131 Final Project

John Wei

Contents

Introduction

One person every 36 seconds dies of heart disease in the United States. It is the leading cause of death in this country and identifying what correlates and indicates the possibility of one and using it to predict is useful to hopefully not fall to the unfortunate statistic of 1 in 4 Americans dying to it. Prevention is possible!

The original data is from the Cleveland data from the UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/heart+Disease I downloaded the formatted version from Kaggle: https://www.kaggle.com/datasets/cherngs/heart-disease-cleveland-uci?resource=download

Purpose

The idea behind this project is to predict whether an individual will have a heart disease with the following indicators:

The data set has 297 observations with 14 variables. Target is the response variable, with the other 13 variables predictors.

1. Age - age in years
2. Sex - (1 = male; 0 = female)
3. CP - chest pain type: Typical angina: chest pain related decrease blood supply to the heart Atypical angina: chest pain not related to heart Non-anginal pain: typically esophageal spasms (non heart related) Asymptomatic: chest pain not showing signs of disease
4. Trestbps - resting blood pressure (in mm Hg on admission to the hospital) anything above 130-140 is typically cause for concern
5. Chol - serum cholestorol in mg/dl serum = LDL + HDL + .2 * triglycerides above 200 is cause for concern
6. Fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false) '>126' mg/dL signals diabetes

7. Restecg - resting electrocardiographic results; 0: Nothing to note 1: ST-T Wave abnormality can range from mild symptoms to severe problems signals non-normal heart beat 2: Possible or definite left ventricular hypertrophy Enlarged heart's main pumping chamber
8. Thalach - maximum heart rate achieved
9. Exang - exercise induced angina (1 = yes; 0 = no)
10. Oldpeak - ST depression induced by exercise relative to rest; looks at stress of heart during exercise - an unhealthy heart will stress more
11. Slope - the slope of the peak exercise ST segment; 0: Upsloping: better heart rate with excercise (uncommon) 1: Flatsloping: minimal change (typical healthy heart) 2: Downsloping: signs of unhealthy heart
12. Ca - number of major vessels (0-3) colored by flourosopy colored vessel means the doctor can see the blood passing through the more blood movement the better (no clots)
13. Thal - thalium stress result; 0: normal 1: fixed defect: used to be defect but ok now 2: reversable defect: no proper blood movement when exercising
14. Condition - have disease or not (1 = yes, 0 = no)

R Markdown

Loading Data and Packages

```
library(ranger)
library(janitor)
library(rpart.plot)
library(ggplot2)
library(tidyverse)
library(tidymodels)
library(tibble)
library(corrplot)
library(yardstick)
library(corrr)
library(pROC)
library(glmnet)
library(ggthemes)
library(vip)
library(xgboost)
library(kknn)
library(psych)
library(dplyr)
library(knitr)
library(haven)
library(lubridate, warn.conflicts = FALSE)
tidymodels_prefer()
set.seed(4167)
```

Let's look at some of our data!

```r
df <- read_csv("heart_cleveland_upload.csv")
head(df)
```

```
## # A tibble: 6 x 14
##     age   sex    cp trestbps  chol   fbs restecg thalach exang oldpeak slope
##   <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl>   <dbl>   <dbl> <dbl>   <dbl> <dbl>
## 1    69     1     0      160   234     1       2     131     0     0.1     1
## 2    69     0     0      140   239     0       0     151     0     1.8     0
## 3    66     0     0      150   226     0       0     114     0     2.6     2
## 4    65     1     0      138   282     1       2     174     0     1.4     1
## 5    64     1     0      110   211     0       2     144     1     1.8     1
## 6    64     1     0      170   227     0       2     155     0     0.6     1
## # ... with 3 more variables: ca <dbl>, thal <dbl>, condition <dbl>
```

Start by cleaning and checking data.

```r
df <- df %>% clean_names()
head(df)
```

```
## # A tibble: 6 x 14
##     age   sex    cp trestbps  chol   fbs restecg thalach exang oldpeak slope
##   <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl>   <dbl>   <dbl> <dbl>   <dbl> <dbl>
## 1    69     1     0      160   234     1       2     131     0     0.1     1
## 2    69     0     0      140   239     0       0     151     0     1.8     0
## 3    66     0     0      150   226     0       0     114     0     2.6     2
## 4    65     1     0      138   282     1       2     174     0     1.4     1
## 5    64     1     0      110   211     0       2     144     1     1.8     1
## 6    64     1     0      170   227     0       2     155     0     0.6     1
## # ... with 3 more variables: ca <dbl>, thal <dbl>, condition <dbl>
```

Then checking if there are missing values; there are not.

```r
sum(is.na(df))
```

```
## [1] 0
```

Checking if responsible variable is balanced; since these two values are close to even, our target column can be considered balanced.

```r
table(df$condition)
```

```
##
##   0   1
## 160 137
```

Converting `sex`, `cp`, `fbs`, `restecg`, `slope`, `thal`, `condition` to factors.

```
df$sex <- factor(df$sex,
labels = c("Female", "Male"))
df$cp <- factor(df$cp,
labels = c("Typical angina", "Atypical angina","Non-anginal", "Asymptomatic"))
df$fbs <- factor(df$fbs,
labels = c("Less than 120 mg/dl", "Greater than 120 mg/dl"))
df$restecg <- factor(df$restecg,
labels = c("Nothing to note", "ST-T wave abnormality", "Possible left ventricular hypertrophy"))
df$exang <- factor(df$exang,
labels = c('No', 'Yes'))
df$slope <- factor(df$slope,
labels = c("Upsloping", "Flatsloping", "Downsloping"))
df$ca <- factor(df$ca)
df$thal <- factor(df$thal,
labels = c("Normal", "Fixed Defect", "Reversible Defect"))
df$condition <- factor(df$condition,
labels = c("No", "Yes"))
head(df)
```

```
## # A tibble: 6 x 14
##     age sex    cp       trestbps  chol fbs   restecg thalach exang oldpeak slope
##   <dbl> <fct>  <fct>       <dbl> <dbl> <fct> <fct>     <dbl> <fct>   <dbl> <fct>
## 1    69 Male   Typical~      160   234 Grea~ Possib~     131 No        0.1 Flat~
## 2    69 Female Typical~      140   239 Less~ Nothin~     151 No        1.8 Upsl~
## 3    66 Female Typical~      150   226 Less~ Nothin~     114 No        2.6 Down~
## 4    65 Male   Typical~      138   282 Grea~ Possib~     174 No        1.4 Flat~
## 5    64 Male   Typical~      110   211 Less~ Possib~     144 Yes       1.8 Flat~
## 6    64 Male   Typical~      170   227 Less~ Possib~     155 No        0.6 Flat~
## # ... with 3 more variables: ca <fct>, thal <fct>, condition <fct>
```

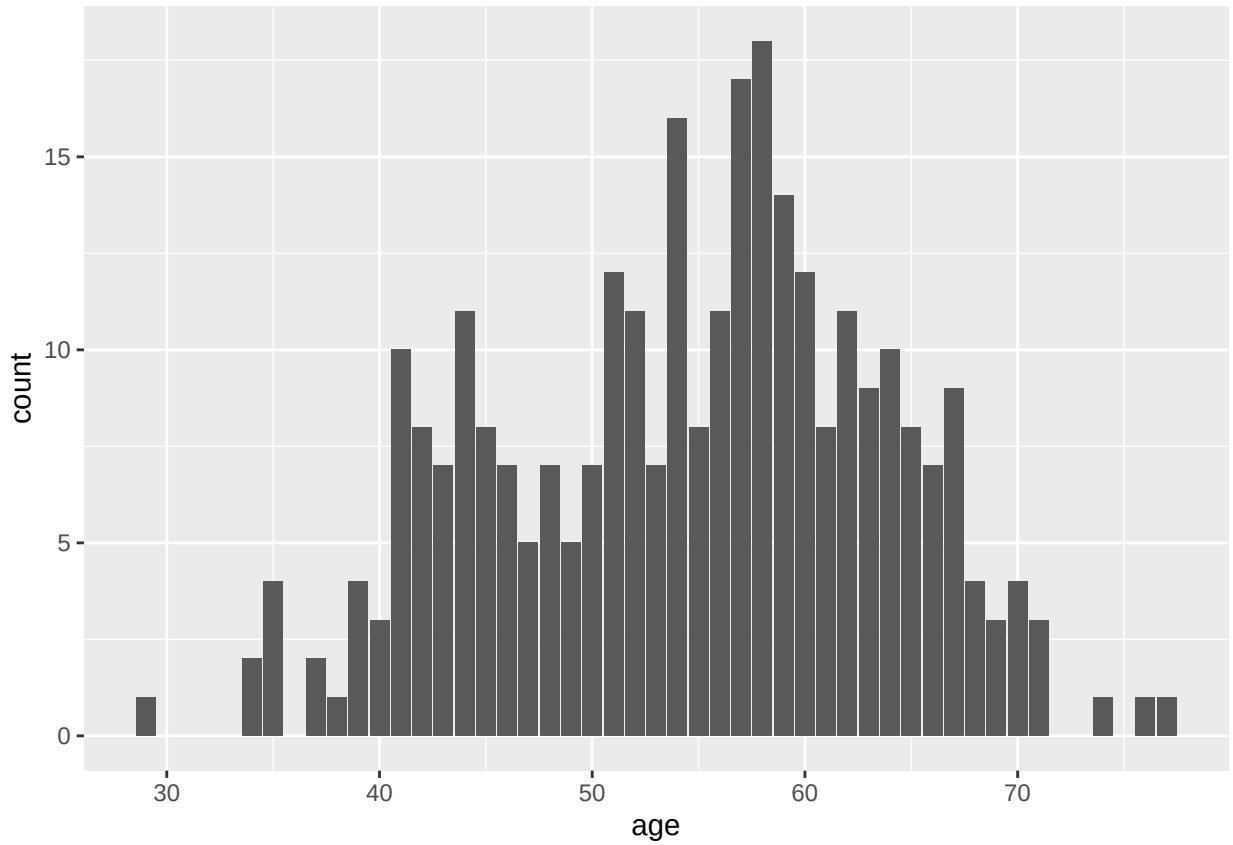
Data Analysis

This exploratory data analysis will be based only on the entire set, which has 297 observations with 14 variables. Each observation represents a single 'df' class, with age and thalach paired in the specific interest of seeing those two variables in potential correlation.

Variable age and thalach (max heart rate)

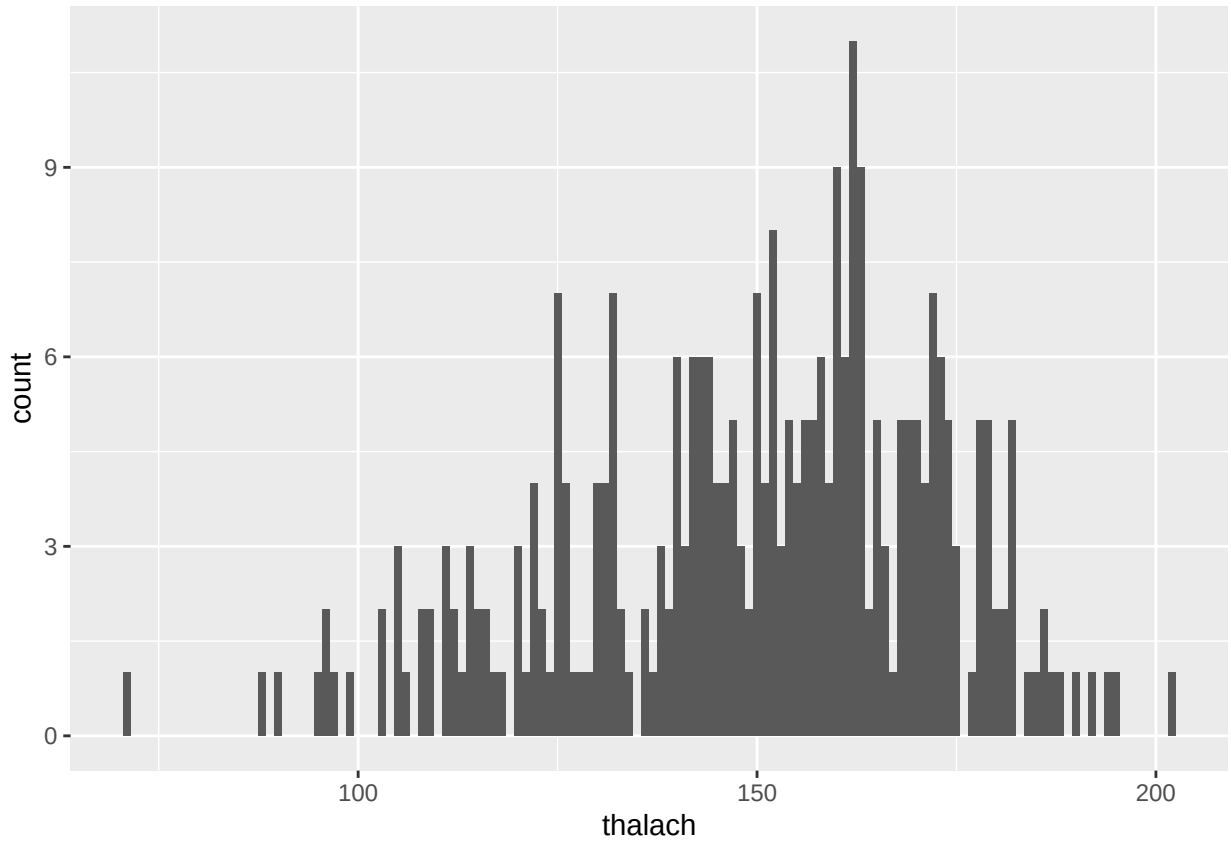
Drawing a plot of variable `age`:

```
df %>%
  ggplot(aes(x = age)) +
  geom_bar()
```



The graph is right skewed, with many respondents between 50-60 and mostly older than 40.

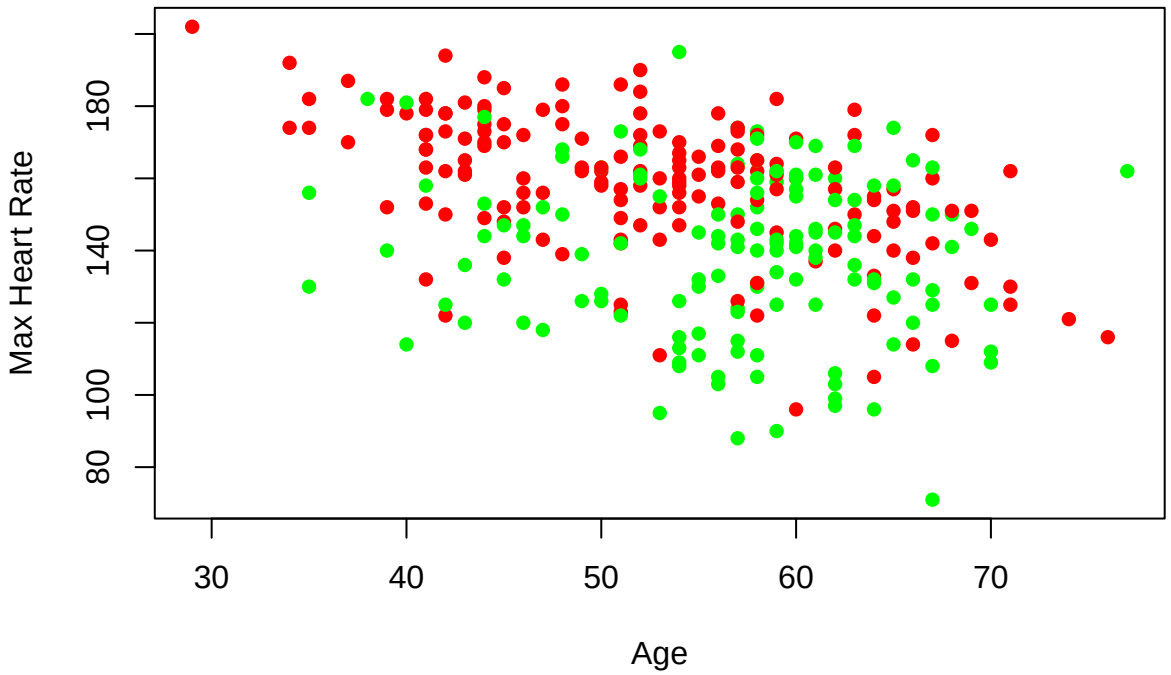
```
df %>%
  ggplot(aes(x = thalach)) +
  geom_bar()
```



The graph is right skewed, with many respondents having a max heart rate between 150 and 200.

Drawing a graph between age, max heart rate, and condition (green: no, red: yes):

```
plot(df$age, df$thalach, pch = 16, col = c('red', 'green')[df$condition], xlab = "Age", ylab = "Max Hea
```



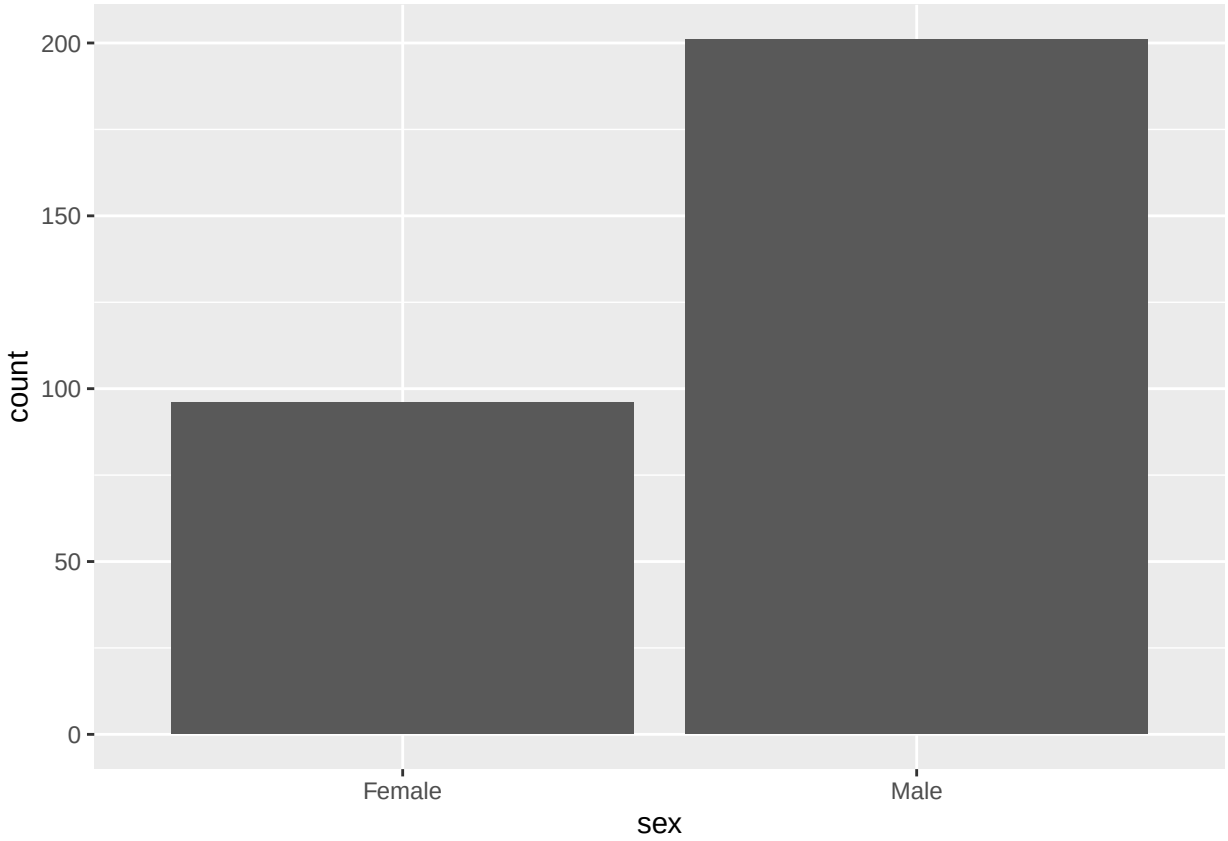
Heart condition seems to go down while age goes up (possibly due to a lack of quantity in respondents), heart rate

tends to go down with age, and the frequency of a heart condition seems to go up with maximum heart rate.

Variable sex

Drawing a plot of variable `sex`:

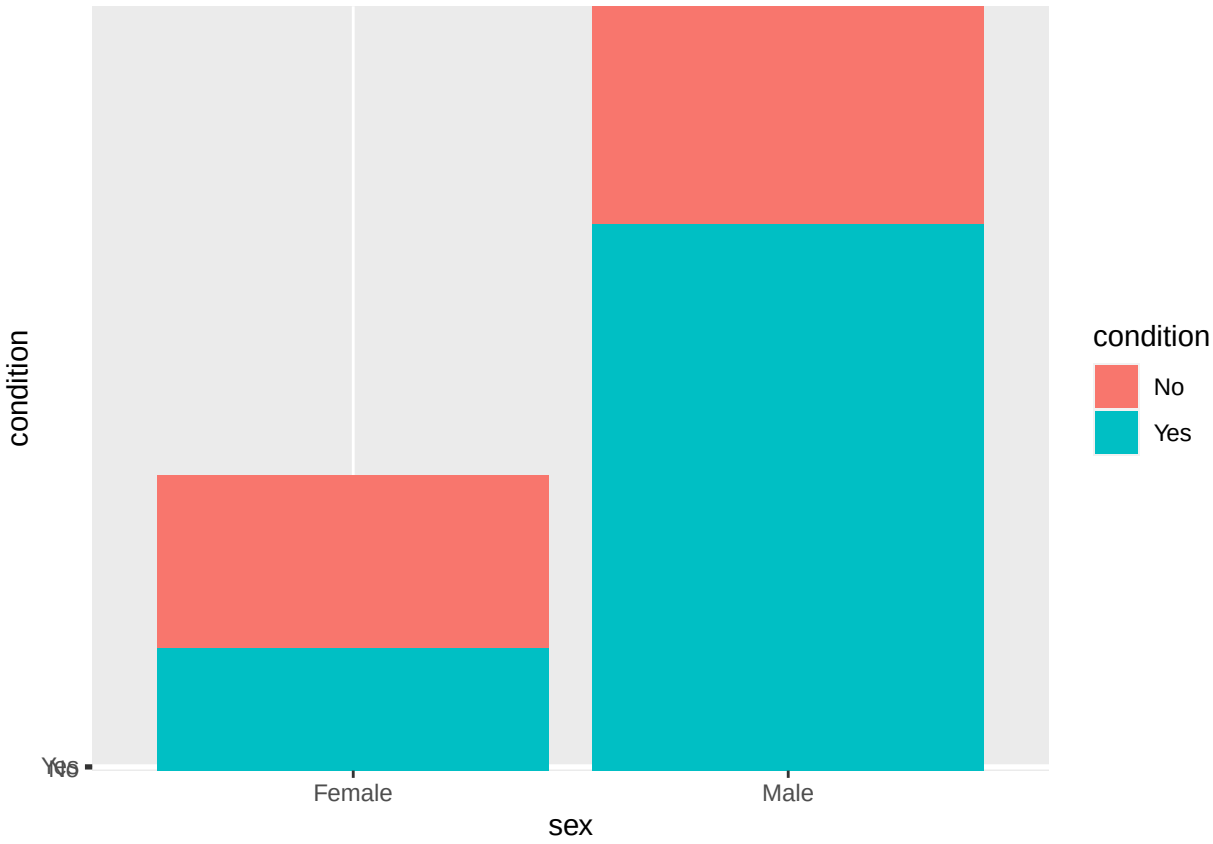
```
df %>%
  ggplot(aes(x = sex)) +
  geom_bar()
```



According to the graph, we find that there are more observations on 'Male' levels than 'Female' levels for variable `sex`.

Drawing a plot of variable `sex` by `condition`:

```
df %>%
  ggplot(aes(x = sex, y = condition, fill= condition)) +
  geom_bar(stat = "identity")
```

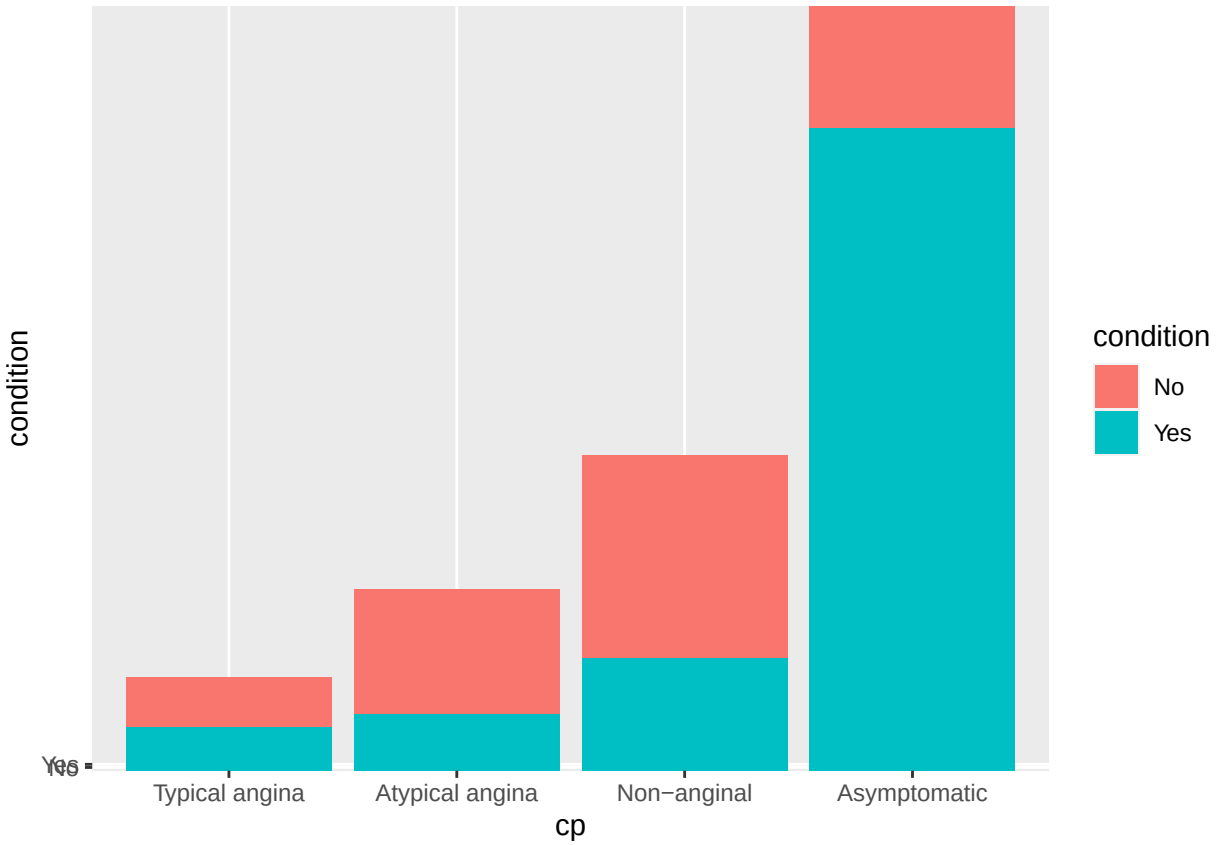


Based on the graph, we see that male individuals are more likely to get heart disease.

Variable cp (chest pain type)

Drawing a plot of variable `cp` vs `condition`:

```
df %>%
  ggplot(aes(x = cp, y = condition, fill = condition)) +
  geom_bar(stat = "identity")
```

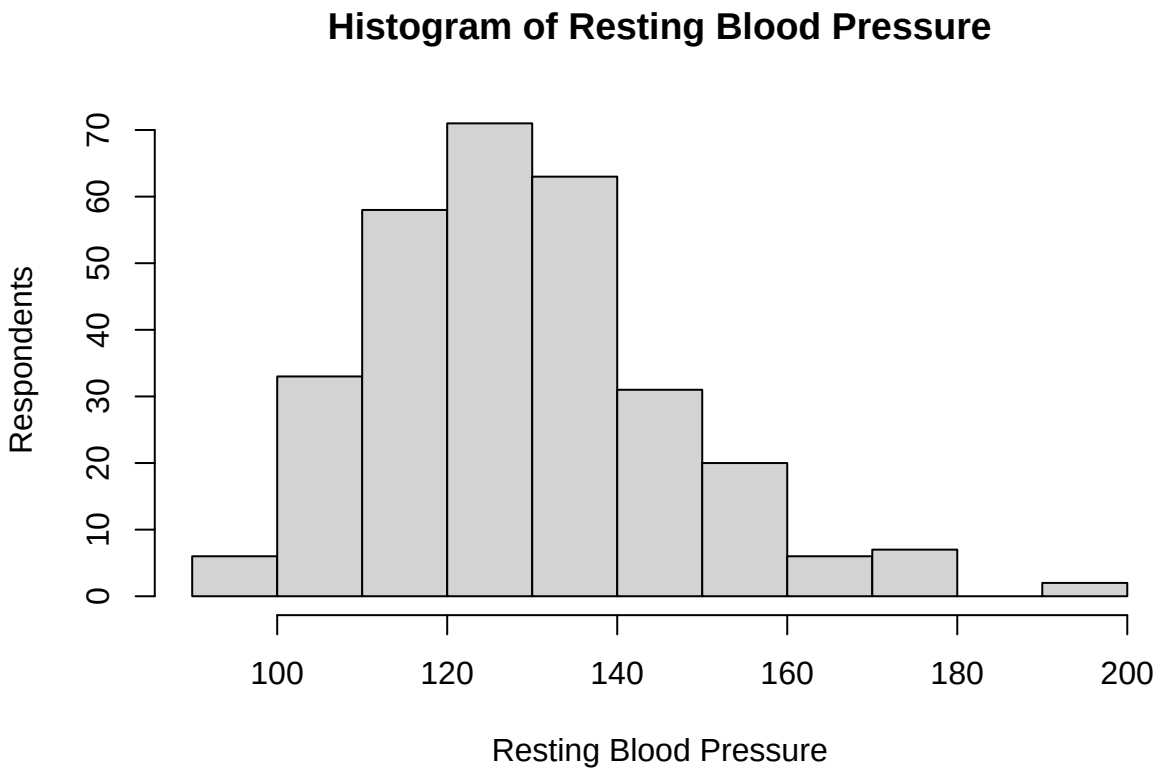


From this graph, we see that in an atypical angina or non-anginal chest pain type, there is least likely to have a heart condition. Typical seems split in between, and asymptomatic chest pain type has the most likely for a heart condition.

Variable trestbps

Drawing a histogram of variable `trestbps`:

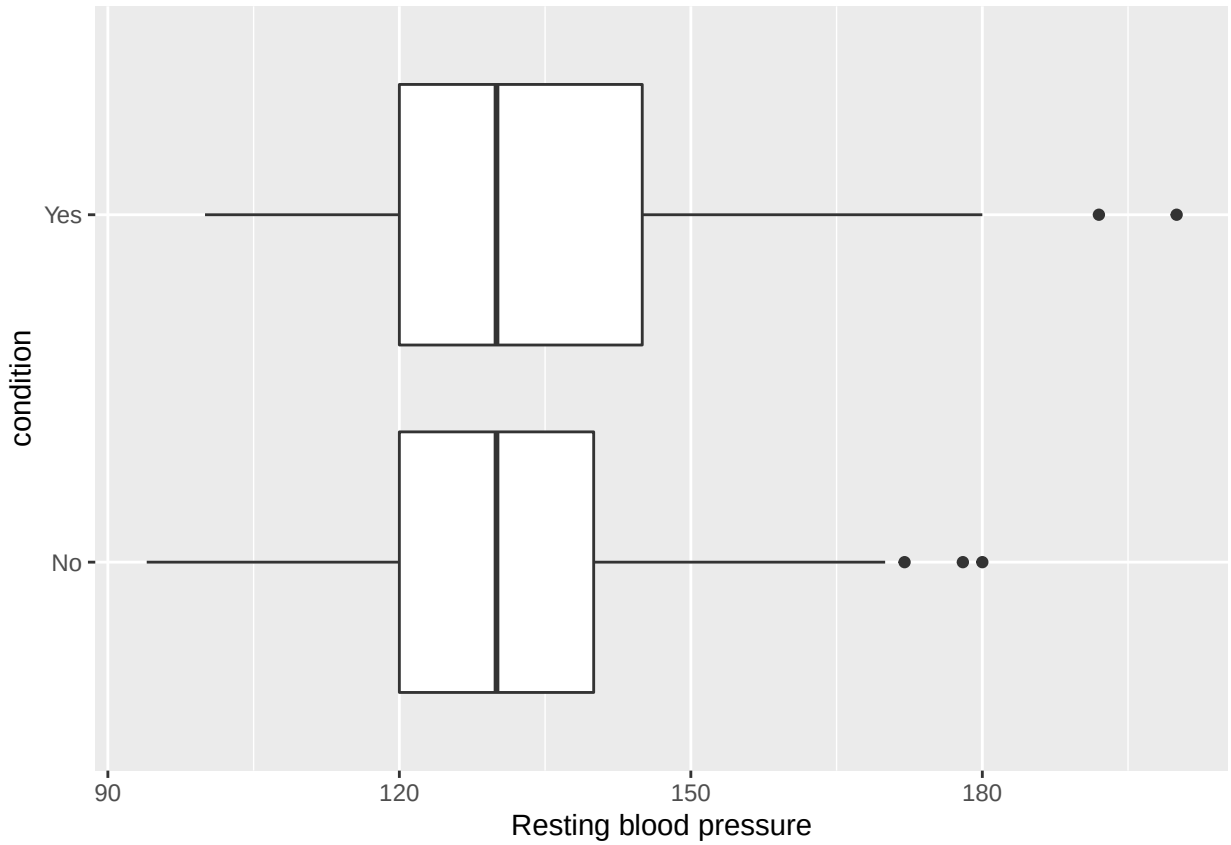
```
hist(df$trestbps, main = paste("Histogram of Resting Blood Pressure"), xlab = 'Resting Blood Pressure',
```



The distribution of `trestbps` is left skewed and has a long right tail. It looks closely like a normal distribution, with a peak around 120-130. Most people have a resting blood pressure below 160.

Drawing a boxplot of variable `trestbps` by `condition`:

```
df %>%
  ggplot(aes(x = trestbps, y= condition))+
  geom_boxplot() +
  xlab("Resting blood pressure")
```

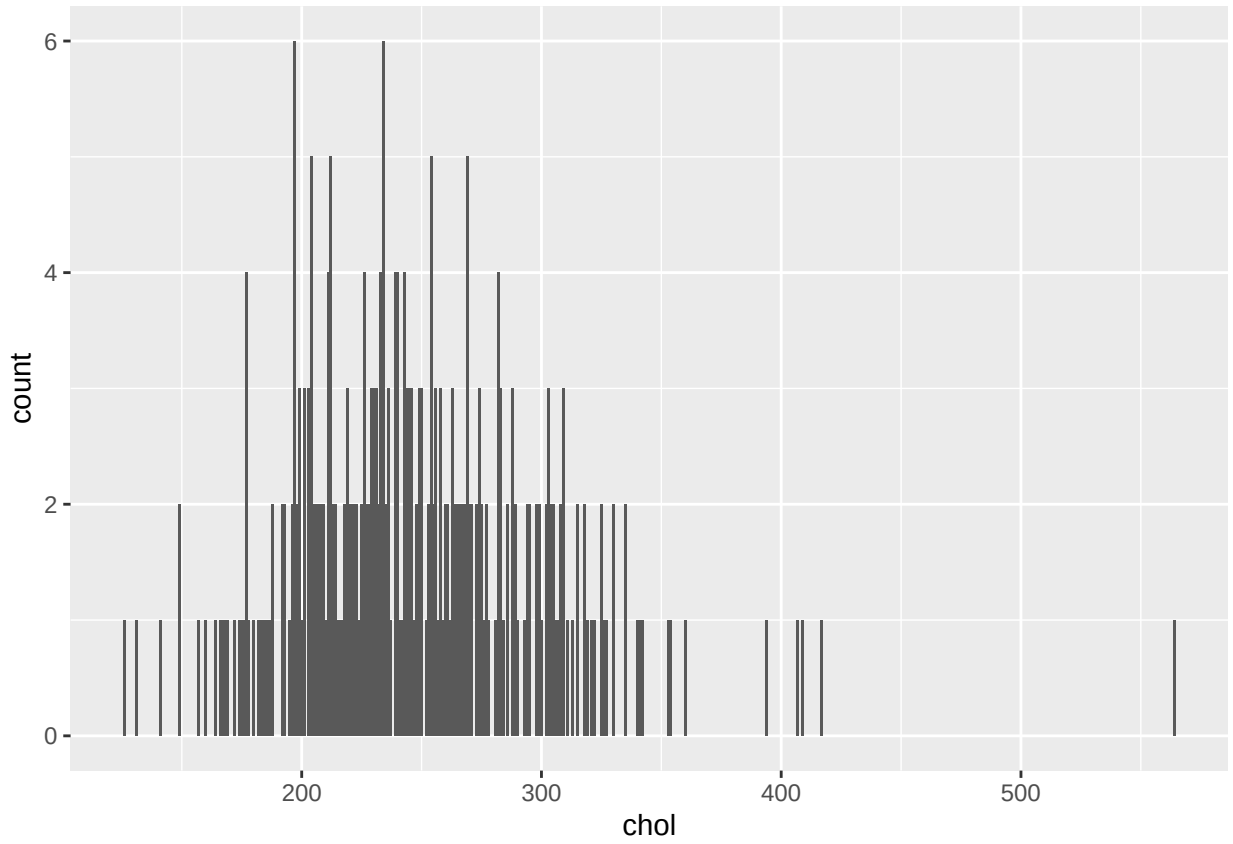


Based on the graph, we can find that individuals who have a higher resting blood pressure are more likely to get heart disease.

Variable chol (cholesterol)

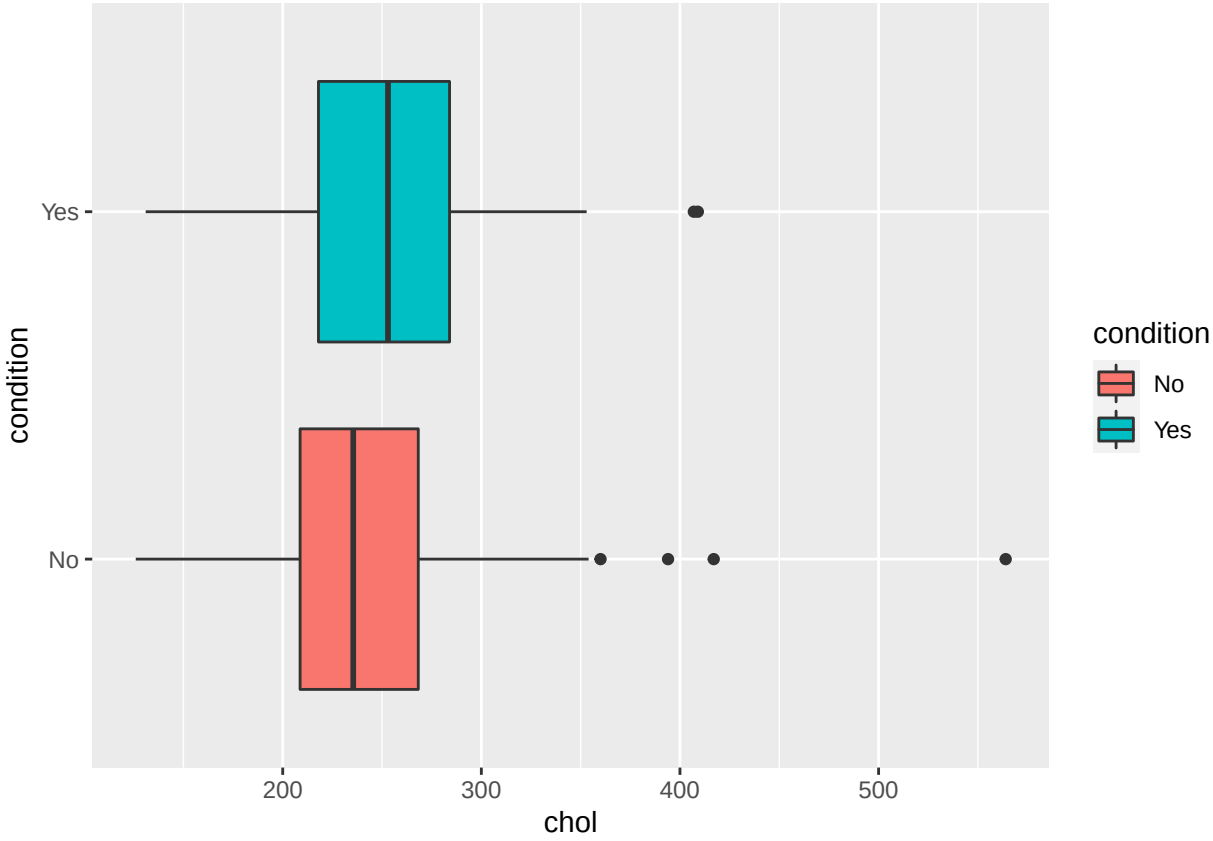
Drawing a plot of variable `chol`:

```
df %>%
  ggplot(aes(x = chol)) +
  geom_bar()
```



Like 'trestbps', the data is left skewed, with a shorter right tail. Most of the cholesterol levels fall between 200-300 mg/dl, which is above the 200 threshhold for "cause of concern".

```
df %>%
  ggplot(aes(x = chol, y = condition, fill = condition)) +
  geom_boxplot()
```

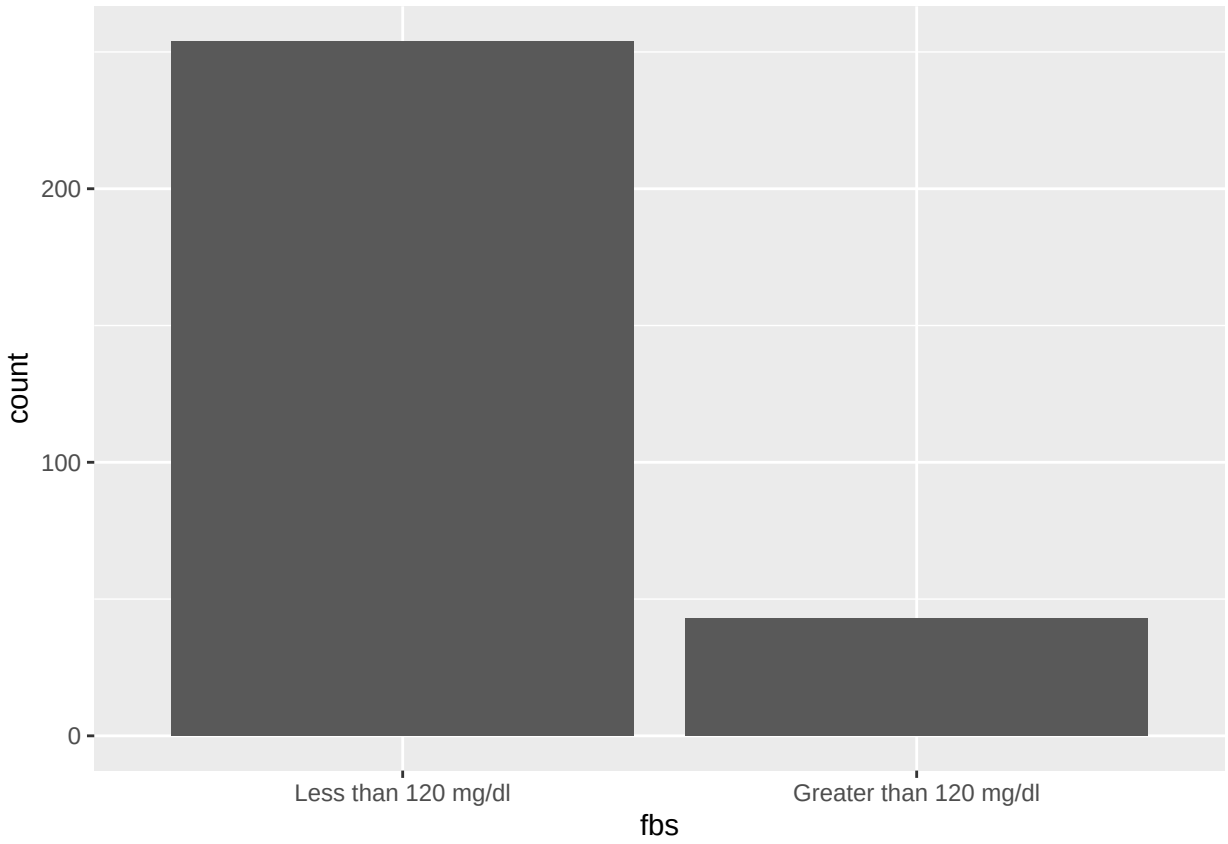


Higher levels of cholerestol seem to indicate more likelihood of having a heart condition.

Variable fbs (fasting blood sugar)

Drawing a plot of variable `fbs`:

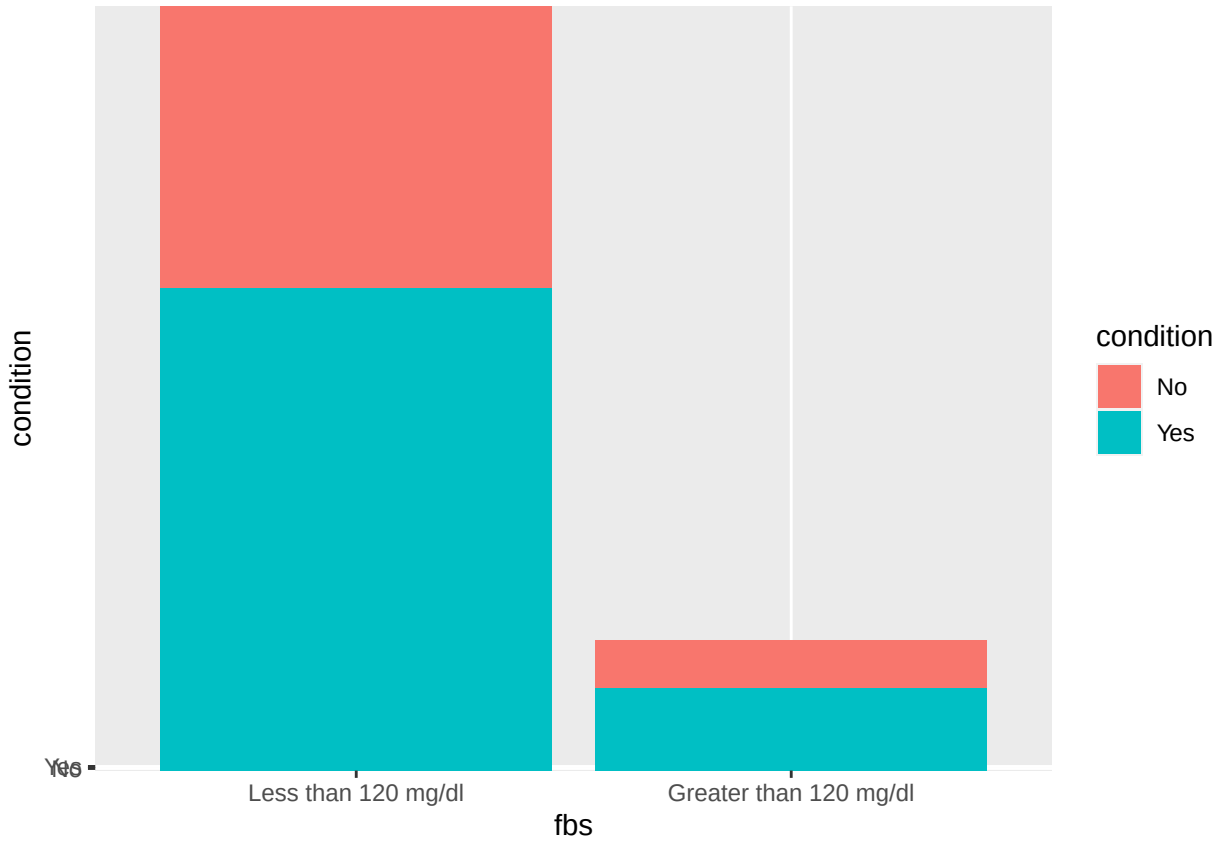
```
df %>%
  ggplot(aes(x = fbs)) +
  geom_bar()
```



The graph shows that there are far more individuals with less than 120 mg/dl fasting blood sugar than individuals with greater than 120 mg/dl. Unfortunately, as the cutoff for diabetes is greater than 126 mg/dl, and the data only shows whether they are above or below 120, it is hard to see the percentage of people who have diabetes, but rather the ratio shows us close (or not) to diabetes.

Drawing a plot of variable `fbs` by `condition`:

```
df %>%
  ggplot(aes( x= fbs, y = condition, fill = condition)) +
  geom_bar(stat="identity")
```

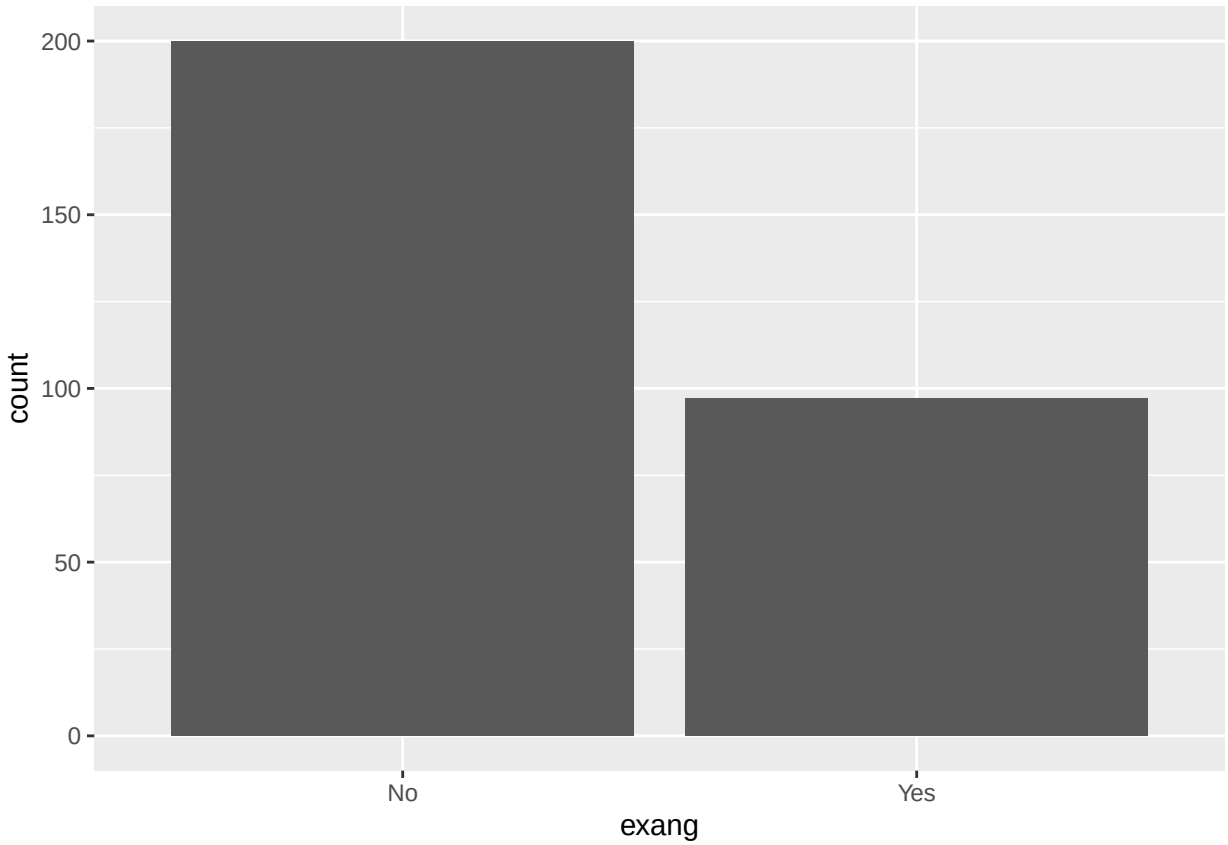


Based on the graph, an individual has just a slight increase in likelihood of heart condition if they are above 120 mg/dl fasting blood sugar, this may be attributed to the data being observed as a below or above threshhold with no indication of just how above/below a persons level may be.

Variable exang (excercise-induced angina)

Drawing plot of variable `exang`:

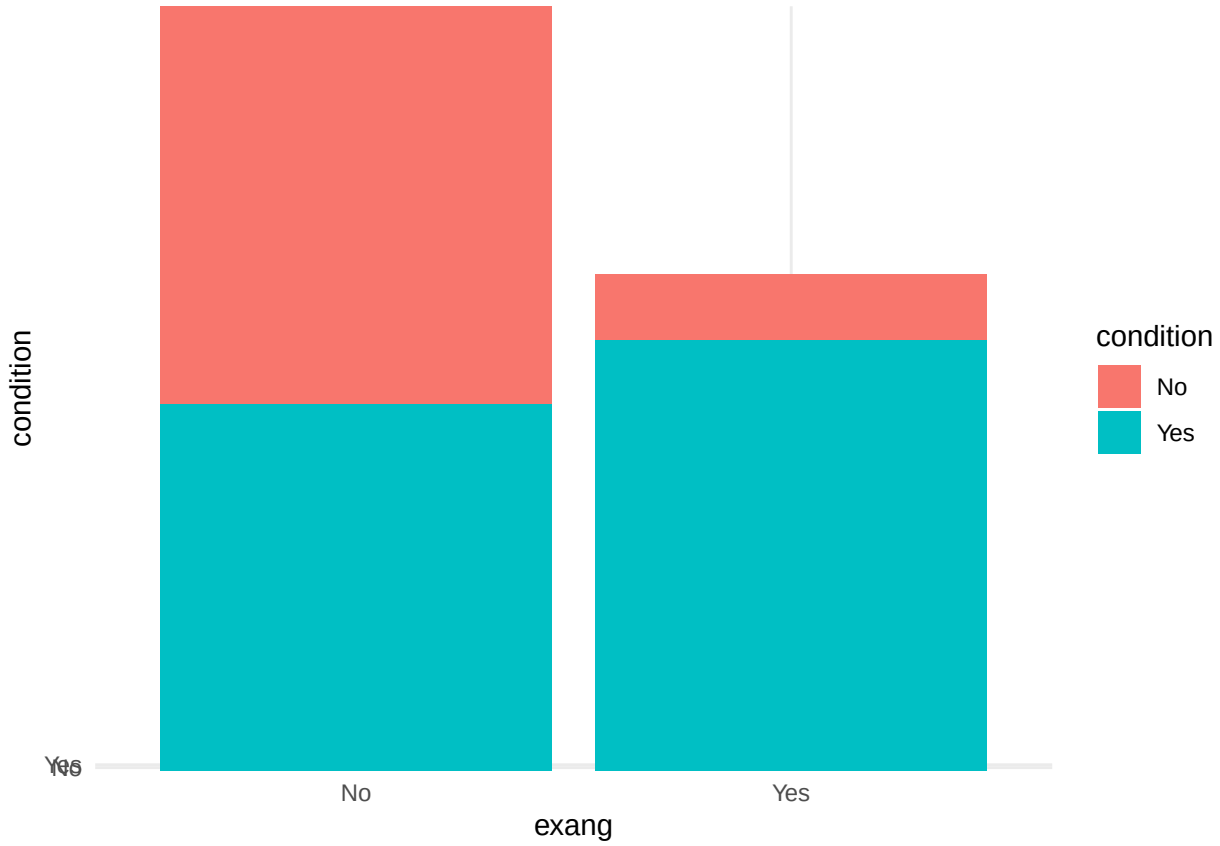
```
df %>%
  ggplot(aes(x = exang)) +
  geom_bar()
```



Around twice the people do not have exercise-induced angina compared to those who do.

Drawing a plot of variable `exang` by `condition`:

```
df %>%
  ggplot(aes(x= exang, y= condition , fill = condition)) +
  geom_bar(stat="identity")+theme_minimal()
```

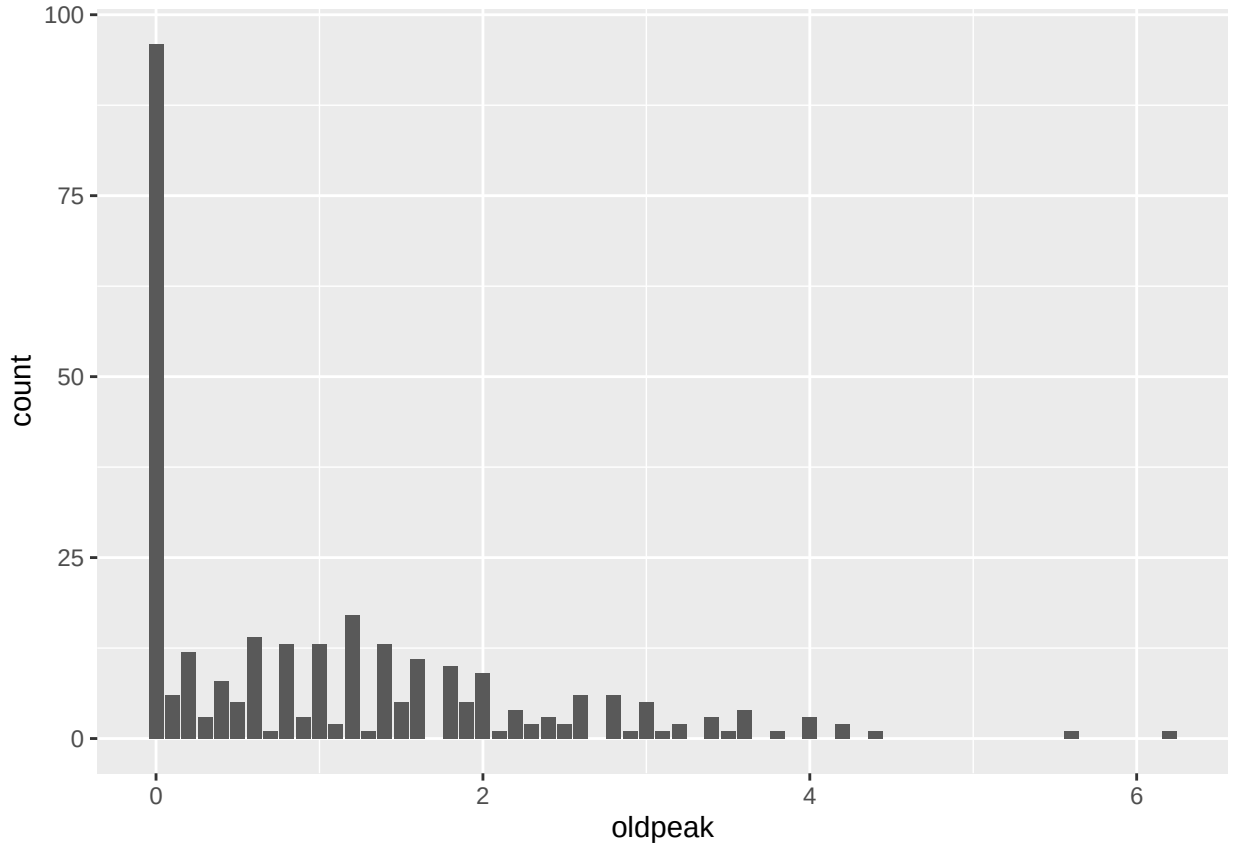


Based on the graph, we can find that the respondent who have exercise-induced angina are much more likely to get heart disease.

Variable oldpeak (ST depression induced by exercise relative to rest)

Drawing a plot of variable `oldpeak`.

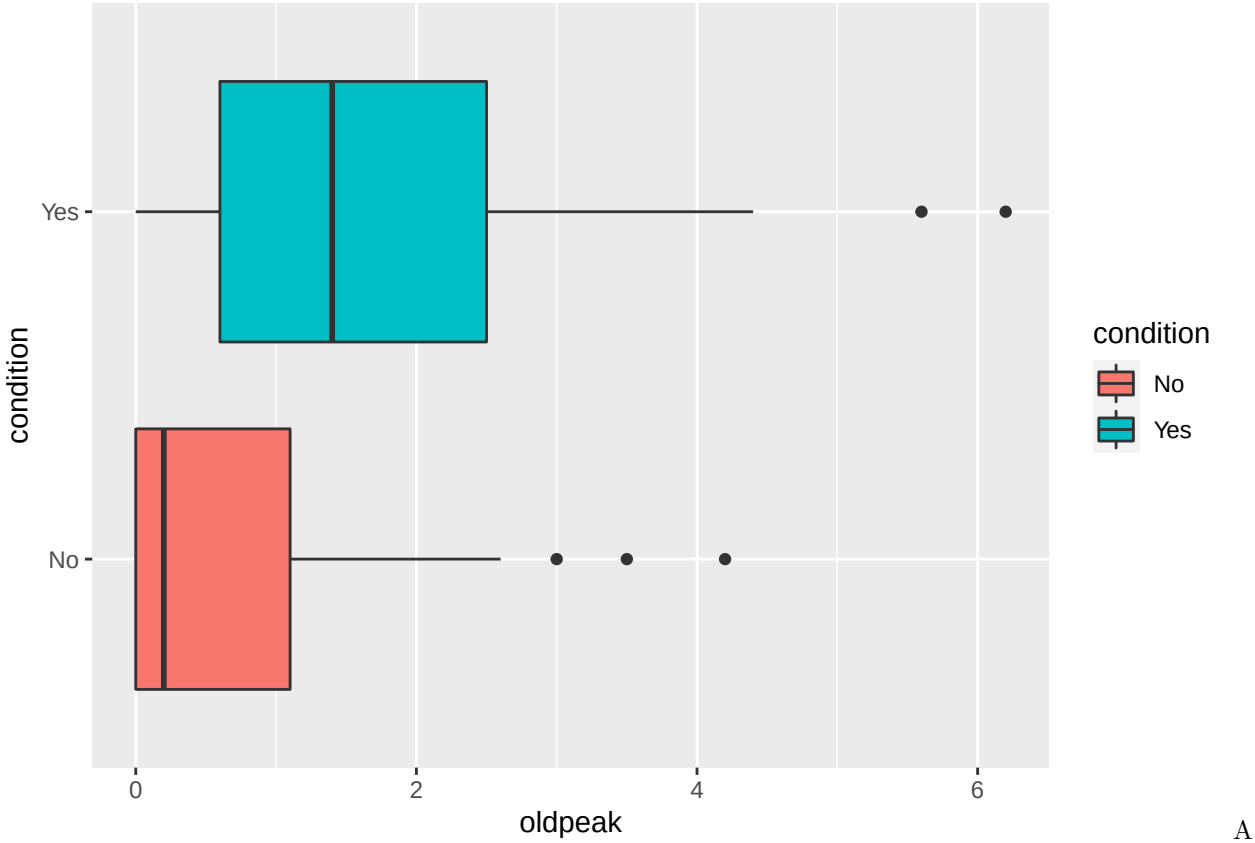
```
df %>%
  ggplot(aes(x = oldpeak)) +
  geom_bar()
```



There are many observations specifically at 0 (which indicates normal stressing during exercise), along with many observations spread between 0 and .4 and a few outliers at around .6.

Drawing a plot of variable `oldpeak` by `condition`:

```
df %>%
  ggplot(aes(x = oldpeak,y = condition, fill = condition)) +
  geom_boxplot()
```

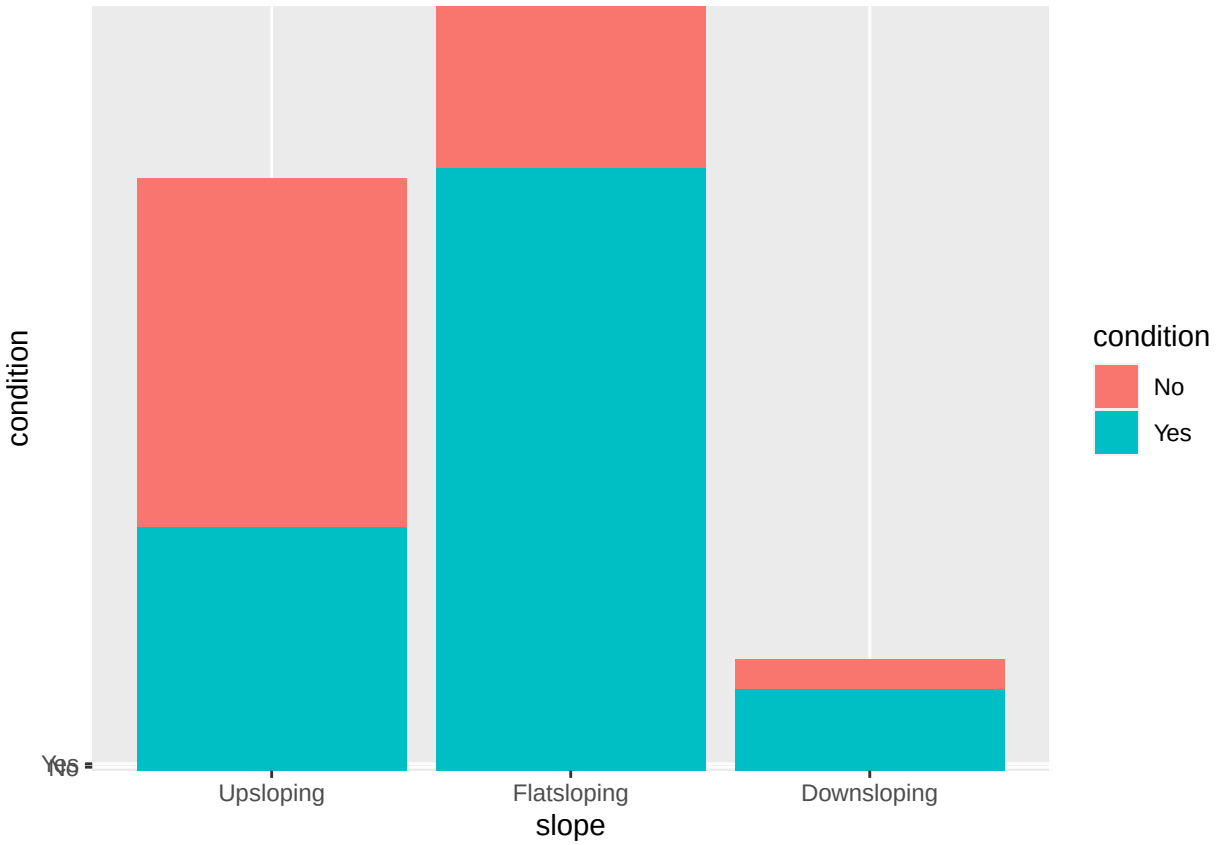


A boxplot shows that when an individuals heart stresses more than usual during exercise, there is higher indication of a heart condition.

Variable slope

Drawing a plot with variable 'slope' vs 'condition'.

```
df %>%
  ggplot(aes(x = slope, y = condition, fill = condition)) +
  geom_bar(stat = "identity")
```

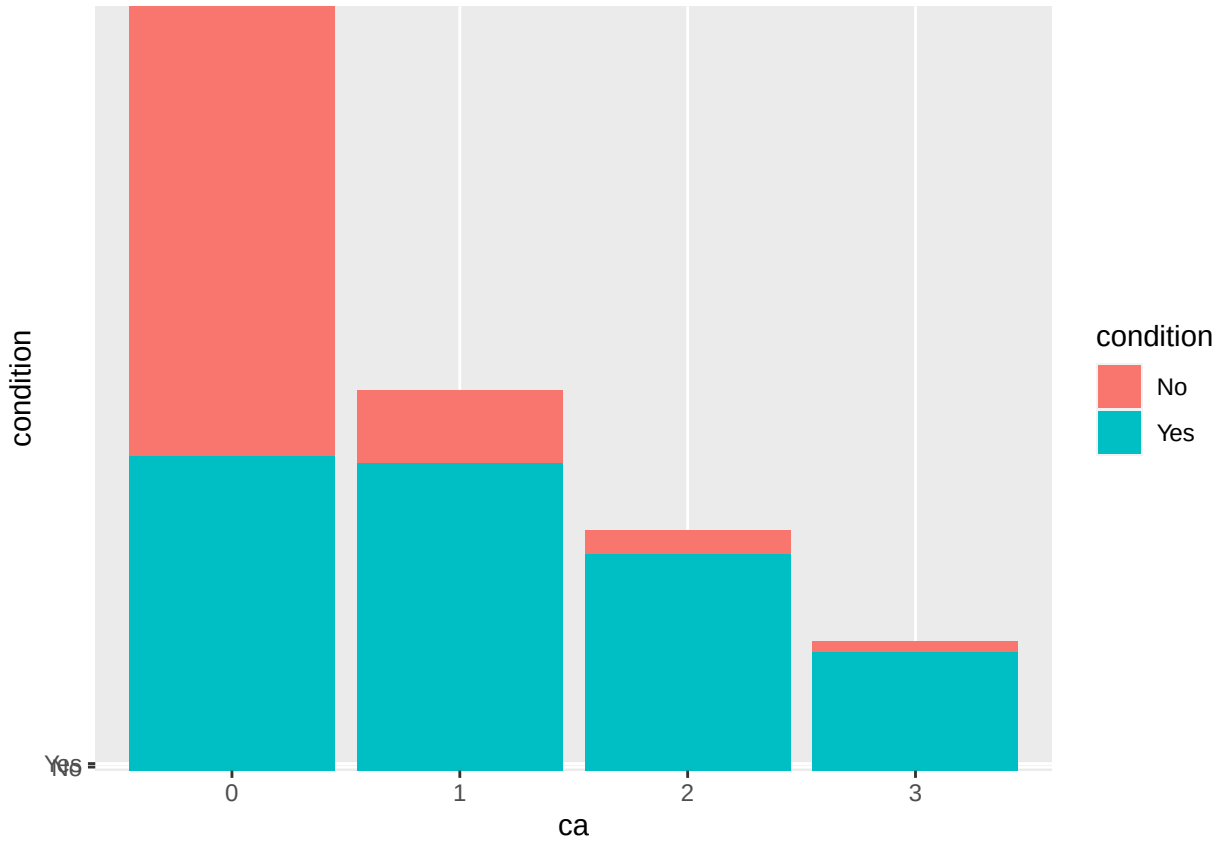


From this graph we see flatsloping (typical healthy heart) as the most common, with a better heart rate from exercise aboute as common, to the least common which is a typically unhealthy heart. Heart conditions are seen most in a typically unhealthy heart and a common heart, with less in a healthier heart.

Variable ca (number of major vessels (0-3) colored by flourosopy)

Drawing a plot with variable `ca` vs `condition`:

```
df %>%
  ggplot(aes(x = ca, y = condition, fill = condition)) +
  geom_bar(stat = "identity")
```

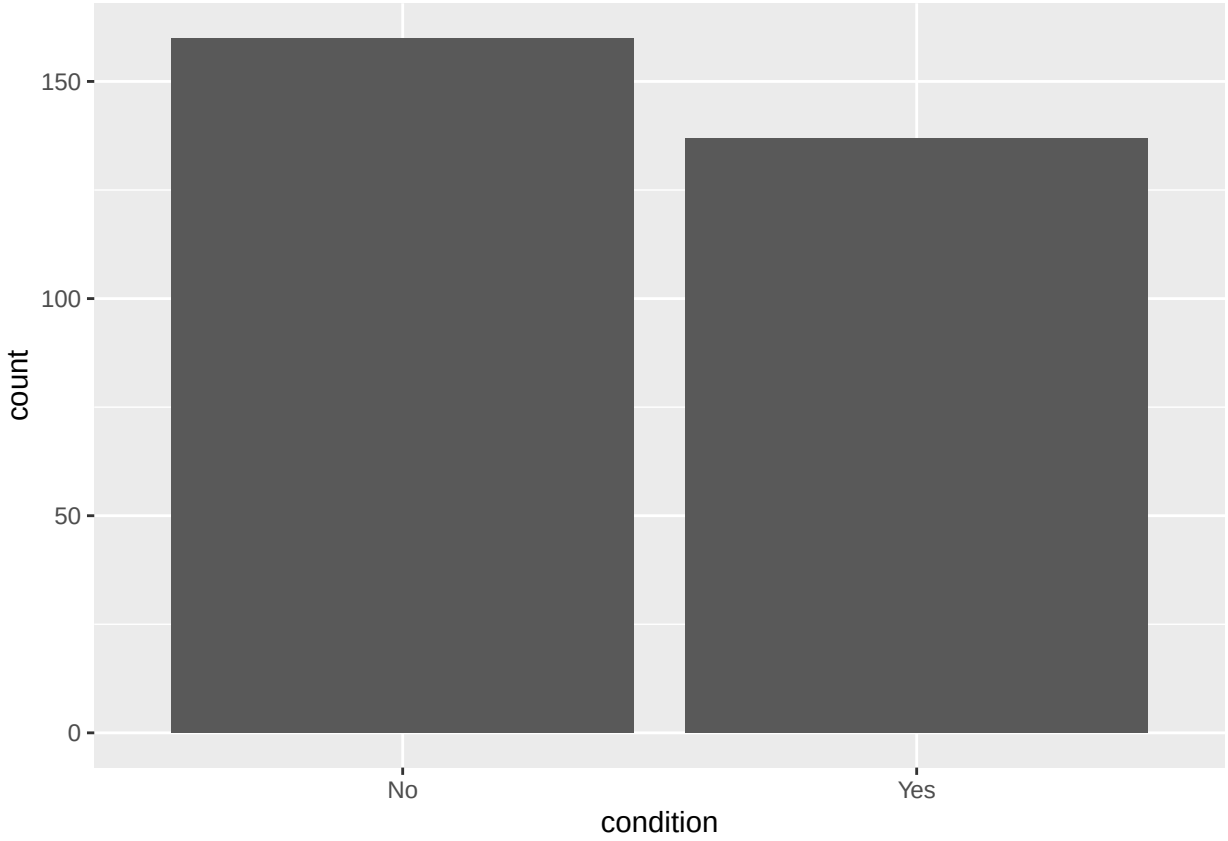


The more number of major vessels colored by flouroscopy, the more likely there is a heart condition. There is a huge jump from not having any major vessels to having a or multiple major vessel in terms of likelihood of a heart condition.

Variable condition (heart disease or not)

Drawing a plot with response variable `condition`:

```
df %>%
  ggplot(aes(x = condition)) +
  geom_bar()
```



From the graph, we find the response variable to be almost balanced. However, since there is a slight skew, when we split the data we will be using stratified sampling.

Data Split

80% of the data will be split for training, and the other 20% for testing.

```
df_split <- df %>%
  initial_split(prop = 0.8, strata = "condition")
df_train <- training(df_split)
df_test <- testing(df_split)
```

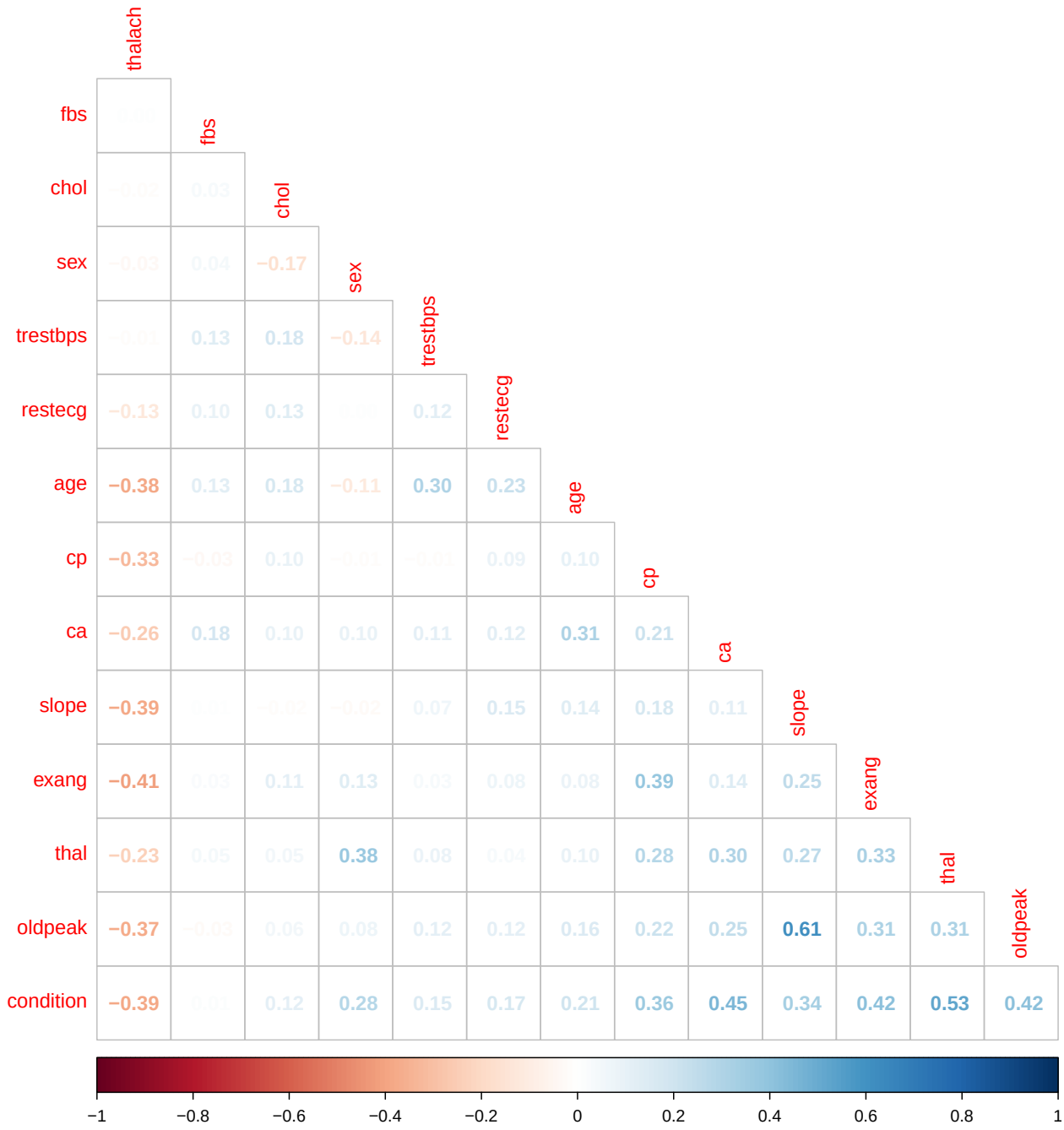
The testing data has 60 observations (just about 20% of the 300 total observations) and training data has the other 237 observations.

Correlation Plot

We create a correlation plot with each character variable turned into a numeric one.

```
df_train_factor <- df_train
df_train_factor[] <- lapply(df_train_factor, factor)
df_train_numeric <- df_train_factor
df_train_numeric[] <- as.data.frame(sapply(df_train_numeric, as.numeric))
```

```
df_train_numeric %>%
  select(where(is.numeric)) %>%
  cor(use = "complete.obs") %>%
  corrplot(method = "number", order = 'FPC', type = "lower", diag = FALSE, tl.cex=1)
```



Relatively strong correlations include slope and oldpeak, thal and condition, and ca and condition. Relatively strong negative correlations include thalach and: exang, condition, oldpeaek, age, and cp.

Model Building

Folding the training set using v-fold cross-validation, with `v = 5` and stratifying on our outcome variable `condition`:

```
df_folds <- vfold_cv(data = df_train, v = 5, strata = condition)
```

As we are working with a classification problem, we will be fitting and tuning these following models:

1. **Random Forest**
2. **Boosted Tree**
3. **Logistic Regression**
4. **K-Nearest Neighbors**

Building the Recipe

```
df_recipe <- recipe(condition ~ age + sex + cp + trestbps + chol + fbs + restecg + thalach + exang + old
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_predictors()) %>%
  step_center(all_predictors())
```

Random Forest

We first set up using a random forest model and workflow. Using the ranger engine and setting importance = "impurity".

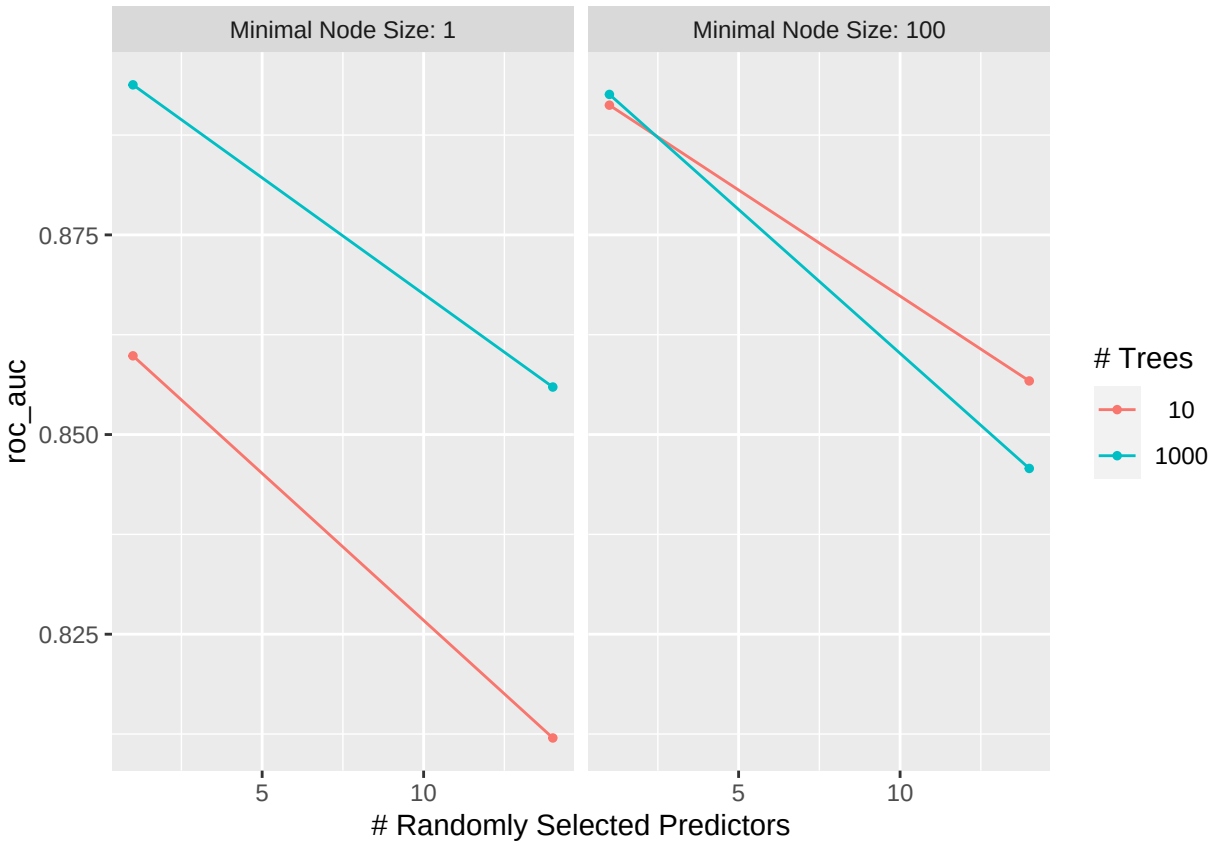
```
rf_spec <- rand_forest(mtry = tune(),trees = tune(), min_n = tune()) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("classification")
rf_wf <- workflow() %>%
  add_model(rf_spec) %>%
  add_recipe(df_recipe)
```

Tuning mtry, trees, and min_n.

```
param_grid_rf <- grid_regular(mtry(range = c(1, 14)),
                              trees(range = c(10, 1000)),
                              min_n(range = c(1, 100)),
                              levels = 2)
```

Tuning the model and print an autoplot() of the results.

```
tune_result <- tune_grid(
  rf_wf,
  resamples = df_folds,
  grid = param_grid_rf,
  metrics = metric_set(roc_auc)
)
autoplot(tune_result)
```



We see from plotting that a higher number of trees leads to a higher accuracy, as well as a lower number of randomly selected predictors and a higher minimal node size.

```
show_best(tune_result) %>% select(-.estimator, -.config)
```

```
## # A tibble: 5 x 7
##    mtry trees min_n .metric  mean     n std_err
##   <int> <int> <int> <chr>   <dbl> <int>   <dbl>
## 1     1  1000     1 roc_auc 0.894     5  0.0215
## 2     1  1000   100 roc_auc 0.893     5  0.0190
## 3     1    10   100 roc_auc 0.891     5  0.0296
## 4     1    10     1 roc_auc 0.860     5  0.0256
## 5    14    10   100 roc_auc 0.857     5  0.0180
```

Using the `show_best()` function, we find the highest mean is .8935. This is with the smallest amount of `mtry` at 1, max amount of `trees` at 1000, and a max amount of `min_n` at 100. This alligns with our previous discovery on the graph.

Boosted trees

Now, we are going to set up a boosted tree model and workflow. Use the xgboost engine. We will tune mtry, trees, and min_n. Using the documentation for boost_tree().

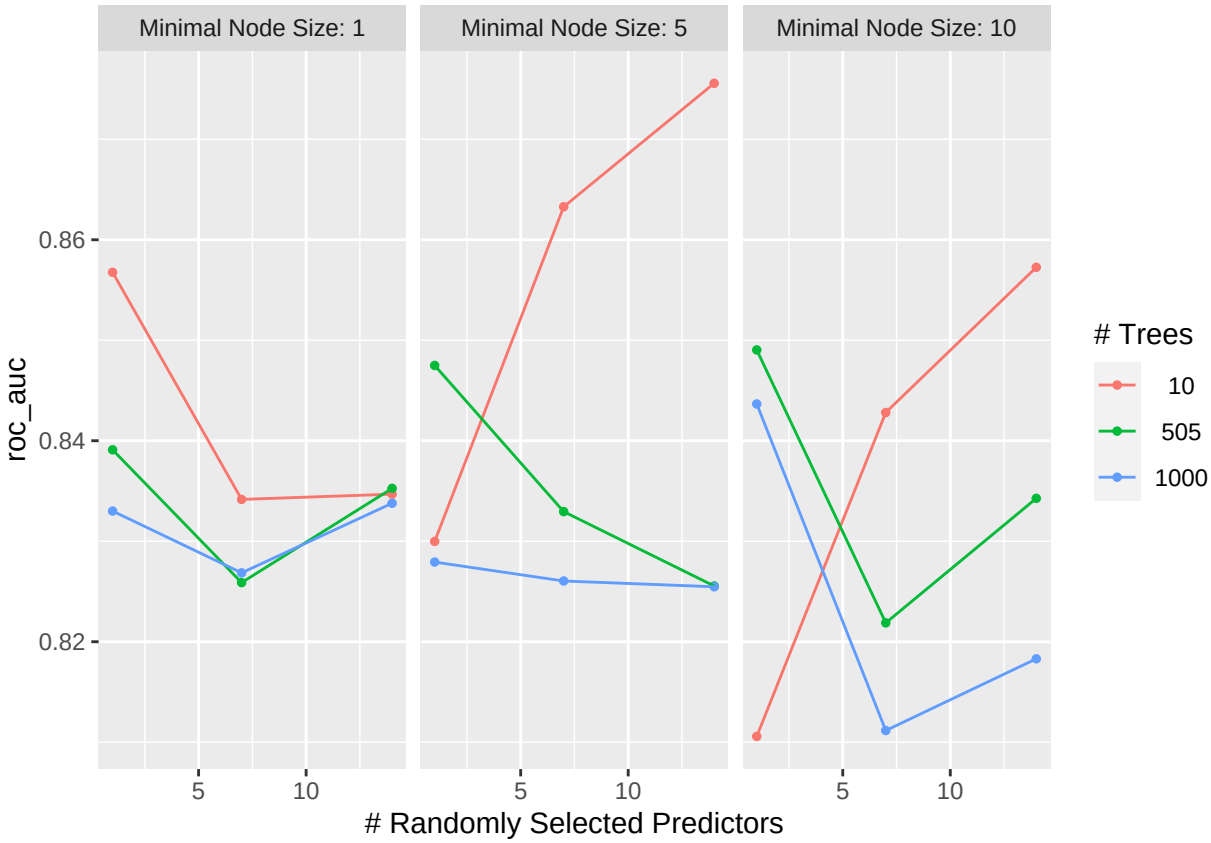
```
boost_tree_spec <- boost_tree(trees = tune(),
                              min_n = tune(),
                              mtry = tune()
```

```
                              ) %>%
  set_engine("xgboost") %>%
  set_mode("classification")
boost_tree_workflow <- workflow() %>%
  add_model(boost_tree_spec) %>%
  add_recipe(df_recipe)
```

```
boost_tree_grid<- grid_regular(mtry(range = c(1, 14)),
                               trees(range = c(10, 1000)),
                               min_n(range = c(1, 10)),
                               levels = 3)
```

Tuning model and print an autoplot() of the results.

```
boost_tune_result <- tune_grid(
  boost_tree_workflow,
  resamples = df_folds,
  grid = boost_tree_grid,
  metrics = metric_set(roc_auc),
)
autoplot(boost_tune_result)
```



We see from plotting that a lower number of trees leads to a highest accuracy, as well as a medium number of randomly selected predictors for the lowest amount of trees (higher number of random selected predictors for higher amount of trees) and a higher minimal node size. I tried keeping the minimal node size consistent for the different models but had to tune each one specifically to show data visually best.

```
show_best(boost_tune_result) %>% select(-.estimator, -.config)
```

```
## # A tibble: 5 x 7
##    mtry trees min_n .metric  mean     n std_err
##   <int> <int> <int> <chr>   <dbl> <int>   <dbl>
## 1    14    10     5 roc_auc 0.876     5  0.0278
## 2     7    10     5 roc_auc 0.863     5  0.0270
## 3    14    10    10 roc_auc 0.857     5  0.0333
## 4     1    10     1 roc_auc 0.857     5  0.0337
## 5     1   505    10 roc_auc 0.849     5  0.0242
```

Using the `show_best()` function, we find the highest mean is .8605. This is with 'mtry' amount of 7 (right in the middle), minimal amount of `trees` at 10, and a `min_n` amount at 5 (also in the middle). This alligns with our previous discovery on the graph.

Logistic Regression

To create a logistics regression model, `logistic_reg()` and the `glm` engine will be used. `fit()` will be used fit the model to the folded data.

```
log_reg <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")
log_wkflow <- workflow() %>%
  add_model(log_reg) %>%
  add_recipe(df_recipe)
```

```
log_fit <- fit_resamples(log_wkflow, df_folds)
```

We use `collect_metrics()` to print the mean and standard errors of the performance metric accuracy across all folds.

```
collect_metrics(log_fit)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.789     5  0.0200 Preprocessor1_Model1
## 2 roc_auc  binary     0.859     5  0.0187 Preprocessor1_Model1
```

Logistic regression returns a roc_auc of .8845.

K Nearest Neighbors

To set up a K-Nearest Neighbor Model and workflow, I will use `nearest_neighbor()` and the `kknn` engine, and tune `neighbors` and setting the mode to `classification`.

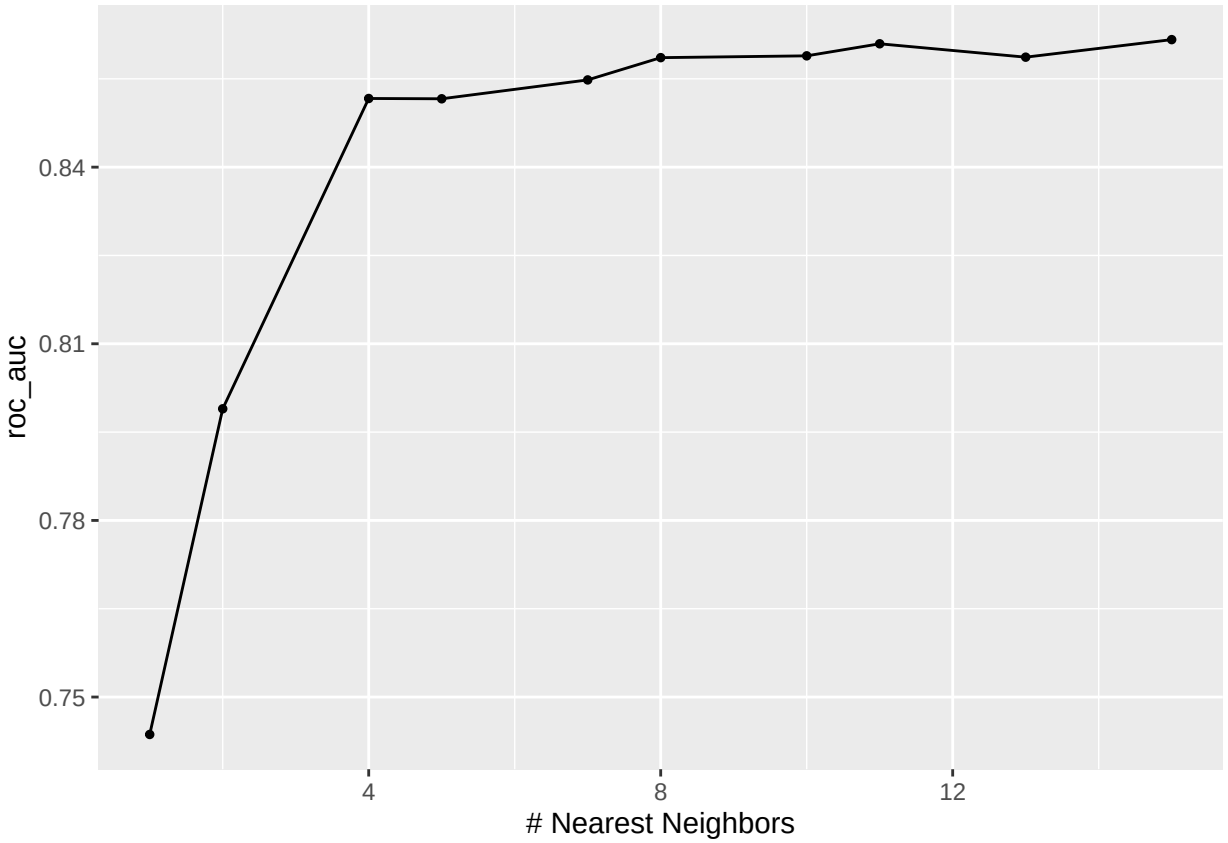
```
knn_model <- nearest_neighbor(neighbors = tune(), mode = "classification") %>%
  set_engine("kknn")

knn_workflow <- workflow() %>%
  add_model(knn_model) %>%
  add_recipe(df_recipe)
```

```
knn_params <- parameters(knn_model)
```

```
## Warning: ‘parameters.model_spec()‘ was deprecated in tune 0.1.6.9003.
## Please use ‘hardhat::extract_parameter_set_dials()‘ instead.
```

```
knn_grid <- grid_regular(knn_params, levels = 10)
```

```
knn_tune <- tune_grid(knn_workflow, resamples = df_folds, grid = knn_grid, metrics = metric_set(roc_auc
autoplot(knn_tune)
```



From the plot we see the roc_auc goes up with higher numbers of nearest neighbors.

```
arrange(collect_metrics(knn_tune),desc(mean))
```

```
## # A tibble: 10 x 7
##    neighbors .metric .estimator  mean     n std_err .config
##        <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1        15 roc_auc binary     0.862     5  0.0153 Preprocessor1_Model10
```

```
##  2        11 roc_auc binary     0.861     5  0.0176 Preprocessor1_Model08
##  3        10 roc_auc binary     0.859     5  0.0185 Preprocessor1_Model07
##  4        13 roc_auc binary     0.859     5  0.0176 Preprocessor1_Model09
##  5         8 roc_auc binary     0.859     5  0.0173 Preprocessor1_Model06
##  6         7 roc_auc binary     0.855     5  0.0179 Preprocessor1_Model05
##  7         4 roc_auc binary     0.852     5  0.0183 Preprocessor1_Model03
##  8         5 roc_auc binary     0.852     5  0.0168 Preprocessor1_Model04
##  9         2 roc_auc binary     0.799     5  0.0172 Preprocessor1_Model02
## 10         1 roc_auc binary     0.744     5  0.0175 Preprocessor1_Model01
```

Using collect_metric() and arrange(), the highest mean accuracy of KNN models amounts to .8593 with 15 neighbors.

Final model

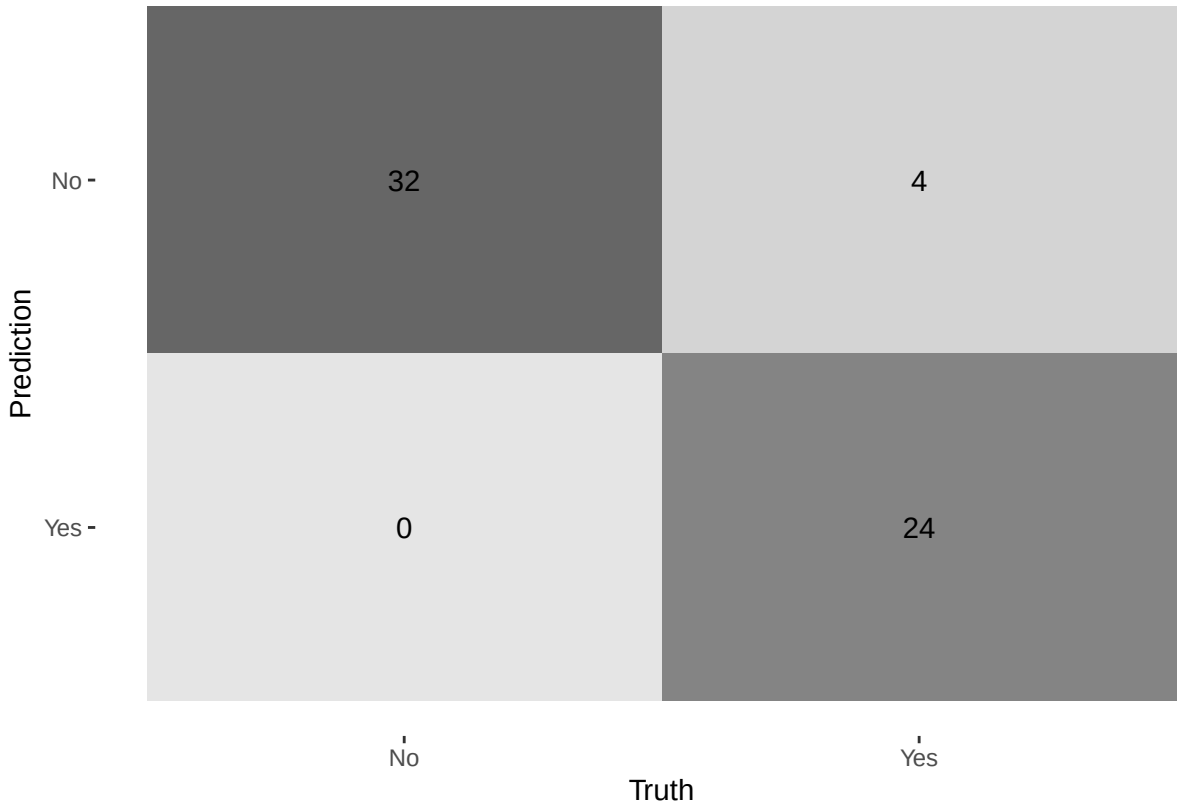
Our best performing model so far is the random forest model. We will now evaluate its performance on the testing set with finalize_workflow().

```
best <- select_best(tune_result, metric= 'roc_auc')
forest_final <- finalize_workflow(rf_wf, best)
forest_final_fit <- fit(forest_final, data = df_train)
augment(forest_final_fit, new_data = df_test) %>%
  accuracy(truth = condition, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.933
```

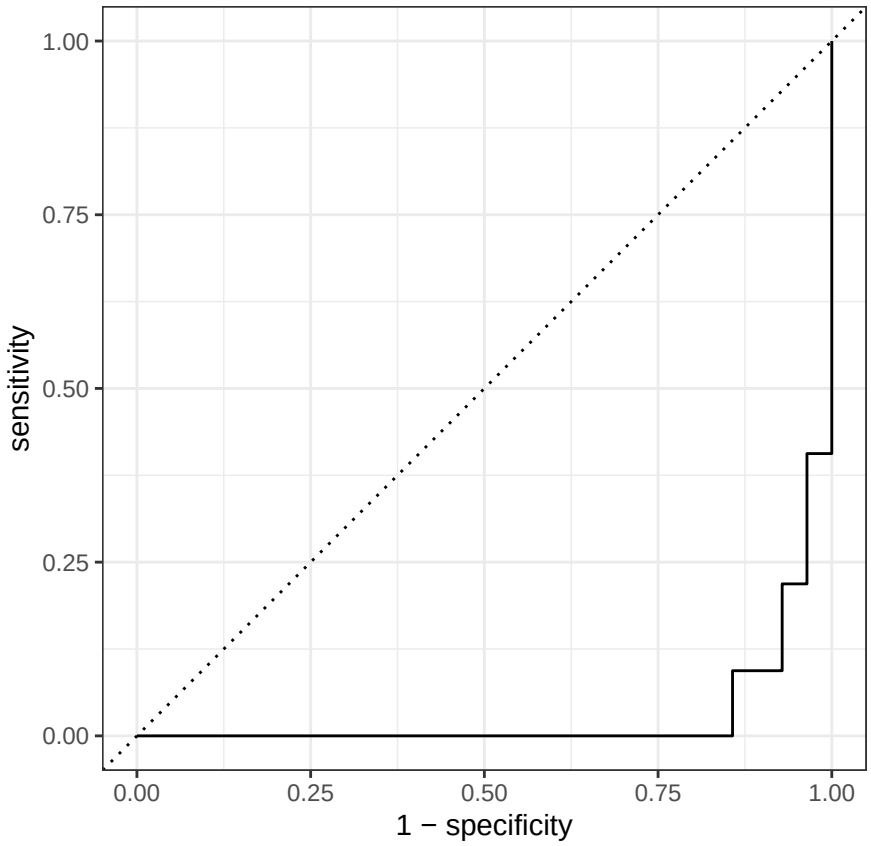
Accuracy of .85 is returned. Now we make a heatmap on the predictions on the testing set.

```
augment(forest_final_fit, new_data = df_test) %>%
  conf_mat(truth = condition, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```



We visualize by making a ROC curve.

```
augment(forest_final_fit, new_data = df_test) %>%
  roc_curve(condition, .pred_Yes) %>%
  autoplot()
```



Calculating AUC of random forest on the testing set; .039 is returned.

```
augment(forest_final_fit, new_data = df_test) %>%
  roc_auc(condition, .pred_Yes)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary        0.0290
```

Now, we get to use our model to predict if an individual has a heart condition or not.

```
test1 <- data.frame(
  age = 80,
  sex = "Male",
  cp = "Typical angina",
  trestbps = 188,
  chol = 288,
  fbs = "Greater than 120 mg/dl",
  restecg = "Nothing to note",
  thalach = 171,
  exang = "Yes",
  oldpeak = 3,
  slope = "Downsloping",
  ca = "2",
  thal = "Normal")
```

```
predict(forest_final_fit, test1)
```

```
## # A tibble: 1 x 1
##   .pred_class
##   <fct>
## 1 Yes
```

```
test2 <- data.frame(
  age = 50,
  sex = "Female",
  cp = "Atypical angina",
  trestbps = 138,
  chol = 138,
  fbs = "Less than 120 mg/dl",
  restecg = "ST-T wave abnormality",
  thalach = 121,
  exang = "No",
  oldpeak = .5,
  slope = "Flatsloping",
  ca = "0",
  thal = "Normal")

predict(forest_final_fit, test2)
```

```
## # A tibble: 1 x 1
##   .pred_class
##   <fct>
## 1 No
```

```
test3 <- data.frame(
  age = 30,
  sex = "Male",
  cp = "Asymptomatic",
  trestbps = 103,
  chol = 108,
  fbs = "Less than 120 mg/dl",
  restecg = "Nothing to note",
  thalach = 101,
  exang = "No",
  oldpeak = 0,
  slope = "Upsloping",
  ca = "3",
  thal = "Fixed defect")

predict(forest_final_fit, test1)
```

```
## # A tibble: 1 x 1
##   .pred_class
##   <fct>
## 1 Yes
```

Conclusion

Overall, the accuracies across all models were high (above .8), with the Random Forest model performing best on the training model with a mean of .8935 and .85 on the testing model. The predictions are satisfactory and reflect well on the EDA - factors such as ca (number of major vessels historically checked by fluoroscopy), slope (heart status in response to exercise), and exang (exercise-induced angina), all of which pertain directly to the heart, logically are the biggest influences in whether an individual has a heart condition or not.

In a second do-over I may have included LDA and QDA forms to the logistic regression models, since the accuracies of the logistic model was the second highest. I would have also chosen a data set with more observations, as the conveniently formatted data set from Kaggle truncated some of the total information from the UCI original source. Lastly, I think I should have tested the ROC-AUC of all the models on the testing set as well to compare their accuracies (I determined the best model solely on the training data set here). All in all, I am satisfied since the models have high accuracy compared to previous models done in the labs and homework assignments. I built a model that I can test on my step-dad since he is always worried about his blood pressure and cholesterol levels.