Project 1 - DS8002

Rafik Matta

Bike Data Set Analysis

Exploratorary Data Analysis

From the hour.csv data set, I have chosen the following 5 features:

- season : season (1:springer, 2:summer, 3:fall, 4:winter)
- weathersit :
- 1: Clear, Few clouds, Partly cloudy, Partly cloudy
- 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
- 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
- 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- temp : Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hourly scale)
- windspeed: Normalized wind speed. The values are divided to 67 (max)
- cnt: count of total rental bikes including both casual and registered

I begin by selecting my features the required data:

```
library(readr)
library("MASS")
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(plotly)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'plotly'
```

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

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
hour <- read_csv("C:/Users/rafik/Google Drive/Education/ds8002/ds8002project/Bike-Sharing-Dataset/hour.csv")
```

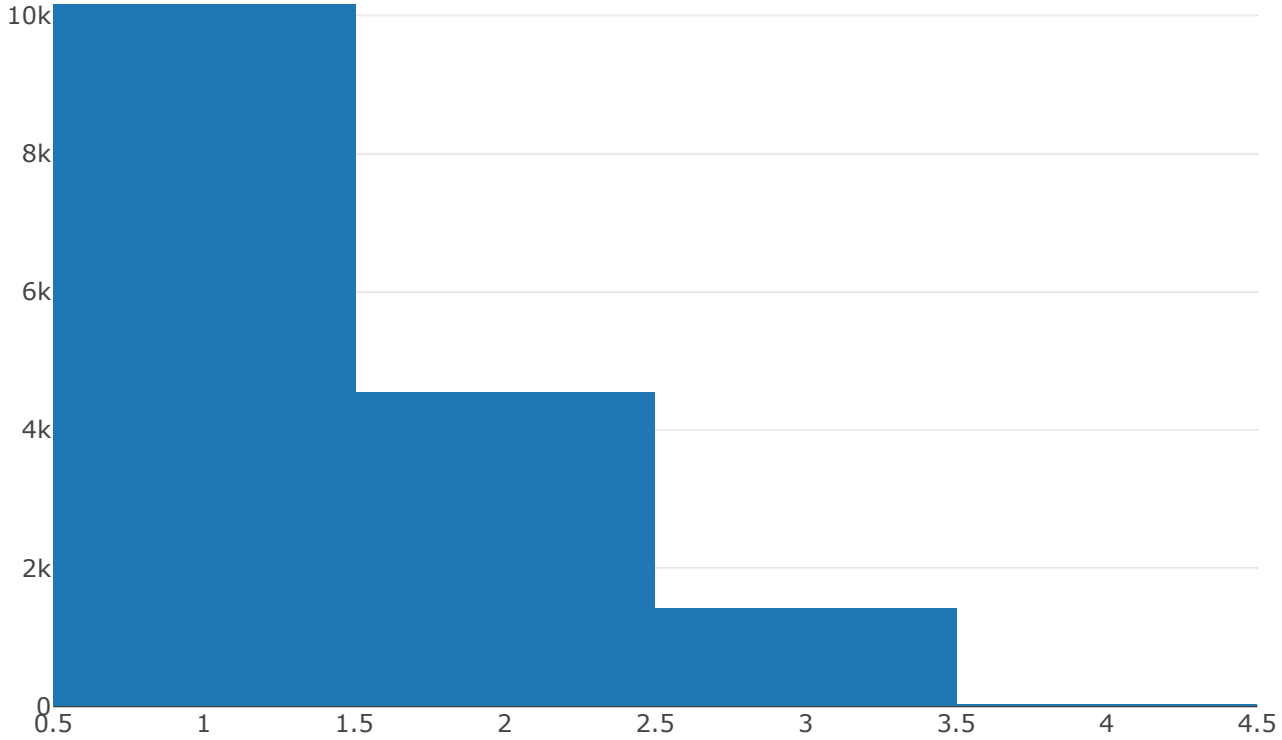
```
## Parsed with column specification:
## cols(
##   instant = col_integer(),
##   dteday = col_date(format = ""),
##   season = col_integer(),
##   yr = col_integer(),
##   mnth = col_integer(),
##   hr = col_integer(),
##   holiday = col_integer(),
##   weekday = col_integer(),
##   workingday = col_integer(),
##   weathersit = col_integer(),
##   temp = col_double(),
##   atemp = col_double(),
##   hum = col_double(),
##   windspeed = col_double(),
##   casual = col_integer(),
##   registered = col_integer(),
##   cnt = col_integer()
## )
```

```
#pick interesting features of bike data
bike_share_data <- data.frame(hour$weathersit,
                              hour$temp,
                              hour$windspeed,
                              hour$season,
                              hour$cnt)
```

Statistics

Weather Situtaion

```
plot_ly(x = bike_share_data$hour.weathersit, type = "histogram")
```




```
summary(bike_share_data$hour.weathersit)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.425   2.000   4.000
```

```
mean(bike_share_data$hour.weathersit)
```

```
## [1] 1.425283
```

```
median(bike_share_data$hour.weathersit)
```

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

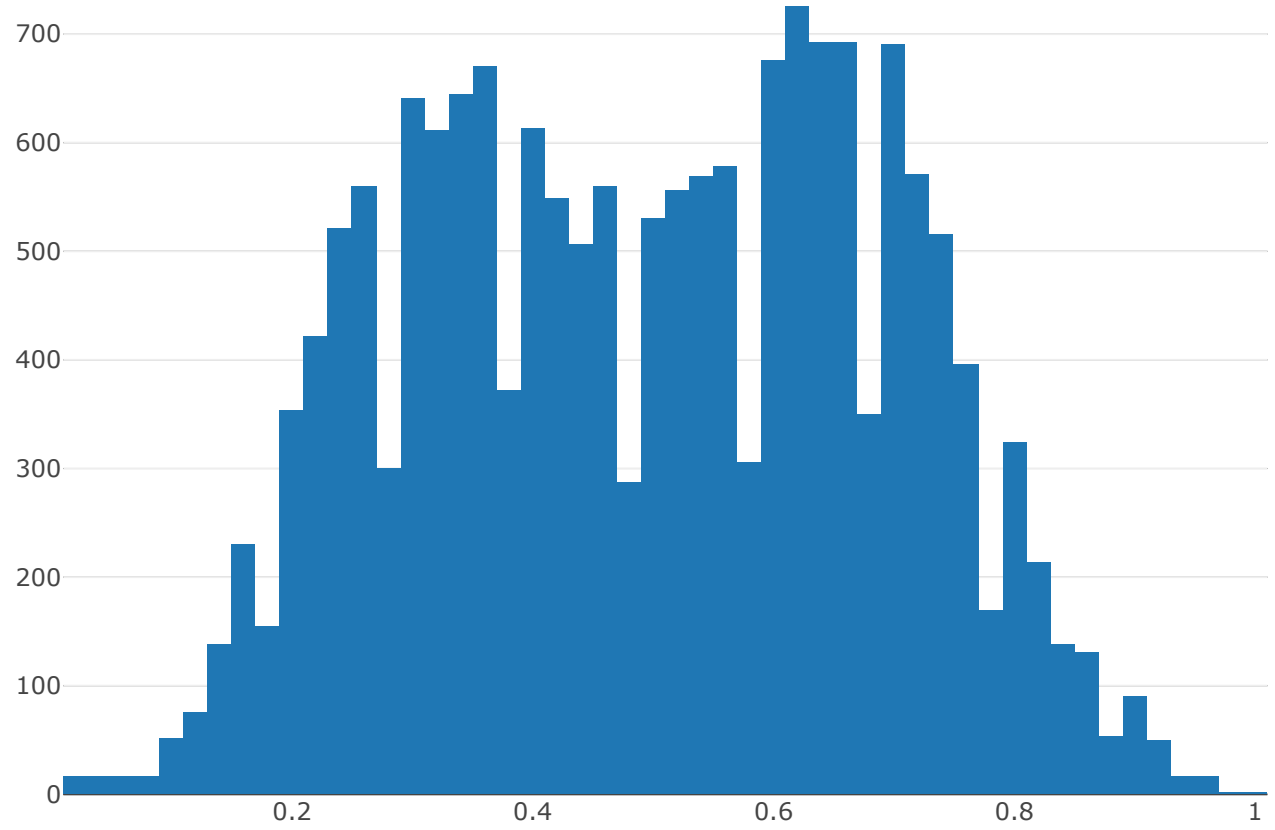
```
var(bike_share_data$hour.weathersit)
```

```
## [1] 0.4087772
```

Here I have made the assumption that I can treat the weather situation as ordinal, where the worse the weather situation the higher it's value. My assumption is that on average the weather is fine but if the weather is very bad it will contribute to a tremendous drop in ridership.

Temperature

```
plot_ly(x = bike_share_data$hour.temp, type = "histogram")
```




```
mean(bike_share_data$hour.temp)
```

```
## [1] 0.4969872
```

```
median(bike_share_data$hour.temp)
```

```
## [1] 0.5
```

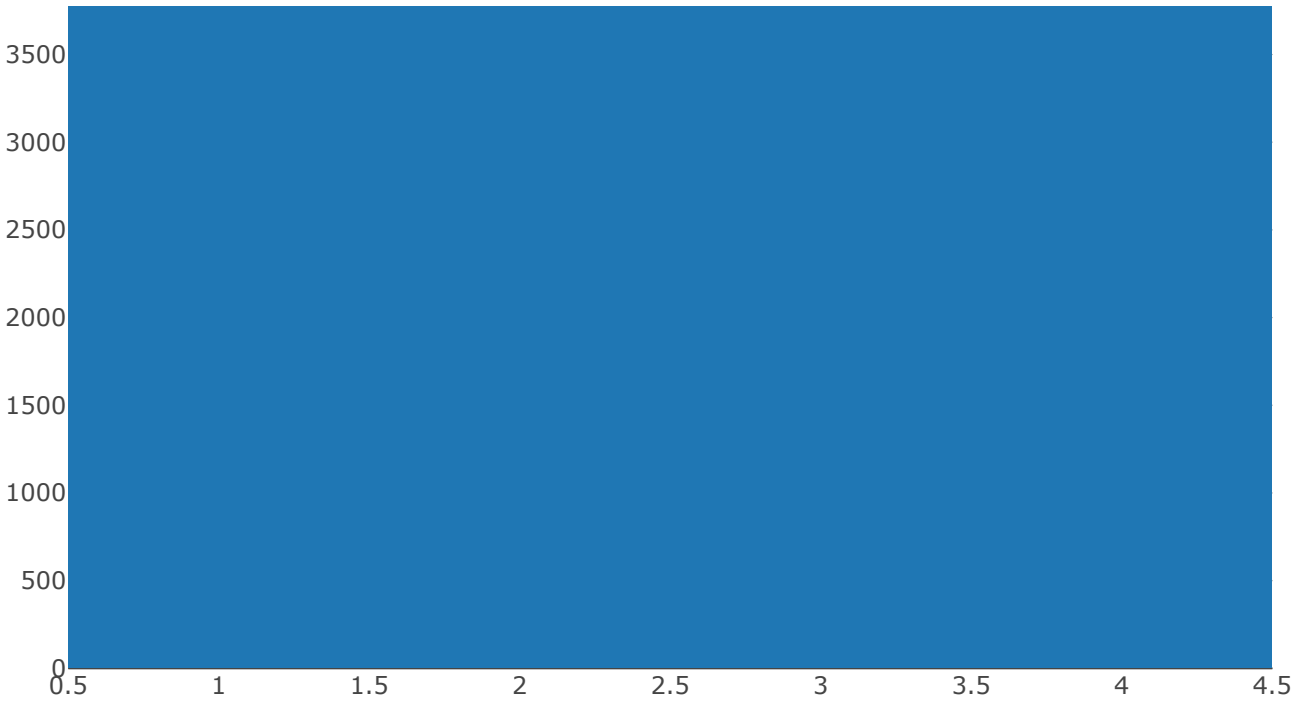
```
var(bike_share_data$hour.temp)
```

```
## [1] 0.03707786
```

As expected the temperature looks normally distributed with low variance. Generally temperature patterns over a day will tend to follow a gaussian distribution.

Season

```
plot_ly(x = bike_share_data$hour.season, type = "histogram")
```




```
mean(bike_share_data$hour.season)
```

```
## [1] 2.50164
```

```
median(bike_share_data$hour.season)
```

```
## [1] 3
```

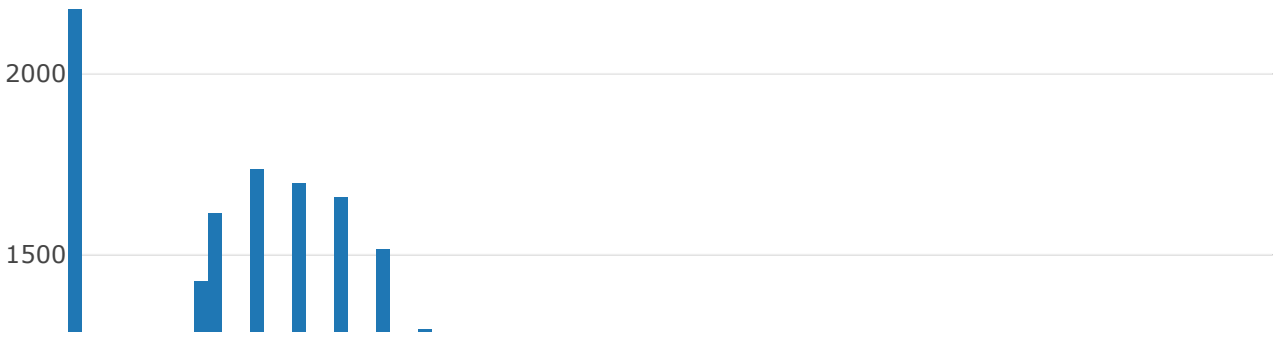
```
var(bike_share_data$hour.season)
```

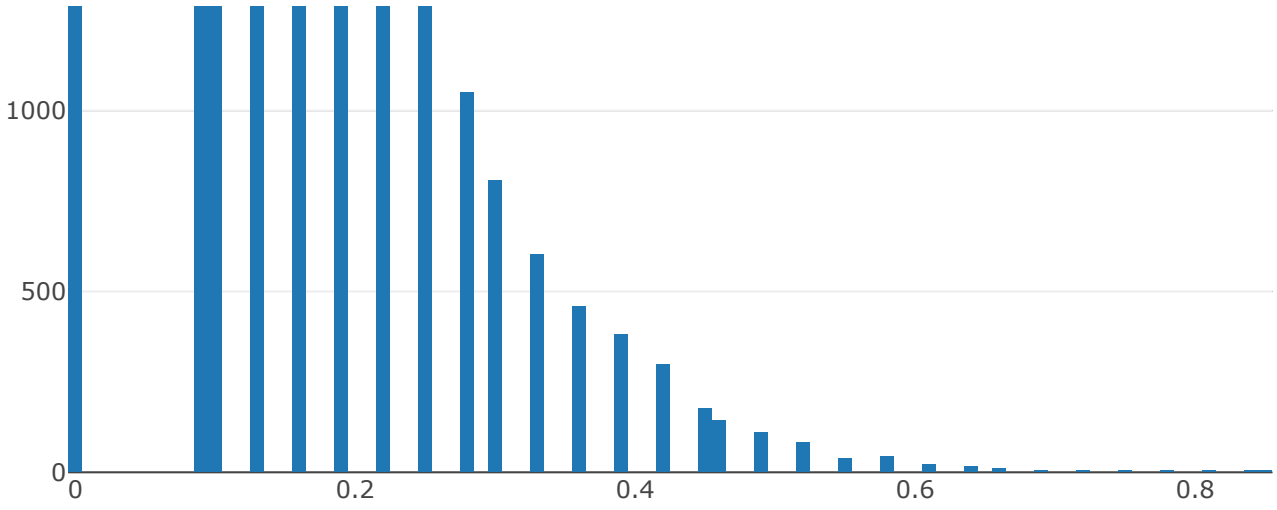
```
## [1] 1.225268
```

Not much needs to be explained here. In the region this data was taken, they have 4 seasons.

Windspeed

```
plot_ly(x = bike_share_data$hour.windspeed, type = "histogram")
```






```
mean(bike_share_data$hour.windspeed)
```

```
## [1] 0.1900976
```

```
median(bike_share_data$hour.windspeed)
```

```
## [1] 0.194
```

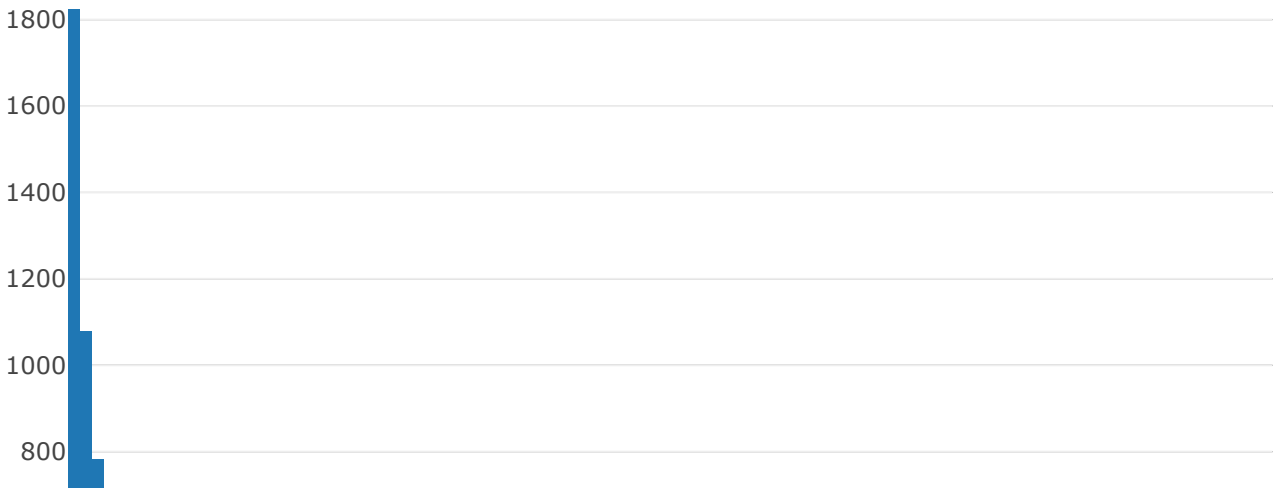
```
var(bike_share_data$hour.windspeed)
```

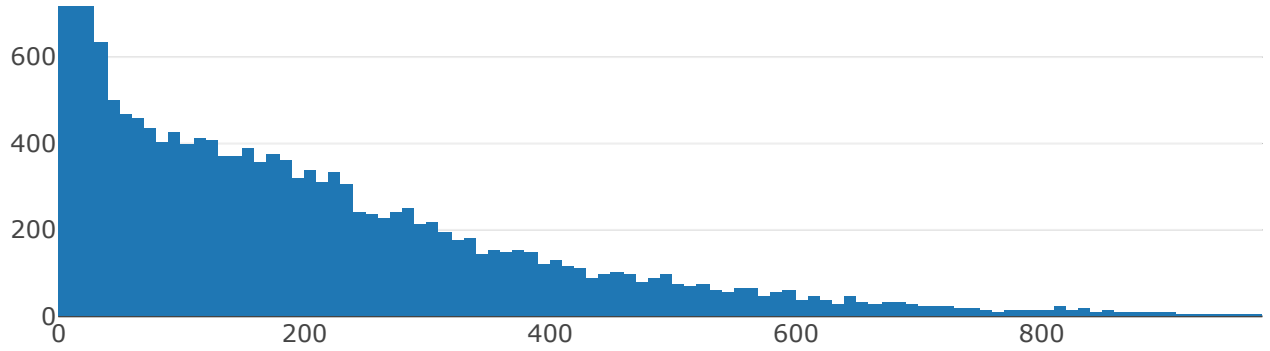
```
## [1] 0.01496713
```

This data is quite interesting. It is heavily skewed to being not too windy. I'm curious to know it's correlation to ridership.

Total Count of Users (Registered and Casual)

```
plot_ly(x = bike_share_data$hour.cnt, type = "histogram")
```






```
mean(bike_share_data$hour.cnt)
```

```
## [1] 189.4631
```

```
median(bike_share_data$hour.cnt)
```

```
## [1] 142
```

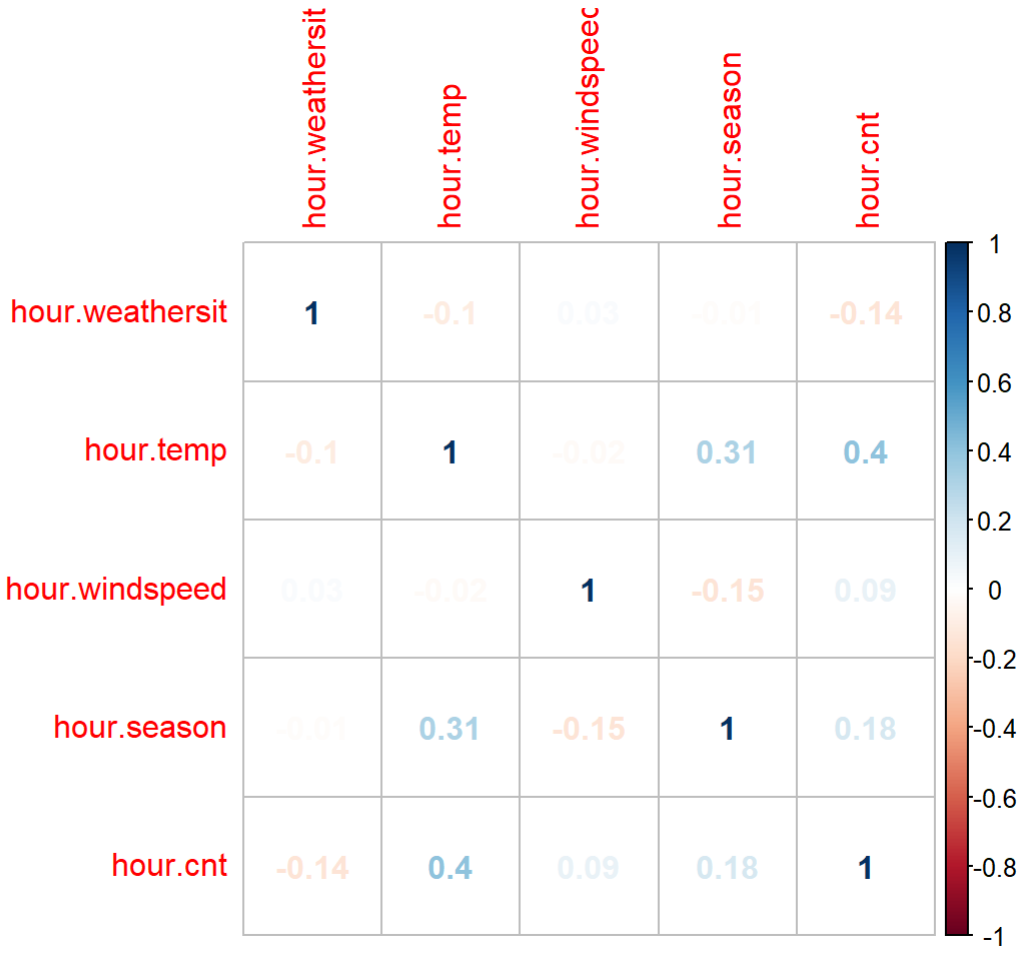
```
var(bike_share_data$hour.cnt)
```

```
## [1] 32901.46
```

This is the total count of riders. There is a much larger skew towards their being less riders in any given hour. I'm considering using this as my output variable.

Plotting the Correlation Matrix:

```
bike_share_matrix <- data.matrix(bike_share_data)
corr_matrix <- cor(bike_share_matrix)
corrplot(corr_matrix, method="number")
```



There is certainly a good correlation (defined as above 0.4/-0.4) between temperature and total count indication that the higher the temperature the more likely people are to be renting bicycles. There is some positive correlation between season and total count as well which is to be expected as it's unlikely people will want to ride bicycles in the cold unless absolutely necessary. There's a moderate correlation between temperature and casual users as well as total count of users. Assuming we can treat weather situation as an ordered variable the correlation between that and other's is weak

For my analysis, I'd like to see if I can make a model that predicts ridership in any given hour based on seasonal and climate variables.

Analysis:

Since there's some strong correlation between some of the variables and the majority can be treated as numeric or non-categorical, I have chosen to apply multivariate linear regression on the data set:

As per eqution (5.35) from the I2ML course text book:

$$r^t = w_0 + w_1 x_1^t + w_2 x_2^t + \ldots + w_d x_d^t + \epsilon$$

The initial step I took was to split the dataset into training and testing data as follows:

```
data_size = length(bike_share_data$hour.weathersit)
train_val = .8*data_size
test_val = train_val + 1

response_var_train <- bike_share_data$hour.cnt[1:train_val]
response_var_test <- bike_share_data$hour.cnt[test_val:data_size]

bike_share_data <- data.frame(rep(1,length(hour$weathersit)),
                              hour$weathersit,
                              hour$temp,
                              hour$windspeed,
                              hour$casual)
bike_share_matrix <- data.matrix(bike_share_data)

bike_share_matrix_train <- bike_share_matrix[1:train_val,]
bike_share_matrix_test <- bike_share_matrix[test_val:data_size,]
```

Now I train the model to determine the weights of the regression. This is based off of the derived formula from the book (5.39):

```
weights = ginv(t(bike_share_matrix_train) %*% bike_share_matrix_train) %*%
          t(bike_share_matrix_train) %*% response_var_train

response_on_training = vector()
for(i in 1:length(response_var_train))
{
  response_on_training[i] = weights[1] +
    bike_share_matrix_train[i,2]*weights[2] +
    bike_share_matrix_train[i,3]*weights[3] +
    bike_share_matrix_train[i,4]*weights[4] +
    bike_share_matrix_train[i,5]*weights[5]
}

#this is the expected error
epsilon = 0
for(i in 1:length(response_var_train))
{
 epsilon = epsilon + response_on_training[i] - weights[1] -
    bike_share_matrix_train[i,2]*weights[2] -
    bike_share_matrix_train[i,3]*weights[3] -
    bike_share_matrix_train[i,4]*weights[4] -
    bike_share_matrix_train[i,5]*weights[5]
}
epsilon = (0.5)*(epsilon^2)
```

An explanation of the vector that is produced. The first value is

$$w_0$$

which is the intercept. The rest of the values are the ordered weights of the variables.

With my weights produced, I can now test my model against the test data:

```
response_test = vector()
for(i in 1:length(response_var_test))
{
  response_test[i] = weights[1] +
    bike_share_matrix_test[i,2]*weights[2] +
    bike_share_matrix_test[i,3]*weights[3] +
    bike_share_matrix_test[i,4]*weights[4] +
    bike_share_matrix_test[i,5]*weights[5] +
    epsilon
}
```

Finally, to test the performance of the model, I will calculate the RMSE and the R-Squared:

```
#RMSE
rmse_of_model = sqrt(sum((response_test-response_var_test)^2))

#R-Squared
ssr = sum((response_test-response_var_test)^2)
ss_total = sum((response_test-mean(response_test))^2)
rsquared = (ss_total - ssr)/ss_total

print("RMSE:")
```

```
## [1] "RMSE:"
```

```
print(rmse_of_model)
```

```
## [1] 10273.4
```

```
print("R-Squared:")
```

```
## [1] "R-Squared:"
```

```
print(rsquared)
```

```
## [1] -0.5999092
```

As can be seen by the above values, it's quite clear that this model performs very poorly. As this model does have an intercept as mentioned above, the RSquared value being negative is an indication of an extremely poor fit to the data.

The RMSE is also very high. As the variance in the resultant output is quite high this is to be expected.

Bank Data Set Analysis

Exploratorary Data Analysis

For the bank data set, I used the larger “bank_full.csv” data set and I chose the following variables:

- age (numeric)
- education (categorical: “tertiary”, “secondary”, “primary”)
- balance: (numeric) clients bank balance
- poutcome: outcome of the previous marketing campaign (categorical: ‘failure’,‘nonexistent’,‘success’)
- y - has the client subscribed a term deposit? (binary: ‘yes’,‘no’)

I begin by preparing the data and selecting my variables:

```
bank_full <- read_delim("C:/Users/rafik/Google Drive/Education/ds8002/ds8002project/bank/bank-fu
ll.csv",
                        ";", escape_double = FALSE, trim_ws = TRUE)
```

```
## Parsed with column specification:
## cols(
##   age = col_integer(),
##   job = col_character(),
##   marital = col_character(),
##   education = col_character(),
##   default = col_character(),
##   balance = col_integer(),
##   housing = col_character(),
##   loan = col_character(),
##   contact = col_character(),
##   day = col_integer(),
##   month = col_character(),
##   duration = col_integer(),
##   campaign = col_integer(),
##   pdays = col_integer(),
##   previous = col_integer(),
##   poutcome = col_character(),
##   y = col_character()
## )
```

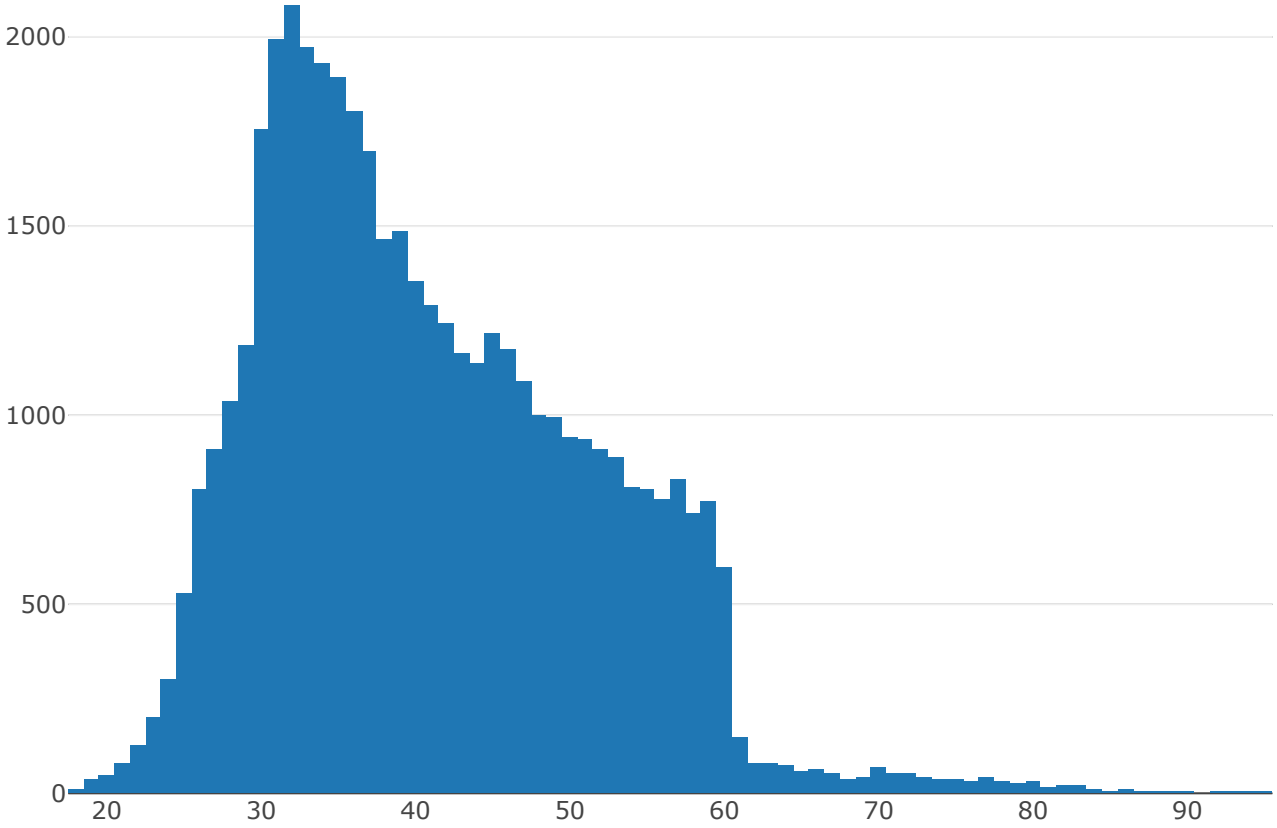
```
#outcome vector indexing
outcome = vector()
for(i in 1:length(bank_full$y))
{
  if(bank_full$y[i] == "yes")
  {
    outcome[i] = 1
  }
  else
  {
    outcome[i] = 0
  }
}

#education vector indexing
education = vector()
for(i in 1:length(bank_full$education))
{
  if(bank_full$education[i] == "tertiary")
  {
    education[i] = 3
  }
  else if (bank_full$education[i] == "secondary")
  {
    education[i] = 2
  }
  else if (bank_full$education[i] == "primary")
  {
    education[i] = 1
  }
  else if (bank_full$education[i] == "unknown")
  {
    education[i] = 0
  }
}

#pick interesting features of bike data
bank_data <- data.frame(bank_full$age,
                        bank_full$balance,
                        education,
                        bank_full$previous,
                        outcome)
```

I converted education into an ordinal variable.

Statistics

Age

```
plot_ly(x = bank_full$age, type = "histogram")
```




```
mean(bank_full$age)
```

```
## [1] 40.93621
```

```
median(bank_full$age)
```

```
## [1] 39
```

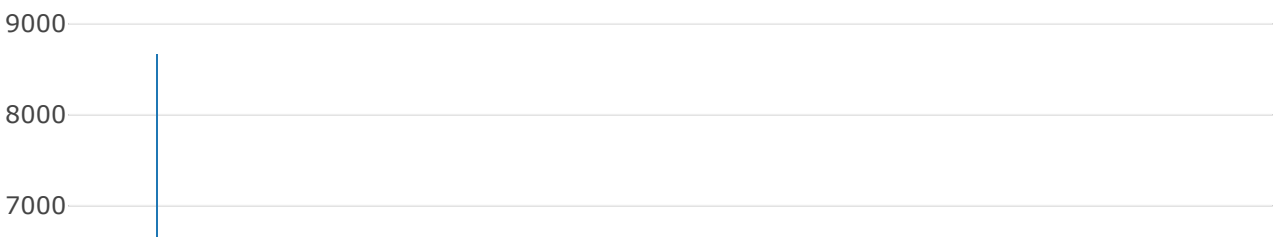
```
var(bank_full$age)
```

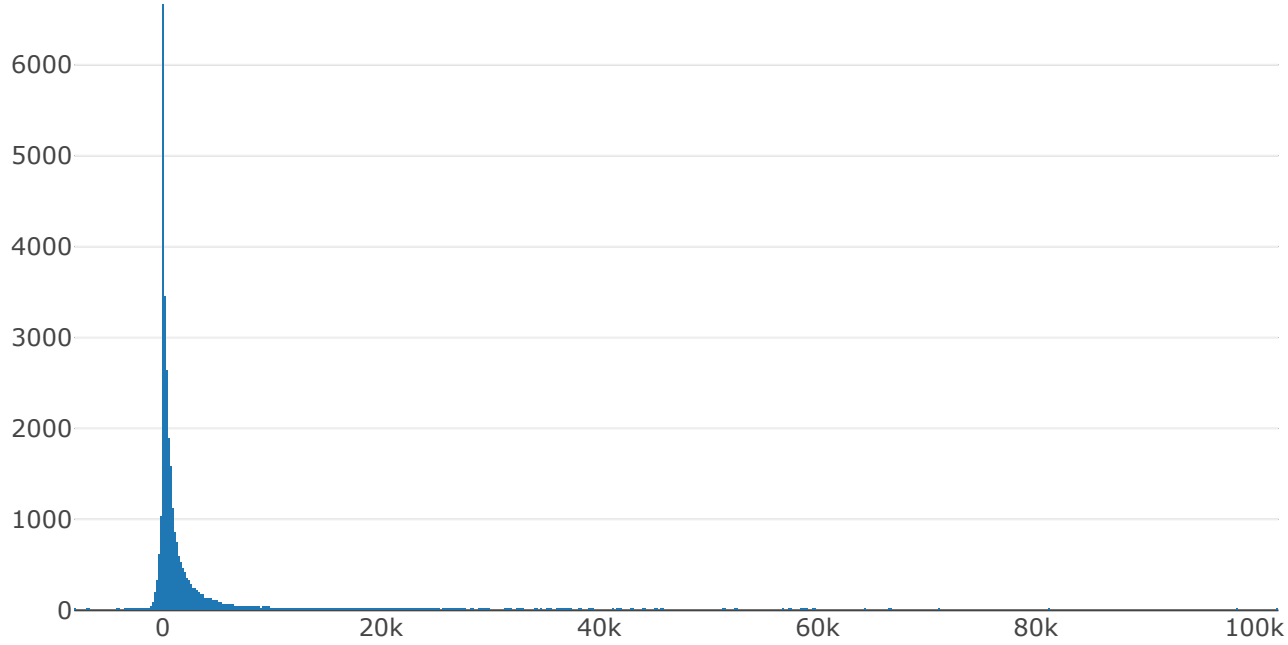
```
## [1] 112.7581
```

Age looks to be normally distributed and that's not too surprising.

Bank Balance

```
plot_ly(x = bank_full$balance, type = "histogram")
```






```
mean(bank_full$balance)
```

```
## [1] 1362.272
```

```
median(bank_full$balance)
```

```
## [1] 448
```

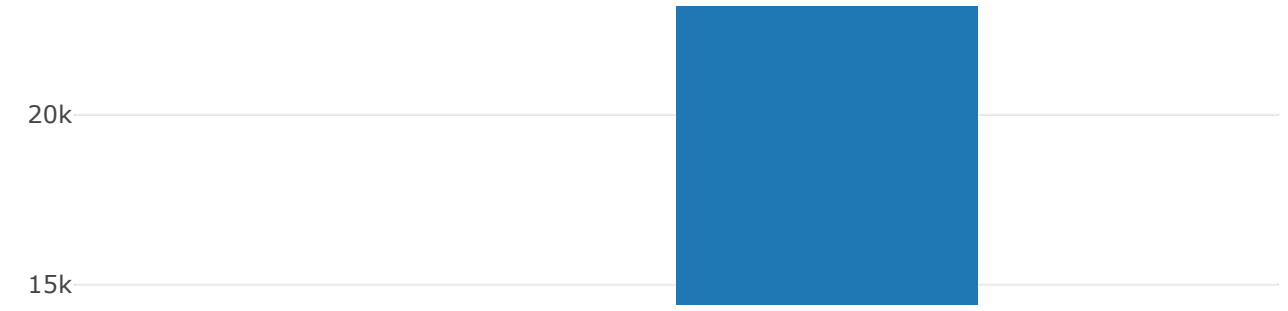
```
var(bank_full$balance)
```

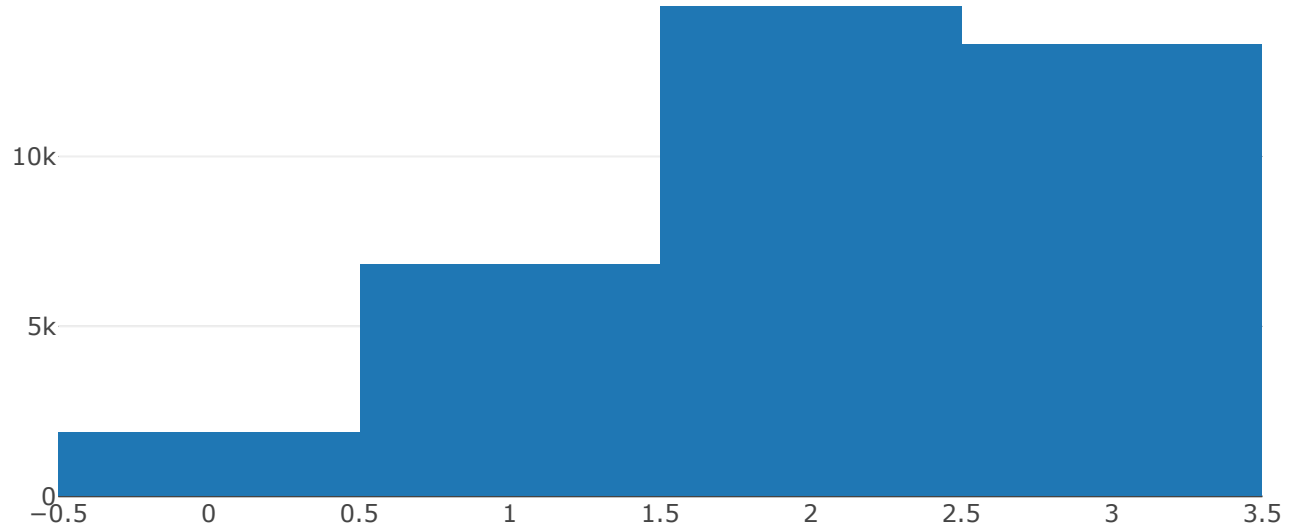
```
## [1] 9270599
```

There is very high variance in the bank balance data which is quite interesting. I'd expect that this variable might have a tremendous influence on the outcome of a lot of things a person does. My assumption is more money means more flexibility to take out loans etc.

Eduation

```
plot_ly(x = education, type = "histogram")
```






```
mean(education)
```

```
## [1] 2.060516
```

```
median(education)
```

```
## [1] 2
```

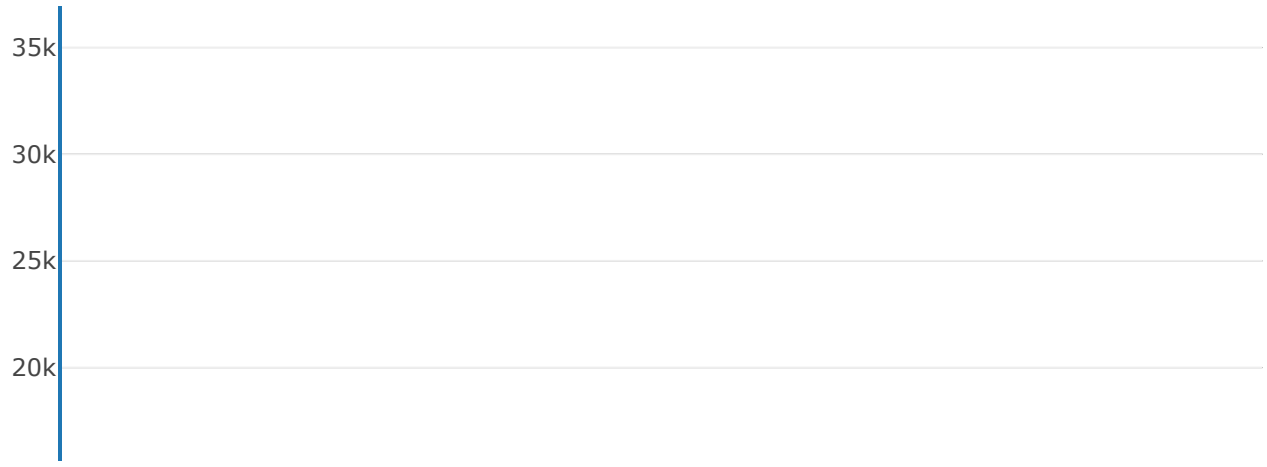
```
var(education)
```

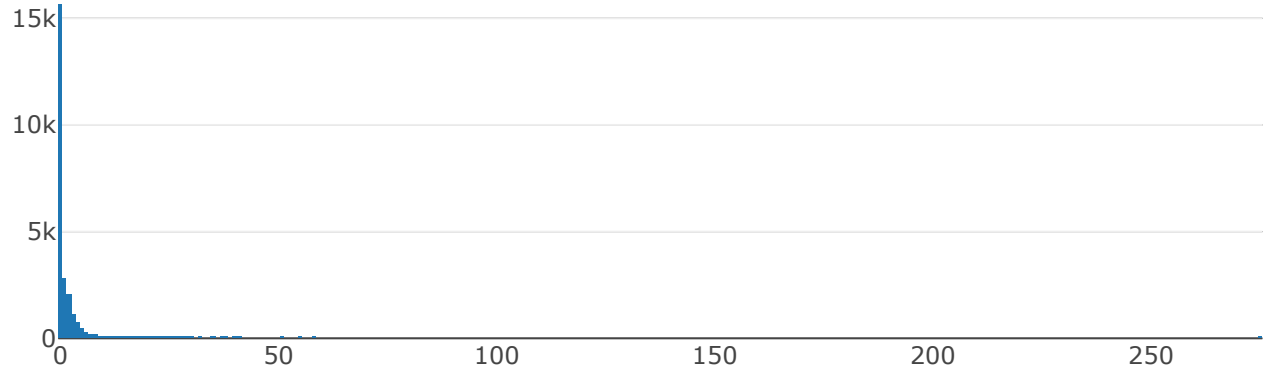
```
## [1] 0.6063797
```

We can safely assume that this data is normal with low variance. I'd expect there to be a high correlation between more education and bank balance.

Previous Outcome of prior campaign

```
plot_ly(x = bank_full$previous, type = "histogram")
```






```
mean(bank_full$previous)
```

```
## [1] 0.5803234
```

```
median(bank_full$previous)
```

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

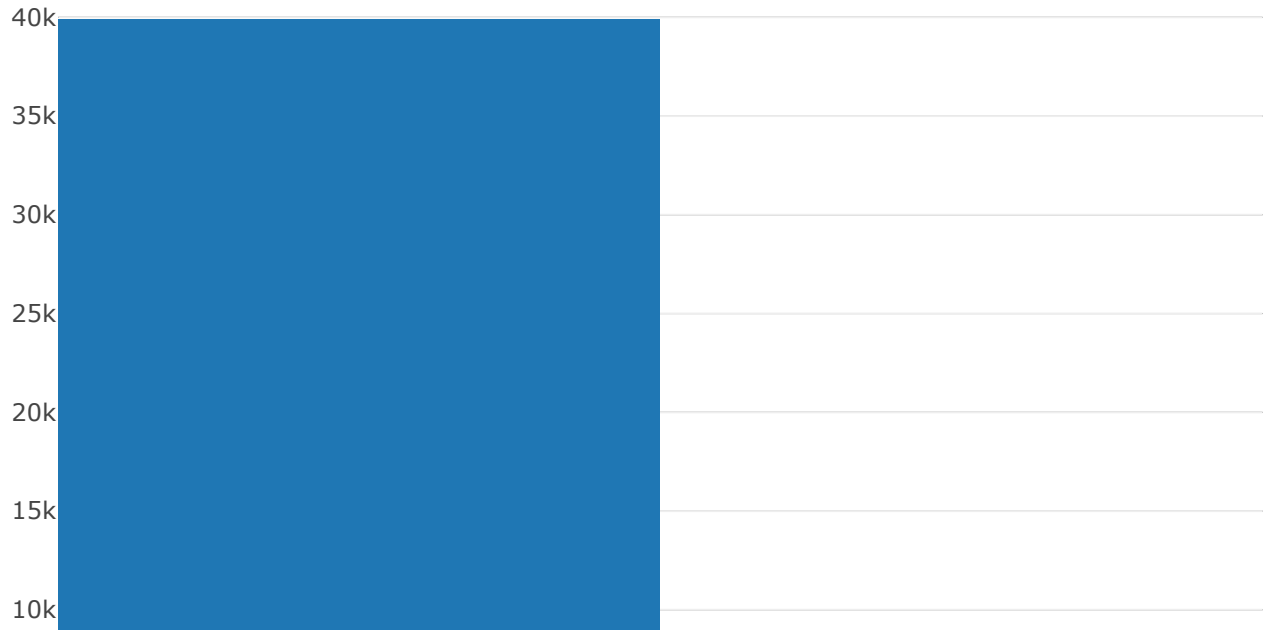
```
var(bank_full$previous)
```

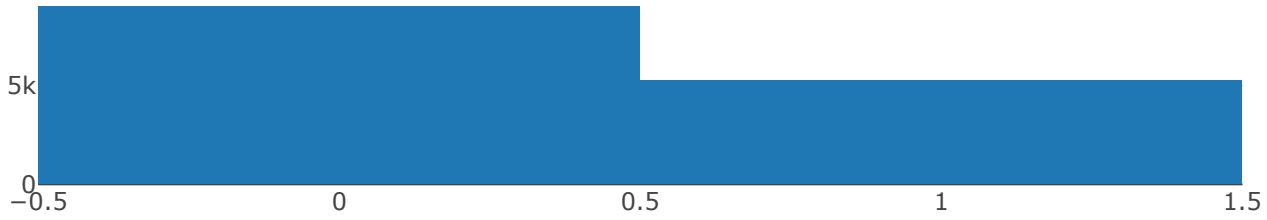
```
## [1] 5.305841
```

Is it possible that the previous outcome is highly correlated to current outcome? In general the data is skewed to a negative outcome. The mean is a meaningless metric here. I would say the mode is much more useful.

Outcome of current campaign

```
plot_ly(x = outcome, type = "histogram")
```






```
mean(outcome) #meaningless metric here
```

```
## [1] 0.1169848
```

```
median(outcome)
```

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

```
var(outcome) #meaningless metric here
```

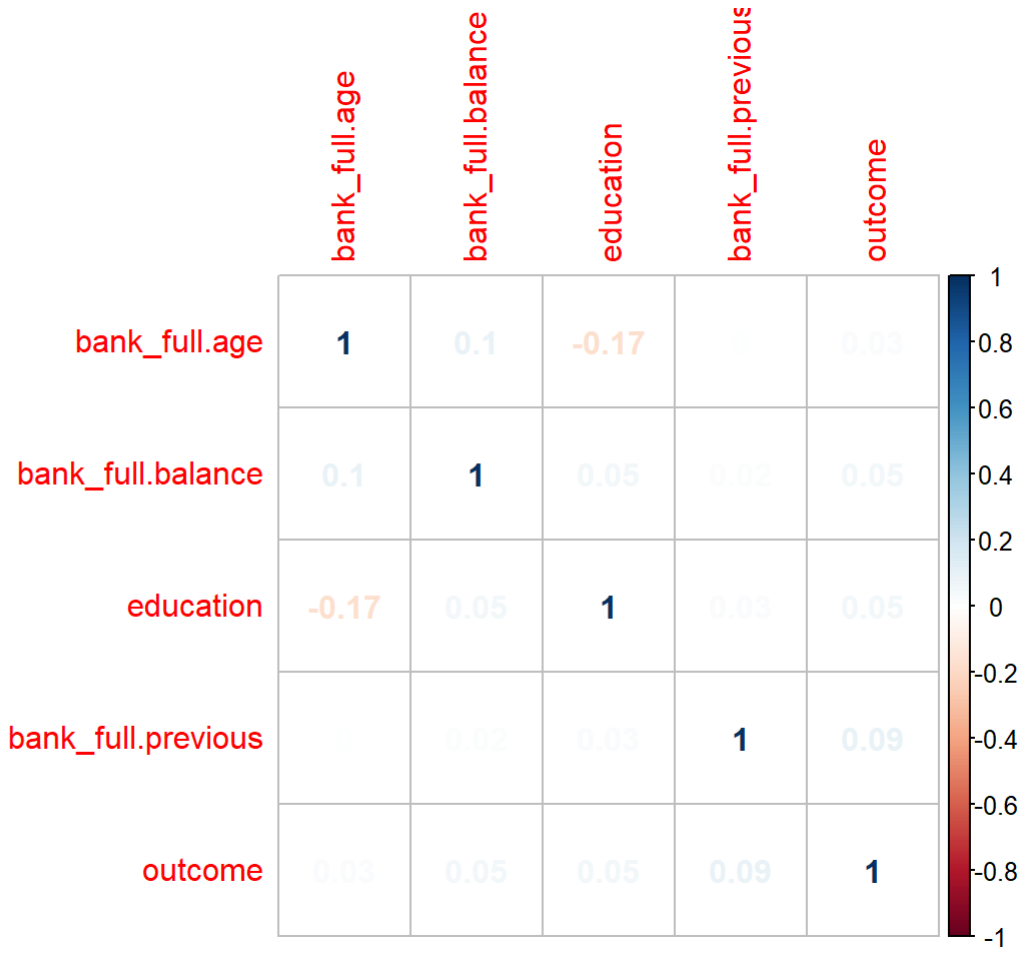
```
## [1] 0.1033016
```

Similarly to previous outcome, the mean is meaningless and mode would be more useful. Nevertheless I'd expect there to be a high correlation between previous outcome and current outcome.

Plotting the Correlation Matrix:

some notes here, the correlation is potentially meaningless as this is a class based problem unless we are looking to create a prediction value that gives a probability of yes/no

```
corr_matrix <- cor(bank_data)
corrplot(corr_matrix,method="number")
```



As expected, the strongest positive correlation between any two variables comes from the age and bank balance. Also a very strong negative correlation between age and education, which might mean that the older generation involved in this campaign tended to not pursue tertiary education. There is also a somewhat moderately positive correlation between the previous campaign outcome and the current one. Some conclusions that can be drawn here are that, generally older people will tend to have more money and less education.

Overall, I'd say these variables aren't strongly correlated which might impact the performance of the model.

Analysis:

I will follow the original intention of the data set and use the Outcome variable as my response variable. As the outcome can be easily classified into two classes (positive/negative outcome), I can apply multivariate classification to the data.

I will make the same assumption about the data as mentioned in the book on pg. 100, such that generally all the variables follow an overall normal distrubtion.

Assuming the likelihoods of the data are as follows:

$$p(x|C_i) \sim N_d(\mu_i, \Sigma_i)$$

Then I can use the discriminant function from equation (5.20):

$$g_i(x) = x^T W_i x + w_i^T x + w_0 i$$

The definitions of the individual terms are available on pg. 101 of the I2ML text. I am using the estimators for mean, covariance matricies and prior probabilities.

I will be solving a quadratic discriminant function seeking the maximum likelihood of a given class. With this in mind I can begin the analysis. The first step as before, is to prepare the data and split it into training and test datasets.

```
#data prep
data_size = length(bank_data$outcome)
train_val = .8*data_size
test_val = train_val + 1

#without response variable
bank_data_matrix = matrix(c(bank_data$bank_full.age,
                            bank_data$bank_full.balance,
                            bank_data$education,
                            bank_data$bank_full.previous),nrow = data_size, ncol = 4)
bank_data_matrix_train = bank_data_matrix[1:train_val,]
bank_data_matrix_test = bank_data_matrix[test_val:data_size,]

#with response variable for some calculations
bank_data_with_outcome = data.matrix(bank_data)
bank_data_train = bank_data_with_outcome[1:train_val,]
bank_data_test = bank_data_with_outcome[test_val:data_size,]
```

Here, I created some basic R functions to allow me to estimate the parameters based on the derived formulas given for esimating the covariance matricies, the class means and the priors.

```
#create estimation functions
cov_custom <- function(x,means)
{
  row_size = length(x[,1])
  column_size = length(x[1,])
  output_matrix = matrix(0,column_size,column_size)
  for(i in 1:row_size)
  {
    output_matrix = output_matrix + (x[i,]-means)%*%t(x[i,]-means)
  }
  output_matrix = output_matrix/row_size
  return(output_matrix)
}

mu_custom <- function(x)
{
  row_size = length(x[,1])
  column_size = length(x[1,])
  mu = vector()
  for(i in 1:column_size)
  {
    mu = c(mu,mean(x[,i]))
  }
  return(mu)
}

priors_custom <- function(x,output_variable)
{
  classes = unique(output_variable)
  row_size = length(x[,1])

  priors_vector = vector()
  for(i in 1:length(classes))
  {
    class = subset(x,output_variable == classes[i])
    class_size = length(class[,1])
    prior = class_size/row_size
    priors_vector = c(priors_vector,prior)
  }
  return(priors_vector)
}
```

Now for the actual analysis. I calculated the number of classes (2), the priors, the

$$\mu$$

for each class and finally the

$$S_i$$

for each class.

```
#determine # of classes in the data
classes = unique(outcome)

#find the priors #find the mean
priors = priors_custom(bank_data_train,bank_data_train[,5])

#find the covariance matrix and the means
s_matrixes = list()
mu = matrix(0,2,4)
for(i in 1:length(classes))
{
   data_subset = subset(bank_data_train,bank_data_train[,5] == classes[i])
   mu[i,] = mu_custom(data_subset[,1:4])
   s_matrixes[[i]] = cov_custom(data_subset[,1:4],mu[i,])
}
```

With my paramters estimated, I implemented the Quadratic Discriminant function from equation (5.20) in the book.

```
#implement qda
discriminant_model <- function(x,mu,s,priors)
{
  discriminants = matrix(0,length(x[,1]),length(priors)+1)
  for(i in 1:length(x[,1]))
  {
    for(j in 1:length(priors))
    {
      wio = (-0.5) * (t(mu[j,])%*%ginv(s[[j]])%*%mu[j,]) - (0.5 * log(det(s[[j]]))) + log(priors
[j])
      w = ginv(s[[j]]) %*% mu[j,]
      Wi = (-0.5) * ginv(s[[j]])
      discriminants[i,j] =  t(x[i,])%*%Wi%*%x[i,] + t(w)%*%x[i,] + wio
    }
    if(discriminants[i,1] > discriminants [i,2])
    {
      discriminants[i,3] = 0
    }
    else
    {
      discriminants[i,3] = 1
    }
  }

  return(discriminants)
}

result = discriminant_model(bank_data_matrix_test,mu,s_matrixes,priors)
```

Now to look at the performance of the model:

```
actual_negative = sum(subset(bank_data$outcome[test_val:data_size],bank_data$outcome==0))
actual_positive = sum(subset(bank_data$outcome[test_val:data_size],bank_data$outcome==1))

#stats on accuracy of prediction

ppp = 0
ppn = 0
npp = 0
npn = 0


for(i in 1:length(bank_data_test[,5]))
{
  if(result[i,3] == bank_data_test[i,5])
  {
    if(bank_data$outcome[i] == 1)
    {
      ppp = ppp + 1
    }
    else
    {
      npn = npn + 1
    }
  }

  if(result[i,3] != bank_data_test[i,5])
  {
    if(bank_data$outcome[i] == 1)
    {
      ppn = ppn + 1
    }
    else
    {
      npp = npp + 1
    }
  }
}

predicted_positive = c(ppp,ppn)
predicted_negative = c(npp,npn)
results_df = data.frame(predicted_positive,predicted_negative)
row.names(results_df) <- c("actual positive", "actual negative")
results_df
```

```
##                 predicted_positive predicted_negative
## actual positive                193               2750
## actual negative                111               5988
```

As we can see from above, the model is not performing well.

The accuracy is 68.35%. The precision is 63.4% for Postive Predicted Postive which is quite low.

I attribute this to the generally low correlation between the variables.

Finally, I applied regularized discriminant analysis to determine if the model might be overfit and approaching linearized discriminant analysis might be more appropriate here:

```
alpha_vector = vector()
accuracy_vector = vector()
precision_vector = vector()

#rda
for(alpha in seq(0,1,by=.1))
{
  s = matrix(0,4,4)
  for(i in 1:length(s_matrixes))
  {
    s = s + s_matrixes[[i]]
  }

  s_matricies = list()
  for(i in 1:length(s_matrixes))
  {
    s_matricies[[i]] = alpha * s_matrixes[[i]] + (1-alpha) * s
  }

  output = discriminant_model(bank_data_matrix_test,mu,s_matricies,priors)

  #stats on accuracy of prediction

  ppp = 0
  ppn = 0
  npp = 0
  npn = 0


  for(i in 1:length(bank_data_test[,5]))
  {
    if(output[i,3] == bank_data_test[i,5])
    {
      if(bank_data$outcome[i] == 1)
      {
        ppp = ppp + 1
      }
      else
      {
        npn = npn + 1
      }
    }

    if(output[i,3] != bank_data_test[i,5])
    {
      if(bank_data$outcome[i] == 1)
      {
        ppn = ppn + 1
      }
      else
      {
        npp = npp + 1
      }
```

```
    }
  }

  alpha_vector = c(alpha_vector,alpha)
  accuracy = (npn+ppp)/(ppp+ppn+npp+npn)
  accuracy_vector = c(accuracy_vector,accuracy)
  precision = (ppp/(ppp+ppn))
  precision_vector = c(precision_vector,precision)

}

alpha_df = data.frame(alpha_vector,accuracy_vector,precision_vector)
alpha_df
```

```
##    alpha_vector accuracy_vector precision_vector
## 1           0.0       0.6841407        0.6414474
## 2           0.1       0.6841407        0.6414474
## 3           0.2       0.6841407        0.6414474
## 4           0.3       0.6841407        0.6414474
## 5           0.4       0.6841407        0.6414474
## 6           0.5       0.6841407        0.6414474
## 7           0.6       0.6841407        0.6414474
## 8           0.7       0.6841407        0.6414474
## 9           0.8       0.6841407        0.6414474
## 10          0.9       0.6840301        0.6414474
## 11          1.0       0.6835877        0.6348684
```

It seems that it made the model perform worse. Therefore a quadratic fit is fine. What is likely happening is that since the correlation of the data is weak, the model is doing a poor job of prediction.