# Capstone_RMD

## Section 1: Introduction

I analyzed the Heart Attack Analysis & Prediction Dataset found on Kaggle at https://www.kaggle.com/rashikrahmanpritom/heart-attack-analysis-prediction-dataset I have transferred this file to a repository on GitHub and saved it as a CSV file.

The goal of this project is to perform machine learning tasks (train and validate a model on the data). There should be two algorithms used, and one must be more complex than regression.

The Heart Attack dataset looks at thirteen variables, including Age, Sex, Cholesterol, and has a binary output variable for whether a heart attack occured. The predictor variables are initially all numeric, but are ultimately either numeric or factors (ie: cholestorol is numeric while sex is a factor). Note that throughout the project or script, I will be using the terms "heart attack" and "heart disease" interchangably.

I performed the following steps: 1. Look at high level summary stats on the dataset 2. Clean the dataset as necessary to train a machine learning algorithm 3. Split the data into training and test sets 4. Train and test a multi-variate logistic regression model 5. Train and test a KNN algorithm 6. Train and test a random forest model 7. State my results and conclusion

I begin by installing and loading required packages and the dataset:

```r
if(!require(readr)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: readr
```

```r
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Loading required package: lattice

## Loading required package: ggplot2
```

```r
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: dplyr

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: corrplot

## corrplot 0.90 loaded
```

```r
if(!require(gridExtra)) install.packages("gridExtra", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: gridExtra

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse

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

## v tibble  3.1.2     v stringr 1.4.0
## v tidyr   1.1.3     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x gridExtra::combine() masks dplyr::combine()
## x dplyr::filter()      masks stats::filter()
## x dplyr::lag()         masks stats::lag()
## x purrr::lift()        masks caret::lift()
```

```r
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: rpart
```

```r
if(!require(rpart.plot)) install.packages("rpart.plot", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: rpart.plot
```

```r
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: randomForest
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:gridExtra':
##
##     combine

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library (readr)
library(caret)
library(dplyr)
library(corrplot)
library(gridExtra)
library(tidyverse)
library(rpart)
library(rpart.plot)
library(randomForest)


urlfile<-"https://raw.githubusercontent.com/danman13/Capstone/main/heart.csv"
heart<-read_csv(url(urlfile))
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   age = col_double(),
##   sex = col_double(),
##   cp = col_double(),
##   trtbps = col_double(),
##   chol = col_double(),
##   fbs = col_double(),
##   restecg = col_double(),
##   thalachh = col_double(),
##   exng = col_double(),
##   oldpeak = col_double(),
##   slp = col_double(),
##   caa = col_double(),
##   thall = col_double(),
##   output = col_double()
## )
```

```r
class(heart)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

```r
heart <- as.data.frame(heart)
class(heart)
```

```
## [1] "data.frame"
```

The dataset is stored in Github and is a tibble initially. I converted it into a dataframe

## Section 2: Methods and Analysis

The next step is to explore the data and note any significant observations. In this section, I will analyze the data, note any trends, and visualize different variables. Since I am predicting output, I want to note correlation between any variables, which can result in colinearity and make it difficult to isolate the effect of a single varibale. I also want to note any blank data cells, since that will impact training of different algorithms. Finally, I want to note any relevant trends or observations.

At a high level, I will be training three different types of algorithms: regression, KNN, and random forests.

Start with a high level data exploration

```r
dim(heart)
```

```
## [1] 303  14
```

```r
head(heart)
```

```
##   age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall output
## 1  63   1  3    145  233   1       0      150    0     2.3   0   0     1      1
## 2  37   1  2    130  250   0       1      187    0     3.5   0   0     2      1
## 3  41   0  1    130  204   0       0      172    0     1.4   2   0     2      1
## 4  56   1  1    120  236   0       1      178    0     0.8   2   0     2      1
## 5  57   0  0    120  354   0       1      163    1     0.6   2   0     2      1
## 6  57   1  0    140  192   0       1      148    0     0.4   1   0     1      1
```

```r
str(heart)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : num  63 37 41 56 57 57 56 44 52 57 ...
##  $ sex     : num  1 1 0 1 0 1 0 1 1 1 ...
##  $ cp      : num  3 2 1 1 0 0 1 1 2 2 ...
##  $ trtbps  : num  145 130 130 120 120 140 140 120 172 150 ...
##  $ chol    : num  233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : num  1 0 0 0 0 0 0 0 1 0 ...
##  $ restecg : num  0 1 0 1 1 1 0 1 1 1 ...
##  $ thalachh: num  150 187 172 178 163 148 153 173 162 174 ...
##  $ exng    : num  0 0 0 0 1 0 0 0 0 0 ...
##  $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ slp     : num  0 0 2 2 2 1 1 2 2 2 ...
##  $ caa     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ thall   : num  1 2 2 2 2 1 2 3 3 2 ...
##  $ output  : num  1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   sex = col_double(),
##   ..   cp = col_double(),
##   ..   trtbps = col_double(),
##   ..   chol = col_double(),
##   ..   fbs = col_double(),
##   ..   restecg = col_double(),
##   ..   thalachh = col_double(),
##   ..   exng = col_double(),
##   ..   oldpeak = col_double(),
##   ..   slp = col_double(),
##   ..   caa = col_double(),
##   ..   thall = col_double(),
##   ..   output = col_double()
##   .. )
```

```
table(heart$output)
```

```
##
##   0   1
## 138 165
```

```
prop.table(table(heart$output))
```

```
##
##         0         1
## 0.4554455 0.5445545
```

This initial summary data shows that there are 303 rows and 14 columns. This is a fairly small dataset, which means right off the bat there is a good chance that whatever model I create may be overly sensitive or have limited applications. Analyzing the first few rows, there are 13 predictors we can use to predict the 14th variable - output. All of the data is in numeric format, even though some variables, such as sex or chest pain, appear to be factors. Finally, we see roughly 55% of people in the dataset have had a heart attack.

Let's reclass these factors from numeric to factor

```
heart$sex <- factor(heart$sex)
levels(heart$sex) <- c("Female", "Male")

heart$cp <- factor(heart$cp)
heart$fbs <- factor(heart$fbs)
heart$restecg <- factor(heart$restecg)
heart$exng <- factor(heart$exng)
heart$slp <- factor(heart$slp)
heart$caa <- factor(heart$caa)
heart$thall <- factor(heart$thall)
heart$output <- factor(heart$output)
```

Next, let's look for missing values that could impact our prediction:

```r
str(heart)
```

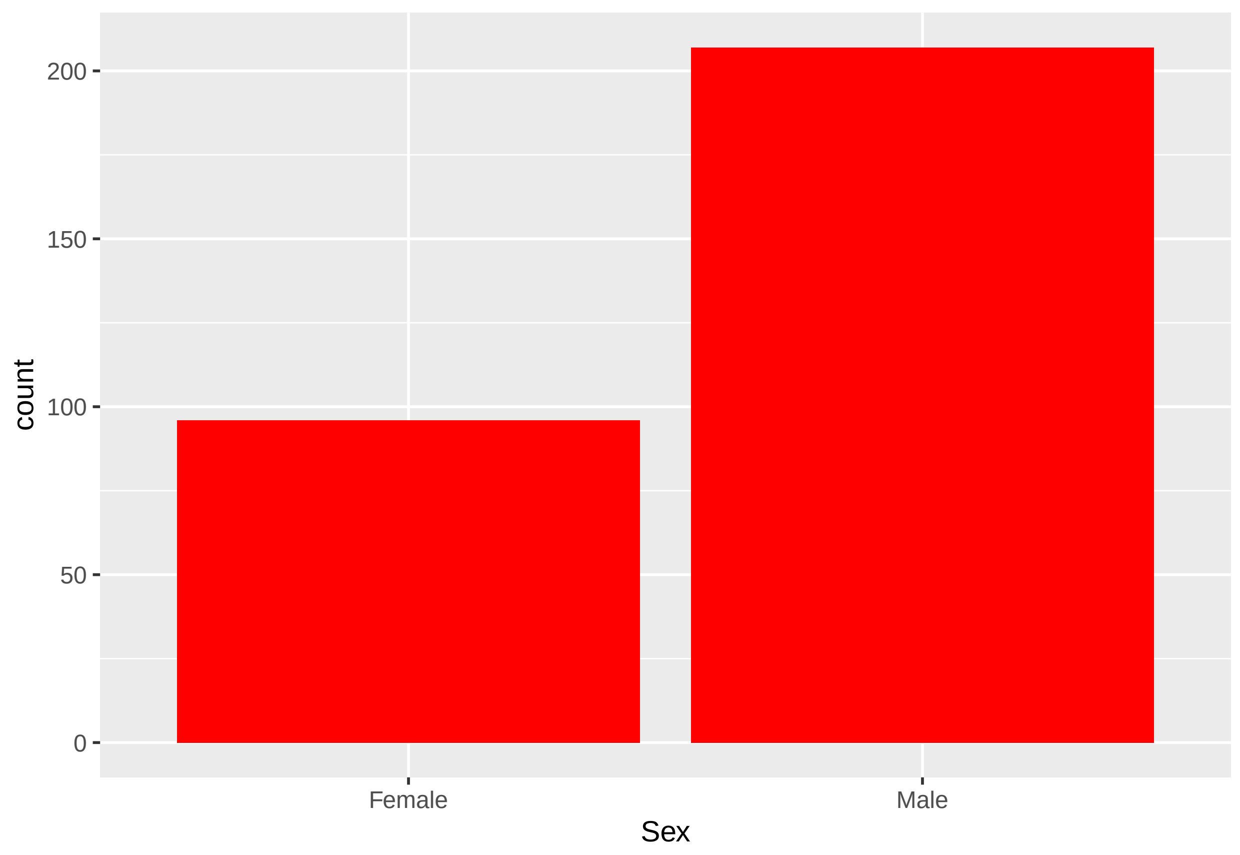
```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : num  63 37 41 56 57 57 56 44 52 57 ...
##  $ sex     : Factor w/ 2 levels "Female","Male": 2 2 1 2 1 2 1 2 2 2 ...
##  $ cp      : Factor w/ 4 levels "0","1","2","3": 4 3 2 2 1 1 2 2 3 3 ...
##  $ trtbps  : num  145 130 130 120 120 140 140 120 172 150 ...
##  $ chol    : num  233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 2 1 ...
##  $ restecg : Factor w/ 3 levels "0","1","2": 1 2 1 2 2 2 1 2 2 2 ...
##  $ thalachh: num  150 187 172 178 163 148 153 173 162 174 ...
##  $ exng    : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...
##  $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ slp     : Factor w/ 3 levels "0","1","2": 1 1 3 3 3 2 2 3 3 3 ...
##  $ caa     : Factor w/ 5 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ thall   : Factor w/ 4 levels "0","1","2","3": 2 3 3 3 3 2 3 4 4 3 ...
##  $ output  : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   sex = col_double(),
##   ..   cp = col_double(),
##   ..   trtbps = col_double(),
##   ..   chol = col_double(),
##   ..   fbs = col_double(),
##   ..   restecg = col_double(),
##   ..   thalachh = col_double(),
##   ..   exng = col_double(),
##   ..   oldpeak = col_double(),
##   ..   slp = col_double(),
##   ..   caa = col_double(),
##   ..   thall = col_double(),
##   ..   output = col_double()
##   .. )
```

```r
anyNA(heart)
```

```
## [1] FALSE
```

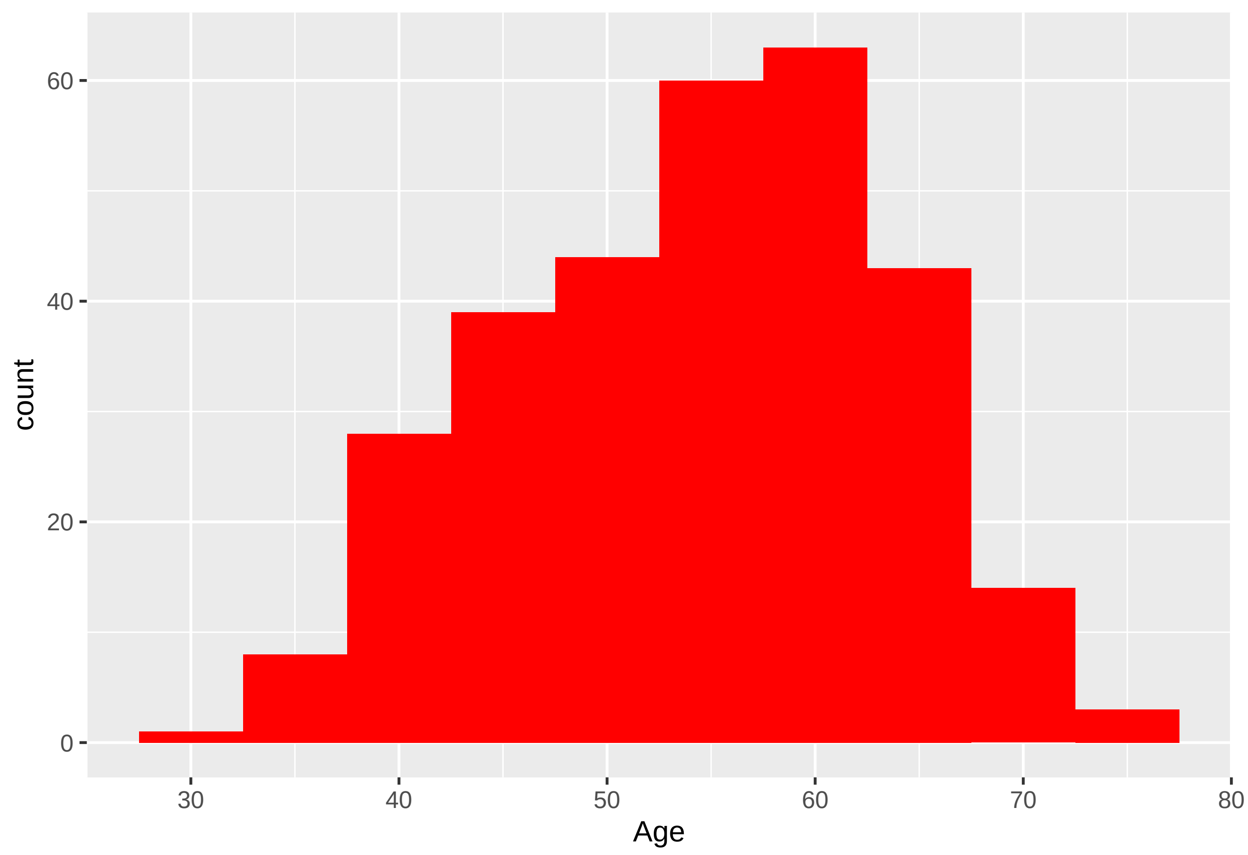
No missing values present. Great! Let's move on. Next, I want to visualize different variables. First one up is sex. We'll plot a bar chart to show the distribution of sex in the dataset, since sex should be a good predictor of heart disease intuitively.

```r
heart %>%
  ggplot(aes(x = factor(sex))) +
  geom_bar(fill = "red") +
  xlab("Sex")
```



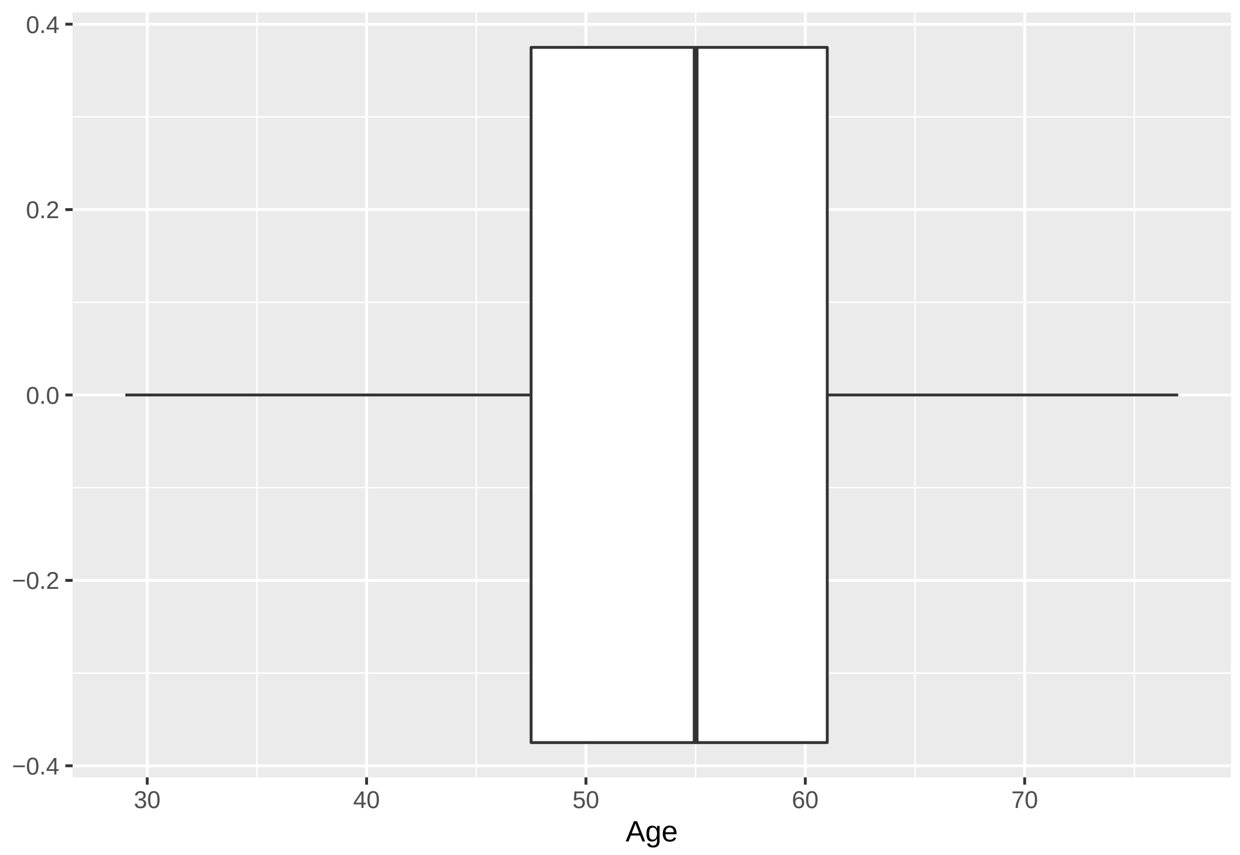
Males are more prevalent in the dataset by roughly 2x Next, I look at age, since this should also be highly correlated with heart disease

```
heart %>%
  ggplot(aes(x = age)) +
  geom_histogram(binwidth = 5, fill = "red") +
  xlab("Age")
```



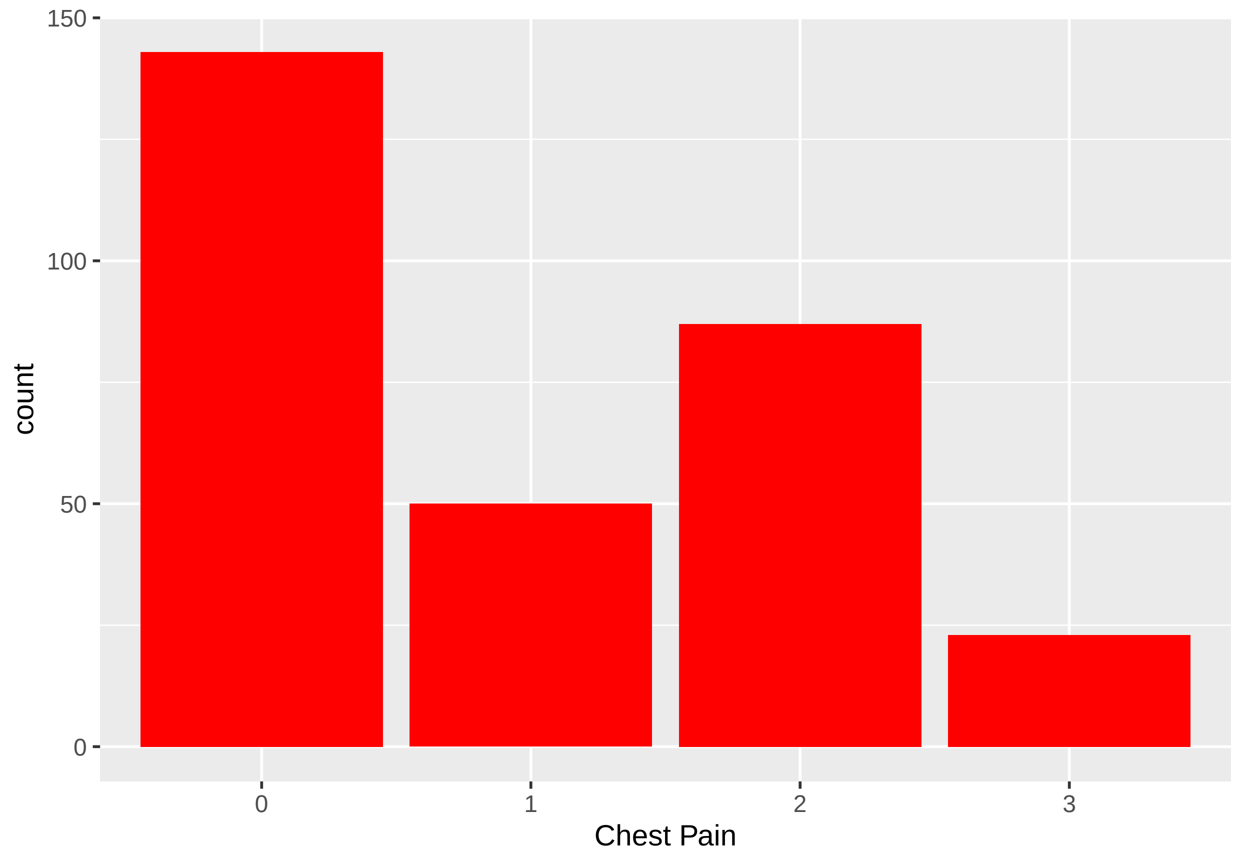
The distribution is skewed slightly younger, but the median age appears to be around 55 as shown in the boxplot

```
heart %>%
  ggplot(aes(x = age)) +
  geom_boxplot() +
  xlab("Age")
```



Finally, let's see chest pain's distribution:

```
heart %>%
  ggplot(aes(x = factor(cp))) +
  geom_bar(fill = "red") +
  xlab("Chest Pain")
```



Chest pain is a factor with 4 levels - 0 is typical angia, 1 is atypical angia, 2 is non-anginal, and 3 is asymptomatic. Based off of this, almost 50% show typical angia pain and about 25%-30% show non-anginal. Asymptomatic is the rarest in the dataset

Next, I want to look at correlation between the numeric data. Highly correlated variables (either positive or negative) could have a big swing on our resulting equation and open us up to increased sensitivity when one variable changes, esentially overfitting. I'll split the dataset into a new dataset that contains only the numeric variables

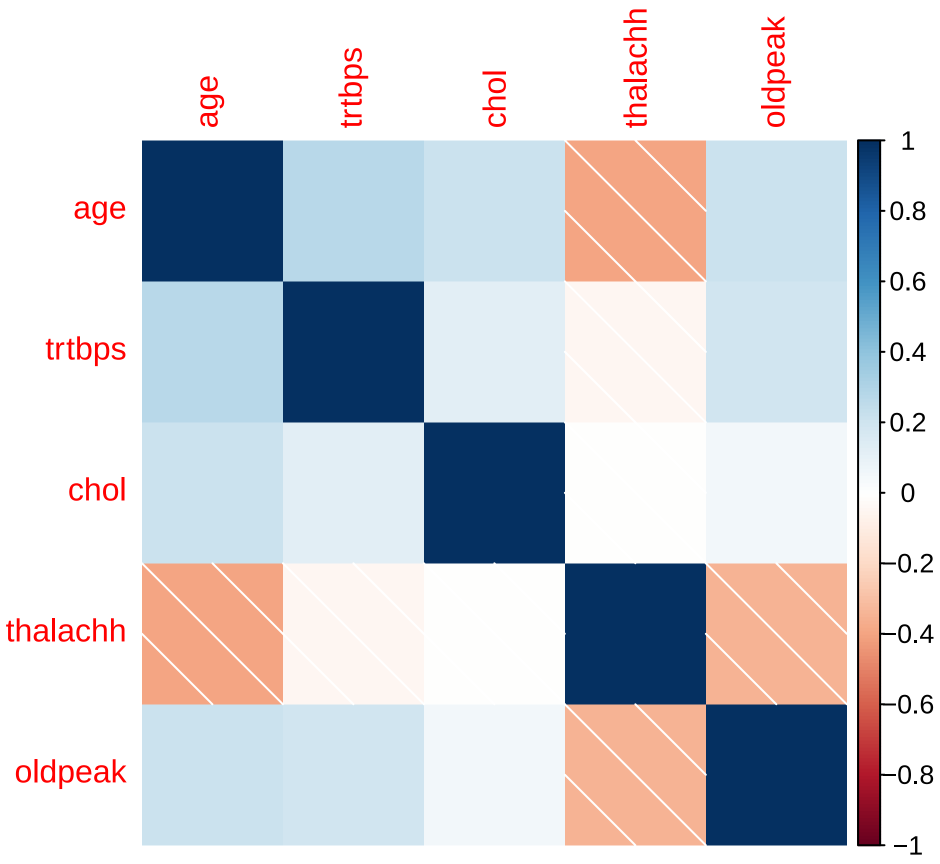
```
numeric_vector <- c(1,4,5,8,10)
heart_numeric <- heart[,numeric_vector]
head(heart_numeric)
```

```
##   age trtbps chol thalachh oldpeak
## 1  63    145  233      150     2.3
## 2  37    130  250      187     3.5
## 3  41    130  204      172     1.4
## 4  56    120  236      178     0.8
## 5  57    120  354      163     0.6
## 6  57    140  192      148     0.4
```

```
corelation <- cor(heart_numeric)
head(corelation)
```

```
##                 age     trtbps        chol     thalachh     oldpeak
## age       1.0000000 0.27935091 0.213677957 -0.398521938  0.21001257
## trtbps    0.2793509  1.00000000  0.123174207 -0.046697728  0.19321647
## chol      0.2136780  0.12317421  1.000000000 -0.009939839  0.05395192
## thalachh -0.3985219 -0.04669773 -0.009939839  1.000000000 -0.34418695
## oldpeak   0.2100126  0.19321647  0.053951920 -0.344186948  1.00000000
```

```
corrplot(corelation, method="shade")
```



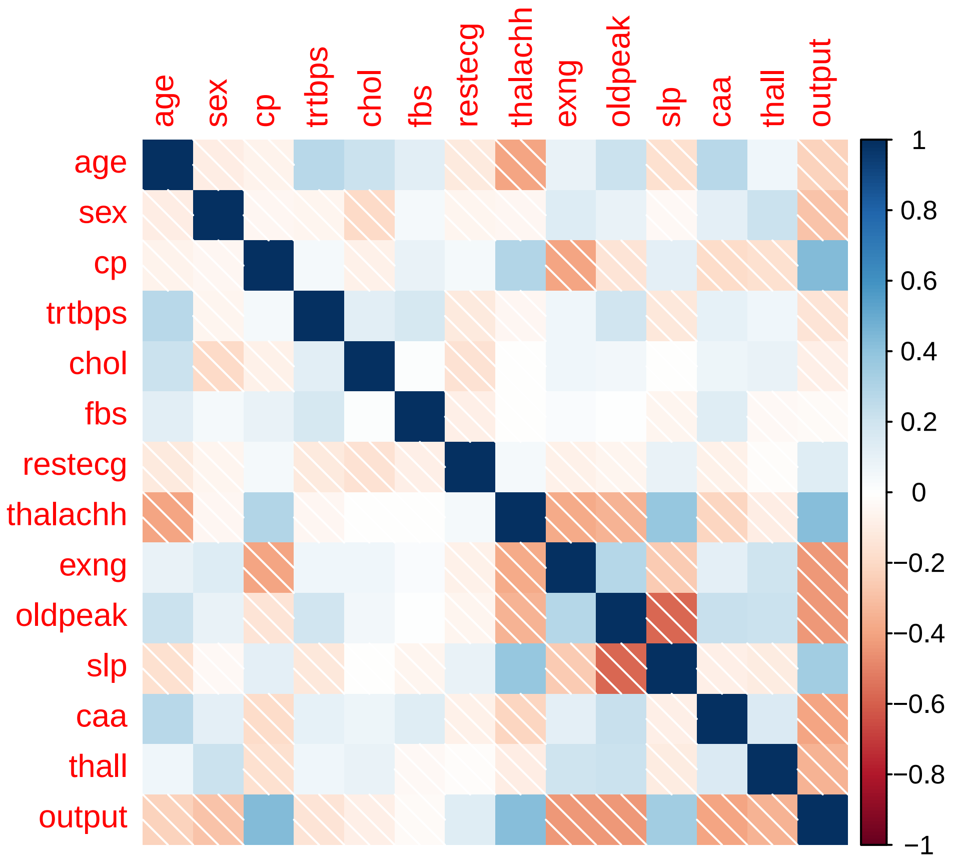
We see some stonger correlation (thalachh vs age or thalachh vs old peak), but these variables aren't too correlated meaning they should be good to go for our model

R gives an error when trying to see correlations between non-numeric data, so to work around this, I'll create another dataset using the original since it initally came as numeric.

```
heart2<-read_csv(url(urlfile))
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   age = col_double(),
##   sex = col_double(),
##   cp = col_double(),
##   trtbps = col_double(),
##   chol = col_double(),
##   fbs = col_double(),
##   restecg = col_double(),
##   thalachh = col_double(),
##   exng = col_double(),
##   oldpeak = col_double(),
##   slp = col_double(),
##   caa = col_double(),
##   thall = col_double(),
##   output = col_double()
## )
```

```
heart2 <- as.data.frame(heart2)
cor_heart2 <- cor(heart2)
corrplot(cor_heart2,method = "shade")
```



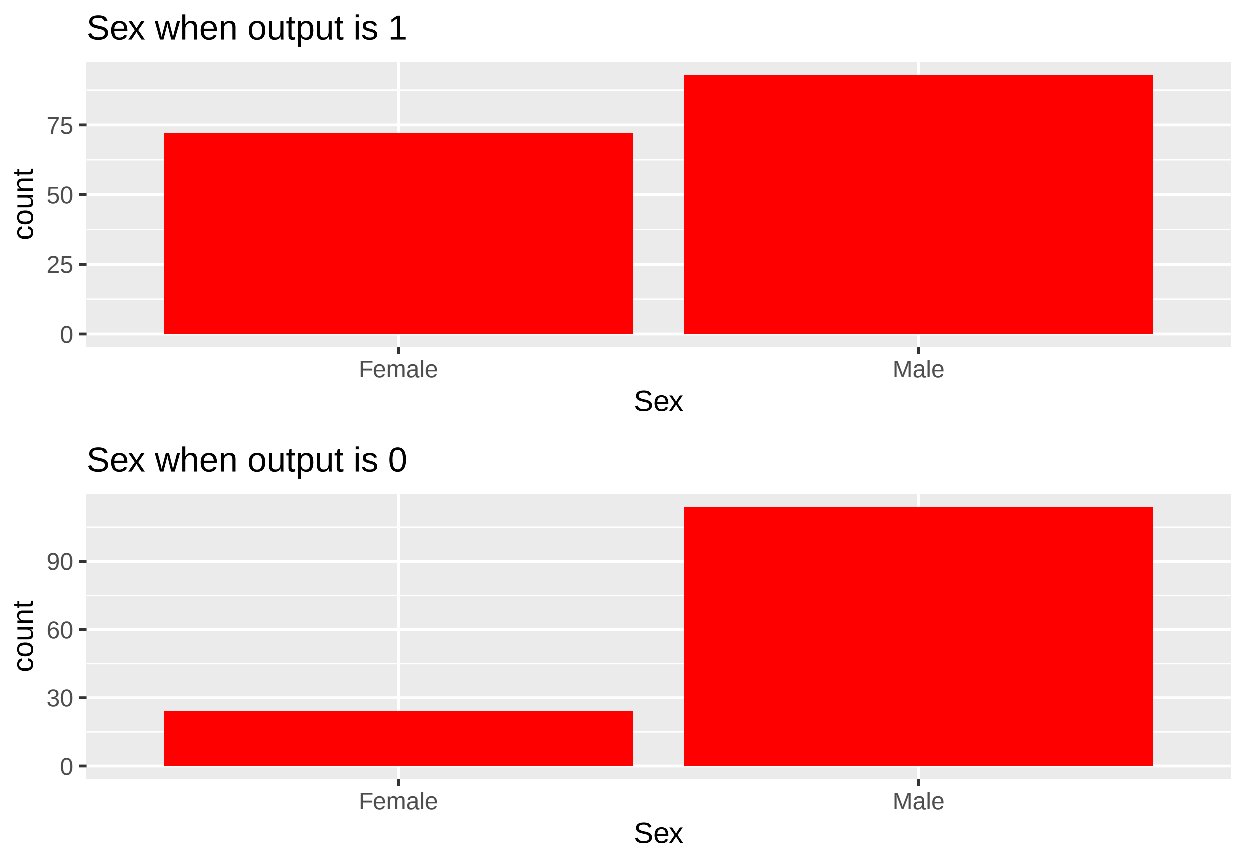
observe that outcome has stronger correlation with certain variables like chest pain, restecg, or old peak, but again there is no strong correlation between two predictors.

Next, I want to see conditional impact on different variables. For instance, what is the distribution of sex when heart disease is present vs the distribution when it isn't.

Let's look first at sex when heart disease is and isn't present

```
sex1 <- heart %>% filter(output == 1) %>%
  ggplot(aes(x = factor(sex)) +
  geom_bar(fill = "Red")+
  xlab("Sex")+
  ggtitle("Sex when output is 1")

sex2 <- heart %>% filter(output == 0) %>%
  ggplot(aes(x = factor(sex))) +
  geom_bar(fill = "Red")+
  xlab("Sex")+
  ggtitle("Sex when output is 0")
grid.arrange(sex1, sex2)
```

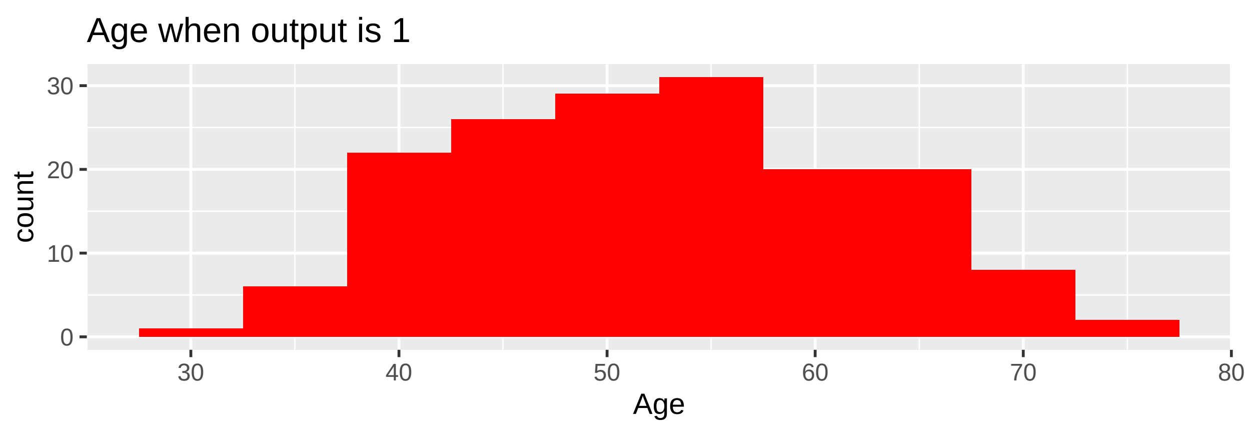


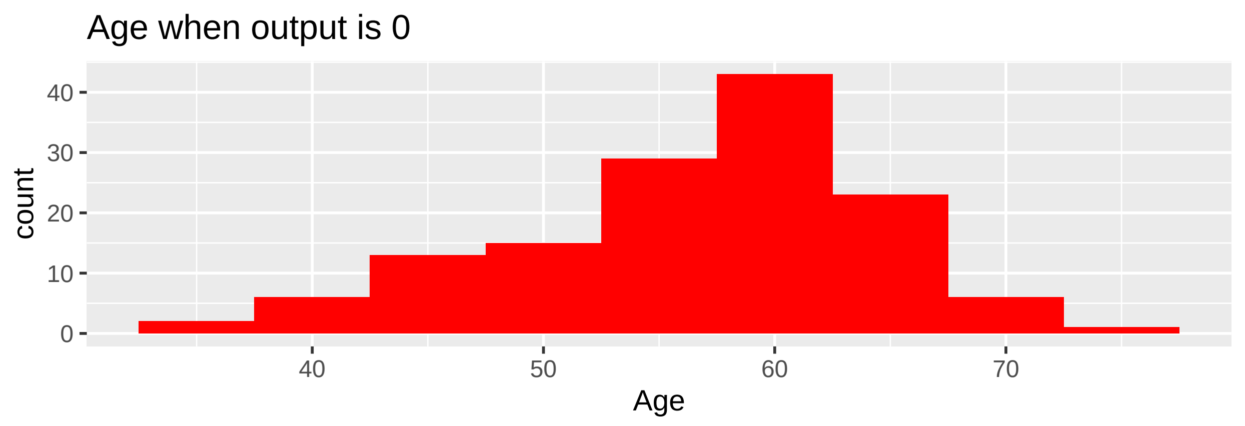
There's a pretty even split when heart disease is present, but it is skewed male when none is present. As a percentage of each sex, female has a higher share of disease in the dataset.

Let's look at age next:

```
age1 <- heart %>% filter(output == 1) %>%
  ggplot(aes(x = age)) +
  geom_histogram(binwidth = 5,fill = "Red")+
  xlab("Age")+
  ggtitle("Age when output is 1")

age2 <- heart %>% filter(output == 0) %>%
  ggplot(aes(x = age)) +
  geom_histogram(binwidth = 5,fill = "Red")+
  xlab("Age")+
  ggtitle("Age when output is 0")
grid.arrange(age1, age2)
```



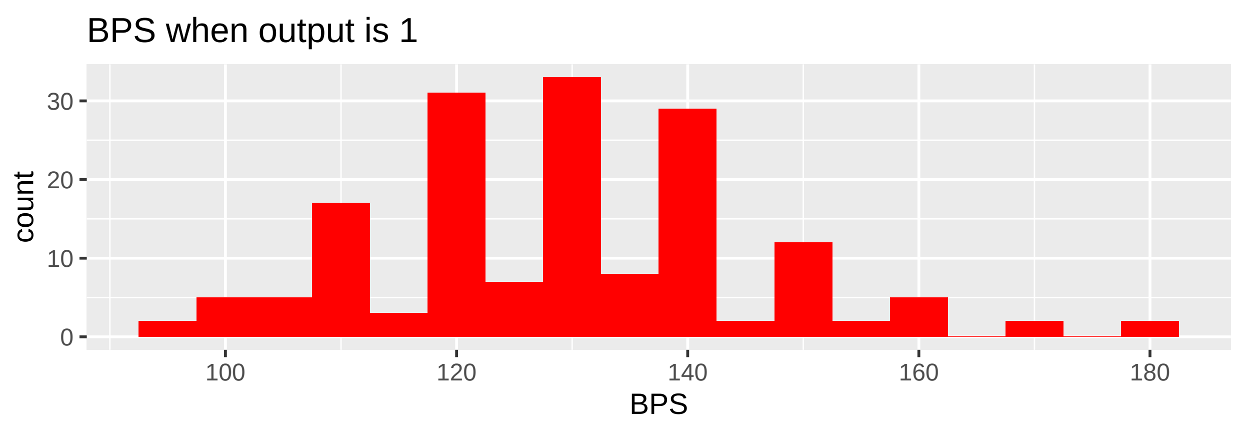


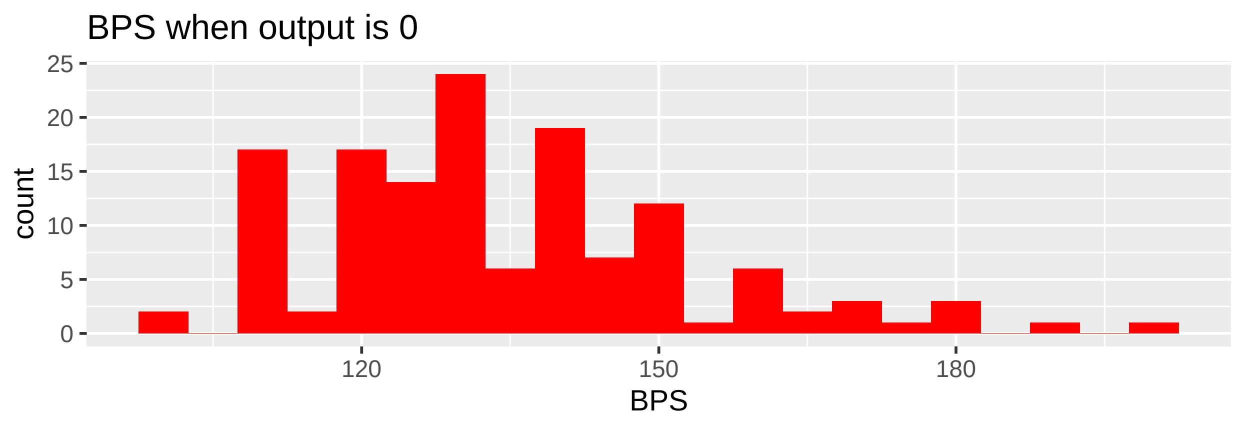
Age skews slightly older when no heart disease is present, which is interesting since the general population tends to skew older when heart disease IS present

Max blood pressure is next:

```
bps1 <- heart %>% filter(output == 1) %>%
  ggplot(aes(x = trtbps)) +
  geom_histogram(binwidth = 5,fill = "Red")+
  xlab("BPS")+
  ggtitle("BPS when output is 1")

bps2 <- heart %>% filter(output == 0) %>%
  ggplot(aes(x = trtbps)) +
  geom_histogram(binwidth = 5,fill = "Red")+
  xlab("BPS")+
  ggtitle("BPS when output is 0")
grid.arrange(bps1, bps2)
```



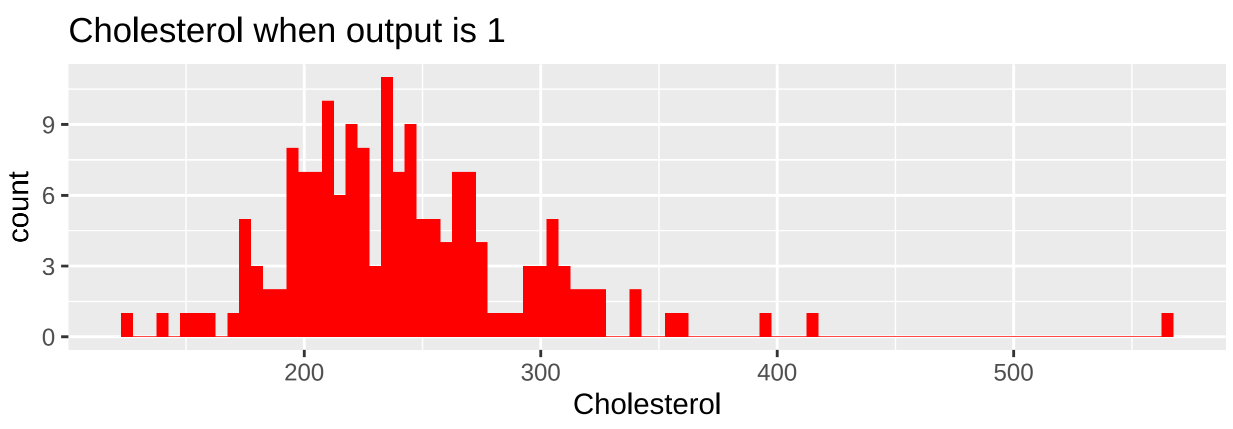


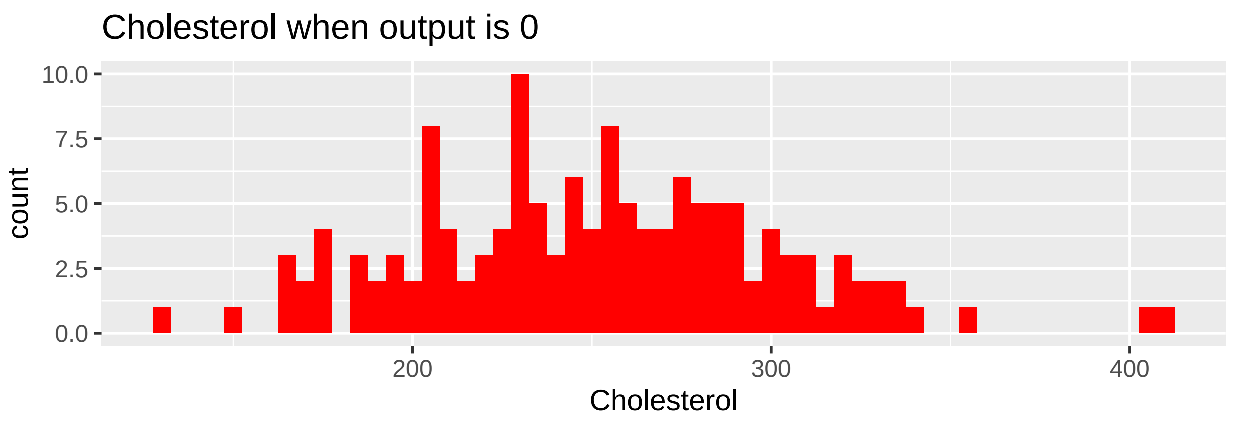
This is not normally distributed for either "yes" or "no" but BPS is slightly higher when heart disease is present.

Let's look at cholesterol next:

```
chol1 <- heart %>% filter(output == 1) %>%
  ggplot(aes(x = chol)) +
  geom_histogram(binwidth = 5,fill = "Red")+
  xlab("Cholesterol")+
  ggtitle("Cholesterol when output is 1")

chol2 <- heart %>% filter(output == 0) %>%
  ggplot(aes(x = chol)) +
  geom_histogram(binwidth = 5,fill = "Red")+
  xlab("Cholesterol")+
  ggtitle("Cholesterol when output is 0")
grid.arrange(chol1, chol2)
```



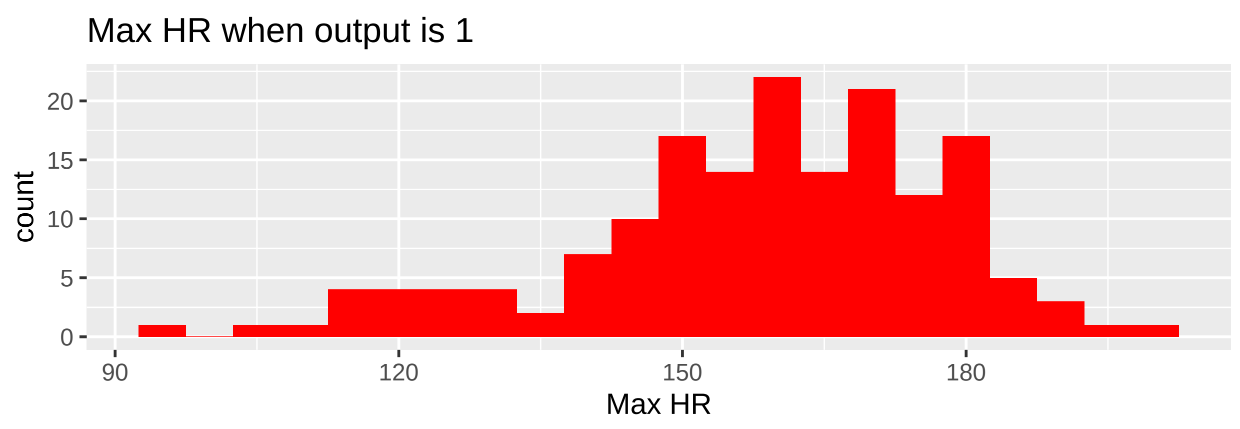


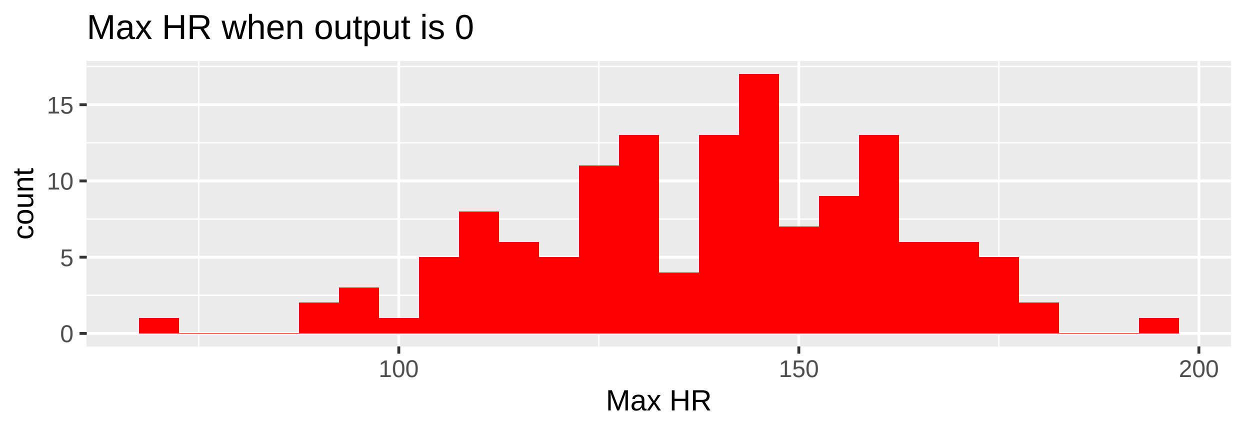
Intersting. Cholesterol is lower when heart disease is present. Could any potential medication play into this? Also, the dataset doesn't say whether this is HDL, LDL, or total cholesterol.

Two more variables to look at. First, max heart rate:

```
maxhr1 <- heart %>% filter(output == 1) %>%
  ggplot(aes(x = thalachh)) +
  geom_histogram(binwidth = 5,fill = "Red")+
  xlab("Max HR")+
  ggtitle("Max HR when output is 1")

maxhr2 <- heart %>% filter(output == 0) %>%
  ggplot(aes(x = thalachh)) +
  geom_histogram(binwidth = 5,fill = "Red")+
  xlab("Max HR")+
  ggtitle("Max HR when output is 0")
grid.arrange(maxhr1, maxhr2)
```



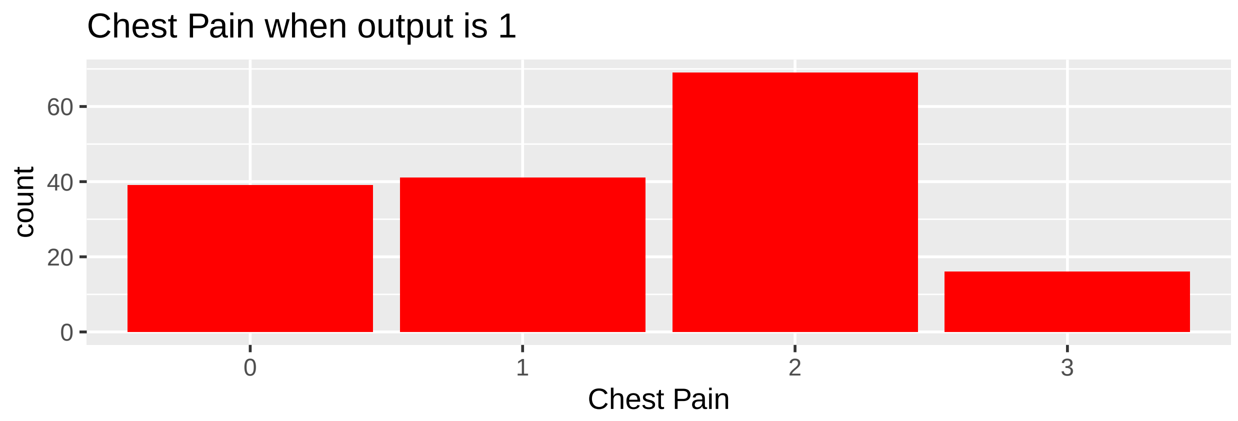


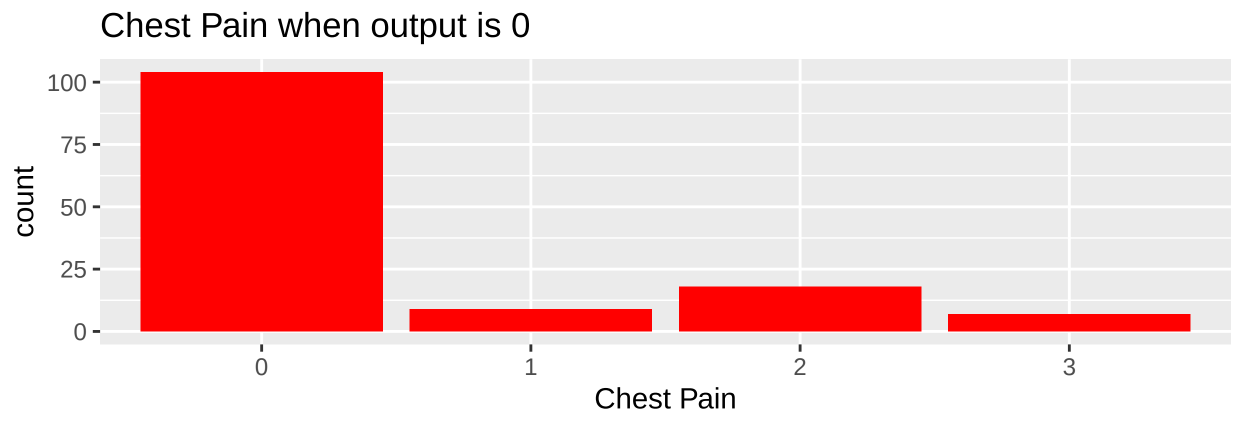
Heart rate is higher when heart disease is present, which is consistent with intuition since the heart would need to work much harder.

Finally, the chest pain distribution:

```
cp1 <- heart %>% filter(output == 1) %>%
  ggplot(aes(x = factor(cp))) +
  geom_bar(fill = "Red")+
  xlab("Chest Pain")+
  ggtitle("Chest Pain when output is 1")

cp2 <- heart %>% filter(output == 0) %>%
  ggplot(aes(x = factor(cp))) +
  geom_bar(fill = "Red")+
  xlab("Chest Pain")+
  ggtitle("Chest Pain when output is 0")
grid.arrange(cp1, cp2)
```





Although typical angia is the most prevalent in the dataset, non-anginal is the most common when someone has heart disease. Maybe if this could be caught early on, heart diease could be prevented (assuming this is a cause rather than effect).

Now that the deep exploration is done, we're going to move on to splitting the data into training and test sets. I chose a 75/25 split for the data since the dataset is so small. I didn't want the validation set to be so small that it didn't have the capability of validating the data.

```r
n.point <- nrow(heart)
sampling_rate <- 0.75
set.seed(123)
train <- sample(1:n.point, sampling_rate * n.point, replace = FALSE)
test <- setdiff(1:n.point, train)
heart_train <- subset(heart[train,])
heart_test <- subset(heart[test,])
heart_train <- as.data.frame(heart_train)
heart_test <- as.data.frame(heart_test)
head(heart_train)
```

```
##     age    sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall
## 179  43   Male  0    120  177   0       0      120    1     2.5   1   0     3
## 14   64   Male  3    110  211   0       0      144    1     1.8   1   0     2
## 195  60   Male  2    140  185   0       0      155    0     3.0   1   0     2
## 118  56   Male  3    120  193   0       0      162    0     1.9   1   0     3
## 299  57 Female  0    140  241   0       1      123    1     0.2   1   0     3
## 229  59   Male  3    170  288   0       0      159    0     0.2   1   0     3
##     output
## 179      0
## 14       1
## 195      0
## 118      1
## 299      0
## 229      0
```

```
head(heart_test)
```

```
##    age    sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall
## 2   37   Male  2    130  250   0       1      187    0     3.5   0   0     2
## 3   41 Female  1    130  204   0       0      172    0     1.4   2   0     2
## 12  48 Female  2    130  275   0       1      139    0     0.2   2   0     2
## 15  58 Female  3    150  283   1       0      162    0     1.0   2   0     2
## 18  66 Female  3    150  226   0       1      114    0     2.6   0   0     2
## 19  43   Male  0    150  247   0       1      171    0     1.5   2   0     2
##    output
## 2       1
## 3       1
## 12      1
## 15      1
## 18      1
## 19      1
```

```
nrow(heart_train)
```

```
## [1] 227
```

```
nrow(heart_test)
```

```
## [1] 76
```

```
str(heart_train)
```

```
## 'data.frame':    227 obs. of  14 variables:
##  $ age     : num  43 64 60 56 57 59 57 57 64 58 ...
##  $ sex     : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 2 2 2 2 1 ...
##  $ cp      : Factor w/ 4 levels "0","1","2","3": 1 4 3 4 1 4 1 1 4 1 ...
##  $ trtbps  : num  120 110 140 120 140 170 152 130 170 100 ...
##  $ chol    : num  177 211 185 193 241 288 274 131 227 248 ...
##  $ fbs     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ restecg : Factor w/ 3 levels "0","1","2": 1 1 1 1 2 1 2 2 1 1 ...
##  $ thalachh: num  120 144 155 162 123 159 88 115 155 122 ...
##  $ exng    : Factor w/ 2 levels "0","1": 2 2 1 1 2 1 2 2 1 1 ...
##  $ oldpeak : num  2.5 1.8 3 1.9 0.2 0.2 1.2 1.2 0.6 1 ...
##  $ slp     : Factor w/ 3 levels "0","1","2": 2 2 2 2 2 2 2 2 2 2 ...
##  $ caa     : Factor w/ 5 levels "0","1","2","3",..: 1 1 1 1 1 1 2 2 1 1 ...
##  $ thall   : Factor w/ 4 levels "0","1","2","3": 4 3 3 4 4 4 4 4 4 3 ...
##  $ output  : Factor w/ 2 levels "0","1": 1 2 1 2 1 1 1 1 2 2 ...
```

```
str(heart_test)
```

```
## 'data.frame':    76 obs. of  14 variables:
##  $ age     : num  37 41 48 58 66 43 51 54 53 39 ...
##  $ sex     : Factor w/ 2 levels "Female","Male": 2 1 1 1 1 2 2 2 1 2 ...
##  $ cp      : Factor w/ 4 levels "0","1","2","3": 3 2 3 4 4 1 3 3 1 3 ...
##  $ trtbps  : num  130 130 130 150 150 150 110 150 130 140 ...
##  $ chol    : num  250 204 275 283 226 247 175 232 264 321 ...
##  $ fbs     : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 1 1 ...
##  $ restecg : Factor w/ 3 levels "0","1","2": 2 1 2 1 2 2 2 1 1 1 ...
##  $ thalachh: num  187 172 139 162 114 171 123 165 143 182 ...
##  $ exng    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ oldpeak : num  3.5 1.4 0.2 1 2.6 1.5 0.6 1.6 0.4 0 ...
##  $ slp     : Factor w/ 3 levels "0","1","2": 1 3 3 3 1 3 3 3 2 3 ...
##  $ caa     : Factor w/ 5 levels "0","1","2","3",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ thall   : Factor w/ 4 levels "0","1","2","3": 3 3 3 3 3 3 3 4 3 3 ...
##  $ output  : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

This splits the data set into a training set of 227 observations and a test set of 76 observations

The first model I'll try is a logistic regression model. I'm going to try multiple iterations - the first with 3 variables, then with 4, then 5, and finally all variables. I'll pick the best model and discuss further before moving to KNN.

```
set.seed(123)
fit_glm <- glm(output ~ sex + age + chol, data = heart_train, family = "binomial")
p_hat_logistic <- predict(fit_glm, heart_test)
y_hat_logistic <- factor(ifelse(p_hat_logistic > 0.5, 1, 0))
fit_glm_result <- confusionMatrix(data = y_hat_logistic, reference = heart_test$output)$overall[1]
fit_glm_result
```

```
##  Accuracy
## 0.6578947
```

```
results <- data.frame()
results <- data_frame(method="Logistic Regression - variables", accuracy = fit_glm_result)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
```

```
results %>% knitr::kable()
```

| method | accuracy |
|---|---:|
| Logistic Regression - variables | 0.6578947 |

The first mdoel used age, sex, and cholesterol as predictors, showing an accuracy of 51% - definitely not great and I think we can do better. Let's add a fourth variable, chest pain:

```
set.seed(123)
fit_glm2 <- glm(output ~ sex + age + chol + cp, data = heart_train, family = "binomial")
p_hat_logistic2 <- predict(fit_glm2, heart_test)
y_hat_logistic2 <- factor(ifelse(p_hat_logistic2 > 0.5, 1, 0))
fit_glm_result2 <- confusionMatrix(data = y_hat_logistic2, reference = heart_test$output)$overall[1]
fit_glm_result2
```

```
##  Accuracy
## 0.8026316
```

```
results <- bind_rows(results, data_frame(method="Logistic Regression - 4 variables", accuracy = fit_glm
results %>% knitr::kable()
```

| method | accuracy |
|---|---:|
| Logistic Regression - variables | 0.6578947 |
| Logistic Regression - 4 variables | 0.8026316 |

By adding chest pain, our model accuracy improves to 77.3%, a significant step up! Can we do better though? Let's add max heart rate and see:

```
set.seed(123)
fit_glm3 <- glm(output ~ sex + age + chol + cp + thalachh, data = heart_train, family = "binomial")
p_hat_logistic3<- predict(fit_glm3, heart_test)
y_hat_logistic3 <- factor(ifelse(p_hat_logistic3 > 0.5, 1, 0))
fit_glm_result3 <- confusionMatrix(data = y_hat_logistic3, reference = heart_test$output)$overall[1]
fit_glm_result3
```

```
##  Accuracy
## 0.8026316
```

```
results <- bind_rows(results, data_frame(method="Logistic Regression - 5 variables", accuracy = fit_glm
results %>% knitr::kable()
```

| method | accuracy |
|---|---:|
| Logistic Regression - variables | 0.6578947 |
| Logistic Regression - 4 variables | 0.8026316 |
| Logistic Regression - 5 variables | 0.8026316 |

This acutually took us back in accuracy

Let's dive deeper into the four variable regression that performed the best

```
confusionMatrix(data = y_hat_logistic2, heart_test$output)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 27  7
##          1  8 34
##
##                Accuracy : 0.8026
##                  95% CI : (0.6954, 0.8851)
##     No Information Rate : 0.5395
##     P-Value [Acc > NIR] : 1.556e-06
##
##                   Kappa : 0.602
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7714
##             Specificity : 0.8293
##          Pos Pred Value : 0.7941
##          Neg Pred Value : 0.8095
##              Prevalence : 0.4605
##          Detection Rate : 0.3553
##    Detection Prevalence : 0.4474
##       Balanced Accuracy : 0.8003
##
##        'Positive' Class : 0
##
```

Looking at the data, there is high sensitivity, which means that we are good at predicting true positives, but low specificity. This model doesn't predict true negatives well. In a healthcare setting, that is worrisome since we'd rather be safe than sorry. Let's look to KNN to see if we can improve on this regression and bring balance.

```r
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
knn_fit <- knn3(output ~ ., data = heart_train, k = 5)
y_hat_knn <- predict(knn_fit, heart_test, type = "class")
knn_results <- confusionMatrix(data = y_hat_knn, reference = heart_test$output)$overall[1]
results <- bind_rows(results, data_frame(method="Knn - k = 3", accuracy = knn_results))
confusionMatrix(data = y_hat_knn, heart_test$output)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 22 18
##          1 13 23
##
##                Accuracy : 0.5921
##                  95% CI : (0.4733, 0.7035)
##     No Information Rate : 0.5395
##     P-Value [Acc > NIR] : 0.2107
```

```
##
##                   Kappa : 0.1876
##
##  Mcnemar's Test P-Value : 0.4725
##
##             Sensitivity : 0.6286
##             Specificity : 0.5610
##          Pos Pred Value : 0.5500
##          Neg Pred Value : 0.6389
##              Prevalence : 0.4605
##          Detection Rate : 0.2895
##    Detection Prevalence : 0.5263
##       Balanced Accuracy : 0.5948
##
##        'Positive' Class : 0
##
```

```
results %>% knitr::kable()
```

| method | accuracy |
|---|---:|
| Logistic Regression - variables | 0.6578947 |
| Logistic Regression - 4 variables | 0.8026316 |
| Logistic Regression - 5 variables | 0.8026316 |
| Knn - k = 3 | 0.5921053 |

Knn produced 71% accuracy, lower than our regression with higher specificity while sacrificing sensitivity. This model really isn't much better at the end of the day. Let's try a random forest model.

```
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
rf_fit <- randomForest(output ~ .,  data=heart_train)
y_hat_rf <- predict(rf_fit, heart_test)
rf_results <- confusionMatrix(data = y_hat_rf, reference = heart_test$output)$overall[1]
rf_results
```

```
##  Accuracy
## 0.8157895
```

```
confusionMatrix(data = y_hat_rf, heart_test$output)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 23  2
##          1 12 39
```

```
## 
##                Accuracy : 0.8158          
##                  95% CI : (0.7103, 0.8955)
##     No Information Rate : 0.5395          
##     P-Value [Acc > NIR] : 4.281e-07       
##                                           
##                   Kappa : 0.6214          
##                                           
##  Mcnemar's Test P-Value : 0.01616         
##                                           
##             Sensitivity : 0.6571          
##             Specificity : 0.9512          
##          Pos Pred Value : 0.9200          
##          Neg Pred Value : 0.7647          
##              Prevalence : 0.4605          
##          Detection Rate : 0.3026          
##    Detection Prevalence : 0.3289          
##       Balanced Accuracy : 0.8042          
##                                           
##        'Positive' Class : 0               
## 
```

```
results <- bind_rows(results, data_frame(method="Random Forest", accuracy = rf_results))
```

This produces an accuracy of ~82%, by far the best model tried. We took a small hit in sensitivity, but were able to boost specificity, although it isn't as high as we'd like.

## Results

```
results %>% knitr::kable()
```

| method | accuracy |
|---|---:|
| Logistic Regression - variables | 0.6578947 |
| Logistic Regression - 4 variables | 0.8026316 |
| Logistic Regression - 5 variables | 0.8026316 |
| Knn - k = 3 | 0.5921053 |
| Random Forest | 0.8157895 |

Our random forest model was the top performer with an accuracy of over 80%. It had a high sensitivity, but specificity could use improvement. I think specificity is the biggest drawback of all these models. The accuracy confidence interval ranged from .71 to almost .9, so there is a chance this model overperforms.

## Conclusion

Looking at the heart attack data, we were able to train a machine learning model with around 80% accruacy. The overall accuracy and sensitivity (true positive rate) were great, but the specificity wasn't what we needed it to be for the model to be useful. In a medical setting, we would rather see a false positive than a false negative, but this model doesn't predict false negatives well. The dataset size is also a limiting factor.

Because we are dealing with limited observations, the overall model might be more sensitive to swings in one variable or outliers than if we had more datapoints. Combined, these really limit our model's usefulness.

The model definitely could be improved. For instance, on all models, we can tweak which predictors we use. We can also use an ensemble method to further normalize noise in our dataset between algorithms.