CMM703 - Data Analysis Coursework

Kaneel Dias

2025-04-14

```
suppressWarnings(suppressMessages({
  library(ggplot2)
  require(gridExtra)
  library(glue)
  library(ggcorrplot)
  library(vcd)
  library(tidyr)
  library(dplyr)
  library(pheatmap)
  library(caTools)
  library(pROC)
  library(shiny)
}))
```

TASK 1: CANDY DATASET

The objective of this exercise would be to determine the effect that the variables contained within the dataset have on the target value `winpercent`. All visualizations included in this report will be made with that objective in mind.

```
candy_data <- read.csv("~/CMM703/candy-data.csv")
```

1.1 Effect of the categorical variables

First we should convert all the categorical columns, from numeric to factor.

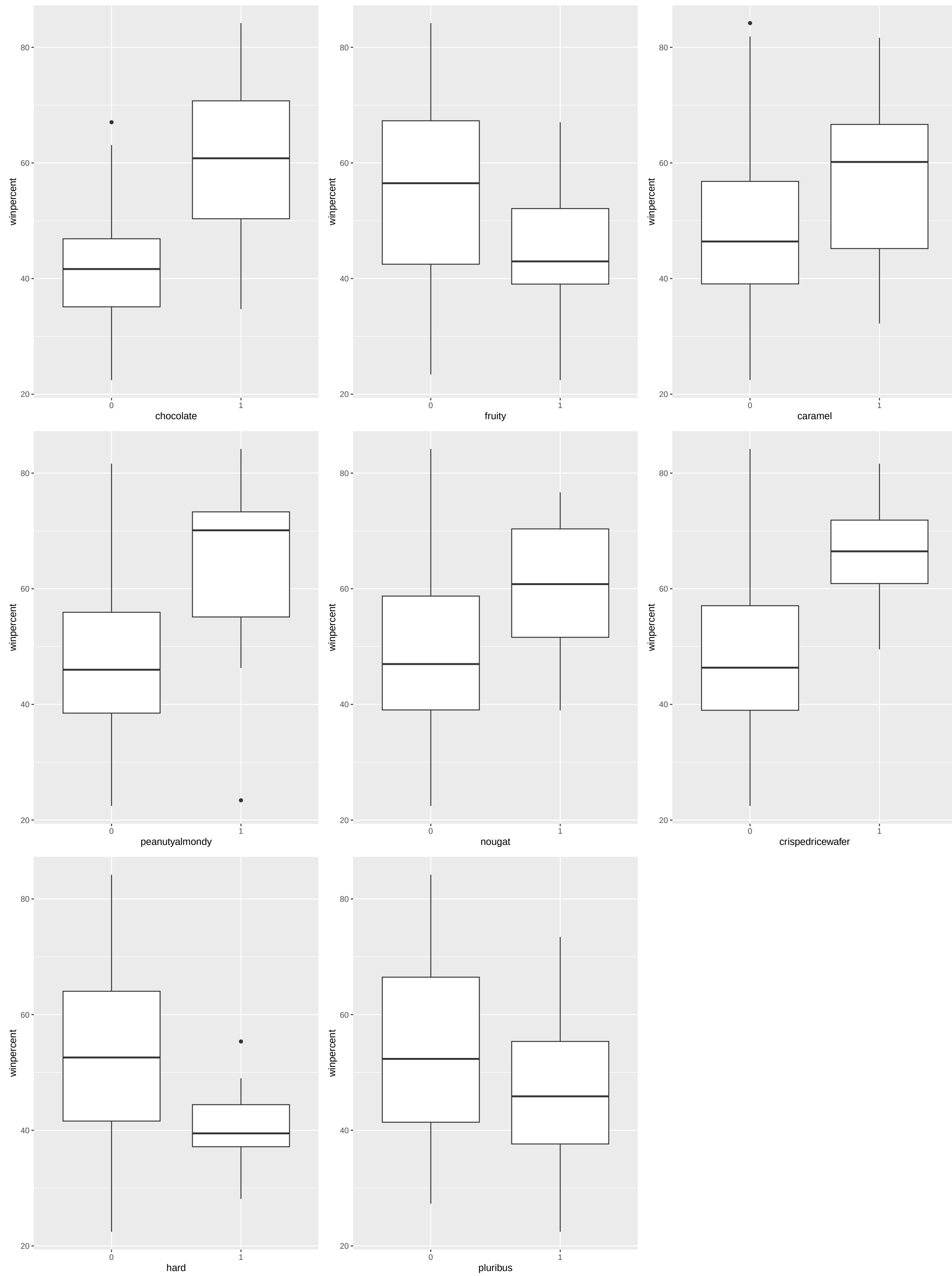
```
candy_data$chocolate <- as.factor(candy_data$chocolate)
candy_data$fruity <- as.factor(candy_data$fruity)
candy_data$caramel <- as.factor(candy_data$caramel)
candy_data$peanutyalmondy <- as.factor(candy_data$peanutyalmondy)
candy_data$nougat <- as.factor(candy_data$nougat)
candy_data$crispedricewafer <- as.factor(candy_data$crispedricewafer)
candy_data$hard <- as.factor(candy_data$hard)
candy_data$bar <- as.factor(candy_data$bar)
candy_data$pluribus <- as.factor(candy_data$pluribus)
```

Next, we can plot boxplots for each categorical variable, and the effect it has on the winning percentage.

```
plot_categorical_boxplot <- function(data, variable) {
  plot <- ggplot(data=data, aes(x=data[,variable], y=winpercent)) +
    xlab(variable) +
    geom_boxplot()
```

```r
  return(plot)
}

plot_all_categorical_boxplots <- function(data) {
  chocolate_plot <- plot_categorical_boxplot(data, "chocolate")
  fruity_plot <- plot_categorical_boxplot(data, "fruity")
  caramel_plot <- plot_categorical_boxplot(data, "caramel")
  peanutyalmondy_plot <- plot_categorical_boxplot(data, "peanutyalmondy")
  nougat_plot <- plot_categorical_boxplot(data, "nougat")
  crispedricewafer_plot <- plot_categorical_boxplot(data, "crispedricewafer")
  hard_plot <- plot_categorical_boxplot(data, "hard")
  pluribus_plot <- plot_categorical_boxplot(data, "pluribus")


  grid.arrange(chocolate_plot, fruity_plot, caramel_plot, peanutyalmondy_plot, nougat_plot, crispedrice
}
```

```r
plot_all_categorical_boxplots(candy_data)
```


winpercent
chocolate
fruity
caramel
peanutyalmondy
nougat
crispedricewafer
hard
pluribus
20
40
60
80
0
1

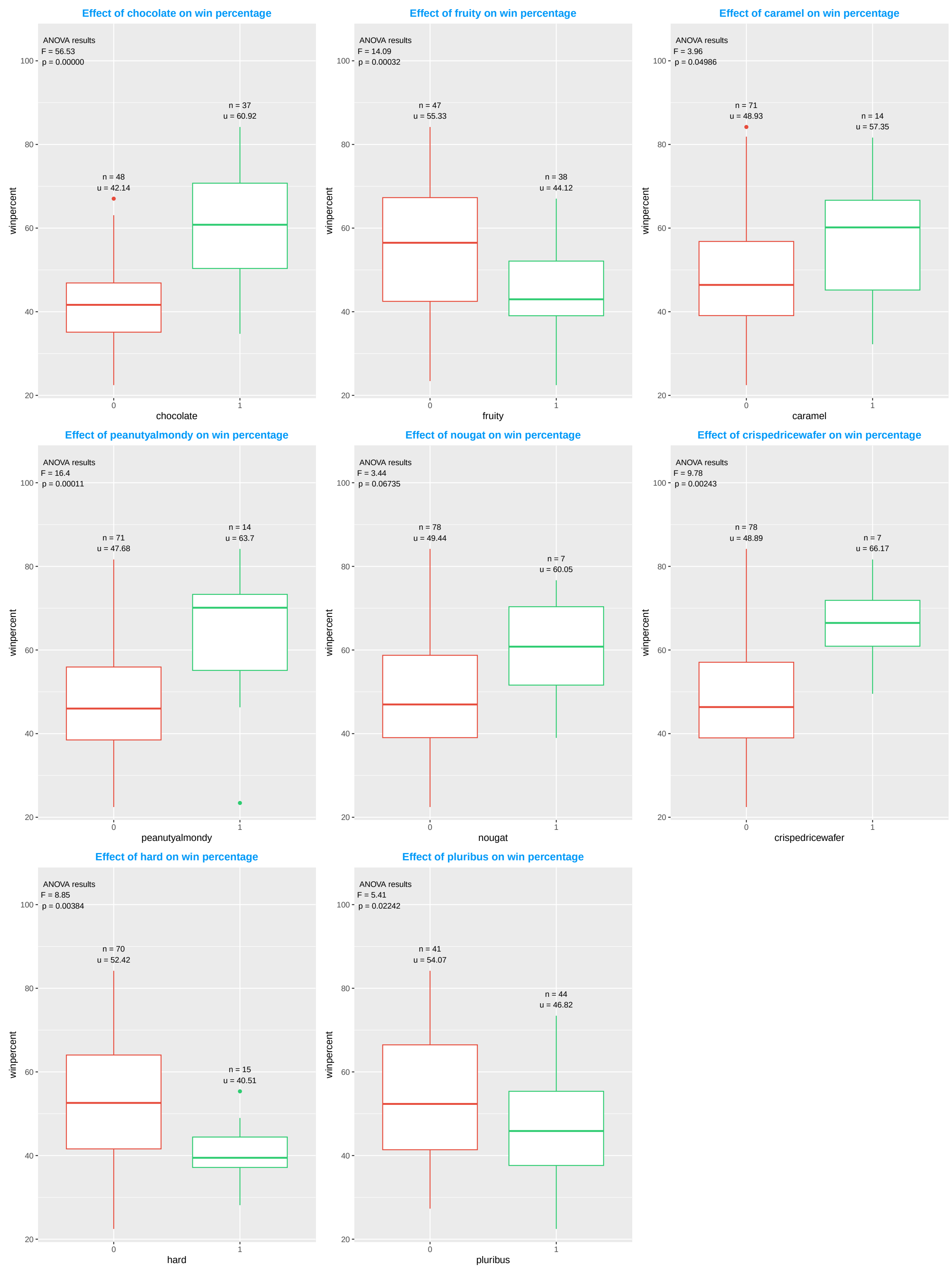
1.1.1 Potential improvements

We can suggest the following improvements
- Visually separate the `contains (1)` and `does not contain (0)` plots using colours
- Indicate for each boxplot, the
- Count of records with that value
- The mean of the `winpercent` value
- Indicate the effect that variable has on the `winpercent` using ANOVA
- Include the F value
- Include the P value

```r
get_summary_stats <- function(y) {
  bxp_stats <- boxplot.stats(y)
  upper_whisker <- bxp_stats$stats[5]
  max_val <- max(c(upper_whisker, bxp_stats$out), na.rm = TRUE)

  n <- length(y)
  q1 <- quantile(y, 0.25, na.rm = TRUE, names = FALSE)
  avg <- mean(y, na.rm = TRUE)
  q3 <- quantile(y, 0.75, na.rm = TRUE, names = FALSE)

  label_str <- paste(
    paste("n =", n),
    paste("u =", round(avg, 2)),
    sep = "\n"
  )

  return(data.frame(
    y = max_val,
    label = label_str
  ))
}

plot_categorical_boxplot_improved <- function(data, variable) {
  anova <- aov(reformulate(variable, "winpercent"), data=data)
  p_val <- summary(anova)[[1]][["Pr(>F)"]][1]
  f_val <- summary(anova)[[1]][["F value"]][1]
  anova_text <- glue("ANOVA results\nF = {round(f_val, 2)}\np = {format(round(p_val, 5), nsmall = 5)}")

  plot <- ggplot(data=data, aes(x=data[,variable], y=winpercent, col=data[,variable])) +
    labs(
      title = glue('Effect of {variable} on win percentage'),
      x = variable
    ) +
    geom_boxplot() +
    scale_color_manual(values = c("#e74c3c", "#2ecc71")) +
    theme(
      legend.position="none",
      plot.title = element_text(color = "#0099f8", size = 12, face = "bold", hjust = 0.5),
    )

  plot <- plot + stat_summary(
    fun.data = get_summary_stats,
    geom = "text",
    hjust = 0.5,
```

```r
    vjust = -0.5,
    size = 3,
    color = "black"
  )

  plot <- plot + annotate(
    geom = "text",
    x = -Inf,
    y = Inf,
    label = anova_text,
    hjust = -0.1,
    vjust = 1.5,
    size = 3,
    color = "black"
  ) +
    scale_y_continuous(expand = expansion(mult = c(0.05, 0.4))) # More space at top


  return(plot)
}

plot_all_categorical_boxplots_improved <- function(data) {
  chocolate_plot <- plot_categorical_boxplot_improved(data, "chocolate")
  fruity_plot <- plot_categorical_boxplot_improved(data, "fruity")
  caramel_plot <- plot_categorical_boxplot_improved(data, "caramel")
  peanutyalmondy_plot <- plot_categorical_boxplot_improved(data, "peanutyalmondy")
  nougat_plot <- plot_categorical_boxplot_improved(data, "nougat")
  crispedricewafer_plot <- plot_categorical_boxplot_improved(data, "crispedricewafer")
  hard_plot <- plot_categorical_boxplot_improved(data, "hard")
  pluribus_plot <- plot_categorical_boxplot_improved(data, "pluribus")


  grid.arrange(chocolate_plot, fruity_plot, caramel_plot, peanutyalmondy_plot, nougat_plot, crispedrice
}
```

```r
plot_all_categorical_boxplots_improved(candy_data)
```



Effect of chocolate on win percentage
ANOVA results
F = 56.53
p = 0.00000
n = 48
u = 42.14
n = 37
u = 60.92
winpercent
chocolate
Effect of fruity on win percentage
ANOVA results
F = 14.09
p = 0.00032
n = 47
u = 55.33
n = 38
u = 44.12
winpercent
fruity
Effect of caramel on win percentage
ANOVA results
F = 3.96
p = 0.04986
n = 71
u = 48.93
n = 14
u = 57.35
winpercent
caramel
Effect of peanutyalmondy on win percentage
ANOVA results
F = 16.4
p = 0.00011
n = 71
u = 47.68
n = 14
u = 63.7
winpercent
peanutyalmondy
Effect of nougat on win percentage
ANOVA results
F = 3.44
p = 0.06735
n = 78
u = 49.44
n = 7
u = 60.05
winpercent
nougat
Effect of crispedricewafer on win percentage
ANOVA results
F = 9.78
p = 0.00243
n = 78
u = 48.89
n = 7
u = 66.17
winpercent
crispedricewafer
Effect of hard on win percentage
ANOVA results
F = 8.85
p = 0.00384
n = 70
u = 52.42
n = 15
u = 40.51
winpercent
hard
Effect of pluribus on win percentage
ANOVA results
F = 5.41
p = 0.02242
n = 41
u = 54.07
n = 44
u = 46.82
winpercent
pluribus

1.1.2 Insights

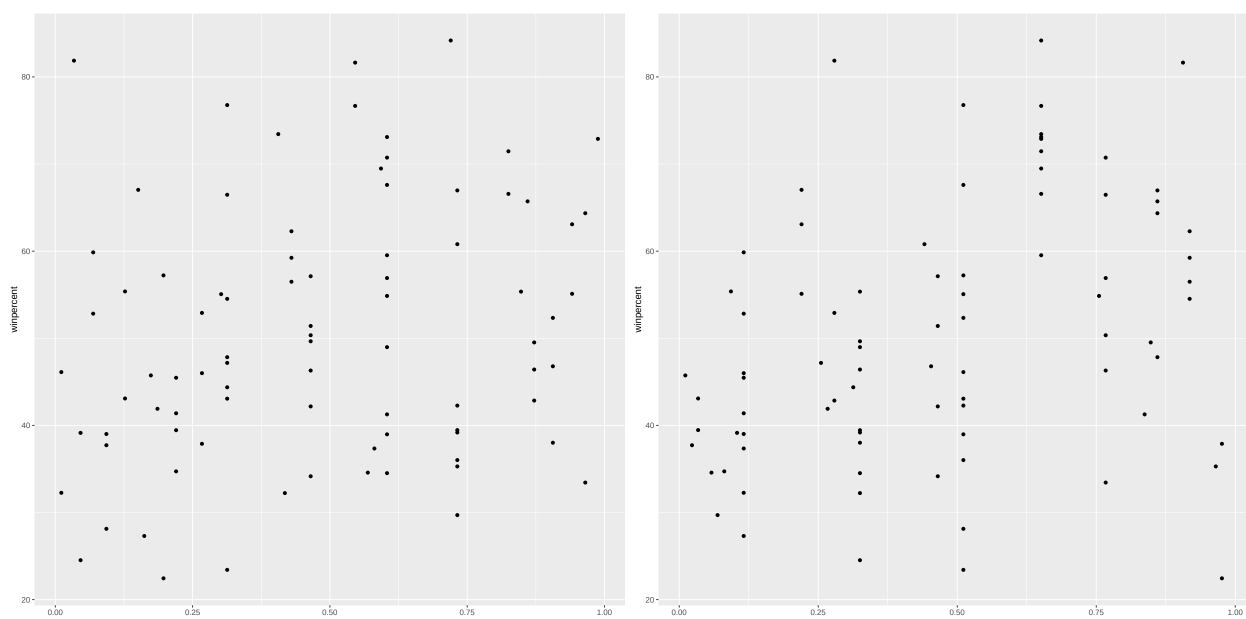
Here we can see that the `chocolate` variable has the highest effect on `winpercent` (highest f-value), and it has a very low p-value as well, indicating that it is most likely to be having an effect. On the other hand, `nougat` has a p-value > 0.05, (as well as a low f-value) which indicates that its effect on the winning percentage is not likely.

1.2 Effect of the numeric variables

There are two numeric, continuous variables: `sugarpercent` and `pricepercent` We can visualize their effect on `winpercent` using scatter plots.

```
plot_numeric_scatterplot <- function(data, variable) {
  plot <- ggplot(data=candy_data, aes(x=data[,variable], y=winpercent)) +
    labs(
      x = variable
    ) +
    geom_point()

  return (plot)
}

plot_all_numerical_scatterplots <- function(data) {
  sugar_plot <- plot_numeric_scatterplot(data, "sugarpercent")
  price_plot <- plot_numeric_scatterplot(data, "pricepercent")



  grid.arrange(sugar_plot, price_plot, nrow = 1)
}
```

```
plot_all_numerical_scatterplots(candy_data)
```




```
get_r2 <- function (x, y) cor(x, y) ^ 2
```

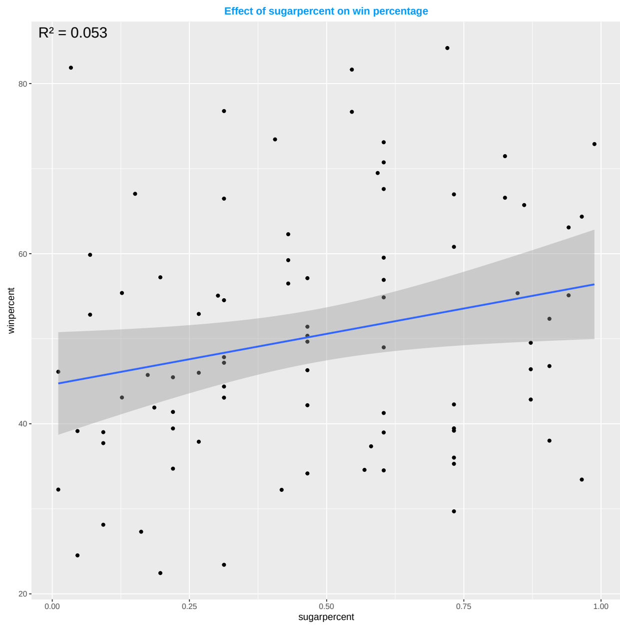
1.2.1 Potential Improvements

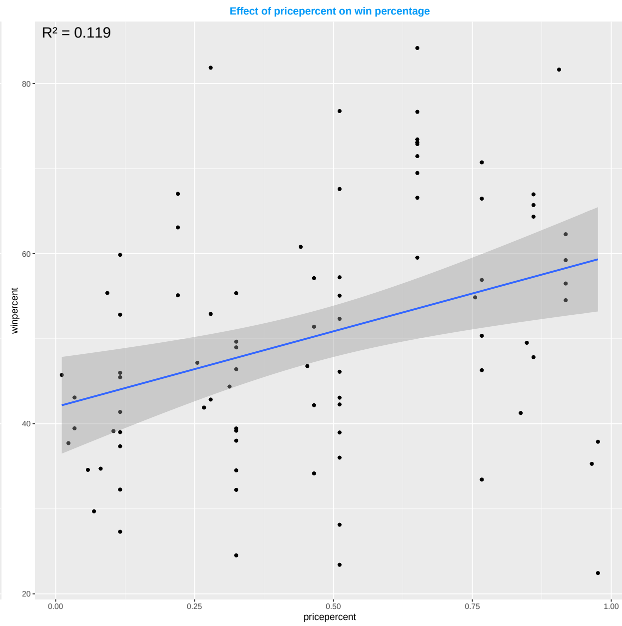
We can suggest the following improvements:
- Add a regression line to be able to view the relationship between the two variables and the winning percentage
- Include the R^2 value (coeffiecient of determination in the chart) to determine whether the they correlate

```
plot_numeric_scatterplot_improved <- function(data, variable) {
  r2 <- get_r2(data[,variable], data$winpercent)
  r2_text <- glue("R² = {format(round(r2, 3), nsmall = 3)}")


  plot <- ggplot(data=candy_data, aes(x=data[,variable], y=winpercent)) +
    labs(
      title = glue('Effect of {variable} on win percentage'),
      x = variable
    ) +
    geom_point() +
    geom_smooth(method=lm) +
    theme(
      legend.position="none",
      plot.title = element_text(color = "#0099f8", size = 12, face = "bold", hjust = 0.5),
    )

  plot <- plot + annotate(
    geom = "text",
    x = -Inf,
    y = Inf,
    label = r2_text,
    hjust = -0.1,
    vjust = 1.5,
    size = 6,
    color = "black"
  )

  return (plot)
}

plot_all_numerical_scatterplots_improved <- function(data) {
  sugar_plot <- plot_numeric_scatterplot_improved(data, "sugarpercent")
  price_plot <- plot_numeric_scatterplot_improved(data, "pricepercent")


  grid.arrange(sugar_plot, price_plot, nrow = 1)
}
```

```
plot_all_numerical_scatterplots_improved(candy_data)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```





1.2.2 Insights

From the above two scatter plots, even though we can see a slight positive correlation with `winpercent` for each of the two variables, they are quite insignificant. Therefore, we can conclude that thereis no significant correlation present.

TASK 2: Bank Churn

2.1 Exploratory Data Analysis

```r
bank_churn_data <- read.csv("~/CMM703/Bank_Churn.csv")
```

After loading the dataset, we can view a summary of all the variables

```r
summary(bank_churn_data)
```

```
##    CustomerId          Surname           CreditScore     Geography        
##  Min.   :15565701   Length:10000       Min.   :350.0   Length:10000      
##  1st Qu.:15628528   Class :character   1st Qu.:584.0   Class :character  
##  Median :15690738   Mode  :character   Median :652.0   Mode  :character  
##  Mean   :15690941                      Mean   :650.5                     
##  3rd Qu.:15753234                      3rd Qu.:718.0                     
##  Max.   :15815690                      Max.   :850.0                     
##     Gender               Age            Tenure          Balance      
##  Length:10000       Min.   :18.00   Min.   : 0.000   Min.   :     0  
##  Class :character   1st Qu.:32.00   1st Qu.: 3.000   1st Qu.:     0  
##  Mode  :character   Median :37.00   Median : 5.000   Median : 97199  
##                     Mean   :38.92   Mean   : 5.013   Mean   : 76486  
##                     3rd Qu.:44.00   3rd Qu.: 7.000   3rd Qu.:127644  
##                     Max.   :92.00   Max.   :10.000   Max.   :250898  
##  NumOfProducts    HasCrCard      IsActiveMember   EstimatedSalary    
##  Min.   :1.00   Min.   :0.0000   Min.   :0.0000   Min.   :    11.58  
##  1st Qu.:1.00   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 51002.11  
##  Median :1.00   Median :1.0000   Median :1.0000   Median :100193.91  
##  Mean   :1.53   Mean   :0.7055   Mean   :0.5151   Mean   :100090.24  
##  3rd Qu.:2.00   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:149388.25  
##  Max.   :4.00   Max.   :1.0000   Max.   :1.0000   Max.   :199992.48  
##      Exited      
##  Min.   :0.0000  
##  1st Qu.:0.0000  
##  Median :0.0000  
##  Mean   :0.2037  
##  3rd Qu.:0.0000  
##  Max.   :1.0000
```

However, it would be much easier to visualize the data through plots

2.1.1 Visualizing numerical data

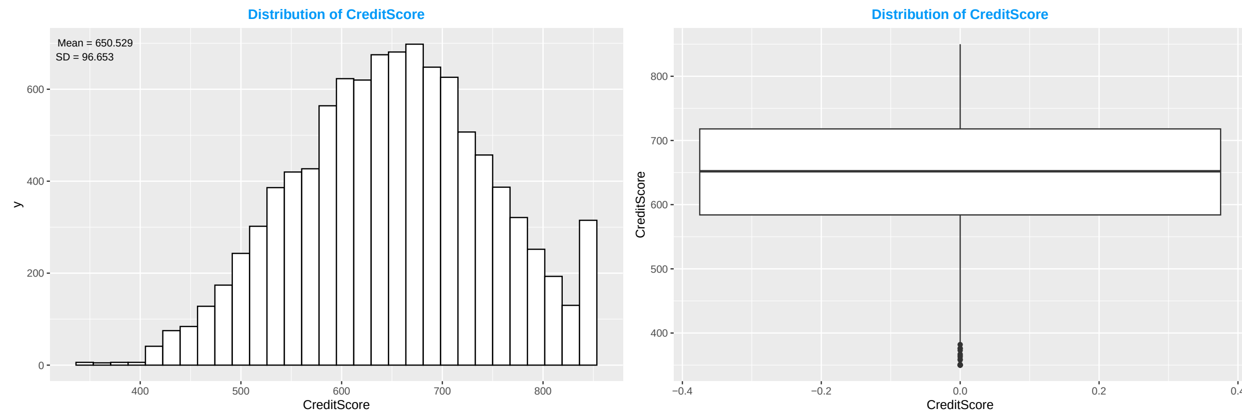
We can visualize numerical data using histograms and box plots.

```r
plot_histogram <- function(data, variable) {
  mean = mean(data[,variable])
  sd = sd(data[,variable])
  summary_text = glue("Mean = {format(round(mean, 3), nsmall = 3)}\nSD = {format(round(sd, 3), nsmall =

  plot <- ggplot(data, aes(x=data[,variable])) +
    geom_histogram(color="black", fill="white") +
    labs(
      title = glue('Distribution of {variable}'),
      x = variable
    ) +
```

```
    theme(
      plot.title = element_text(color = "#0099f8", size = 12, face = "bold", hjust = 0.5),
    )

  plot <- plot + annotate(
    geom = "text",
    x = -Inf,
    y = Inf,
    label = summary_text,
    hjust = -0.1,
    vjust = 1.5,
    size = 3,
    color = "black"
  )


  return (plot)
}


plot_boxplot <- function(data, variable) {
  plot <- ggplot(data=data, aes(y=data[,variable])) +
    labs(
      title = glue('Distribution of {variable}'),
      x = variable,
      y = variable
    ) +
    geom_boxplot() +
    theme(
      plot.title = element_text(color = "#0099f8", size = 12, face = "bold", hjust = 0.5),
    )


  return (plot)
}

plot_numerical <- function(data, variable) {
  histogram <- suppressMessages(plot_histogram(data, variable))
  boxplot <- plot_boxplot(data, variable)

  grid.arrange(histogram, boxplot, nrow = 1)
}
```

```
plot_numerical(bank_churn_data, "CreditScore")
```

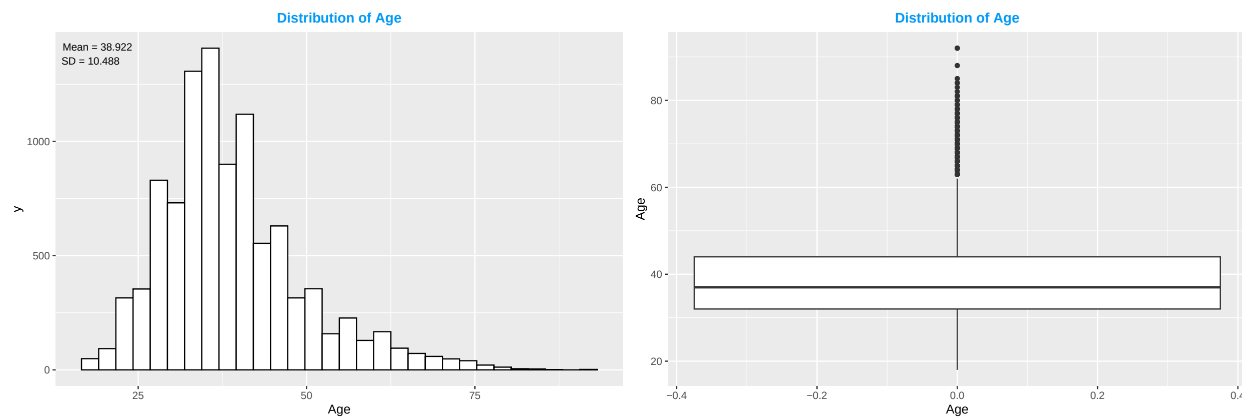
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The credit score seems to have a fairly normal distribution of values.

```
plot_numerical(bank_churn_data, "Age")
```

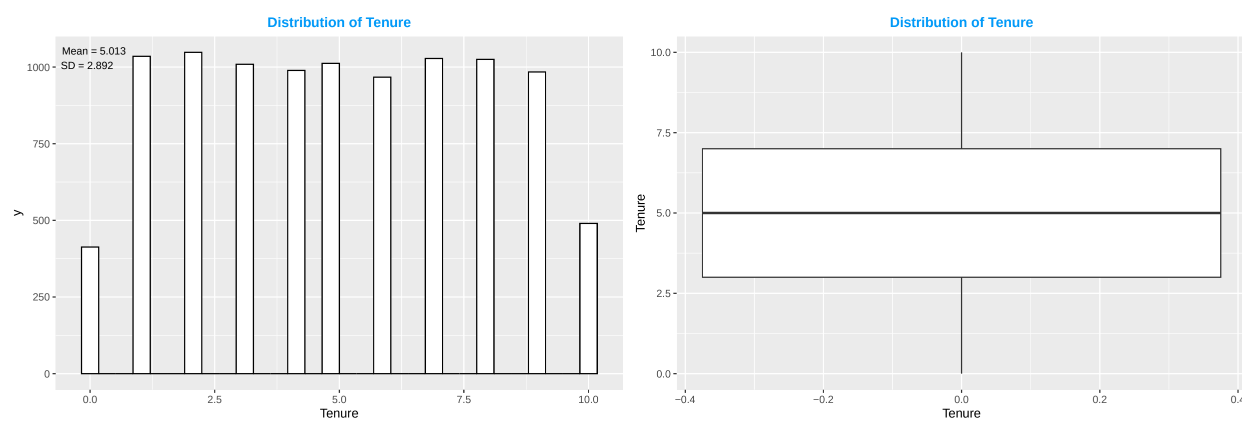
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The age also has a somewhat normal distribution, but with some irregularities.

```
plot_numerical(bank_churn_data, "Tenure")
```

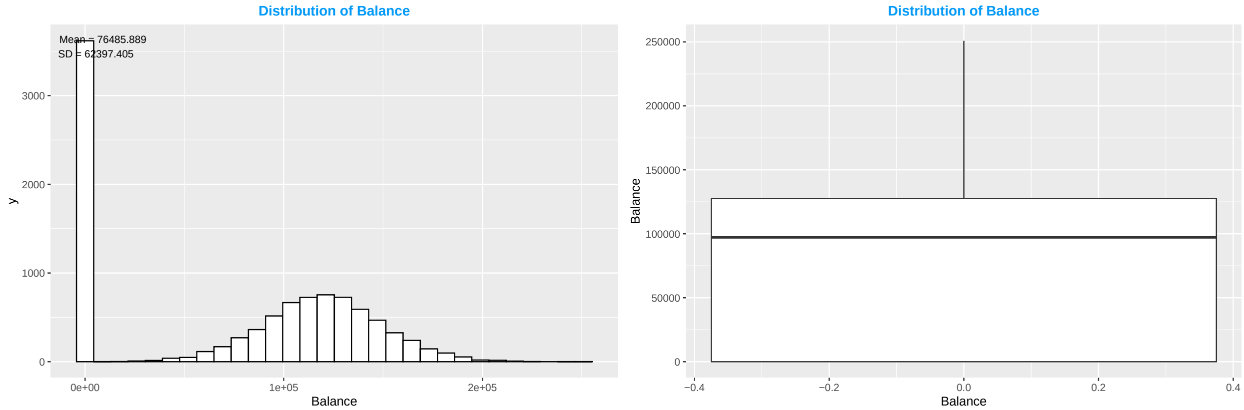
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



There appears to be no pattern to the tenure, with there being around 1000 records for each year.

```r
plot_numerical(bank_churn_data, "Balance")
```

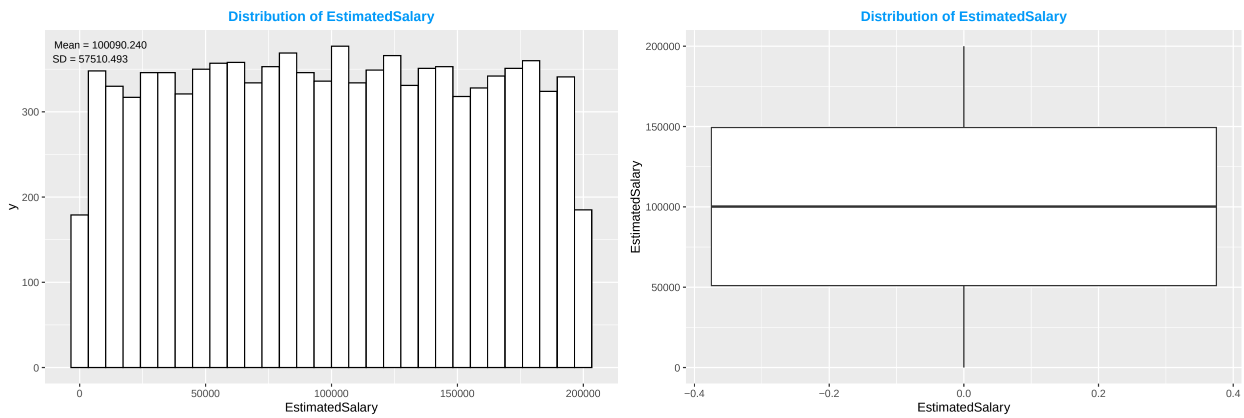
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The balance follows a normal distribution as well. However there is a peak at 0.

```r
plot_numerical(bank_churn_data, "EstimatedSalary")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



There appears to be no pattern to the Estimated Salary as well.

2.1.2 Visualizing categorical data

First, we should convert all categorical columns to the correct data type.

```r
bank_churn_data$Geography <- as.factor(bank_churn_data$Geography)
bank_churn_data$Gender <- as.factor(bank_churn_data$Gender)
bank_churn_data$NumOfProducts <- as.factor(bank_churn_data$NumOfProducts)
bank_churn_data$HasCrCard <- as.factor(bank_churn_data$HasCrCard)
bank_churn_data$IsActiveMember <- as.factor(bank_churn_data$IsActiveMember)
bank_churn_data$Exited <- as.factor(bank_churn_data$Exited)
```

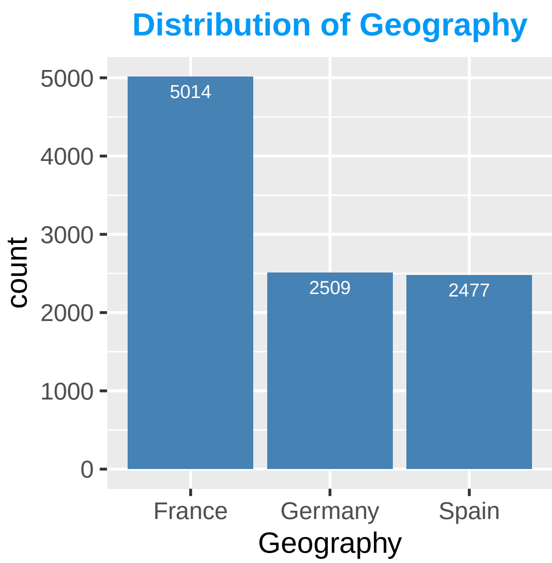
Now we can plot their distribution using bar charts.

```r
plot_bar_chart <- function(data, variable) {
  ggplot(data=data, aes(x=data[,variable])) +
    geom_bar(stat="count", fill="steelblue") +
    geom_text(stat="count", aes(label=..count..), vjust=1.6, color="white", size=2.5) +
```

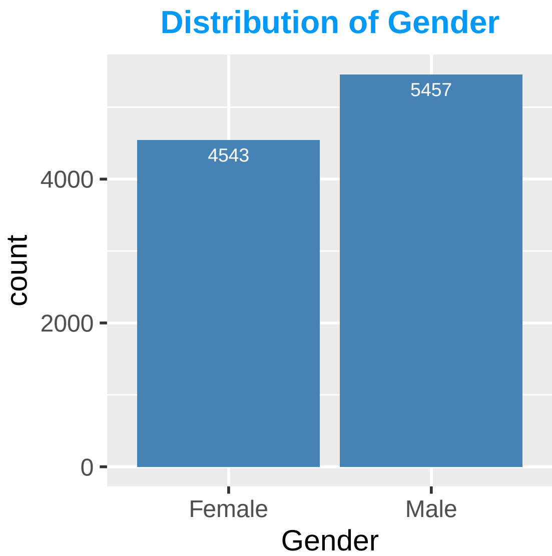
```
    labs(
      title = glue('Distribution of {variable}'),
      x = variable,
      y = "count"
    ) +
    theme(
      plot.title = element_text(color = "#0099f8", size = 12, face = "bold", hjust = 0.5),
    )
}
```

```
plot_bar_chart(bank_churn_data, "Geography")
```

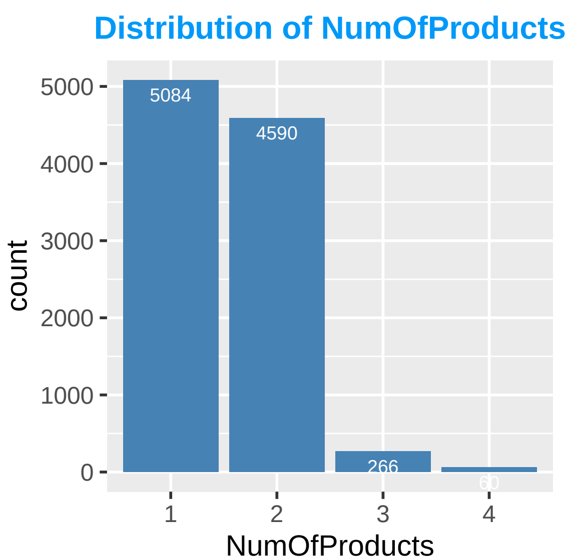
```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



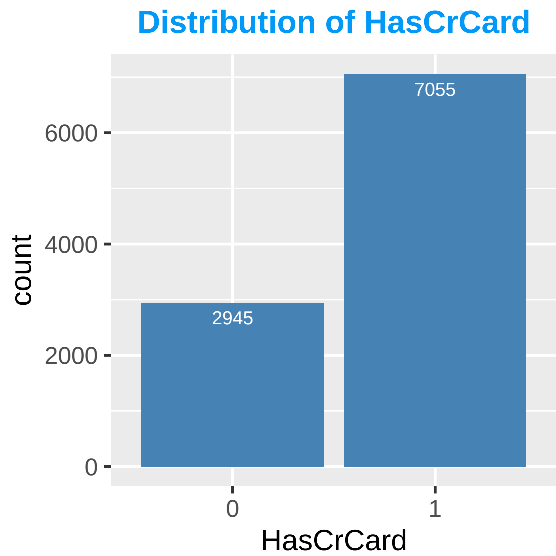

```
plot_bar_chart(bank_churn_data, "Gender")
```



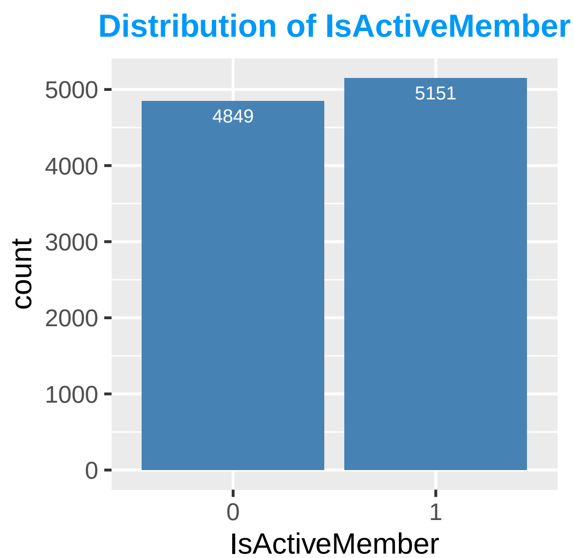

```r
plot_bar_chart(bank_churn_data, "NumOfProducts")
```



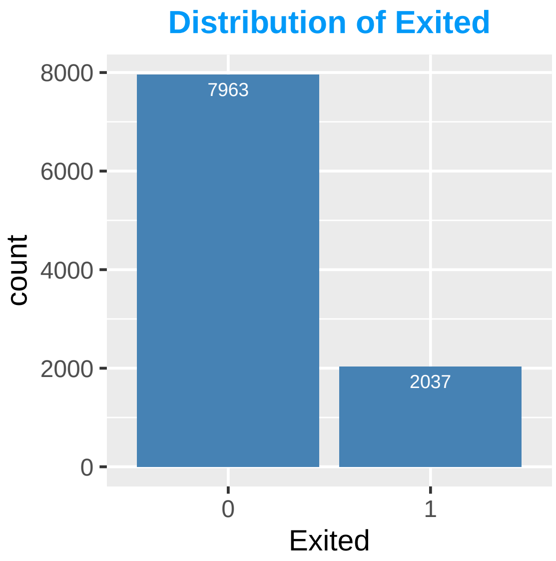

```r
plot_bar_chart(bank_churn_data, "HasCrCard")
```




```r
plot_bar_chart(bank_churn_data, "IsActiveMember")
```




```r
plot_bar_chart(bank_churn_data, "Exited")
```



2.1.3 Identifying Numeric-Numeric Correlations

We can calculate the correlations between numeric variables.

```
numeric_only <- subset(bank_churn_data, select=c("CreditScore", "Age", "Tenure", "Balance", "EstimatedSa
correlations <- cor(numeric_only)
round(correlations, 3)
```

```
##                 CreditScore    Age Tenure Balance EstimatedSalary
## CreditScore           1.000 -0.004  0.001   0.006          -0.001
## Age                  -0.004  1.000 -0.010   0.028          -0.007
## Tenure                0.001 -0.010  1.000  -0.012           0.008
## Balance               0.006  0.028 -0.012   1.000           0.013
## EstimatedSalary      -0.001 -0.007  0.008   0.013           1.000
```

```
ggcorrplot(correlations)
```

As we can see, there does not seem to be any significant correlations between the numerical variables

2.1.4 Identifying Categorical-Categorical Correlations

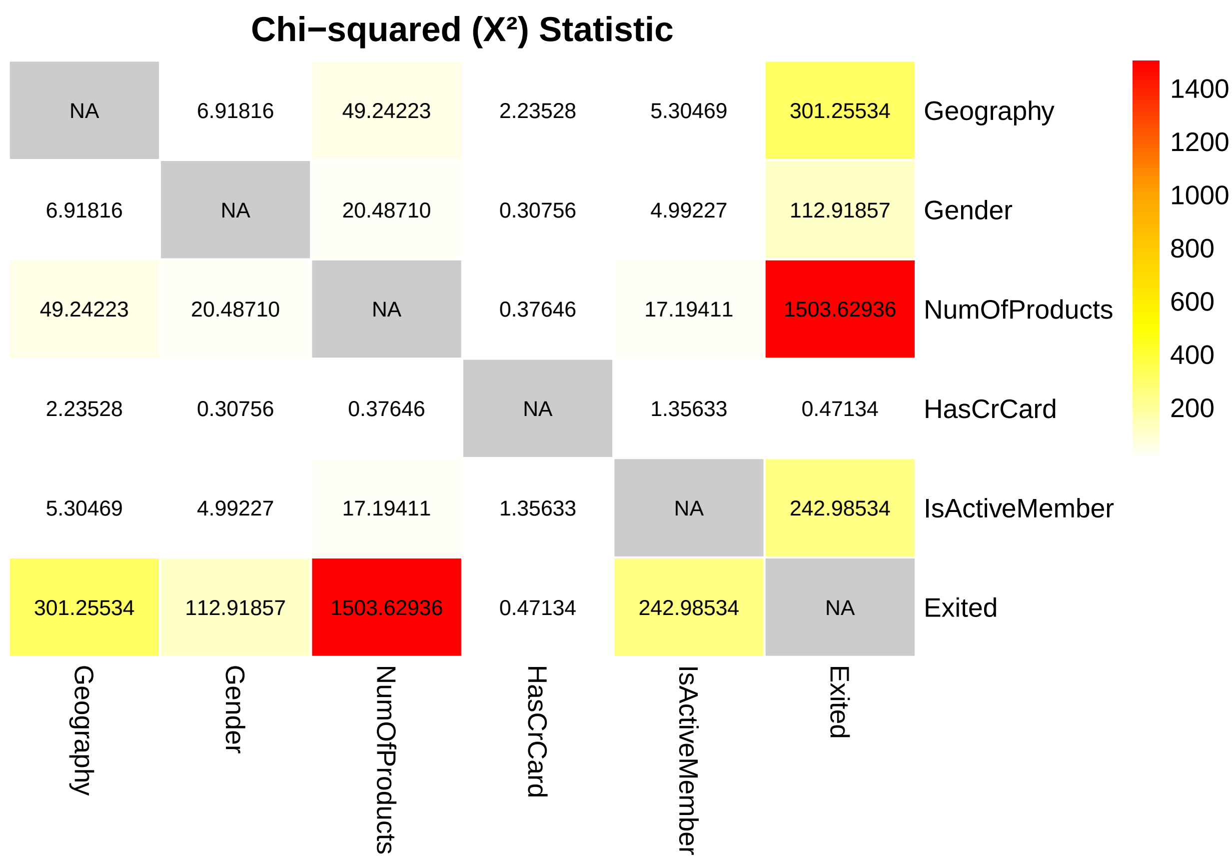
We can calculate the correlations between categorical variables using pairwise Chi-squared tests.

```
categorical_columns <- c("Geography", "Gender", "NumOfProducts", "HasCrCard", "IsActiveMember", "Exited

pairwise_p_vals <- matrix(nrow=6, ncol=6)
pairwise_chi_square <- matrix(nrow=6, ncol=6)
rownames(pairwise_p_vals) <- categorical_columns
colnames(pairwise_p_vals) <- categorical_columns
rownames(pairwise_chi_square) <- categorical_columns
colnames(pairwise_chi_square) <- categorical_columns

for (i in 1:5) {
  for (j in (i+1):6) {
    contingency_table <- table(bank_churn_data[,categorical_columns[i]], bank_churn_data[,categorical_c
    chi_square_test <- chisq.test(contingency_table)

    p_value <- chi_square_test$p.value
    total_chi_square <- chi_square_test$statistic

    pairwise_p_vals[[i, j]] <- p_value
    pairwise_p_vals[[j, i]] <- p_value

    pairwise_chi_square[[i, j]] <- total_chi_square
```

```
    pairwise_chi_square[[j, i]] <- total_chi_square
  }
}
```

The higher the Chi-squared value, the more significant the correlation.

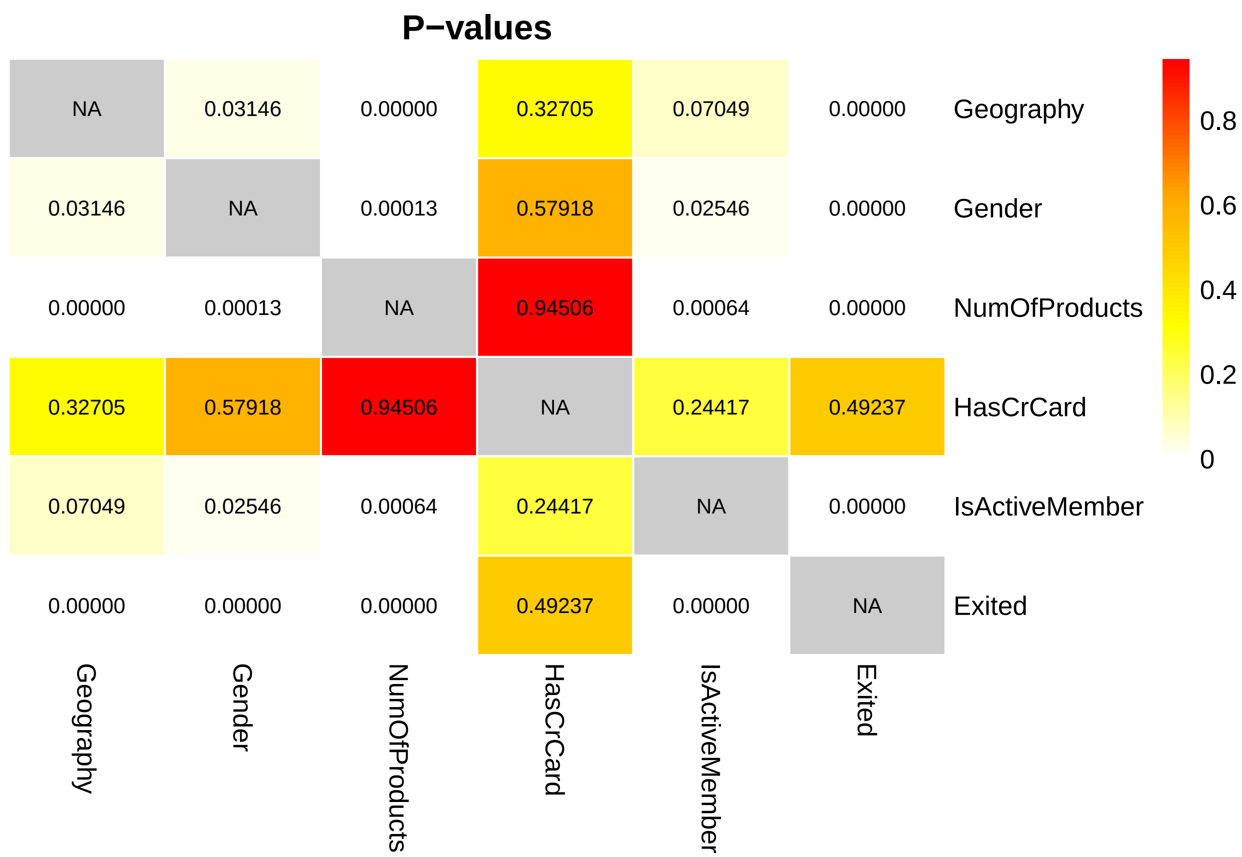
```
pheatmap(pairwise_chi_square,
         color = colorRampPalette(rev(c("red", "orange", "yellow", "white")))(100),
         display_numbers = TRUE,
         number_format = "%.5f",
         number_color = "black",
         na_col = "grey80",
         main = "Chi-squared (X²) Statistic",
         fontsize_number = 8,
         border_color = "white",
         cluster_rows = FALSE,
         cluster_cols = FALSE)
```



If the p-value is less than 0.05, we can consider it significant.

```
pheatmap(pairwise_p_vals,
         color = colorRampPalette(rev(c("red", "orange", "yellow", "white")))(100),
         display_numbers = TRUE,
         number_format = "%.5f",
         number_color = "black",
         na_col = "grey80",
         main = "P-values",
```

```
        fontsize_number = 8,
        border_color = "white",
        cluster_rows = FALSE,
        cluster_cols = FALSE)
```



2.1.5 Identifying Numerical-Categorical Correlations

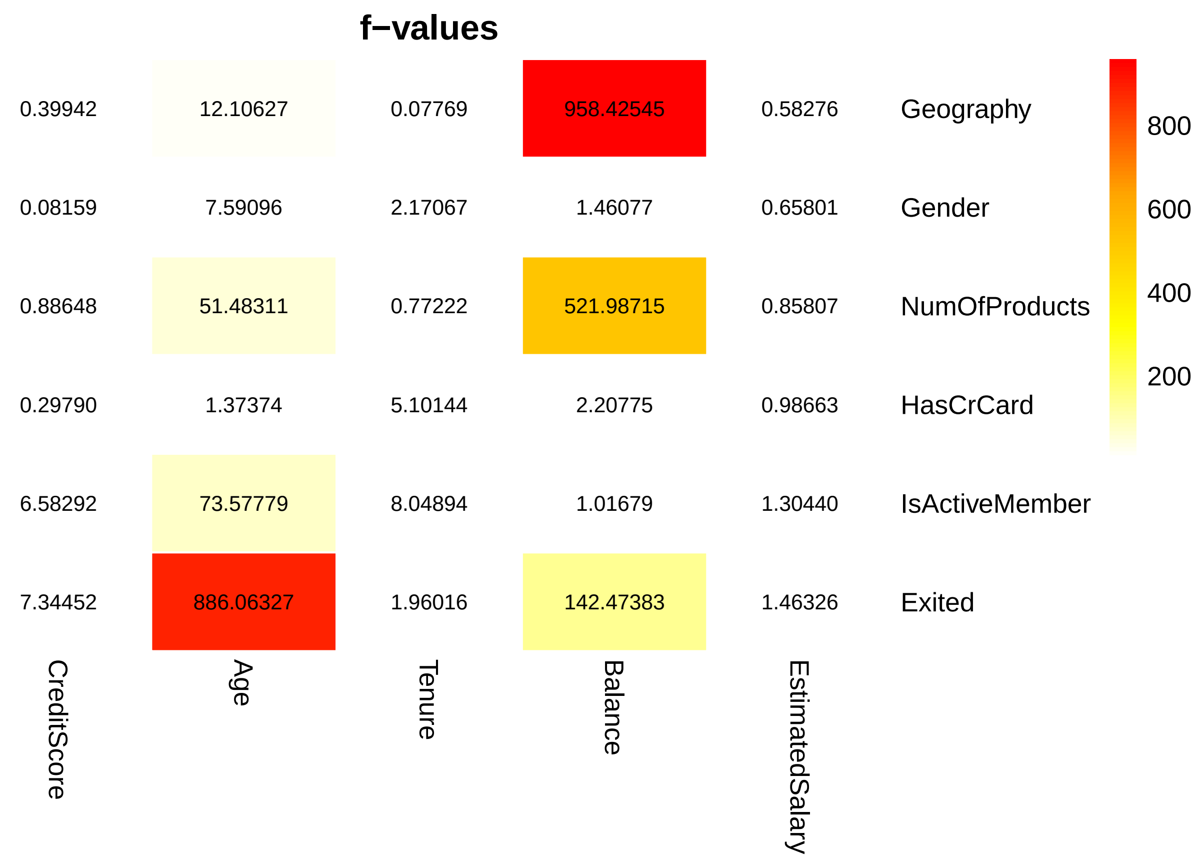
We can determine the correlation between numerical and categorical variables using pairwise ANOVA tests.

```
categorical_columns <- c("Geography", "Gender", "NumOfProducts", "HasCrCard", "IsActiveMember", "Exited
numeric_columns <- c("CreditScore", "Age", "Tenure", "Balance", "EstimatedSalary")

pairwise_p_vals <- matrix(nrow=6, ncol=5)
pairwise_f_vals <- matrix(nrow=6, ncol=5)
rownames(pairwise_p_vals) <- categorical_columns
colnames(pairwise_p_vals) <- numeric_columns
rownames(pairwise_f_vals) <- categorical_columns
colnames(pairwise_f_vals) <- numeric_columns


for (categorical_column in categorical_columns) {
  for (numeric_column in numeric_columns) {
    anova <- aov(reformulate(categorical_column, numeric_column), data=bank_churn_data)
    p_val <- summary(anova)[[1]][["Pr(>F)"]][1]
    f_val <- summary(anova)[[1]][["F value"]][1]
```

```
    pairwise_p_vals[[categorical_column, numeric_column]] <- p_val
    pairwise_f_vals[[categorical_column, numeric_column]] <- f_val
  }
}
```

The higher the f-value is, the more significant the relationship.

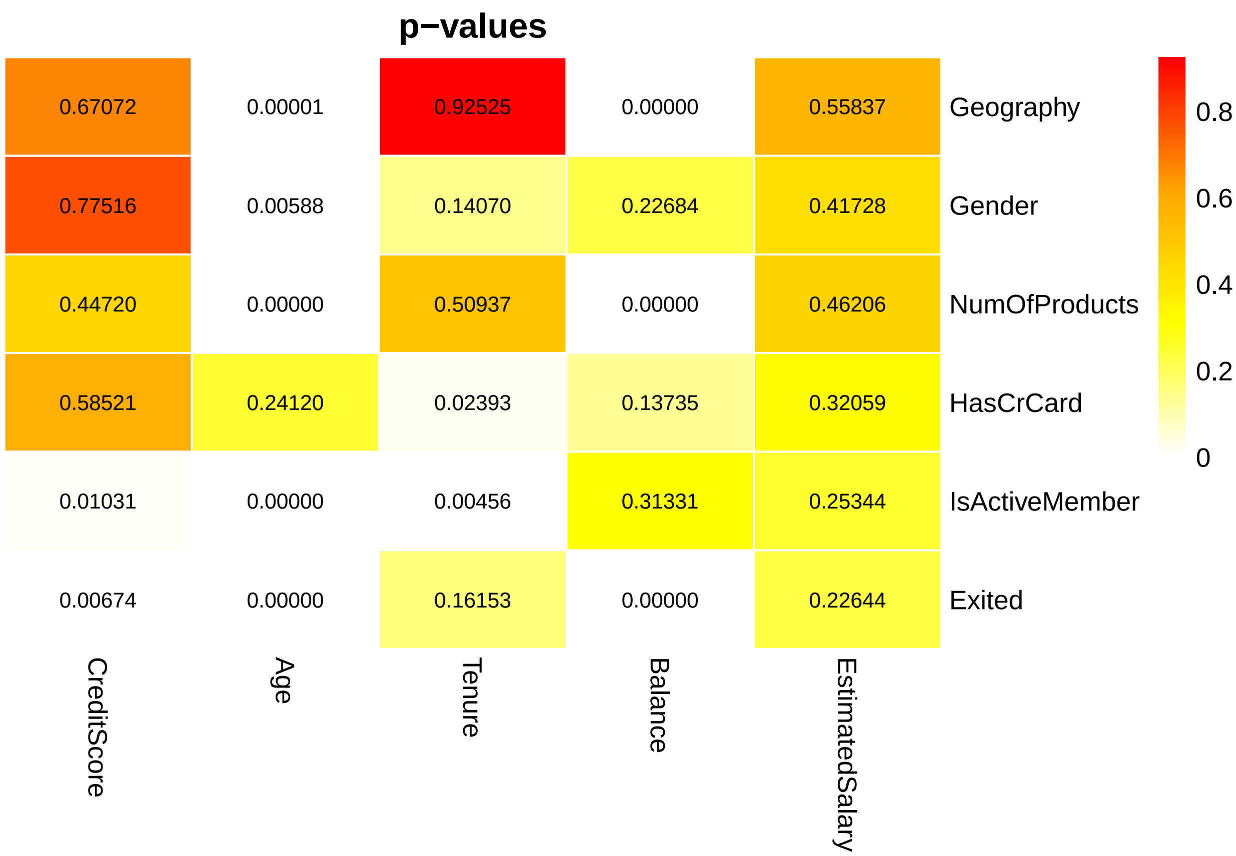
```
pheatmap(pairwise_f_vals,
         color = colorRampPalette(rev(c("red", "orange", "yellow", "white")))(100),
         display_numbers = TRUE,
         number_format = "%.5f",
         number_color = "black",
         na_col = "grey80",
         main = "f-values",
         fontsize_number = 8,
         border_color = "white",
         cluster_rows = FALSE,
         cluster_cols = FALSE)
```



If the p-value is less than 0.05, we can consider it significant

```
pheatmap(pairwise_p_vals,
         color = colorRampPalette(rev(c("red", "orange", "yellow", "white")))(100),
         display_numbers = TRUE,
         number_format = "%.5f",
         number_color = "black",
         na_col = "grey80",
```

```
         main = "p-values",
         fontsize_number = 8,
         border_color = "white",
         cluster_rows = FALSE,
         cluster_cols = FALSE)
```



2.1 Predictive Logistic Regression Model for Churn (Exited)

We first split the data into training and testing datasets at an 80/20 ratio.

```
split <- sample.split(bank_churn_data$Exited, SplitRatio = 0.8)
train_data <- subset(bank_churn_data, split == TRUE)
test_data <- subset(bank_churn_data, split == FALSE)
```

Next we develop the model. From our earlier analysis, we determined that the following variables have the most significant impact on `Exited`
- `Balance`
- `NumOfProducts`
- `Geography`
- `Gender`
- `Age`
- `IsActiveMember`

```
model <- glm(Exited ~ Balance + NumOfProducts + Geography + Gender + Age + IsActiveMember, data = train_
```

We can view the summary of the model.

```r
summary(model)
```

```
##
## Call:
## glm(formula = Exited ~ Balance + NumOfProducts + Geography +
##     Gender + Age + IsActiveMember, family = binomial(link = "logit"),
##     data = train_data)
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -3.340e+00  1.494e-01 -22.355  < 2e-16 ***
## Balance           -4.828e-07  6.321e-07  -0.764    0.445
## NumOfProducts2    -1.542e+00  7.958e-02 -19.379  < 2e-16 ***
## NumOfProducts3     2.331e+00  1.932e-01  12.065  < 2e-16 ***
## NumOfProducts4     1.639e+01  1.920e+02   0.085    0.932
## GeographyGermany   9.552e-01  8.052e-02  11.863  < 2e-16 ***
## GeographySpain     7.492e-02  8.456e-02   0.886    0.376
## GenderMale        -4.992e-01  6.539e-02  -7.634 2.28e-14 ***
## Age                6.995e-02  3.057e-03  22.884  < 2e-16 ***
## IsActiveMember1   -1.053e+00  6.870e-02 -15.327  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8088.9  on 7999  degrees of freedom
## Residual deviance: 6022.2  on 7990  degrees of freedom
## AIC: 6042.2
##
## Number of Fisher Scoring iterations: 14
```

2.2 Getting predictions for test dataset

We use the threshold value as 0.5, and make a set of prediction on our test dataset

```r
test_probabilities <- predict(model, newdata = test_data, type = "response")
predicted_exited <- ifelse(test_probabilities > 0.5, 1, 0)
```

We can compare the predicted values against our actual values and build the confusion matrix.

```r
actual_exited <- test_data$Exited
conf_matrix <- table(Actual = actual_exited, Predicted = predicted_exited)
print(conf_matrix)
```

```
##       Predicted
## Actual    0    1
##      0 1533   60
##      1  231  176
```

Here, we see that our model seems to have an issue with misclassificaion of false values (high number of false negatives). And we can plot the performance metrics to analyze the performance of our model.

```r
TP <- conf_matrix[2, 2]
TN <- conf_matrix[1, 1]
FP <- conf_matrix[1, 2]
FN <- conf_matrix[2, 1]
```

```r
accuracy <- (TP + TN) / sum(conf_matrix)
precision <- TP / (TP + FP)
sensitivity <- TP / (TP + FN)
specificity <- TN / (TN + FP)
f1_score <- 2 * (precision * sensitivity) / (precision + sensitivity)

print(glue("Accuracy: {accuracy}"))
```

```
## Accuracy: 0.8545
```

```r
print(glue("Precision: {precision}"))
```

```
## Precision: 0.745762711864407
```

```r
print(glue("Sensitivity: {sensitivity}"))
```

```
## Sensitivity: 0.432432432432432
```

```r
print(glue("Specificity: {specificity}"))
```

```
## Specificity: 0.962335216572505
```

```r
print(glue("F1 score: {f1_score}"))
```

```
## F1 score: 0.547433903576983
```

Even though the accuracy of the model may be high, it still does not seem to perform that well when the Churn is false (as indicated by the slow Sensitivity). We can also plot the ROC curve and calculate the area under it.

```r
roc_curve <- roc(response = actual_exited, predictor = test_probabilities)
```

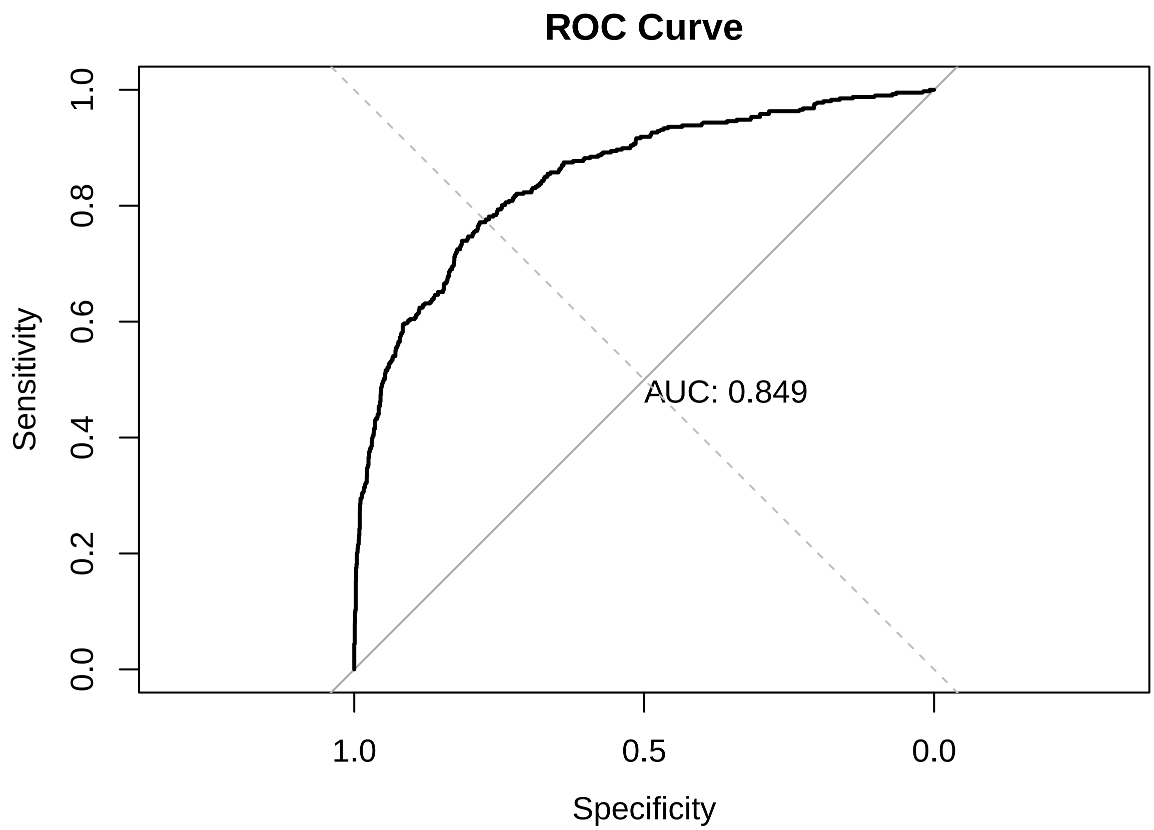
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
auc_value <- auc(roc_curve)
print(glue("AUC: {round(auc_value, 4)}"))
```

```
## AUC: 0.8491
```

```r
plot(roc_curve, main = "ROC Curve", print.auc = TRUE)
abline(a=0, b=1, lty=2, col="gray")
```



2.4 Predicting Tenure

From our earlier analysis, we can see that there are few to no variables which show significant correllation with `Tenure`. Hence we will use all variables in the dataset for our model.

```
tenure_model = lm(Tenure ~ CreditScore + Geography + Gender + Age + Balance + NumOfProducts + HasCrCard
summary(tenure_model)
```

```
##
## Call:
## lm(formula = Tenure ~ CreditScore + Geography + Gender + Age +
##     Balance + NumOfProducts + HasCrCard + +IsActiveMember + EstimatedSalary,
##     data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.3230 -2.2867 -0.0149  2.4909  5.3645
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       5.091e+00  2.751e-01  18.507   <2e-16 ***
## CreditScore      -5.184e-05  3.343e-04  -0.155   0.8768
## GeographyGermany -2.209e-02  8.603e-02  -0.257   0.7974
## GeographySpain    8.425e-03  7.922e-02   0.106   0.9153
## GenderMale        9.050e-02  6.495e-02   1.393   0.1635
## Age              -3.633e-03  3.111e-03  -1.168   0.2429
## Balance          -3.292e-07  6.115e-07  -0.538   0.5904
```

```
## NumOfProducts2    9.413e-02  7.181e-02   1.311   0.1900
## NumOfProducts3    1.226e-01  2.079e-01   0.590   0.5555
## NumOfProducts4    4.555e-01  4.115e-01   1.107   0.2683
## HasCrCard1        1.400e-01  7.062e-02   1.982   0.0475 *
## IsActiveMember1  -1.635e-01  6.496e-02  -2.517   0.0118 *
## EstimatedSalary   2.759e-07  5.610e-07   0.492   0.6229
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.886 on 7987 degrees of freedom
## Multiple R-squared:  0.0024, Adjusted R-squared:  0.0009014
## F-statistic: 1.601 on 12 and 7987 DF,  p-value: 0.08367
```

After training the model on our training dataset, we can evaluate it against our test dataset.

```
test_predictions <- predict(tenure_model, newdata = test_data)
actual_tenure = test_data$Tenure

rmse <- sqrt(mean((actual_tenure - test_predictions)^2))
print(glue("RMSE = {rmse}"))
```

```
## RMSE = 2.91322968982937
```

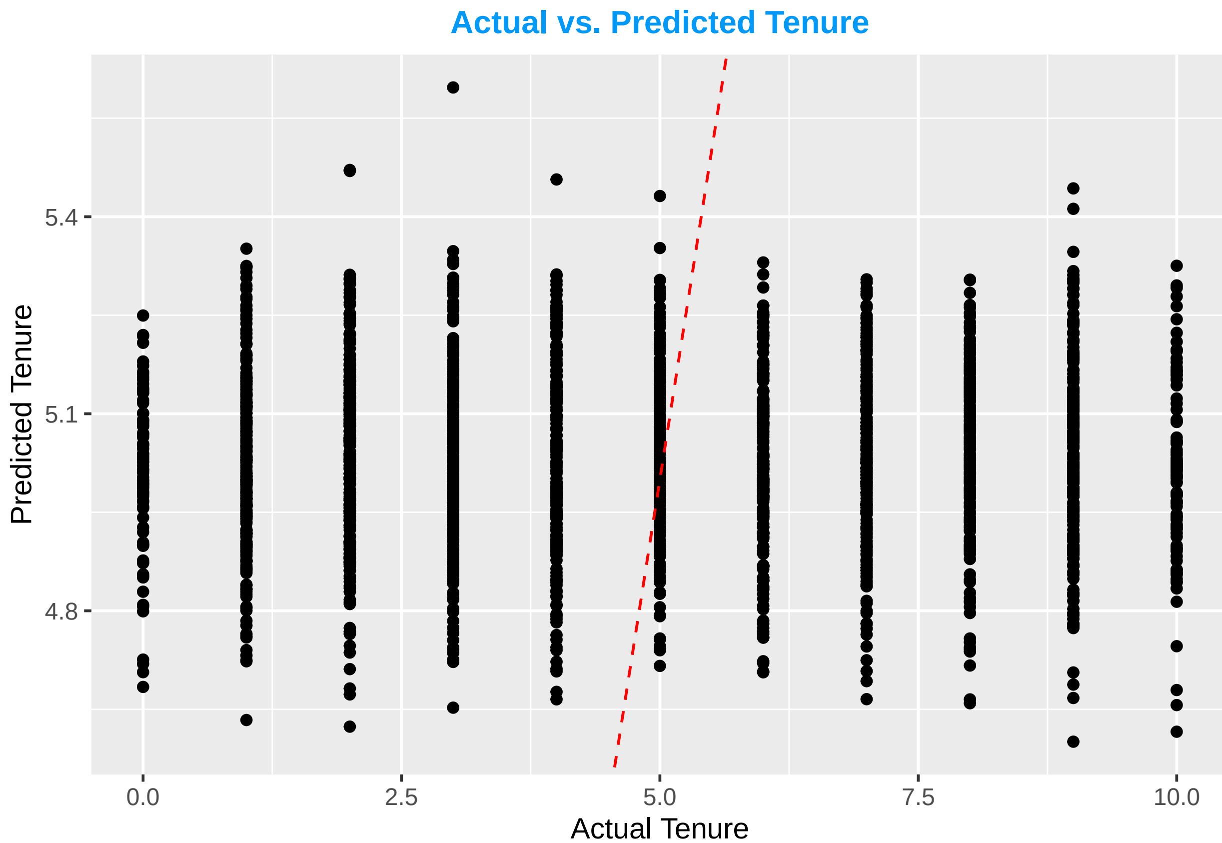
The root mean squared error we obtain is 2.9, which is not great considering that this would cover over 50% of the values in `Tenure`.

```
rss <- sum((test_predictions - actual_tenure)^2)
tss <- sum((actual_tenure - mean(actual_tenure))^2)
rsq_test <- 1 - (rss / tss)
print(glue("R² = {round(rsq_test, 4)}"))
```

```
## R² = -0.0012
```

The value we obtain for R^2 is also very low. This indicates that this model does not perform well.

```
ggplot(data = test_data, aes(x = Tenure, y = test_predictions)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +
  labs(title = "Actual vs. Predicted Tenure",
       x = "Actual Tenure",
       y = "Predicted Tenure") +
  theme(
    plot.title = element_text(color = "#0099f8", size = 12, face = "bold", hjust = 0.5),
  )
```



We can also plot our predicted values against the actual values. Here we can see that our model is mostly predicting values between 4.6 and 5.4. But the complete range of values fall between 0 and 10. We can infer from this, that our model does not have sufficient data to make accurate predictions.

TASK 3: Functions

3.1 Function to identify qualitative and quantitative variable

Here, we simply check for the type of each column. Integer and numeric types correspond to quantitative data, while factor and logical types correspond to qualitative data. Character types are of neither type.

```
quantitative_or_qualitative <- function(data) {
  quantitatve <- c()
  qualitative <- c()

  for (column_name in names(data)){
    column_type = class(data[,column_name])

    if (column_type == "integer" || column_type == "numeric" ) {
      quantitatve <- c(quantitatve, column_name)
    } else if (column_type == "factor" || column_type == "logical") {
      qualitative <- c(qualitative, column_name)
    }

  }

  return (list(quantitatve=quantitatve, qualitative=qualitative))
}

quantitative_or_qualitative(bank_churn_data)
```

```
## $quantitatve
## [1] "CustomerId"      "CreditScore"     "Age"             "Tenure"
## [5] "Balance"         "EstimatedSalary"
##
## $qualitative
## [1] "Geography"      "Gender"         "NumOfProducts"  "HasCrCard"
## [5] "IsActiveMember" "Exited"
```

3.2 Handle missing values

```
handle_missing_values <- function(data) {
  Mode <- function(data) {
    unique_values <- unique(data)
    unique_values[which.max(tabulate(match(data, unique_values)))]
  }

  for (column_name in names(data)){
    column_type = class(data[,column_name])

    if (column_type == "integer" || column_type == "numeric") {
      data[is.na(data[,column_name]), column_name] <- mean(data[,column_name], na.rm = TRUE)
    } else if (column_type == "factor" || column_type == "logical") {
      data[is.na(data[,column_name]), column_name] <- Mode(data[,column_name])
    }

  }

  return (data)
```

```
}
```

To test this method, we add a new row to the `bank_churn` dataset with some missing values.

```
bank_churn_new <- bank_churn_data
bank_churn_new[nrow(bank_churn_new) + 1,] <- list(1, "Dias", 999, NA, "Male", NA, NA, 100, NA, 1, 1, 99
```

And we ensure all columns are of the correct type

```
bank_churn_new$CustomerId <- as.character(bank_churn_new$CustomerId)
bank_churn_new$Surname  <- as.character(bank_churn_new$Surname )
bank_churn_new$CreditScore <- as.integer(bank_churn_new$CreditScore)
bank_churn_new$Geography <- as.factor(bank_churn_new$Geography)
bank_churn_new$Gender <- as.factor(bank_churn_new$Gender)
bank_churn_new$Tenure <- as.integer(bank_churn_new$Tenure)
bank_churn_new$Age <- as.integer(bank_churn_new$Age)
bank_churn_new$Balance  <- as.integer(bank_churn_new$Balance )
bank_churn_new$NumOfProducts  <- as.factor(bank_churn_new$NumOfProducts)
bank_churn_new$HasCrCard  <- as.factor(bank_churn_new$HasCrCard)
bank_churn_new$IsActiveMember   <- as.factor(bank_churn_new$IsActiveMember )
bank_churn_new$EstimatedSalary   <- as.numeric(bank_churn_new$EstimatedSalary )
bank_churn_new$Exited   <- as.factor(bank_churn_new$Exited )
```

Here we can see the last row of data has NA values

```
tail(bank_churn_new, 3)
```

```
##       CustomerId   Surname CreditScore Geography Gender Age Tenure Balance
## 9999    15682355 Sabbatini         772   Germany   Male  42      3   75075
## 10000   15628319    Walker         792    France Female  28      4  130142
## 10001          1      Dias         999      <NA>   Male  NA     NA     100
##       NumOfProducts HasCrCard IsActiveMember EstimatedSalary Exited
## 9999              2         1              0        92888.52      1
## 10000             1         1              0        38190.78      0
## 10001          <NA>         1              1          999.00      0
```

We run the function to handle missing values

```
bank_churn_new <- handle_missing_values(bank_churn_new)
```

And now, we can see that the missing values have been replaced by the mean or the mode.

```
tail(bank_churn_new, 3)
```

```
##       CustomerId   Surname CreditScore Geography Gender     Age Tenure Balance
## 9999    15682355 Sabbatini         772   Germany   Male 42.0000 3.0000   75075
## 10000   15628319    Walker         792    France Female 28.0000 4.0000  130142
## 10001          1      Dias         999    France   Male 38.9218 5.0128     100
##       NumOfProducts HasCrCard IsActiveMember EstimatedSalary Exited
## 9999              2         1              0        92888.52      1
## 10000             1         1              0        38190.78      0
## 10001             1         1              1          999.00      0
```

3.3 Outlier detection

In order to do this, we iterate through all the columns, and identify values which are either 1.5 IQR away from Q1/Q3, or 3 standard deviations away from the mean.

The possible values for method are `IQR` and `SD`.

```
detect_outliers <- function(data, method = "IQR") {
  outliers_list <- list()

  for (column_name in names(data)){
    column_type = class(data[,column_name])
    if (!(column_type == "integer" || column_type == "numeric")) {
      next
    }

    vals <- data[,column_name]

    if (method == "IQR") {
      q1 <- quantile(vals, 0.25, na.rm = TRUE)
      q3 <- quantile(vals, 0.75, na.rm = TRUE)
      iqr <- q3 - q1
      lower_bound <- q1 - 1.5*iqr
      upper_bound <- q3 + 1.5*iqr

      outliers <- vals[vals < lower_bound | vals > upper_bound]
      outliers_list[[column_name]] <- outliers
    } else if (method == "SD") {
      mean <- mean(vals, na.rm = TRUE)
      sd <- sd(vals, na.rm = TRUE)

      lower_bound <- mean - 3*sd
      upper_bound <- mean + 3*sd
      outliers <- vals[vals < lower_bound | vals > upper_bound]
      outliers_list[[column_name]] <- outliers
    } else {
      stop("Invalid method")
    }
  }

  return (outliers_list)
}
```

Here we check for outliers on the `bank_churn` dataset using IQR.

```
detect_outliers(bank_churn_new)
```

```
## $CreditScore
##  [1] 376 376 363 359 350 350 358 351 365 367 350 350 382 373 350 999
##
## $Age
##   [1] 66 75 65 73 65 72 67 67 79 80 68 75 66 66 70 63 72 64 64 70 67 82 63 69 65
##  [26] 69 64 65 74 67 66 67 63 70 71 72 67 74 76 66 63 66 68 67 63 71 66 69 73 65
##  [51] 66 64 69 64 77 74 65 70 67 69 67 74 69 74 74 64 63 63 70 74 65 72 77 66 65
##  [76] 74 88 63 71 63 64 67 70 68 72 71 66 75 67 73 69 76 63 85 67 74 76 66 69 66
## [101] 72 63 71 63 74 67 72 72 66 84 71 66 63 74 69 84 67 64 68 66 77 70 67 79 67
## [126] 76 73 66 67 64 73 76 72 64 71 63 70 65 66 65 80 66 63 63 63 63 66 74 69 63
## [151] 64 76 75 68 69 77 64 66 74 71 67 68 64 68 70 64 75 66 64 78 65 74 64 64 71
## [176] 77 79 70 81 64 68 68 63 79 66 64 70 69 71 72 66 68 63 71 72 72 64 78 75 65
## [201] 65 67 63 68 71 73 64 66 71 69 71 66 76 69 73 64 64 75 73 71 72 63 67 68 73
## [226] 67 64 63 92 65 75 67 71 64 66 64 66 67 77 92 67 63 66 66 68 65 72 71 76 63
## [251] 67 67 66 67 63 65 70 72 77 74 72 73 77 67 71 64 72 81 76 69 68 74 64 64 71
```

```
## [276] 68 63 67 63 64 76 63 63 68 67 72 70 81 67 73 66 68 71 66 63 75 69 64 69 70
## [301] 71 71 66 70 63 64 65 63 67 71 67 65 66 63 73 66 64 72 71 69 67 64 81 73 63
## [326] 67 74 83 69 71 78 63 70 69 72 70 63 74 80 69 72 67 76 71 67 71 78 63 63 68
## [351] 64 70 78 69 68 64 64 77 77
##
## $Tenure
## numeric(0)
##
## $Balance
## integer(0)
##
## $EstimatedSalary
## numeric(0)
```

And here we check using SD.

```
detect_outliers(bank_churn_new, method="SD")
```

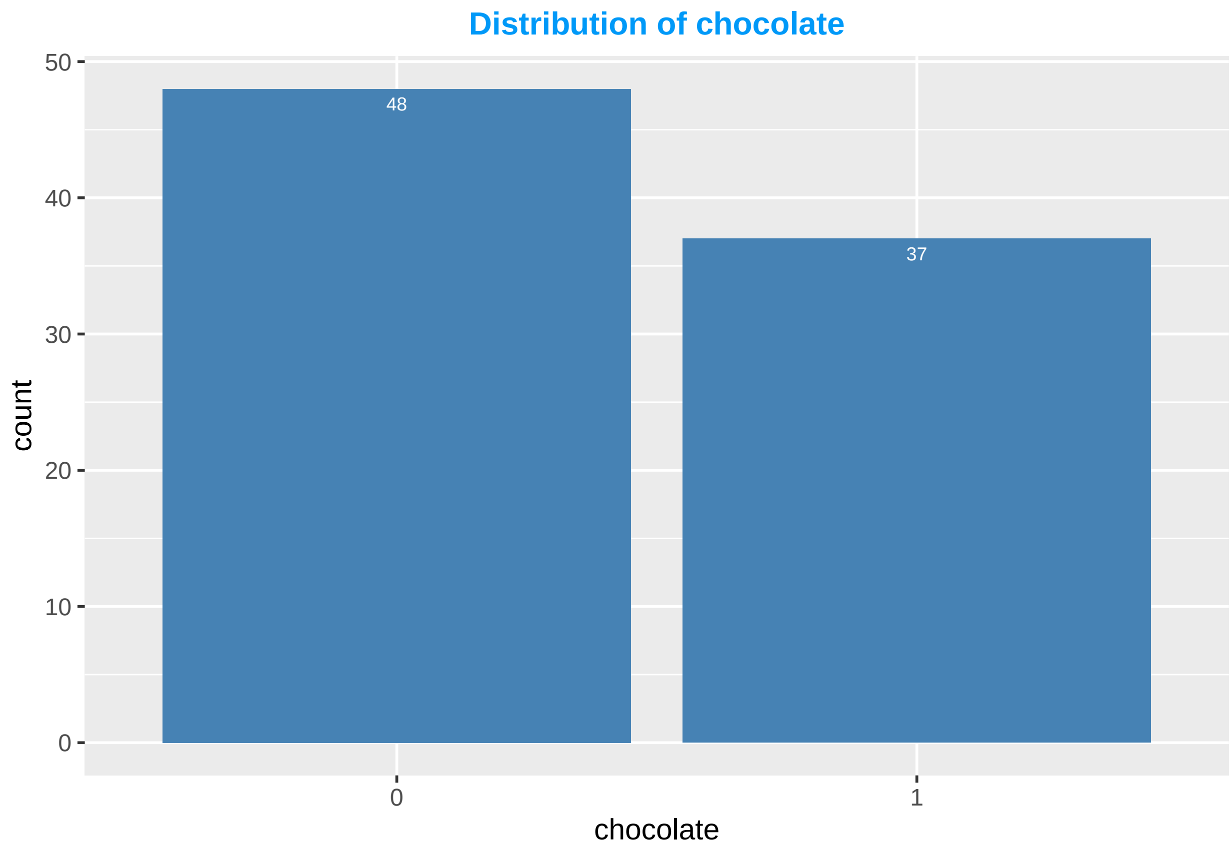
```
## $CreditScore
## [1] 359 350 350 358 351 350 350 350 999
##
## $Age
##   [1] 75 73 72 79 80 75 72 82 74 71 72 74 76 71 73 77 74 74 74 74 74 72 77 74 88
##  [26] 71 72 71 75 73 76 85 74 76 72 71 74 72 72 84 71 74 84 77 79 76 73 73 76 72
##  [51] 71 80 74 76 75 77 74 71 75 78 74 71 77 79 81 79 71 72 71 72 72 78 75 71 73
##  [76] 71 71 76 73 75 73 71 72 73 92 75 71 77 92 72 71 76 72 77 74 72 73 77 71 72
## [101] 81 76 74 71 76 72 81 73 71 75 71 71 71 73 72 71 81 73 74 83 71 78 72 74 80
## [126] 72 76 71 71 78 78 77 77
##
## $Tenure
## numeric(0)
##
## $Balance
## integer(0)
##
## $EstimatedSalary
## numeric(0)
```

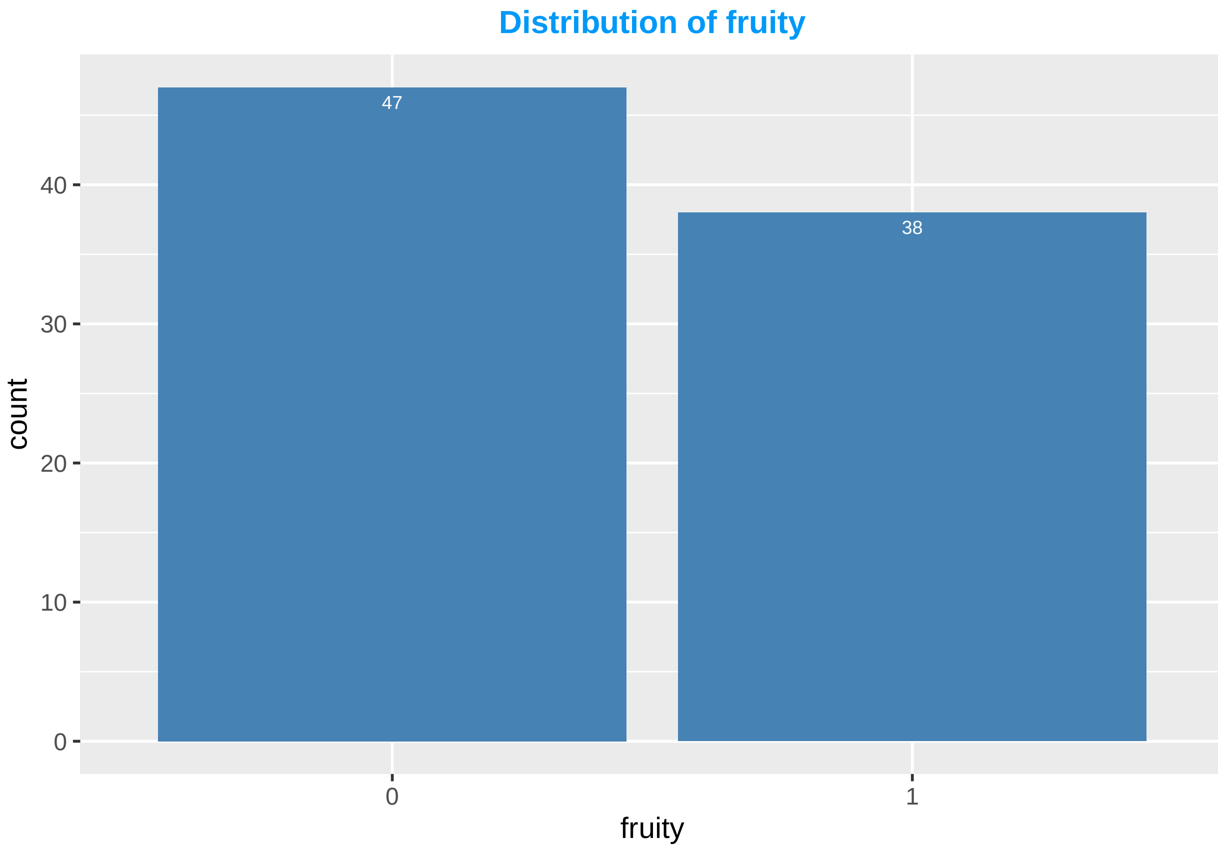
3.4 Visualize

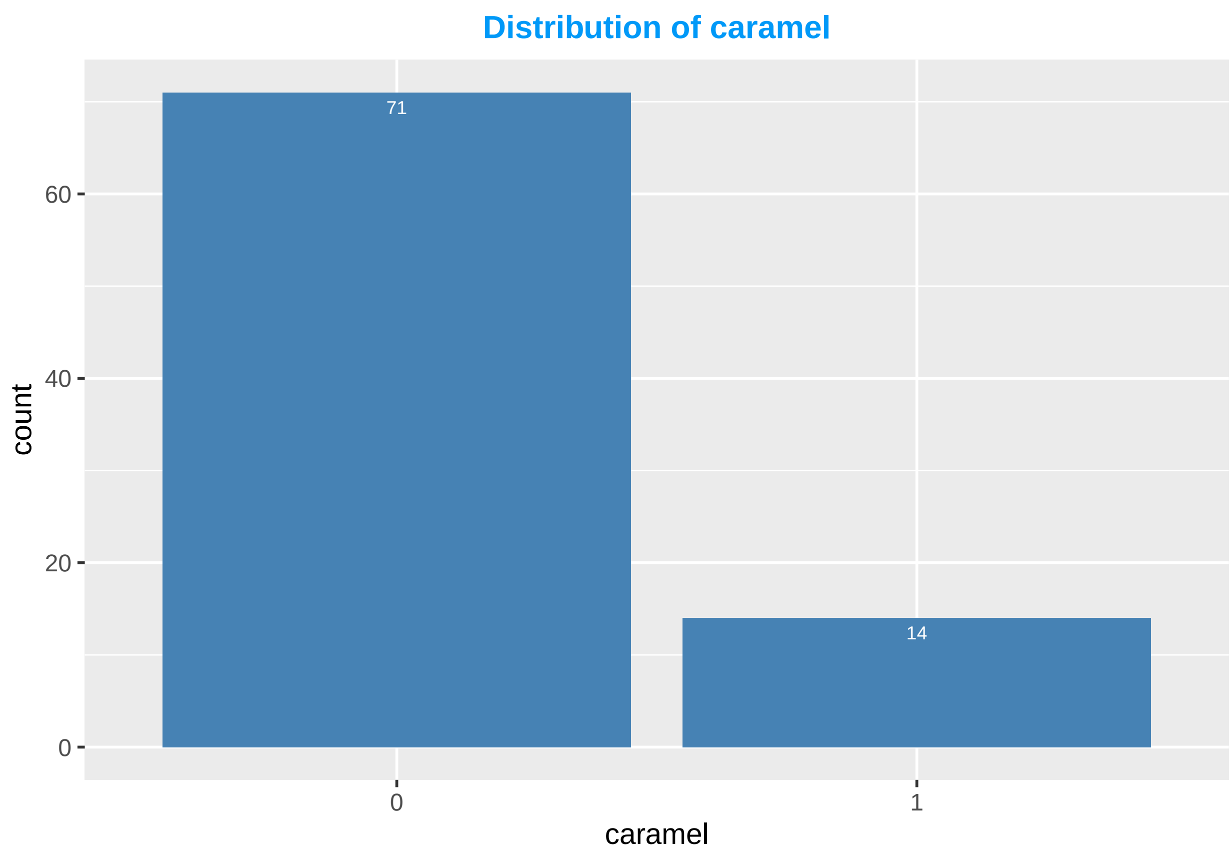
In order to visualize the data, we re-use two methods used in task 2. We have two separate types of visualizations for qualitative data and quantitative data.

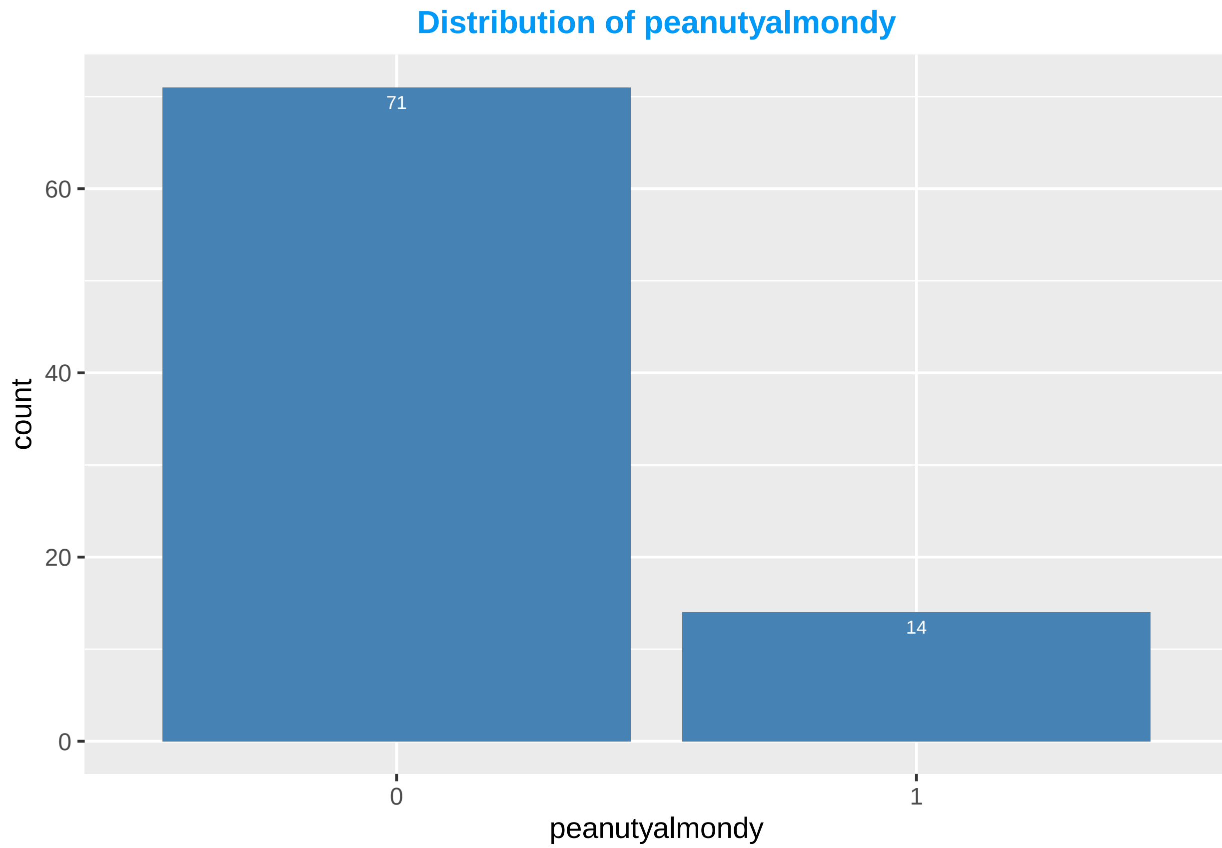
```
visualize <- function(data) {
  for (column_name in names(data)){
    column_type = class(data[,column_name])

    if (column_type == "integer" || column_type == "numeric") {
      #+ fig.height = 3, fig.width = 6
      plot_numerical(data, column_name)
    } else if (column_type == "factor" || column_type == "logical") {
      #+ fig.height = 3, fig.width = 3
      print(plot_bar_chart(data, column_name))
    }
  }
}
```

```r
visualize(candy_data)
```

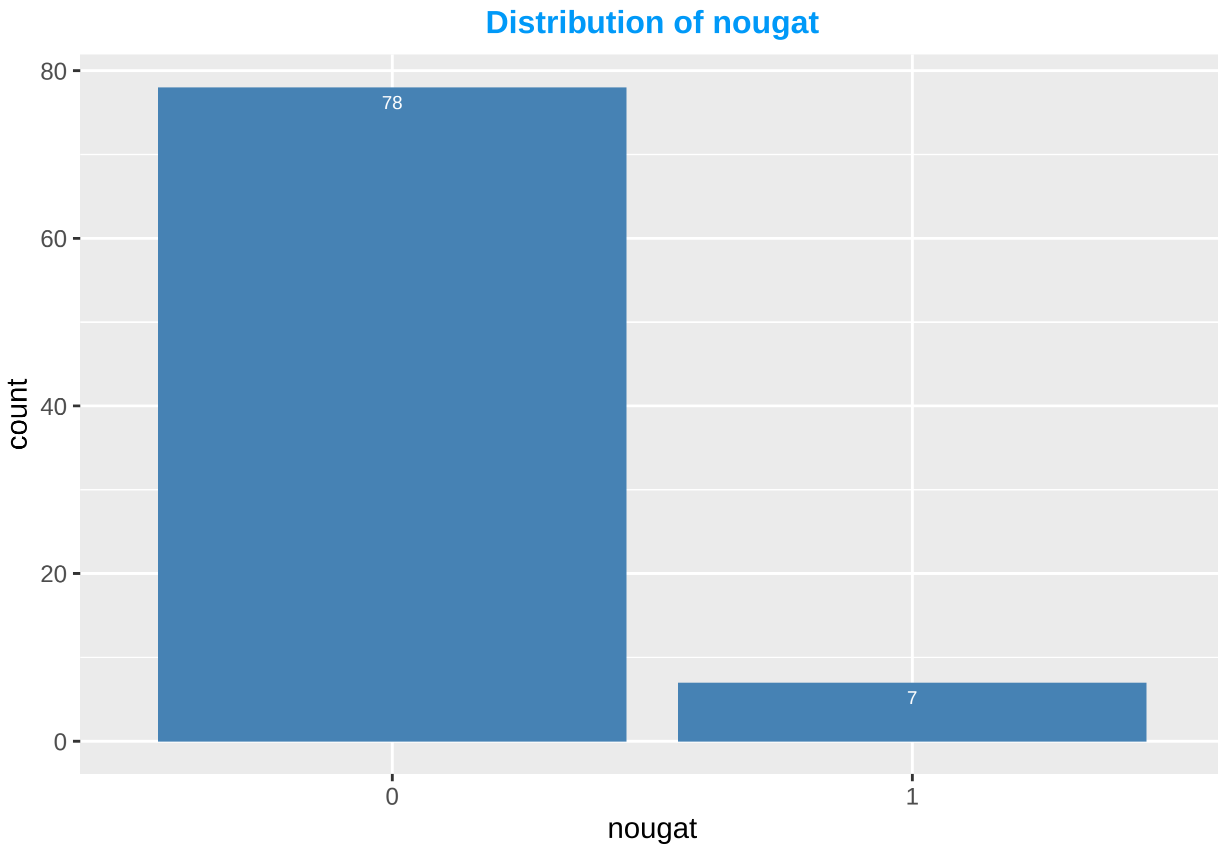


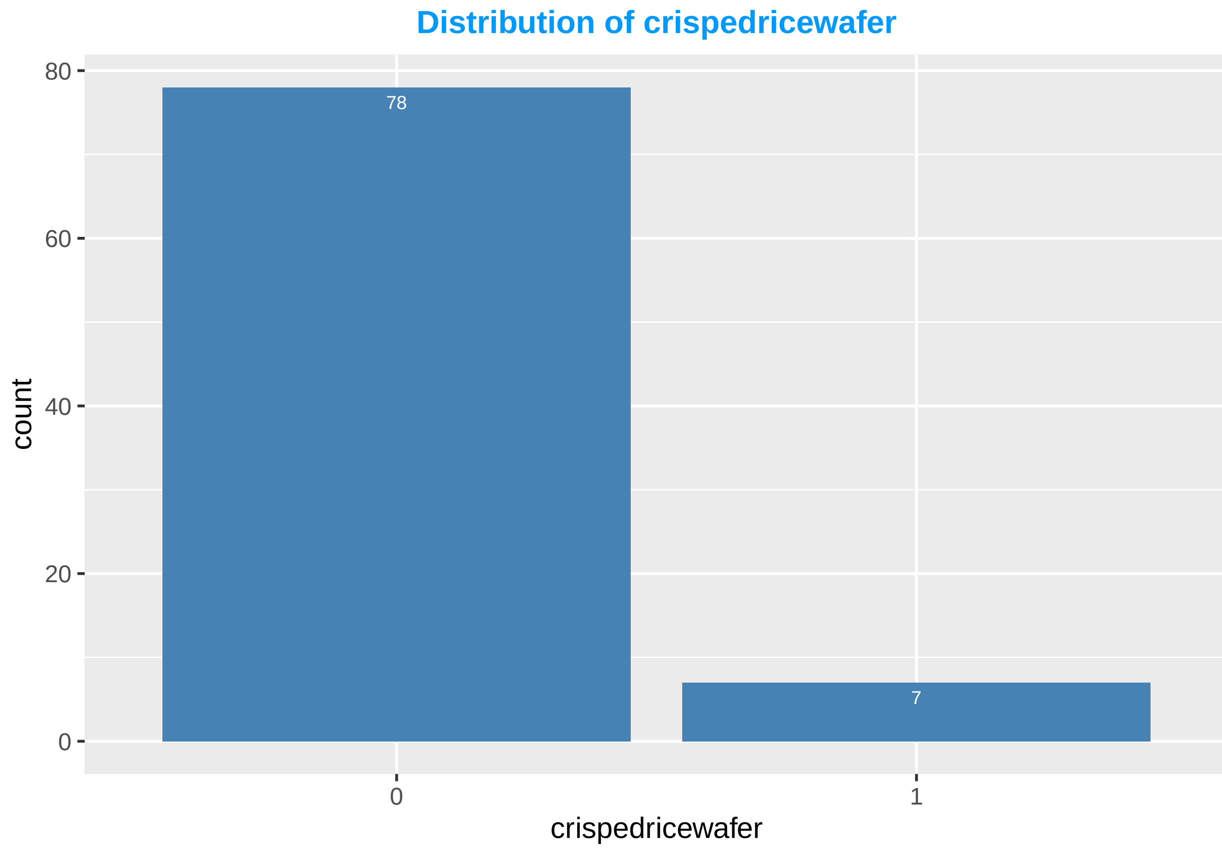

Distribution of fruity
47
38
count
40
30
20
10
0
0
1
fruity

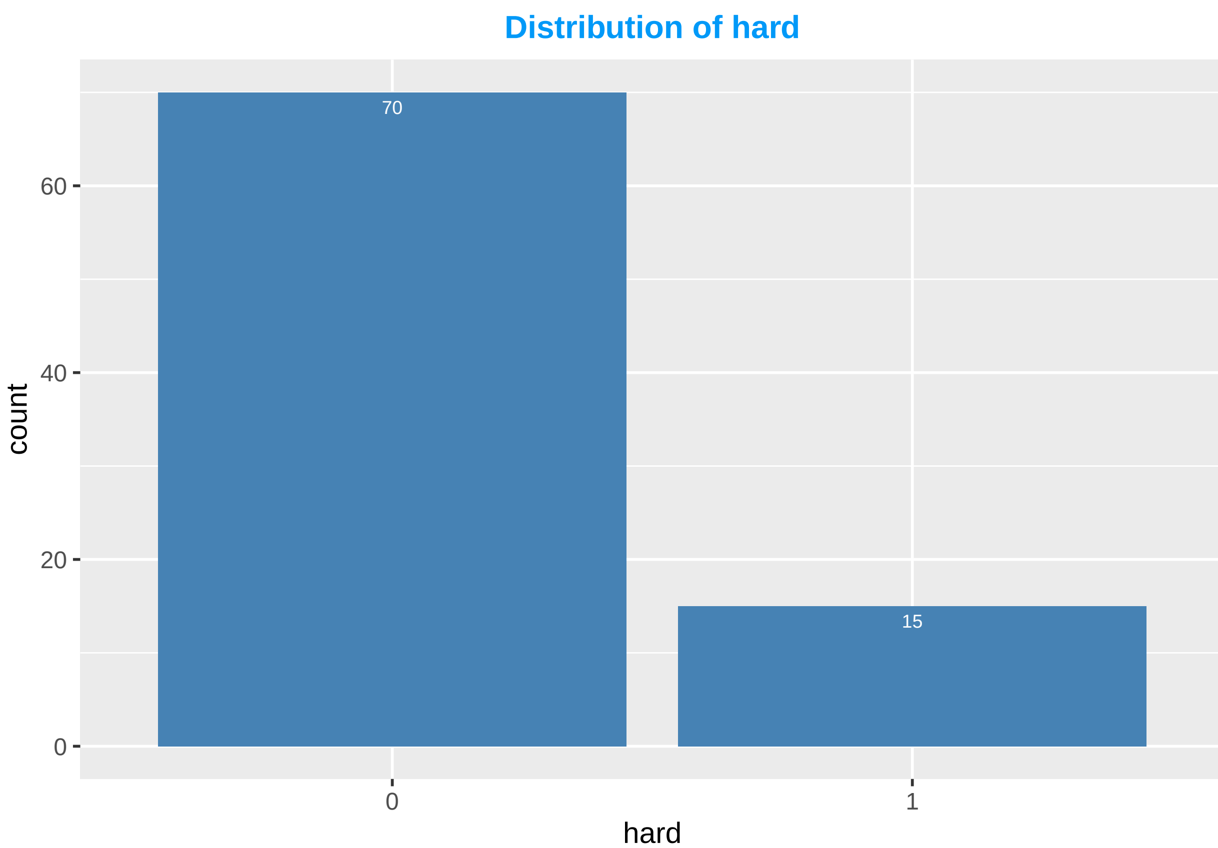

Distribution of caramel
71
14
60
40
20
0
count
0
1
caramel

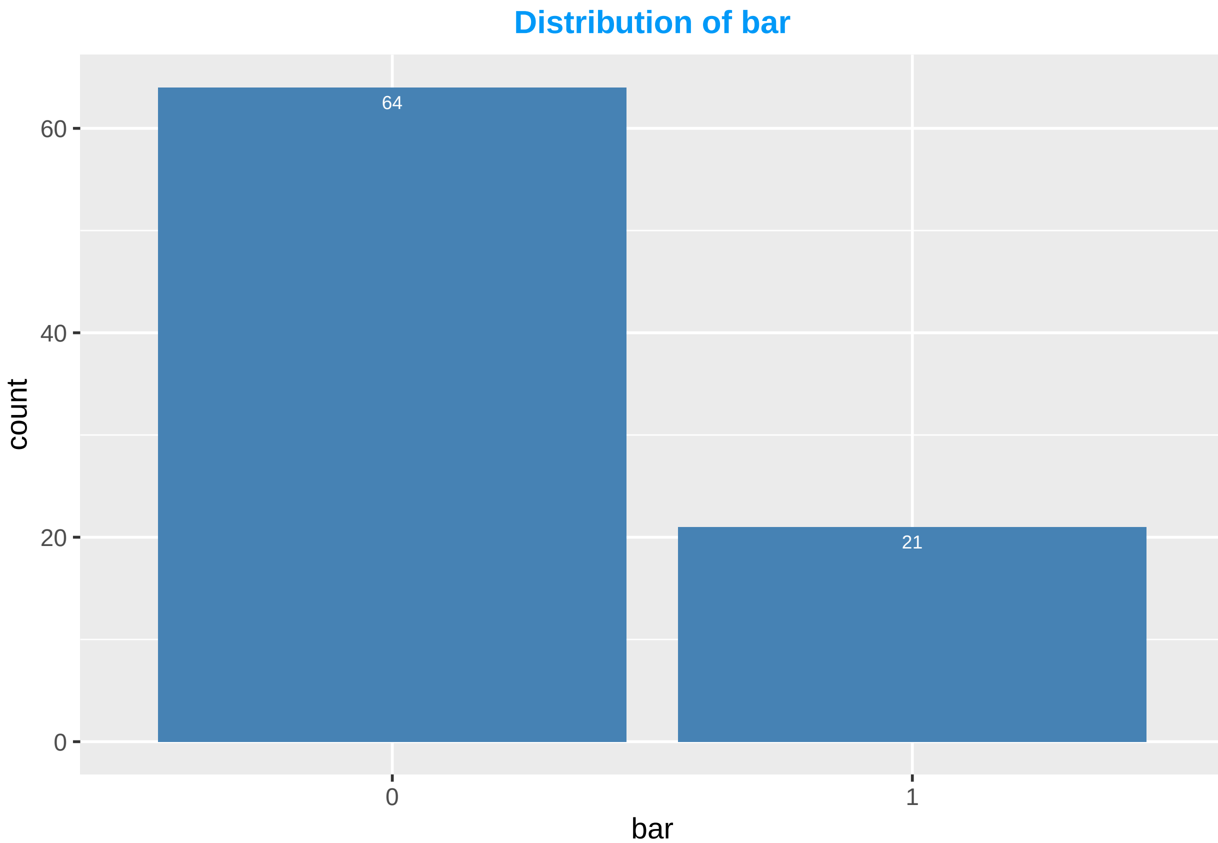


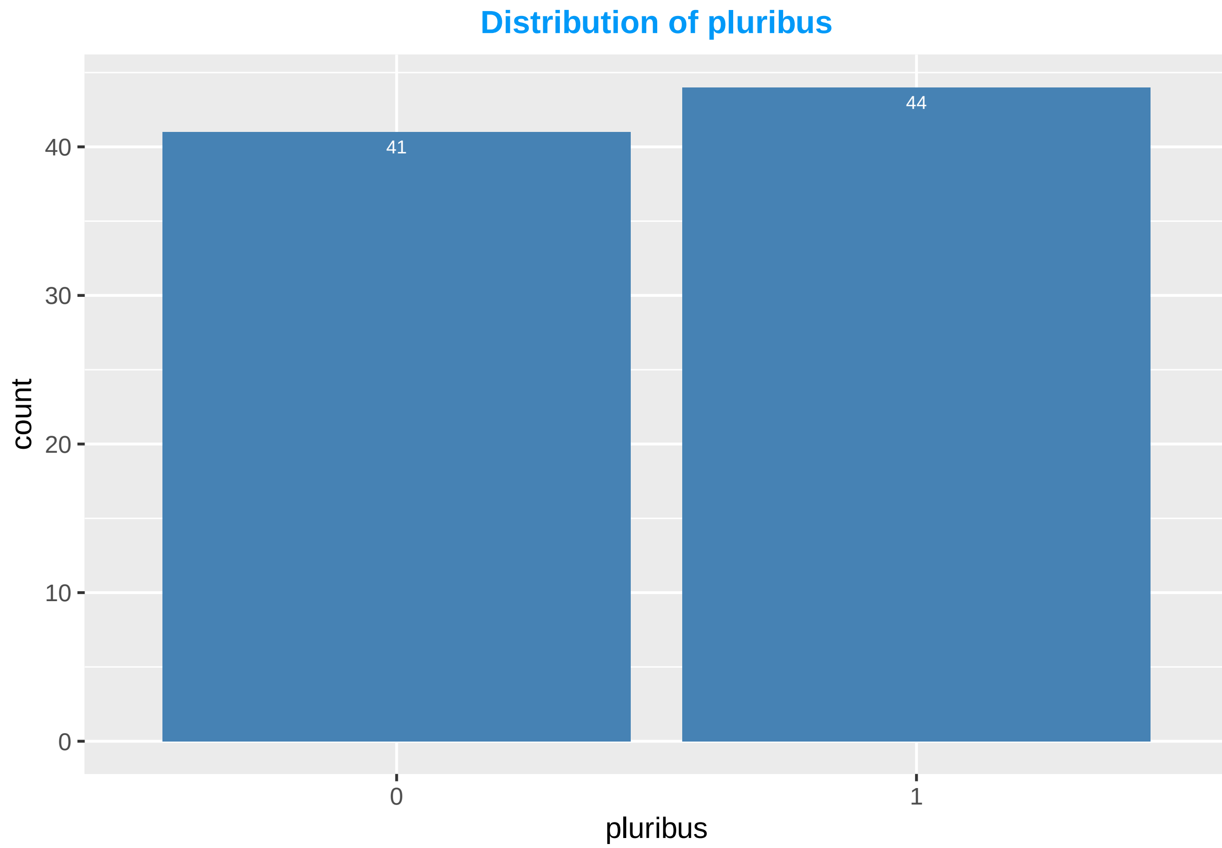
Distribution of peanutyalmondy
71
14
60
40
20
0
count
0
1
peanutyalmondy


Distribution of nougat
80
60
40
20
0
count
78
7
0
1
nougat

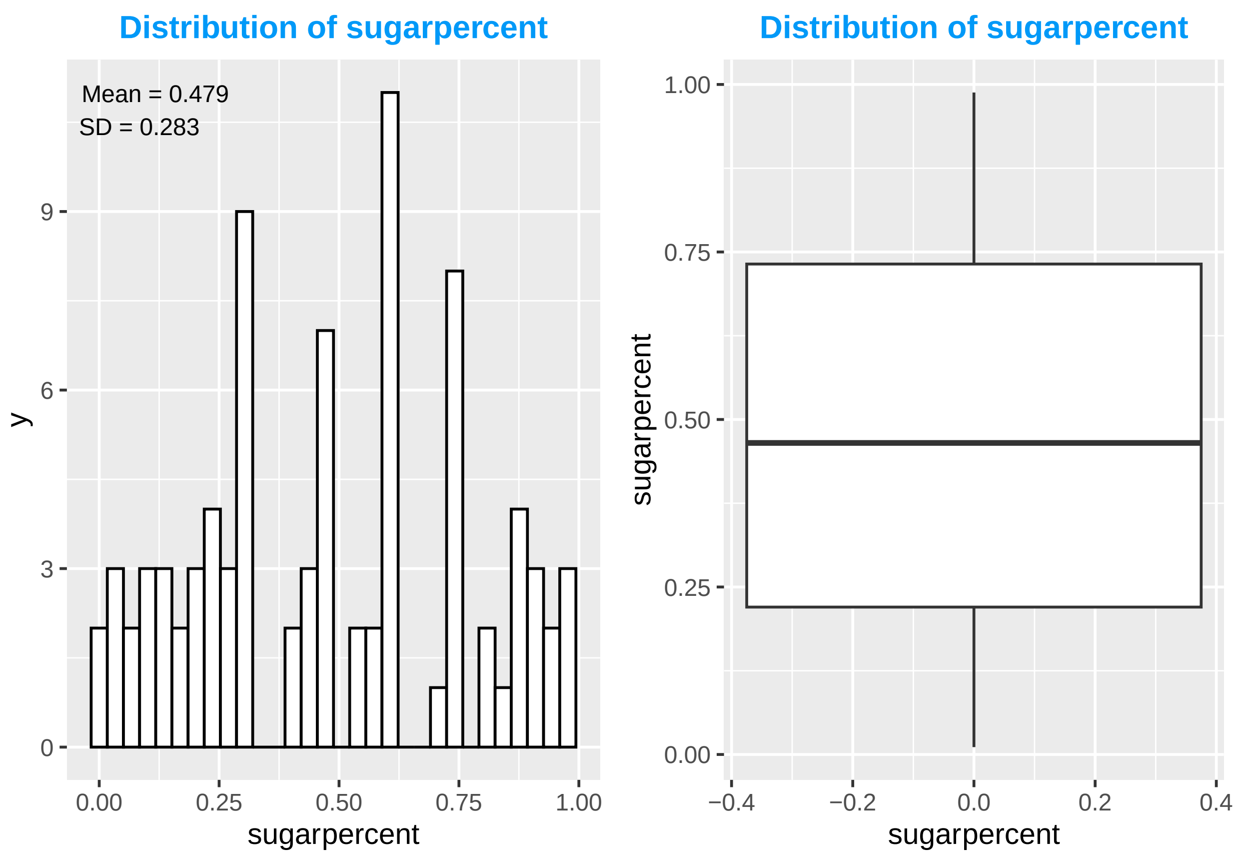

Distribution of crispedricewafer
80
60
40
20
0
count
78
7
0
1
crispedricewafer


Distribution of hard
70
15
60
40
20
0
count
0
1
hard

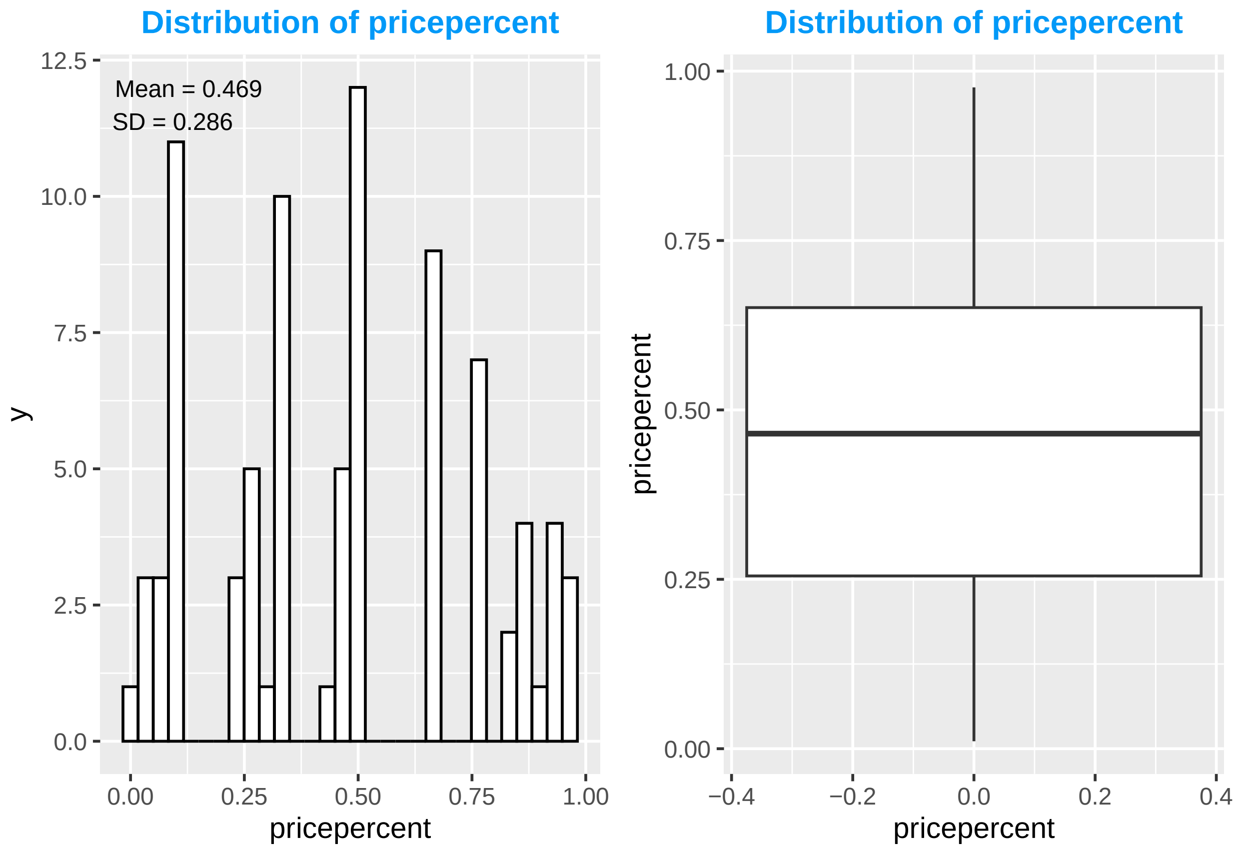

Distribution of bar
64
21
60
40
20
0
count
0
1
bar



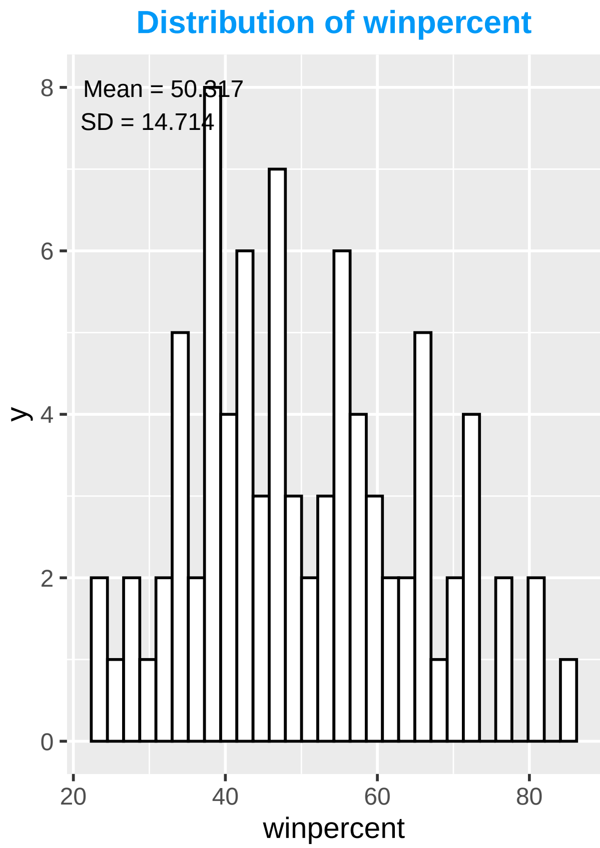

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

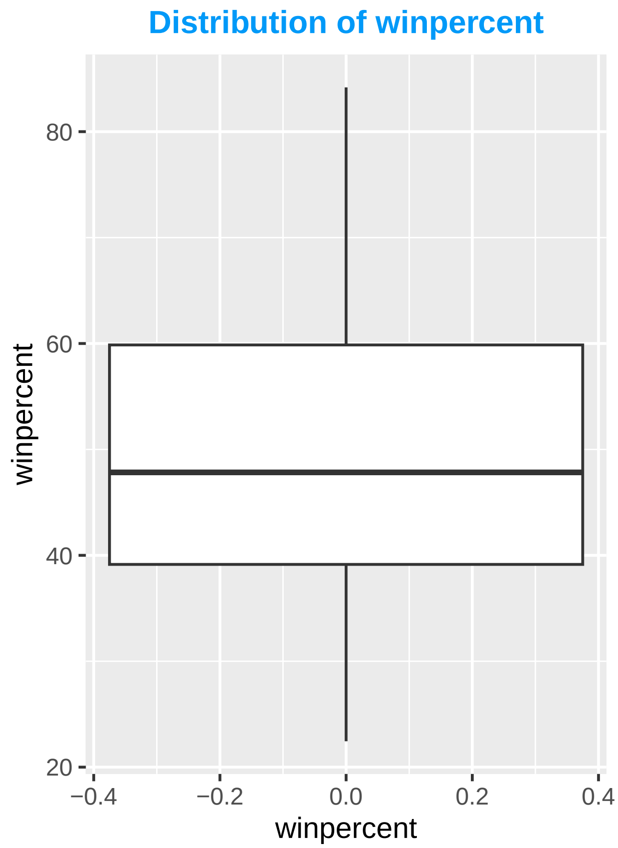



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```





3.5 Predictive Model

Here, we use the same two models we used in Task 2. Depending on the data type, we use either a linear regression model or a logistic regression model. We use every variable able (except for the target variable) to predict the response. And we train/test against the dataset split in an 80/20 ratio.

```
predictive_model <- function(data, variable) {

  response_str <- variable

  variables <- c()
  for (column_name in names(data)) {
    if (column_name == variable) {
      next
    }

    column_type = class(data[,column_name])
    if (column_type == "character") {
      next
    }

    variables <- c(variables, column_name)
  }
  variables_str <- paste(variables, collapse=" + ")
  formula_str <- paste(response_str, " ~ ", variables_str)
  formula <- as.formula(formula_str)
```

```r
split <- sample.split(bank_churn_data$Exited, SplitRatio = 0.8)
train_data <- subset(data, split == TRUE)
test_data <- subset(data, split == FALSE)

column_type = class(data[,variable])
if (column_type == "integer" || column_type == "numeric") {
  model <- lm(formula, data=train_data)
  summary <- summary(model)

  test_predictions <- predict(model, newdata = test_data)
  actual <- test_data[,variable]
  rmse <- sqrt(mean((actual - test_predictions)^2))

  rss <- sum((test_predictions - actual_tenure)^2)
  tss <- sum((actual_tenure - mean(actual_tenure))^2)
  rsq_test <- 1 - (rss / tss)

  pva_plot <- ggplot(data = test_data, aes(x = test_data[,variable], y = test_predictions)) +
    geom_point() +
    geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +
    labs(title = "Actual vs. Predicted",
         x = glue("Actual"),
         y = "Predicted") +
    theme(
      plot.title = element_text(color = "#0099f8", size = 12, face = "bold", hjust = 0.5),
    )

  return (list(
    summary=summary,
    rmse = rmse,
    rsq_test = rsq_test,
    plot = pva_plot
  ))

} else if (column_type == "factor" || column_type == "logical") {
  model <- glm(formula, data=train_data, family=binomial(link = "logit"))
  summary <- summary(model)

  test_probabilities <- predict(model, newdata = test_data, type = "response")
  predicted <- ifelse(test_probabilities > 0.5, 1, 0)

  actual <- test_data[,variable]
  conf_matrix <- table(Actual = actual, Predicted = predicted)

  TP <- conf_matrix[2, 2]
  TN <- conf_matrix[1, 1]
  FP <- conf_matrix[1, 2]
  FN <- conf_matrix[2, 1]

  accuracy <- (TP + TN) / sum(conf_matrix)
  precision <- TP / (TP + FP)
  sensitivity <- TP / (TP + FN)
  specificity <- TN / (TN + FP)
```

```r
    f1_score <- 2 * (precision * sensitivity) / (precision + sensitivity)

    roc_curve <- roc(response = actual, predictor = test_probabilities)
    roc_plot <- plot(roc_curve, main = "ROC Curve", print.auc = TRUE)

    return(list(
      summary=summary,
      conf_matrix=conf_matrix,
      accuracy=accuracy,
      precision=precision,
      sensitivity=sensitivity,
      specificity=specificity,
      f1_score=f1_score,
      plot=roc_plot
    ))
  } else {
    stop("Invalid response variable")
  }
}
```

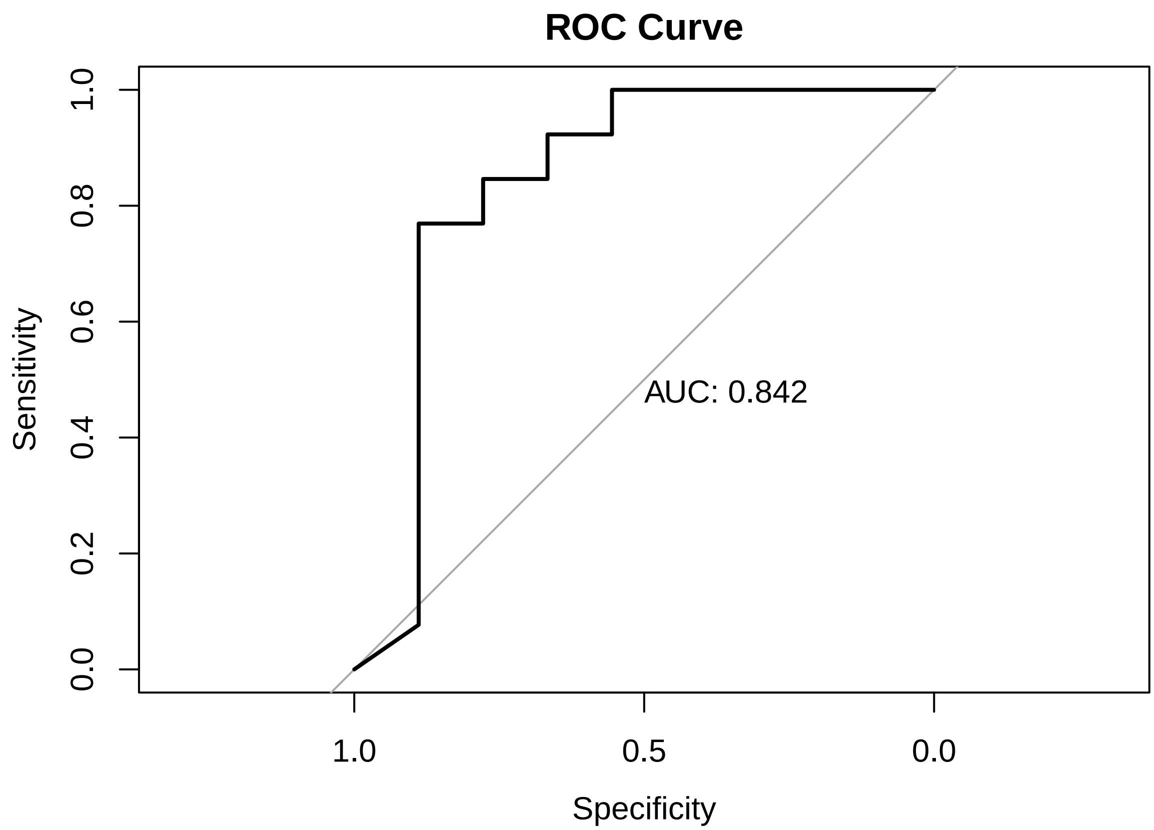
For qualitative data, we print the
- Summary of the model
- Confusion matrix
- Accuracy, precision, sensitivity, specificity, f1 score
- ROC curve
- AUC value

```r
results <- predictive_model(candy_data, "chocolate")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

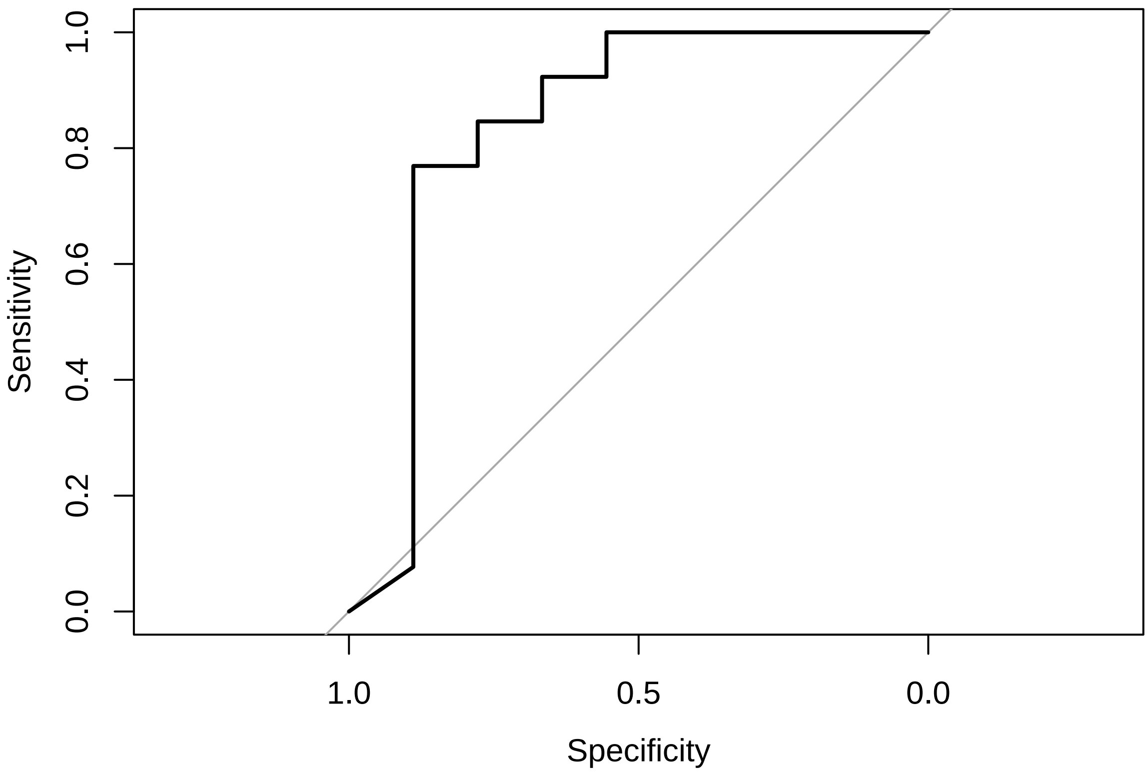



```r
print(results)
```

```
## $summary
##
## Call:
## glm(formula = formula, family = binomial(link = "logit"), data = train_data)
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -13.7408     6.1805  -2.223   0.0262 *
## fruity1             -7.6130     3.1316  -2.431   0.0151 *
## caramel1            -6.0297     3.7853  -1.593   0.1112
## peanutyalmondy1      9.8303  7194.4580   0.001   0.9989
## nougat1              2.5016 18056.9568   0.000   0.9999
## crispedricewafer1   -1.9983 15738.6213   0.000   0.9999
## hard1                3.8936     2.5984   1.498   0.1340
## bar1                17.7098 13645.3058   0.001   0.9990
## pluribus1           -0.5768     1.7139  -0.337   0.7365
## sugarpercent        -0.7454     2.9967  -0.249   0.8036
## pricepercent         3.3379     2.5367   1.316   0.1882
## winpercent           0.3088     0.1382   2.234   0.0255 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 83.731  on 62  degrees of freedom
## Residual deviance: 14.895  on 51  degrees of freedom
##   (7937 observations deleted due to missingness)
## AIC: 38.895
##
## Number of Fisher Scoring iterations: 20
##
##
## $conf_matrix
##       Predicted
## Actual  0  1
##      0  7  2
##      1  2 11
##
## $accuracy
## [1] 0.8181818
##
## $precision
## [1] 0.8461538
##
## $sensitivity
## [1] 0.8461538
##
## $specificity
## [1] 0.7777778
##
## $f1_score
## [1] 0.8461538
##
## $plot
##
## Call:
## roc.default(response = actual, predictor = test_probabilities)
##
## Data: test_probabilities in 9 controls (actual 0) < 13 cases (actual 1).
## Area under the curve: 0.8419
```

```r
plot(results$plot)
```

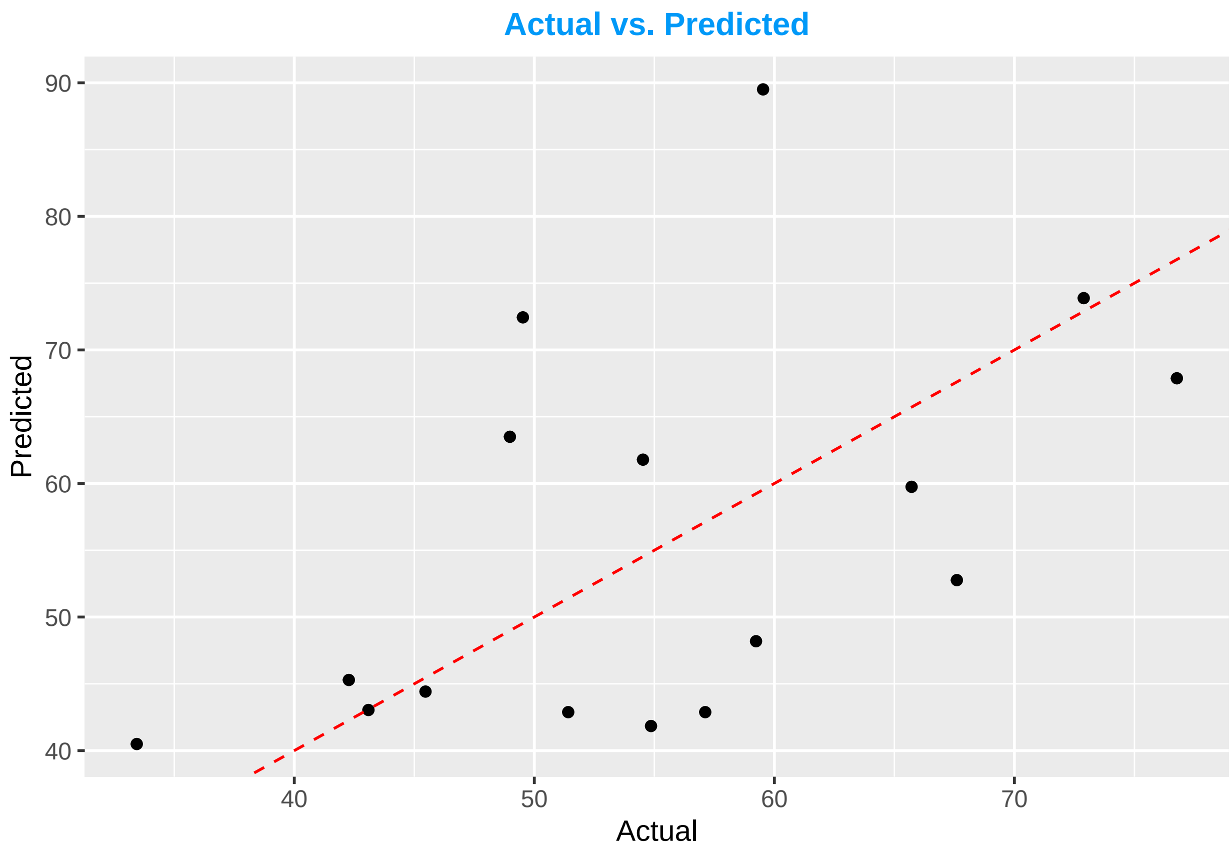


For quantitative data, we print the
- Summary of the model
- RMSE
- R^2 value
- Scatter plot of predicted vs actual values

```r
results <- predictive_model(candy_data, "winpercent")
print(results)
```

```
## $summary
##
## Call:
## lm(formula = formula, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.303  -6.731   0.408   5.949  25.346
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        33.119      4.508   7.347 8.38e-10 ***
## chocolate1         21.161      4.455   4.750 1.42e-05 ***
## fruity1             9.990      4.112   2.430 0.018290 *
## caramel1            4.952      4.308   1.150 0.255099
## peanutyalmondy1    14.825      4.207   3.524 0.000847 ***
## nougat1            -0.762      6.681  -0.114 0.909594
## crispedricewafer1  17.410      7.156   2.433 0.018134 *
```

```
## hard1                -4.318      3.649  -1.183 0.241629
## bar1                 -2.697      6.095  -0.443 0.659761
## pluribus1            -1.107      3.312  -0.334 0.739341
## sugarpercent         10.511      5.009   2.098 0.040311 *
## pricepercent         -8.619      6.062  -1.422 0.160548
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.62 on 57 degrees of freedom
##   (7931 observations deleted due to missingness)
## Multiple R-squared:  0.5903, Adjusted R-squared:  0.5112
## F-statistic: 7.464 on 11 and 57 DF,  p-value: 9.55e-08
##
##
## $rmse
## [1] NA
##
## $rsq_test
## [1] NA
##
## $plot

## Warning: Removed 1984 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



3.6 R Shiny App

This is a fairly basic Shiny app with the following steps. - The user uploads a file
- We display the first 5 rows as well as the qualitative/quantitative variables
- The user can click on a button to calculate and view the outliers
- The user can click on a button to visualize the data
- The user can select a variable, and click on a button to run a model and view metrics/plots.

You can view a screenshot of the application here.

```r
ui <- fluidPage(
  titlePanel("Analyze dataset"),
  fileInput(inputId = "upload", label = "Upload dataset", accept = "text/csv"),
  hr(),
  tableOutput("contents"),

  conditionalPanel(
    condition = "output.contents",
    textOutput("quantitative_vars"),
    textOutput("qualitative_vars"),

    hr(),
    selectInput(inputId = "outlier_removal_method", label = "Select outlier removal method", choices = c
    actionButton("detect_outliers", "Detect outliers"),
    verbatimTextOutput("outliers"),

    hr(),
    actionButton("visualize", "Visualize"),
    verbatimTextOutput("visualization"),

    hr(),
    uiOutput("variable_select_ui"),
    actionButton("run_model", "Run Model"),
    verbatimTextOutput("model"),
    plotOutput("plot")

  ),

  conditionalPanel(
    condition = "output.outliers",
  )
)

server <- function(input, output) {
  reactive_dataset  <- reactive({
    req(input$upload)

    tryCatch(
      {
        dataset <- read.csv(input$upload$datapath)
        for (column_name in names(dataset)){
          count = length(unique(dataset[,column_name]))
          if (count <= 3) {
            dataset[,column_name] <- as.factor(dataset[,column_name])
            next
```

```r
        }
      }

      dataset <- handle_missing_values(dataset)
      return (dataset)
    },
    error = function(e) {
      showNotification(paste("Error reading file:", e$message), type = "error")
      stop(safeError(e))
    }
  )
})

output$contents <- renderTable({
  dataset <- reactive_dataset()
  req(dataset)

  return (head(dataset))
})

output$quantitative_vars <- renderText({
  dataset <- reactive_dataset()
  req(dataset)

  variable_types <- quantitative_or_qualitative(dataset)
  glue("Quantitative variables: {paste(variable_types$quantitatve, collapse=', ')}")
})

output$qualitative_vars <- renderText({
  dataset <- reactive_dataset()
  req(dataset)

  variable_types <- quantitative_or_qualitative(dataset)
  glue("Qualitative variables: {paste(variable_types$qualitative, collapse=', ')}")
})

output$variable_select_ui <- renderUI({
  dataset <- reactive_dataset()
  req(dataset)

  variables <- colnames(dataset)
  selectInput("variable_select", "Select Variable:", choices = variables)
})

outlier_results <- eventReactive(input$detect_outliers, {
  dataset <- reactive_dataset()
  req(dataset)

  method <- input$outlier_removal_method
  req(method)

  return (detect_outliers(dataset, method))
})
```

```r
  output$outliers <- renderPrint({
    print(outlier_results())
  })

  outlier_visualization_results <- eventReactive(input$visualize, {
    dataset <- reactive_dataset()
    req(dataset)

    return (visualize(dataset))
  })

  output$visualization <- renderPrint({
    print(outlier_visualization_results())
  })

  model_results <- eventReactive(input$run_model, {
    dataset <- reactive_dataset()
    req(dataset)

    variable <- input$variable_select
    req(variable)

    print(variable)
    return (predictive_model(dataset, variable))
  })

  output$model <- renderPrint({
    print(model_results())
  })

  output$plot <- renderPlot({
    results <- model_results()
    plot(results$plot)
  })

}

shinyApp(ui = ui, server = server)
```