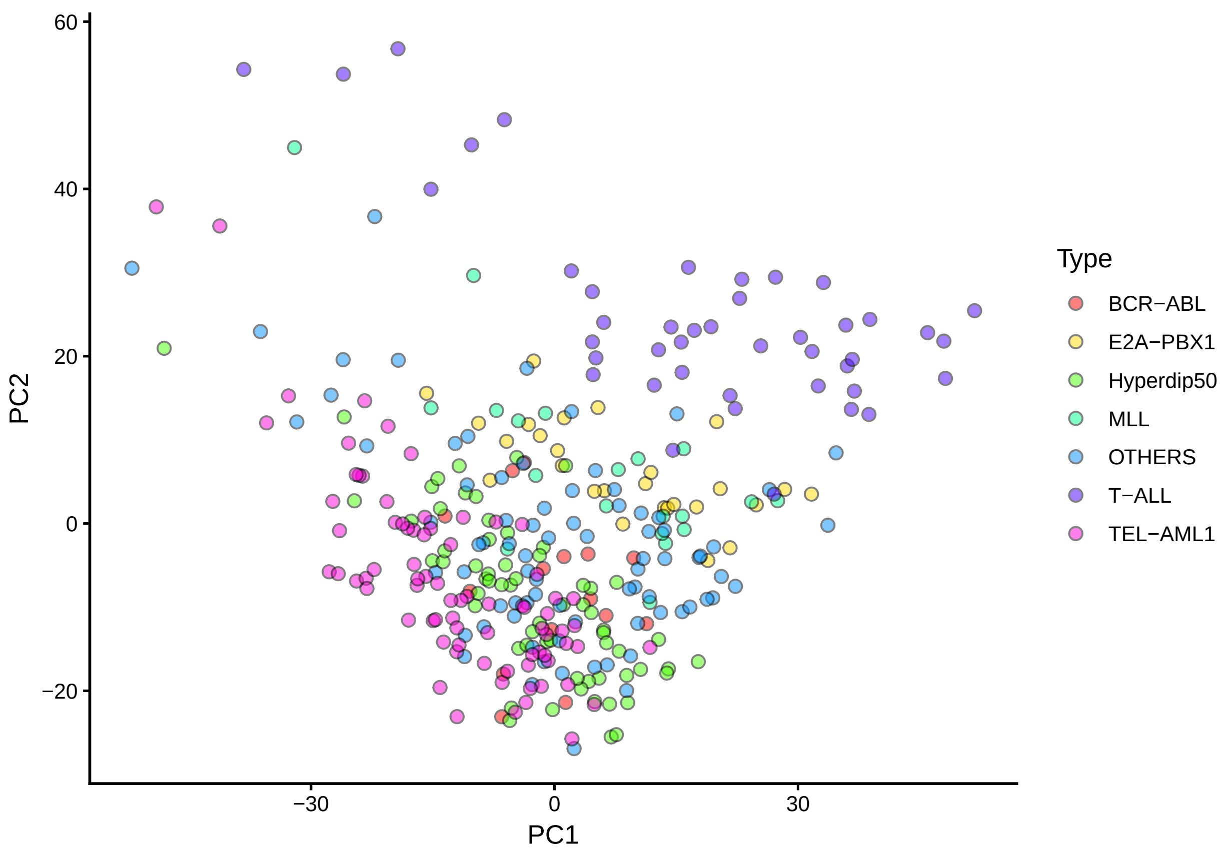
5.3 Use the results of PCA to project the data into the first two principal component dimensions. `prcomp` returns this dimension reduced data in the first columns of `x`. Plot the data as a scatter plot using plot function with `col = plot_colors` where `plot_colors` is defined:

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

```
plot(leuk_pca$x[,1:2], col = plot_colors)
```




```r
#Sorry, but I prefer ggplot2
leuk_pca$x %>%
  as_tibble() %>%
  mutate(Type = leukemia_data$Type) %>%
  ggplot(aes(x = PC1, y = PC2, fill = Type)) +
  geom_point(size = 2, alpha = 0.5) +
  scale_fill_manual(values = rainbow_colors)
```



5.3.1 Which group is most clearly separated from the others along the PC1 axis?

The `T-ALL` (purple) cluster is the furthest appart in the PC1, with coordinates at about (30, 20).

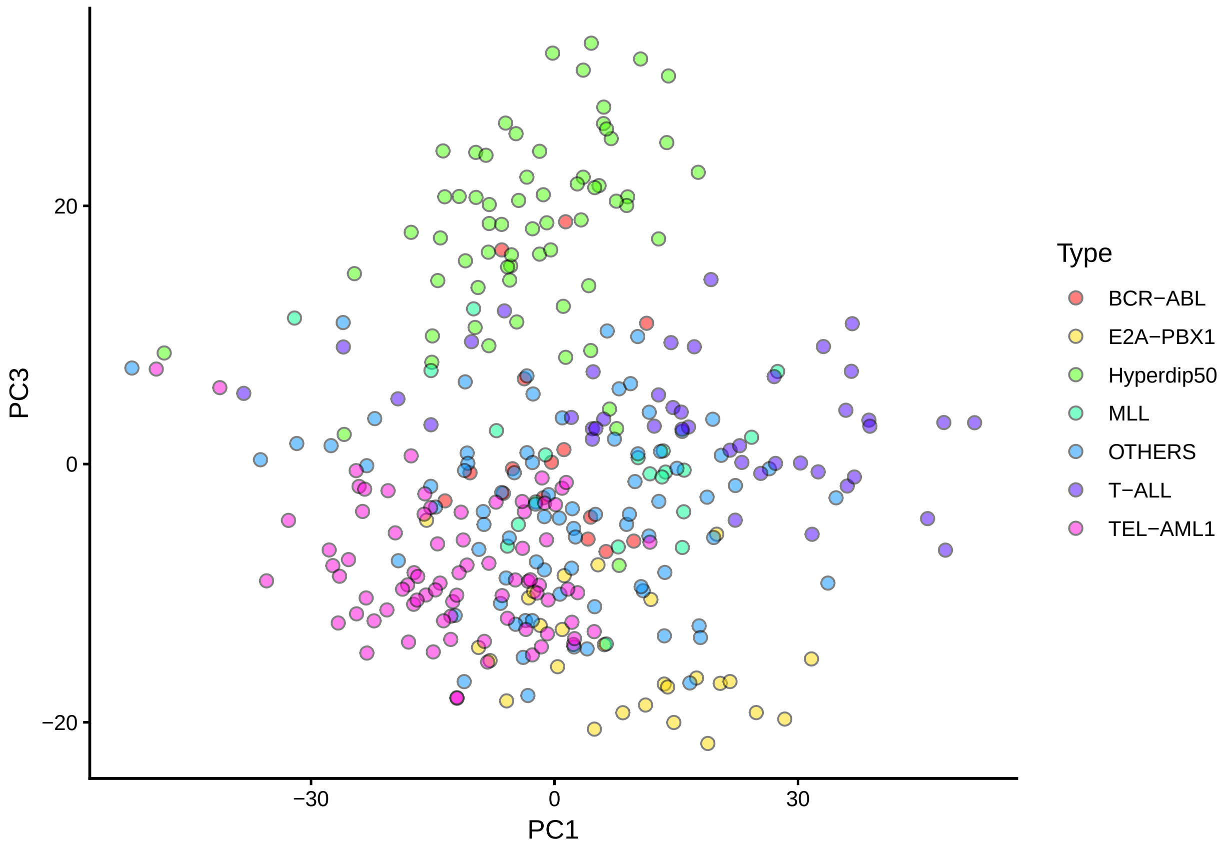
5.3.2 Which genes have the highest absolute loadings for PC1 (the genes that have the largest weights in the weighted average used to create the new variable PC1)?

```
#Get the genes with the top 10 absolute loadings ofr PC1
leuk_pca$rotation[,1] %>%
  abs() %>%
  sort(decreasing = T) %>%
  head()
```

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

5.4 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).

```
leuk_pca$x %>%
  as_tibble() %>%
  mutate(Type = leukemia_data$Type) %>%
  ggplot(aes(x = PC1, y = PC3, fill = Type)) +
  geom_point(size = 2, alpha = 0.5) +
  scale_fill_manual(values = rainbow_colors)
```



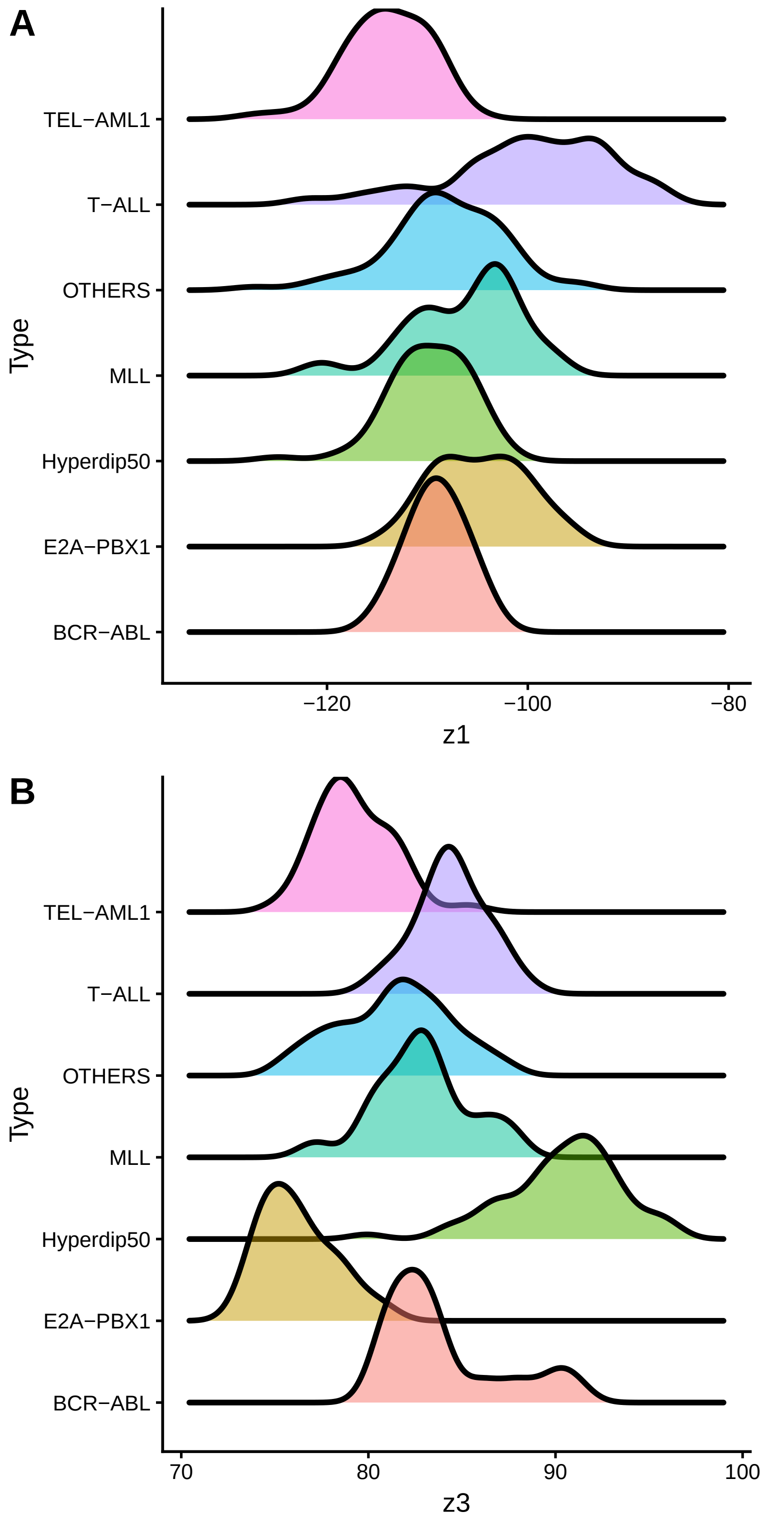
5.5 **For this part we will be using the `ggridges` library. Create a new tibble where the first column (call it `z1`) is the projection of the data onto the first principal component and the second column is the leukemia subtype (`Type`). Use `ggplot` with `geom_density_ridges` to create multiple stacked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

```
leuk_dat <- leukemia_data %>%
  select(-Type) %>%
  as.matrix()

z1 <- leuk_dat %*% leuk_pca$rotation[,1]
z3 <- leuk_dat %*% leuk_pca$rotation[,3]

projections <- tibble(z1 = as.vector(z1),
                      z3 = as.vector(z3),
                      Type = leukemia_data$Type)

z1_plot <- ggplot(data = projections, aes(x = z1, y = Type, fill = Type)) +
  geom_density_ridges() +
  theme(legend.position = "None")

z3_plot <- ggplot(data = projections, aes(x = z3, y = Type, fill = Type)) +
  geom_density_ridges() +
  theme(legend.position = "None")
```

The figure below shows that Hyperdip50 and OTHERS have similar distributions when projected to PC1. However, in PC3 these two types are now different.

```
cowplot::plot_grid(z1_plot, z3_plot,
                   ncol = 1,
                   labels = "AUTO")
```

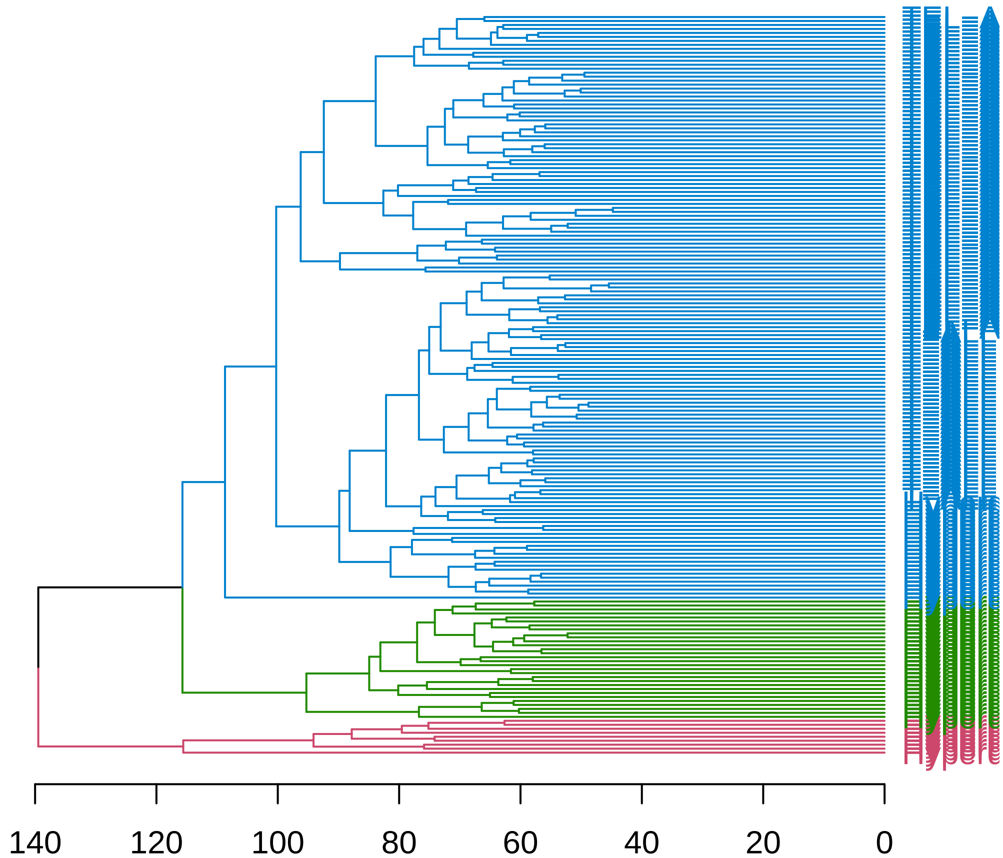
```
## Picking joint bandwidth of 2.07

## Picking joint bandwidth of 0.942
```


A
TEL−AML1
T−ALL
OTHERS
MLL
Hyperdip50
E2A−PBX1
BCR−ABL
Type
−120
−100
−80
z1
B
TEL−AML1
T−ALL
OTHERS
MLL
Hyperdip50
E2A−PBX1
BCR−ABL
Type
70
80
90
100
z3

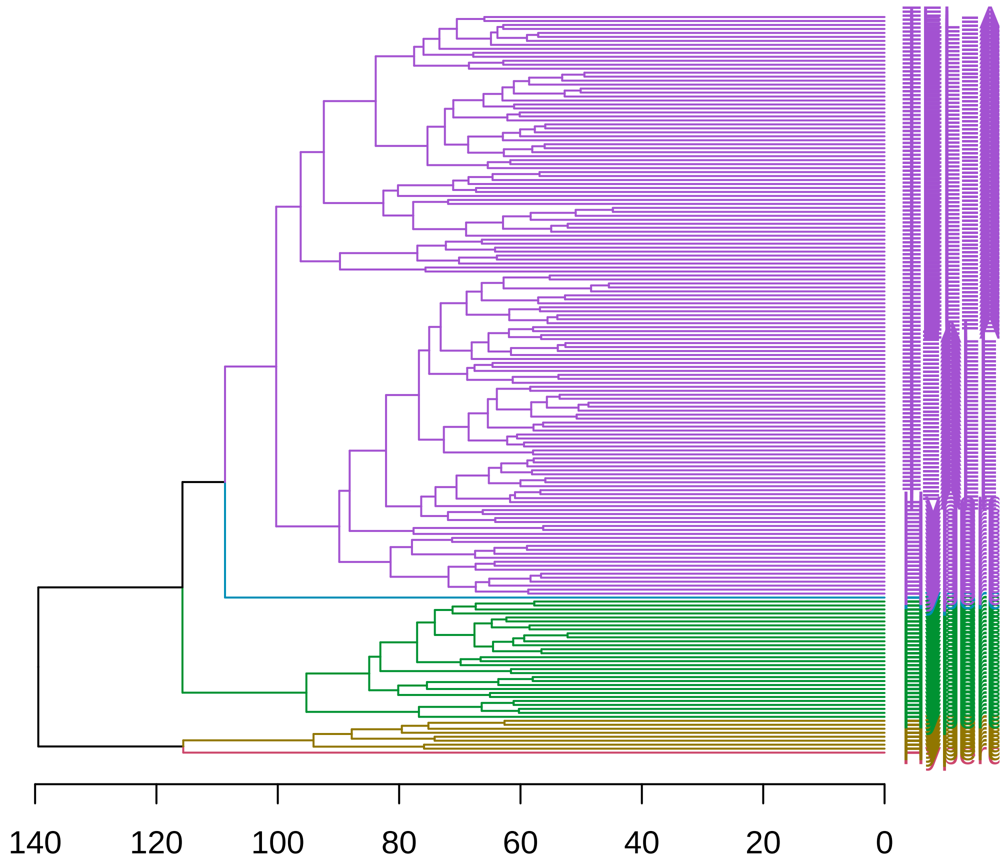
5.6 **Use the `filter` command to create a new `tibble` `leukemia_subset` by subsetting to include only rows for which `Type` is either `T-ALL`, `TEL-AML1`, or `Hyperdip50`. Compute a euclidean distance matrix between the subjects using the `dist` function and then run hierarchical clustering using complete 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 counterclockwise 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.**

```
leukemia_subset <- leukemia_data %>%
  filter(Type %in% c("T-ALL", "TEL-AML1", "Hyperdip50")) %>%
  arrange(Type)

leukemia_hclust <- leukemia_subset %>%
  select(-Type) %>%
  scale() %>%
  dist() %>%
  hclust(method = "complete")
```

```
leukemia_hclust %>%
  as.dendrogram() %>%
  color_labels(k = 3) %>%
  color_branches(k = 3) %>%
  set_labels(labels = leukemia_subset$Type) %>%
  plot(horiz = TRUE)
```



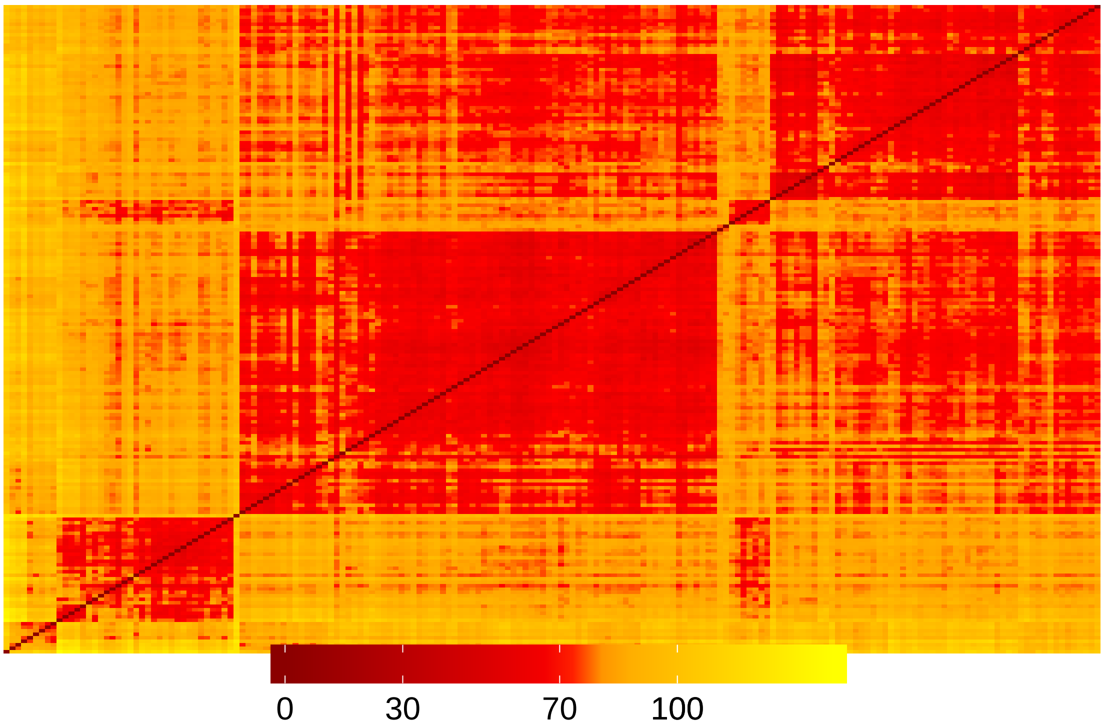

```r
leukemia_hclust %>%
  as.dendrogram() %>%
  color_labels(k = 5) %>%
  color_branches(k = 5) %>%
  set_labels(labels = leukemia_subset$Type) %>%
  plot(horiz = TRUE)
```



5.7 **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.**

```
leuk_dist <- leukemia_subset %>%
  select(-Type) %>%
  scale() %>%
  dist() %>%
```

```r
  as.matrix()

superheat(leuk_dist[leukemia_hclust$order, leukemia_hclust$order],
          heat.pal = c("dark red", "red", "orange", "yellow"))
```




```r
leukemia_subset$Type[leukemia_hclust$order]
```

```
##   [1] Hyperdip50 TEL-AML1   TEL-AML1   T-ALL      T-ALL      T-ALL
##   [7] T-ALL      T-ALL      T-ALL      T-ALL      T-ALL      T-ALL
##  [13] T-ALL      T-ALL      T-ALL      T-ALL      T-ALL      T-ALL
##  [19] T-ALL      T-ALL      T-ALL      T-ALL      T-ALL      T-ALL
##  [25] T-ALL      T-ALL      T-ALL      T-ALL      T-ALL      T-ALL
##  [31] T-ALL      T-ALL      T-ALL      T-ALL      T-ALL      T-ALL
##  [37] T-ALL      T-ALL      T-ALL      Hyperdip50 TEL-AML1   TEL-AML1
##  [43] TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1
##  [49] TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   Hyperdip50
##  [55] Hyperdip50 TEL-AML1   TEL-AML1   TEL-AML1   Hyperdip50 TEL-AML1
##  [61] Hyperdip50 TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1
##  [67] TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1
##  [73] TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1
##  [79] TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1
##  [85] TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1
##  [91] TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1
##  [97] TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1
## [103] TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1
## [109] TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1
```

```
## [115] TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1   TEL-AML1
## [121] TEL-AML1   Hyperdip50 Hyperdip50 T-ALL      T-ALL      T-ALL
## [127] T-ALL      T-ALL      T-ALL      T-ALL      Hyperdip50 Hyperdip50
## [133] Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50
## [139] Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50
## [145] Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50
## [151] Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50
## [157] Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50
## [163] Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50
## [169] Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50
## [175] Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50
## [181] Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50 Hyperdip50
## Levels: BCR-ABL E2A-PBX1 Hyperdip50 MLL OTHERS T-ALL TEL-AML1
```

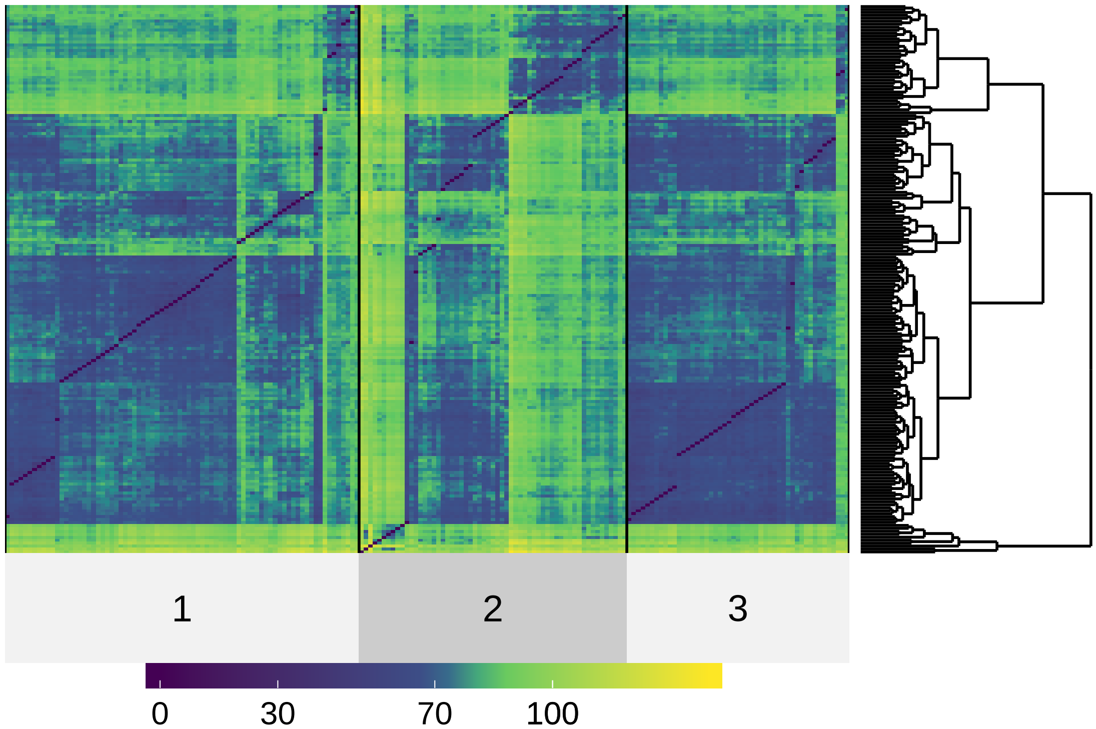
Leukemia types `T-ALL` and `TEL-AML1` seem to be similar to each other, more than `Hyperdip50`.

5.8 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.

Sorry about this one, I don't think the instructions were clear enough. But I think I created something that makes sense.

```
leukemia_kmeans <- leukemia_subset %>%
  select(-Type) %>%
  scale() %>%
  dist() %>%
  kmeans(centers = 3)
```

```
superheat(leuk_dist[leukemia_hclust$order, leukemia_hclust$order],
          row.dendrogram = T,
          clustering.method = "kmeans",
          membership.cols = leukemia_kmeans$cluster,
          pretty.order.cols = T)
```



1
2
3
0
30
70
100