Viper Mortality

Vivi B. Ngo

```
# Packages you definitely need
library(ggplot2)
library(dplyr)
library(ranger) # for looking up help on Conceptual Problem
library(MASS) # see Applied Problem 2
library(parsnip)
library(rsample)
# Please load any other packages you need either all in this chunk
# or in the chunk you need it for
library(keras)
library(tensorflow)
library(ISLR2)
library(esquisse)



# Data you need for applied problems
SoCalRent <- readr::read_csv("SoCalRent.csv")
viper_train <- readr::read_csv("viper_train.csv")
viper_test <- readr::read_csv("viper_test.csv")
```

Snakebite Mortality

Gopalakrishnan and colleagues (2022) attempted to build a model to predict mortality from snakebite based on predictors that could be measured within 48 hours of admission to a hospital or clinic.

The `viper_train` dataset on Canvas contains information about 239 patients from southern India who were diagnosed with having been poisoned by a viper bite. These patients were used to fit their model. The `viper_test` dataset on Canvas contains the same information about a separate set of 140 patients, who were used to validate the model. Please see the `viper_dictionary` file on Canvas for an explanation of each of the variables.

Your goal on this problem is to predict whether a patient will die from snakebite poisoning. Do not expect to get anywhere near as good results as the authors; I could not find the values of two very important predictors in their validation set and so they have been removed from both datasets. Simply show me what you have learned in this class. I will be awarding points for doing the following things correctly:

- Changing the `Outcome` variable appropriately, or creating a new response based on `Outcome`, to reflect that this is a classification problem

```
viper_test$Outcome <- factor(viper_test$Outcome)
viper_train$Outcome <- factor(viper_train$Outcome)
```

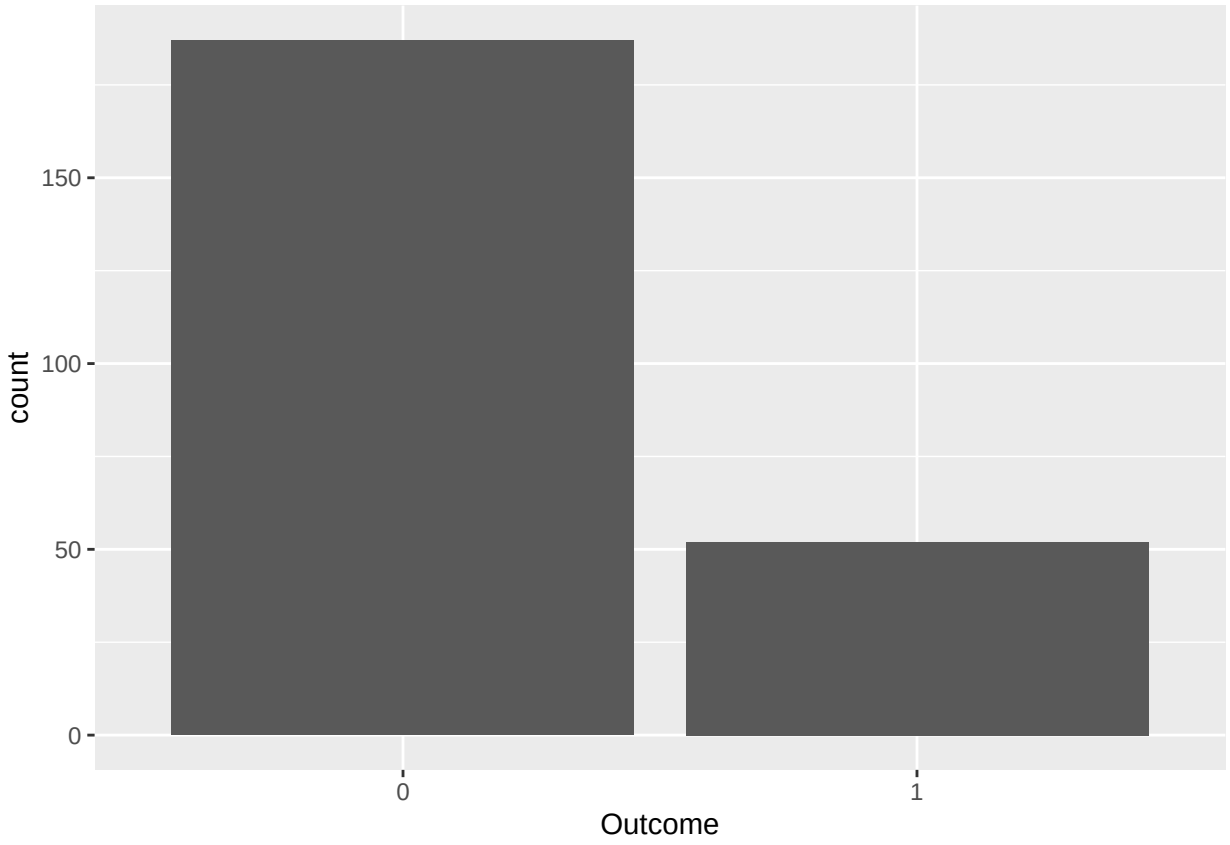
```
#viper_test$Sex <- factor(viper_test$Sex)
#viper_train$Sex <- factor(viper_train$Sex)
#viper_test$Outcome<-ifelse(viper_test$Outcome=="0",0,1)
#viper_train$Outcome<-ifelse(viper_train$Outcome=="0",0,1)

viper_test$Sex<-ifelse(viper_test$Sex=="M",1,0)
viper_train$Sex<-ifelse(viper_train$Sex=="M",1,0)
```

- Performing exploratory data analysis on the training set and documenting your findings

```
#eda on response of interest: outcome
viper_train %>% group_by(Outcome)%>%count()
```

```
## # A tibble: 2 x 2
## # Groups:   Outcome [2]
##   Outcome     n
##   <fct>   <int>
## 1 0         187
## 2 1          52
```

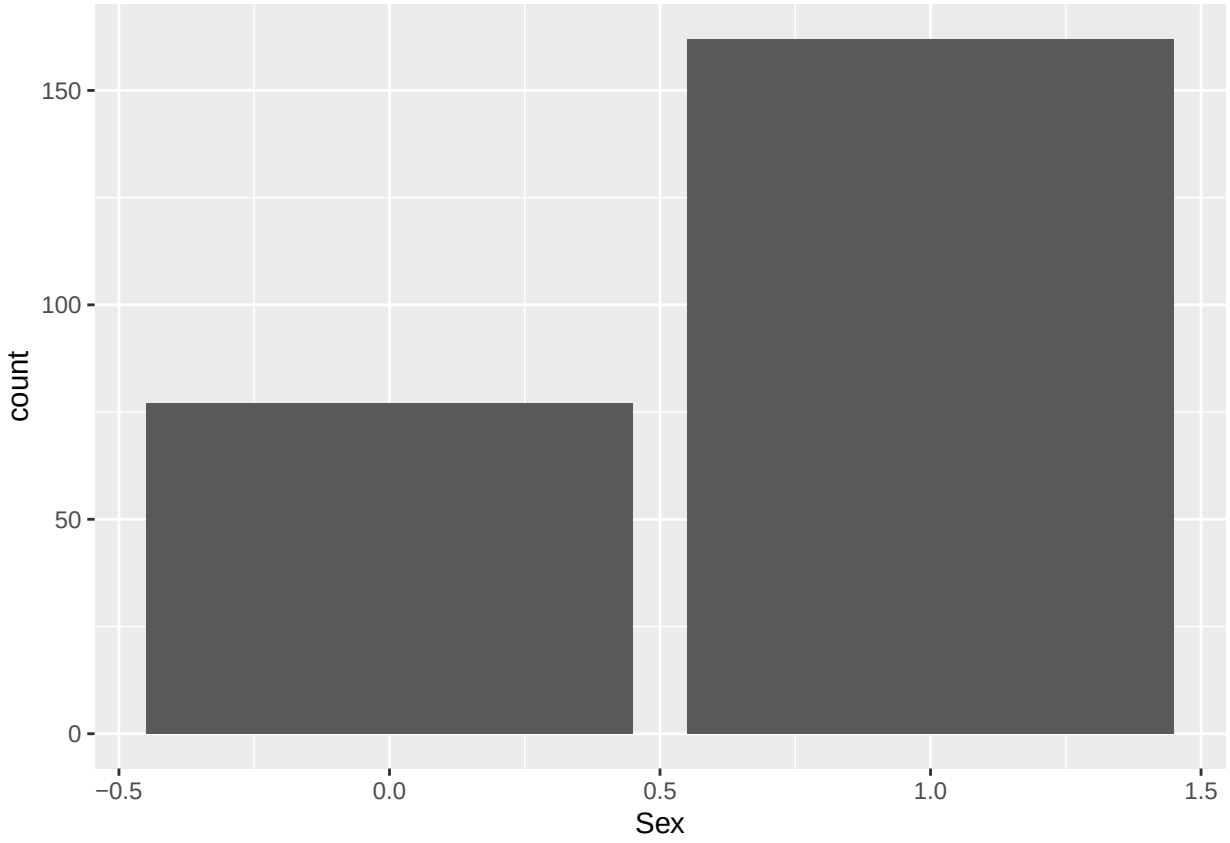
```
#we have 187 people who lived a snake bite and 52 who died in this training set
ggplot(viper_train, aes(x=Outcome)) + geom_bar()
```



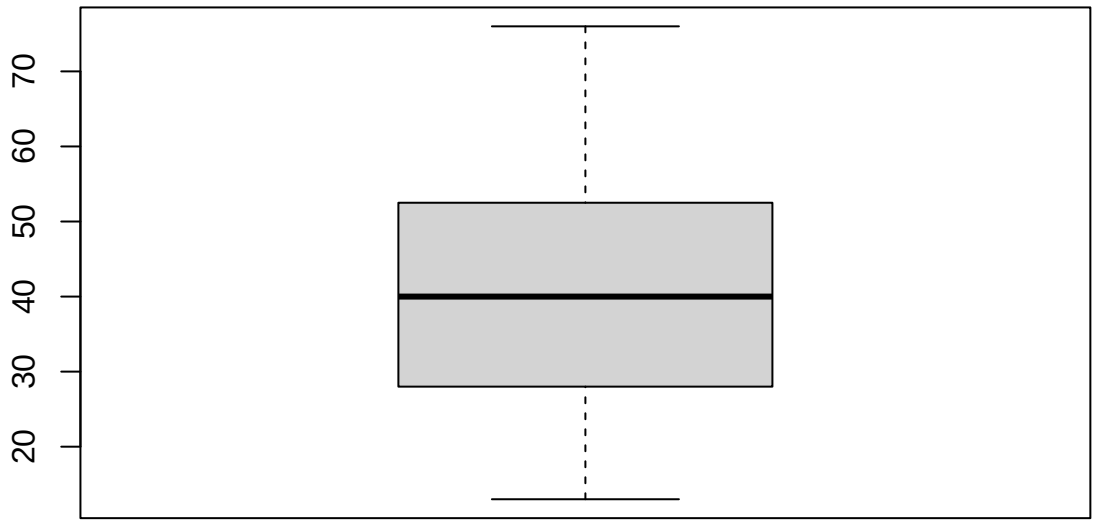
```
#eda on predictors
#77 females and 162 males in the training set
viper_train %>% group_by(Sex)%>%count()
```

```
## # A tibble: 2 x 2
## # Groups:   Sex [2]
##     Sex     n
##   <dbl> <int>
## 1     0    77
## 2     1   162
```

```
ggplot(viper_train, aes(x=Sex)) + geom_bar()
```




```
#age
age_boxplot <- boxplot(viper_train$Age)
```

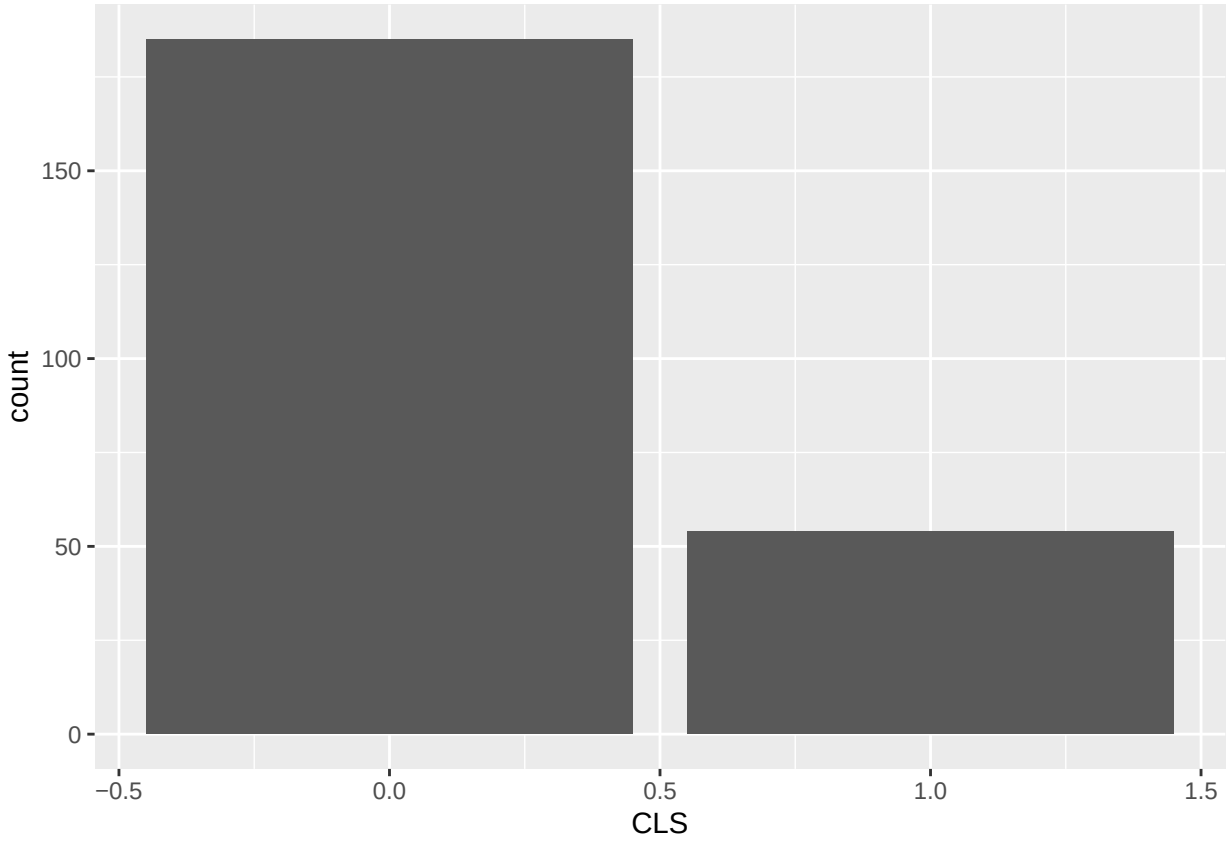



```
age_hist <- hist(viper_train$Age, xlab = 'patient age')
```


```
#CLS capillary leak syndrome
# no(0) = 185, yes(1) = 54
viper_train %>% group_by(CLS)%>%count()
```

```
## # A tibble: 2 x 2
## # Groups:   CLS [2]
##     CLS     n
##   <dbl> <int>
## 1     0   185
## 2     1    54
```

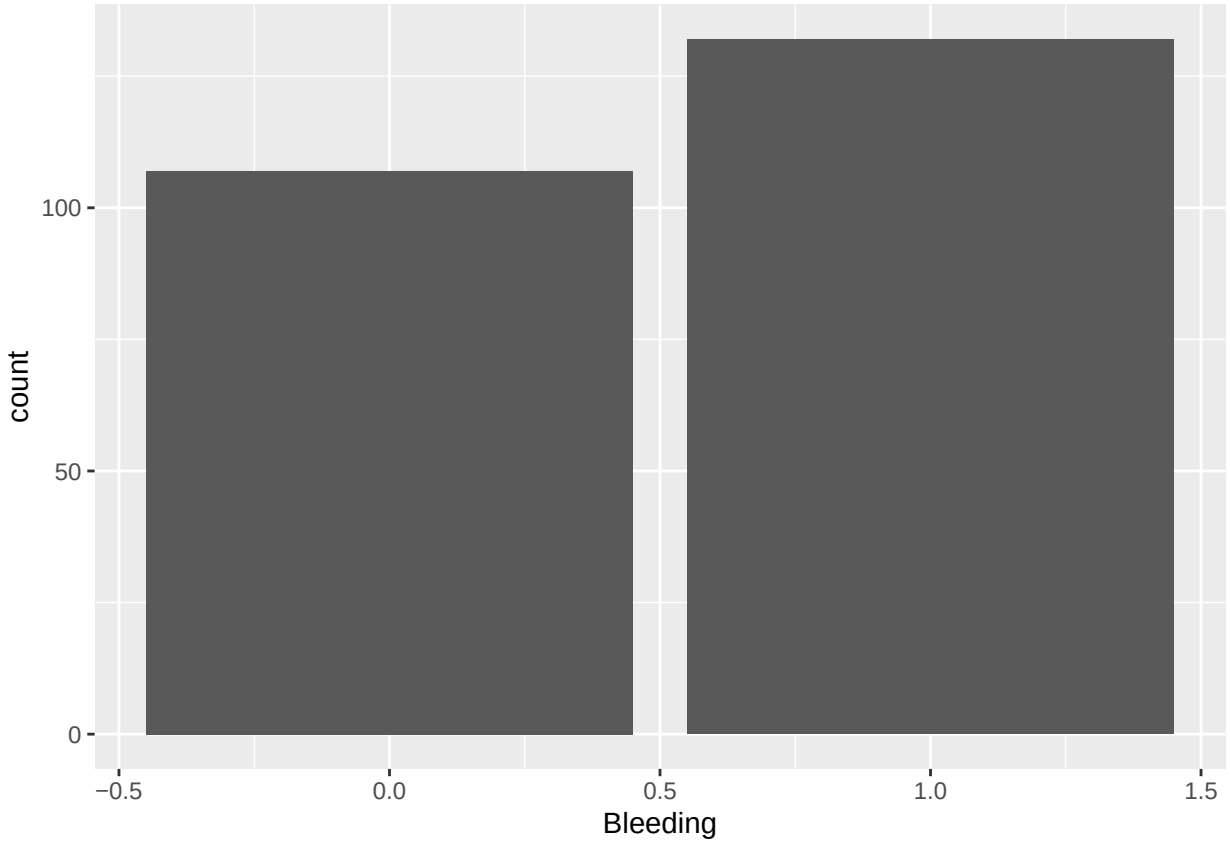
```
ggplot(viper_train, aes(x=CLS)) + geom_bar()
```



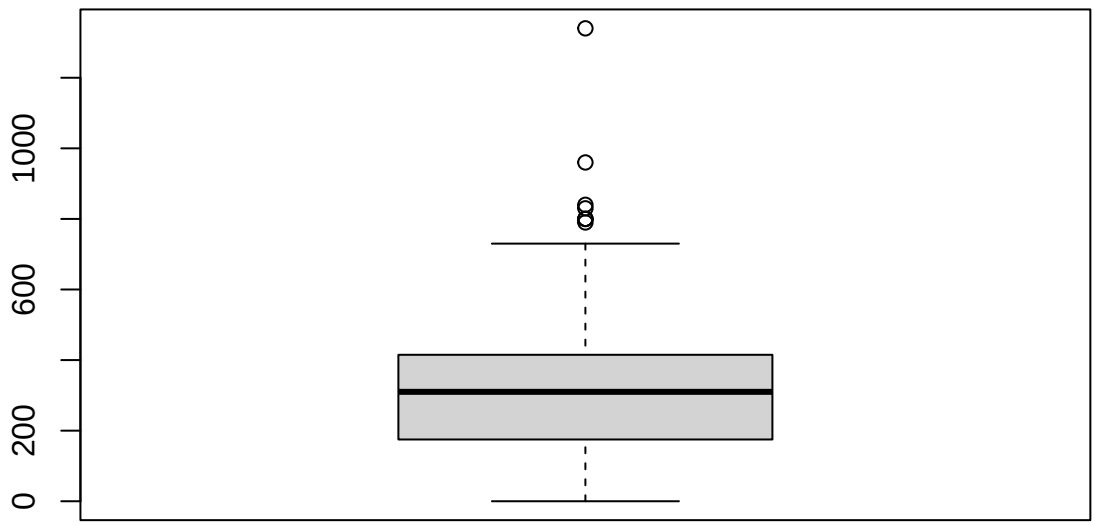

```
#Bleeding
# no bleeding = 107, yes bleeding = 132
viper_train %>% group_by(Bleeding)%>%count()
```

```
## # A tibble: 2 x 2
## # Groups:   Bleeding [2]
##   Bleeding     n
##      <dbl> <int>
## 1        0   107
## 2        1   132
```

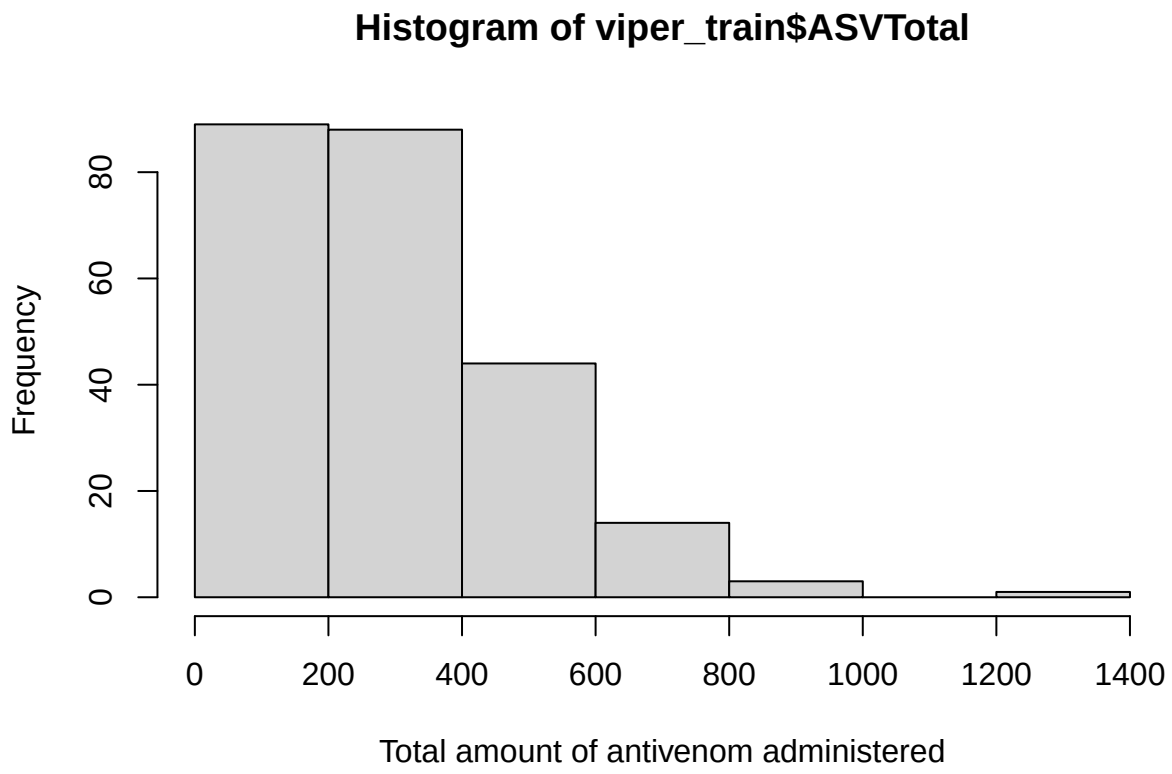
```
ggplot(viper_train, aes(x=Bleeding)) + geom_bar()
```



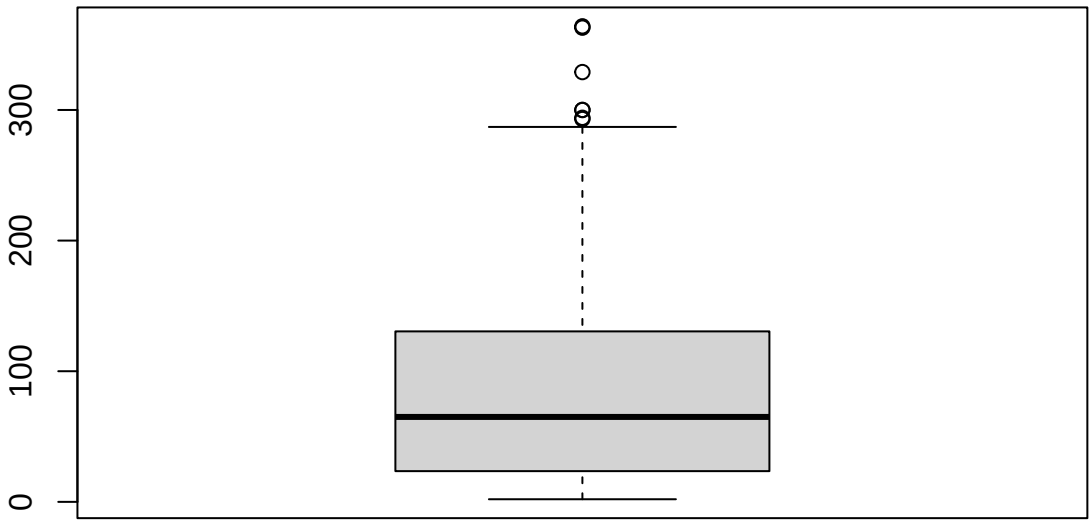

```
#ASVTotal
asv_boxplot <- boxplot(viper_train$ASVTotal)
```



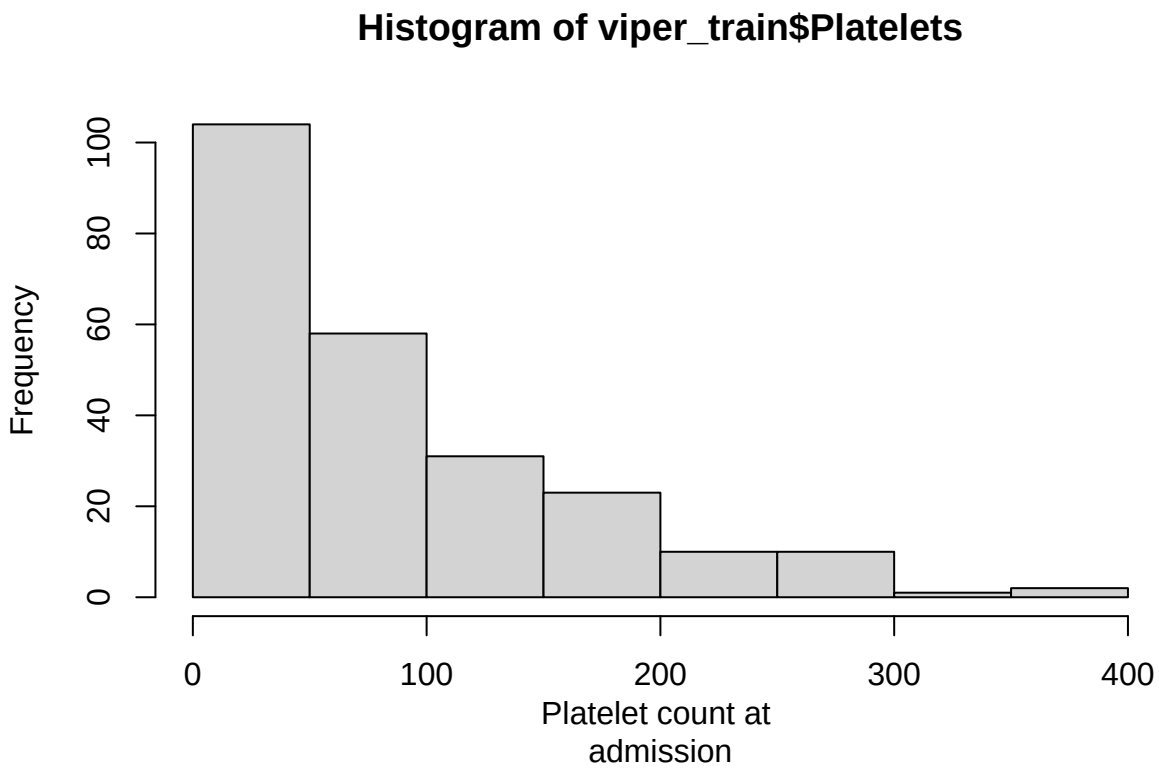

```
asv_hist <- hist(viper_train$ASVTotal, xlab = 'Total amount of antivenom administered')
```



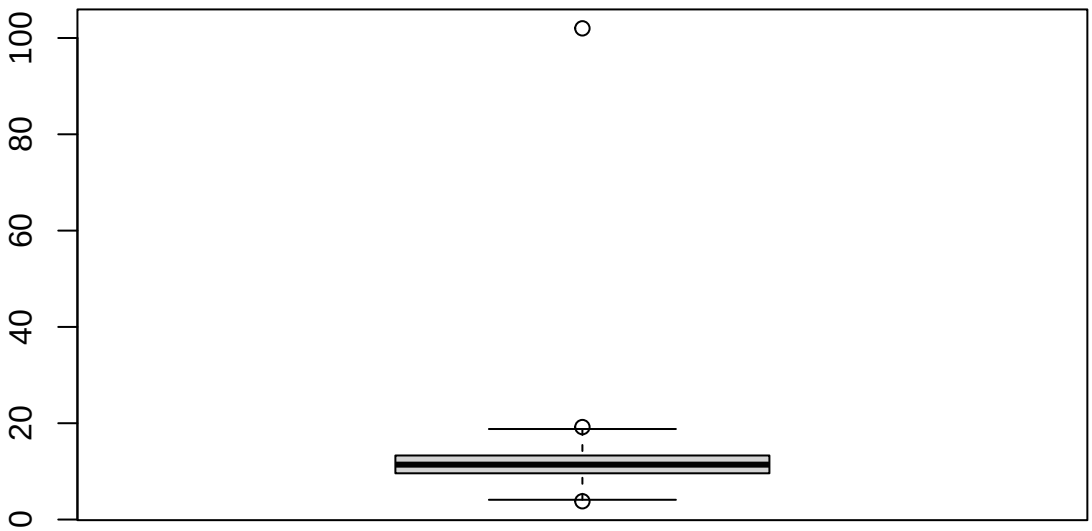

```
#platelets
plate_boxplot <- boxplot(viper_train$Platelets)
```



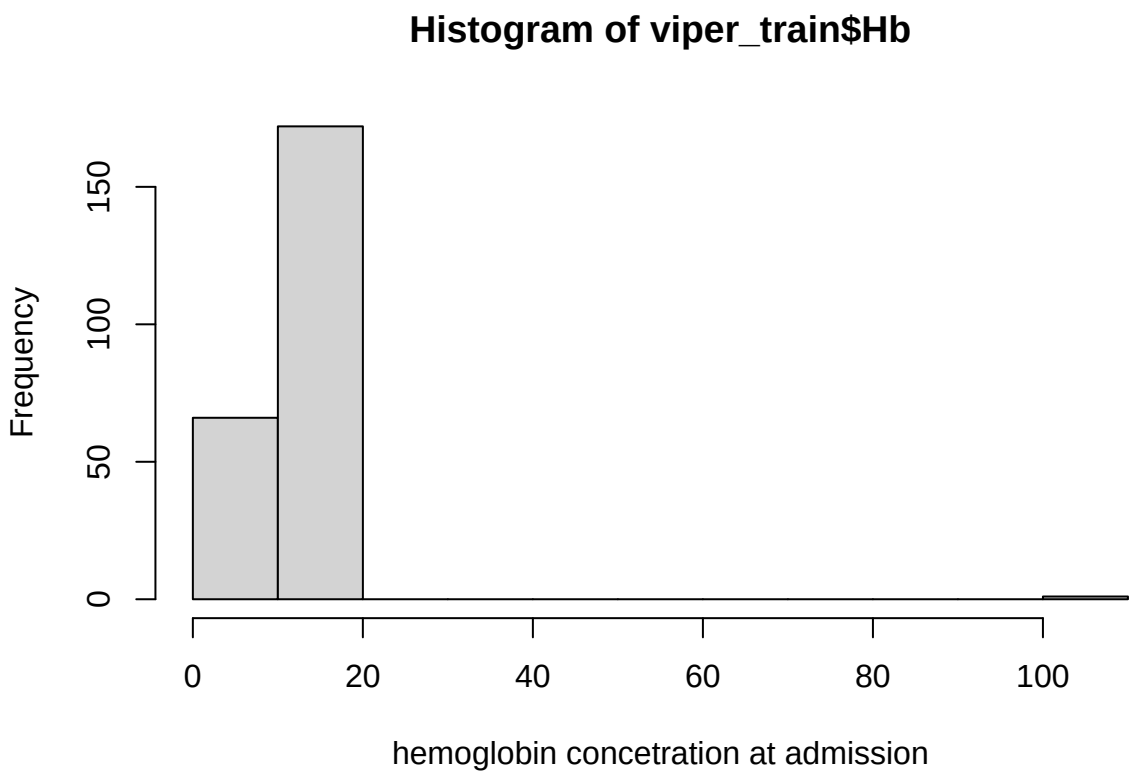

```
plate_hist <- hist(viper_train$Platelets, xlab = 'Platelet count at
admission')
```



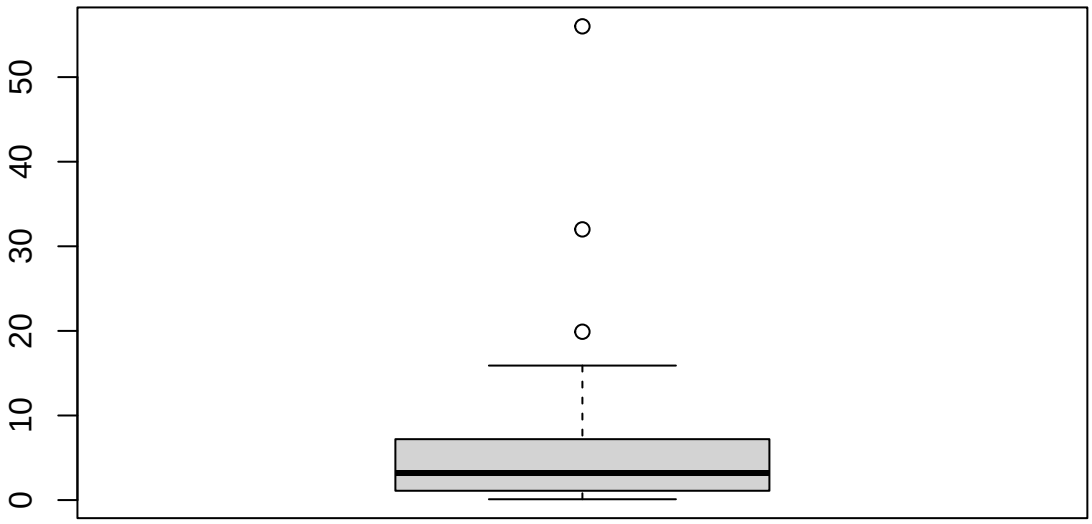

```
#Hb

hb_boxplot <- boxplot(viper_train$Hb)
```



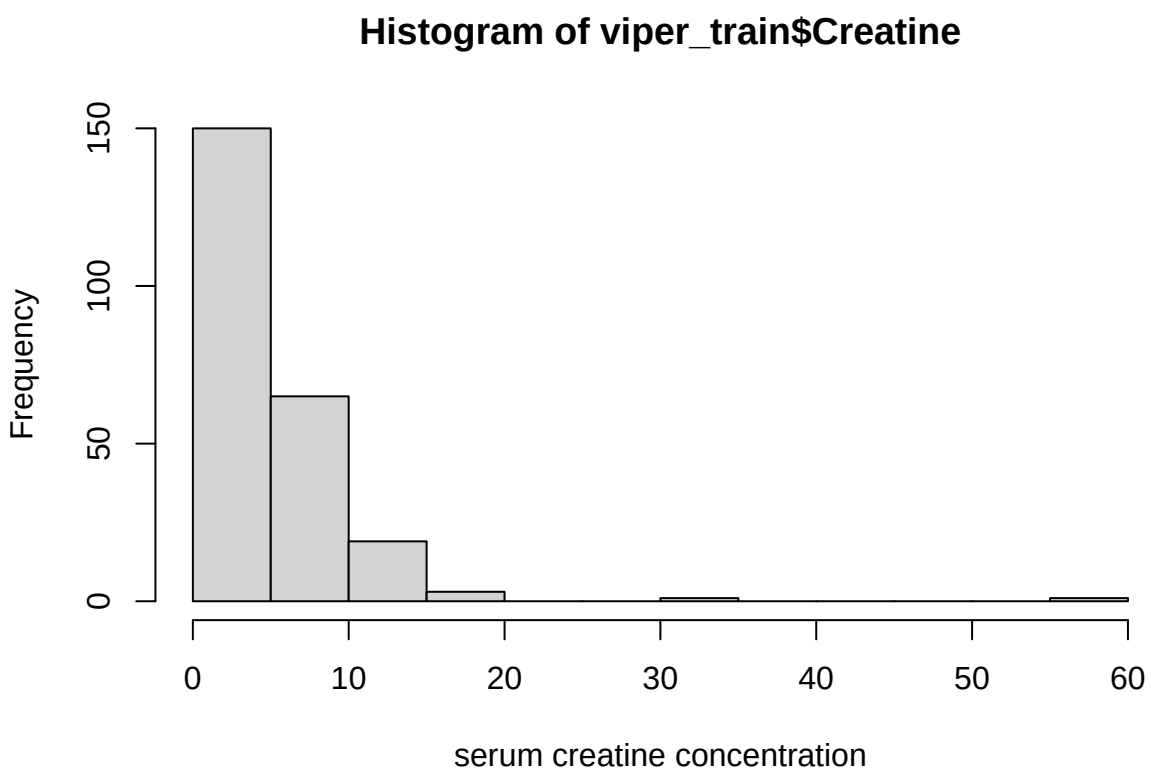

```
hb_hist <- hist(viper_train$Hb, xlab = 'hemoglobin concetration at admission')
```



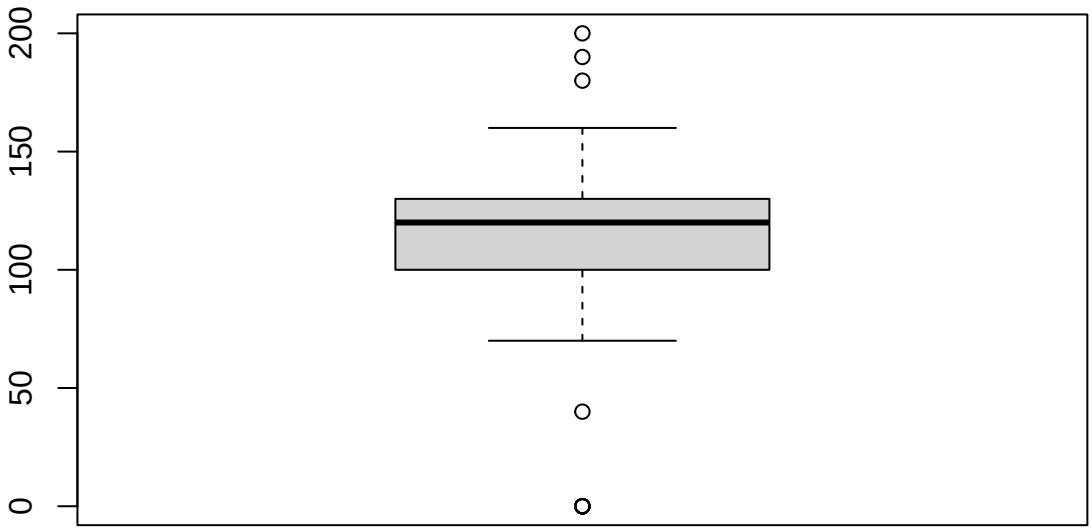

```
#creatine
creatine_boxplot <- boxplot(viper_train$Creatine)
```



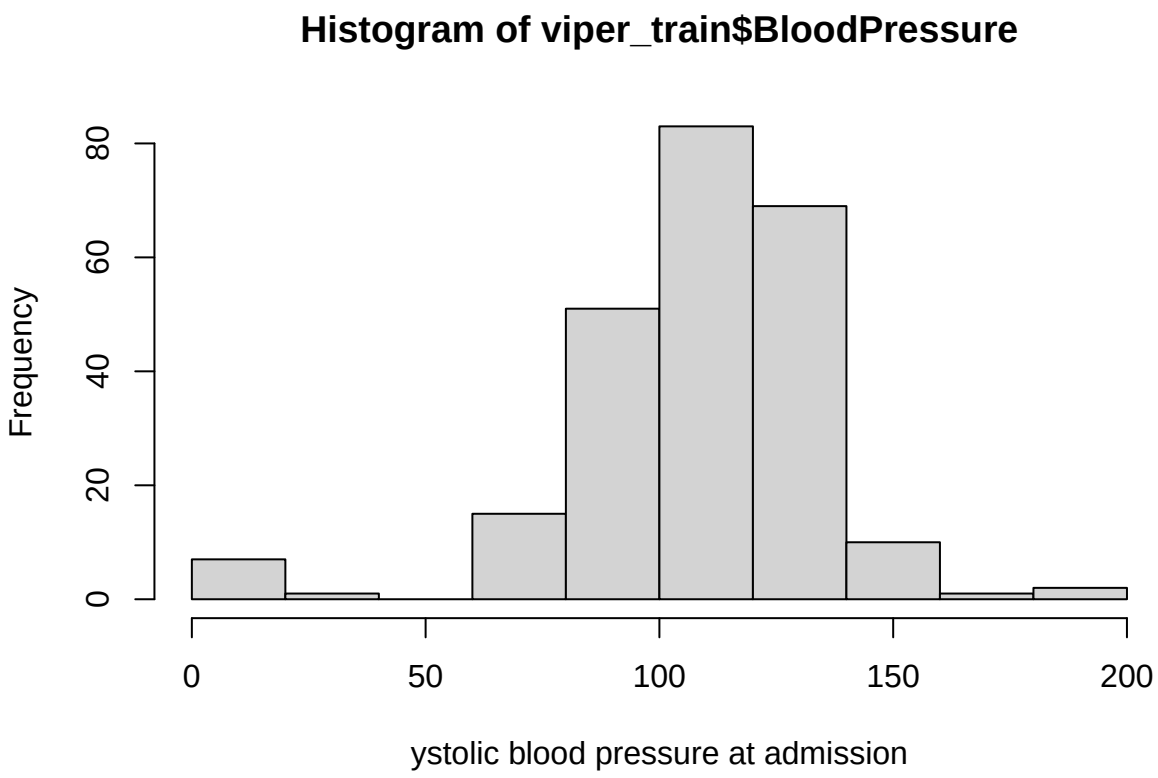

```
creatine_hist <- hist(viper_train$Creatine, xlab = 'serum creatine concentration')
```




```
#blood pressure
#is it possible to have 200bp?
#can dead people have more than 0 bp?
bp_boxplot <- boxplot(viper_train$BloodPressure)
```




```
bp_hist <- hist(viper_train$BloodPressure, xlab = 'ystolic blood pressure at admission')
```



- Performing backward stepwise selection using BIC as a selection criterion and obtaining an estimate of the test error rate on the selected logistic regression model (HINT: `regsubsets` will not work for logistic regression, but you can use the `stepAIC` function in the `MASS` package instead to do stepwise selection. The output of `stepAIC` will be a `glm` object, which means you can do prediction without the workarounds)

```
library(MASS)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.6     v purrr   0.3.4
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

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

```
set.seed(10)


outcome_test <- viper_test$Outcome
```

```r
bglm <- glm(Outcome~.,data=viper_train, family = 'binomial')
backward <- stepAIC(bglm, data=viper_train,method='backward')
```

```
## Start:  AIC=114.95
## Outcome ~ Age + Sex + CLS + Bleeding + ASVTotal + Platelets +
##     Hb + Creatine + BloodPressure
##
##                 Df Deviance    AIC
## - Creatine       1   95.008 113.01
## - Age            1   95.263 113.26
## - Platelets      1   95.530 113.53
## - Sex            1   96.910 114.91
## <none>               94.951 114.95
## - Hb             1   98.794 116.79
## - ASVTotal       1  102.381 120.38
## - BloodPressure  1  117.582 135.58
## - CLS            1  119.912 137.91
## - Bleeding       1  122.572 140.57
##
## Step:  AIC=113.01
## Outcome ~ Age + Sex + CLS + Bleeding + ASVTotal + Platelets +
##     Hb + BloodPressure
##
##                 Df Deviance    AIC
## - Age            1   95.339 111.34
## - Platelets      1   95.904 111.90
## - Sex            1   96.982 112.98
## <none>               95.008 113.01
## - Hb             1   98.933 114.93
## - ASVTotal       1  102.650 118.65
## - BloodPressure  1  118.002 134.00
## - CLS            1  120.162 136.16
## - Bleeding       1  122.963 138.96
##
## Step:  AIC=111.34
## Outcome ~ Sex + CLS + Bleeding + ASVTotal + Platelets + Hb +
##     BloodPressure
##
##                 Df Deviance    AIC
## - Platelets      1   96.233 110.23
## <none>               95.339 111.34
## - Sex            1   97.724 111.72
## - Hb             1   99.717 113.72
## - ASVTotal       1  102.727 116.73
## - BloodPressure  1  118.310 132.31
## - CLS            1  120.870 134.87
## - Bleeding       1  124.021 138.02
##
## Step:  AIC=110.23
## Outcome ~ Sex + CLS + Bleeding + ASVTotal + Hb + BloodPressure
##
##                 Df Deviance    AIC
## <none>               96.233 110.23
```

```
## - Sex            1   98.405 110.41
## - Hb             1  101.192 113.19
## - ASVTotal       1  103.579 115.58
## - BloodPressure  1  121.528 133.53
## - CLS            1  125.266 137.27
## - Bleeding       1  127.859 139.86
```

```
back_pred <- predict(backward, viper_test)



#back_error <- sum((outcome_test/sum(backward$residuals))^2)
```

- Fitting at least two of the following four types of models and obtaining an estimate of the test error rate on each model: k-nearest neighbors, generative models (LDA/QDA/naive Bayes), tree-based methods (bagging/random forests/boosting), neural networks

```
set.seed(10)

#viper_test$Sex<-ifelse(viper_test$Sex=="M",1,0)
#viper_train$Sex<-ifelse(viper_train$Sex=="M",1,0)
#boosted tree



x_scale_test <- data.frame(
outcome = as.numeric(viper_test$Outcome),
sex = as.numeric(viper_test$Sex),
age = scale(viper_test$Age),
cls = scale(viper_test$CLS),
bleeding = scale(viper_test$Bleeding),
asvTotal = scale(viper_test$ASVTotal),
platelets = scale(viper_test$Platelets),
hb = scale(viper_test$Hb),
creatine = scale(viper_test$Creatine),
bp = scale(viper_test$BloodPressure)
) %>% as.matrix()


x_scale_train <- data.frame(
outcome = as.numeric(viper_train$Outcome),
sex = as.numeric(viper_train$Sex),
age = scale(viper_train$Age),
cls = scale(viper_train$CLS),
bleeding = scale(viper_train$Bleeding),
asvTotal = scale(viper_train$ASVTotal),
platelets = scale(viper_train$Platelets),
hb = scale(viper_train$Hb),
creatine = scale(viper_train$Creatine),
bp = scale(viper_train$BloodPressure)
) %>% as.matrix()
```

```r
#viper_test$Sex <- as.numeric(viper_test$Sex)

#1/2 (fast n dirty random forest)
library(ranger)
viper_rf<- ranger(Outcome~., data=viper_train, importance = "permutation", seed=758)
viper_rf
```

```
## Ranger result
##
## Call:
##  ranger(Outcome ~ ., data = viper_train, importance = "permutation",      seed = 758)
##
## Type:                             Classification
## Number of trees:                  500
## Sample size:                      239
## Number of independent variables:  9
## Mtry:                             3
## Target node size:                 1
## Variable importance mode:         permutation
## Splitrule:                        gini
## OOB prediction error:             10.88 %
```

```r
#estimate the test MSE :3333333
#(find the default test)
viper_class <- predict(viper_rf, data=viper_test)$predictions

#predict each tree and then avg the preds
viper_predictions <- predict(viper_rf, data = viper_test, predict.all = TRUE)

rf_probs <- apply(viper_predictions$predictions-1,1,mean)



#neural net
library(keras)


nn_1layer <- keras_model_sequential() %>%
layer_dense(units = 10, activation = "relu",
input_shape = ncol(x_scale_train)) %>%
layer_dropout(rate = 0.4) %>%
layer_dense(units = 1, activation = "sigmoid")
```

```
## Loaded Tensorflow version 2.9.0
```

```r
nn_1layer %>% compile(loss = "binary_crossentropy",
optimizer = optimizer_rmsprop(),
metrics = list("accuracy"))

outcome_train <- as.numeric(viper_train$Outcome)
```

```
nn_fit <- nn_1layer %>% fit(x = x_scale_train,
y = outcome_train)

nn_preds <- predict(nn_1layer,x=x_scale_test)
nn_class <- if_else(nn_preds >= 0.5, '0','1')

library(yardstick)
```

```
## Warning: package 'yardstick' was built under R version 4.1.3
```

```
## For binary classification, the first factor level is assumed to be the event.
## Use the argument `event_level = "second"` to alter this as needed.
```

```
##
## Attaching package: 'yardstick'
```

```
## The following object is masked from 'package:readr':
##
##     spec
```

```
## The following object is masked from 'package:keras':
##
##     get_weights
```

```
prediction_df <- data.frame(
  actual = viper_test$Outcome,
  rf = viper_class,
  nn = factor(nn_class, levels = c('0','1'))
)
```

- Further investigating the prediction accuracy of at least one model (e.g., by creating a confusion matrix or a ROC Curve, computing sensitivity/specificity, etc.)

```
#nnet
#nn
(nn_conf<- conf_mat(prediction_df, truth = actual, estimate = nn))
```

```
##            Truth
## Prediction   0   1
##          0 111  16
##          1   9   4
```

```
(rf_conf<- conf_mat(prediction_df, truth = actual, estimate = rf))
```

```
##            Truth
## Prediction   0   1
##          0 113  14
##          1   7   6
```

- Selecting a best model and justifying your choice; in particular, you should explain why your model is better than a very stupid model that predicts no one will die

My best model is accurate 85% of the time, but at least it is still better than predicting no one will die(?)

```
summary(nn_conf, event_level = "second")
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.821
##  2 kap                  binary         0.146
##  3 sens                 binary         0.2
##  4 spec                 binary         0.925
##  5 ppv                  binary         0.308
##  6 npv                  binary         0.874
##  7 mcc                  binary         0.151
##  8 j_index              binary         0.125
##  9 bal_accuracy         binary         0.562
## 10 detection_prevalence binary         0.0929
## 11 precision            binary         0.308
## 12 recall               binary         0.2
## 13 f_meas               binary         0.242
```

```
table(prediction_df$actual,prediction_df$nn)
```

```
##
##       0   1
##   0 111   9
##   1  16   4
```

```
summary(rf_conf,event_level = 'second')
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.85
##  2 kap                  binary         0.283
##  3 sens                 binary         0.3
##  4 spec                 binary         0.942
##  5 ppv                  binary         0.462
##  6 npv                  binary         0.890
##  7 mcc                  binary         0.291
##  8 j_index              binary         0.242
##  9 bal_accuracy         binary         0.621
## 10 detection_prevalence binary         0.0929
## 11 precision            binary         0.462
## 12 recall               binary         0.3
## 13 f_meas               binary         0.364
```