

CMM703 - Data Analysis Coursework

Kaneel Dias

2025-04-17

```
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()  
}
```

```

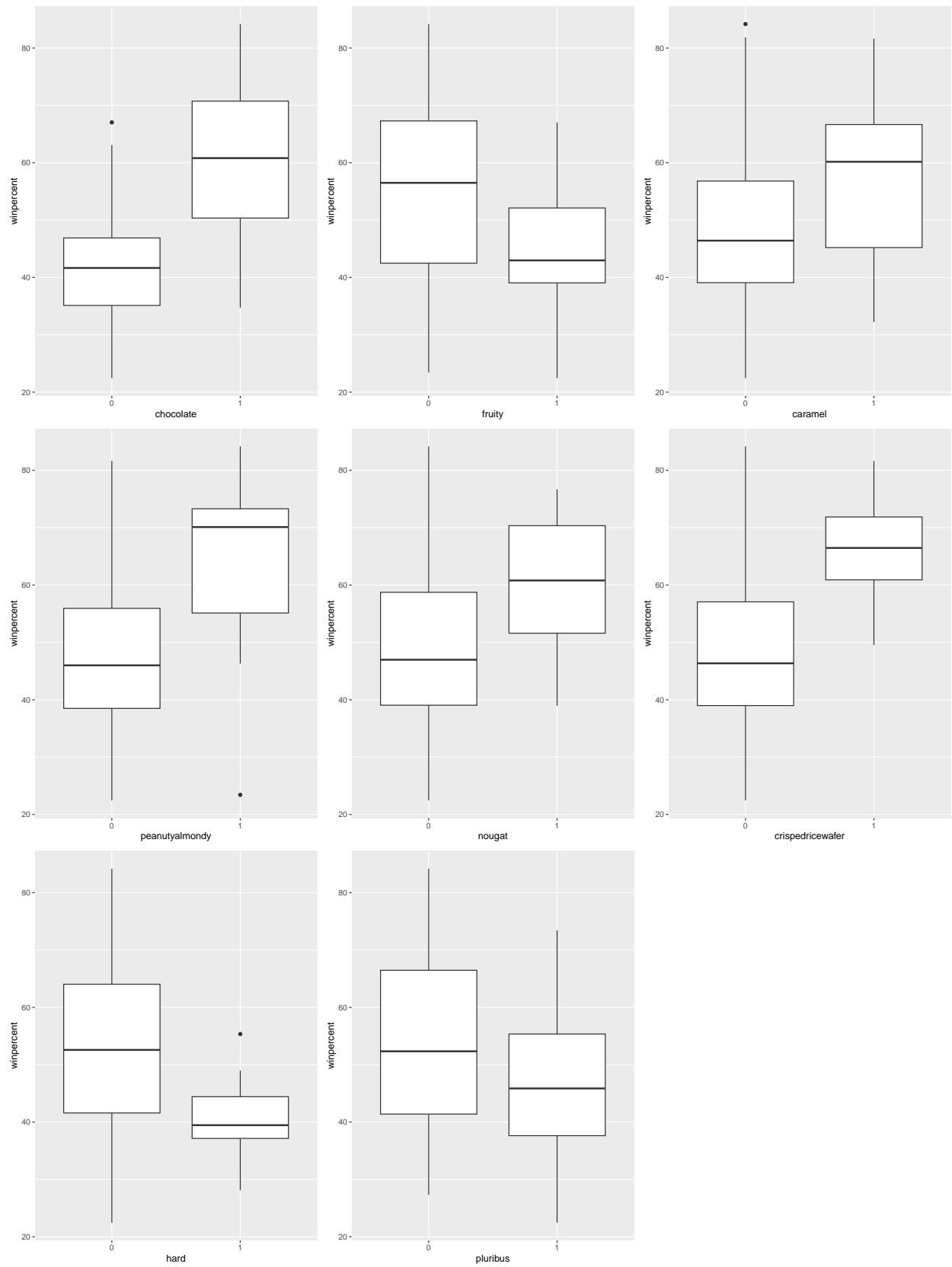
    return(plot)
}

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, crispedricewafer_plot, hard_plot, pluribus_plot)
}

plot_all_categorical_boxplots(candy_data)

```



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

```
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,
```

```

    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

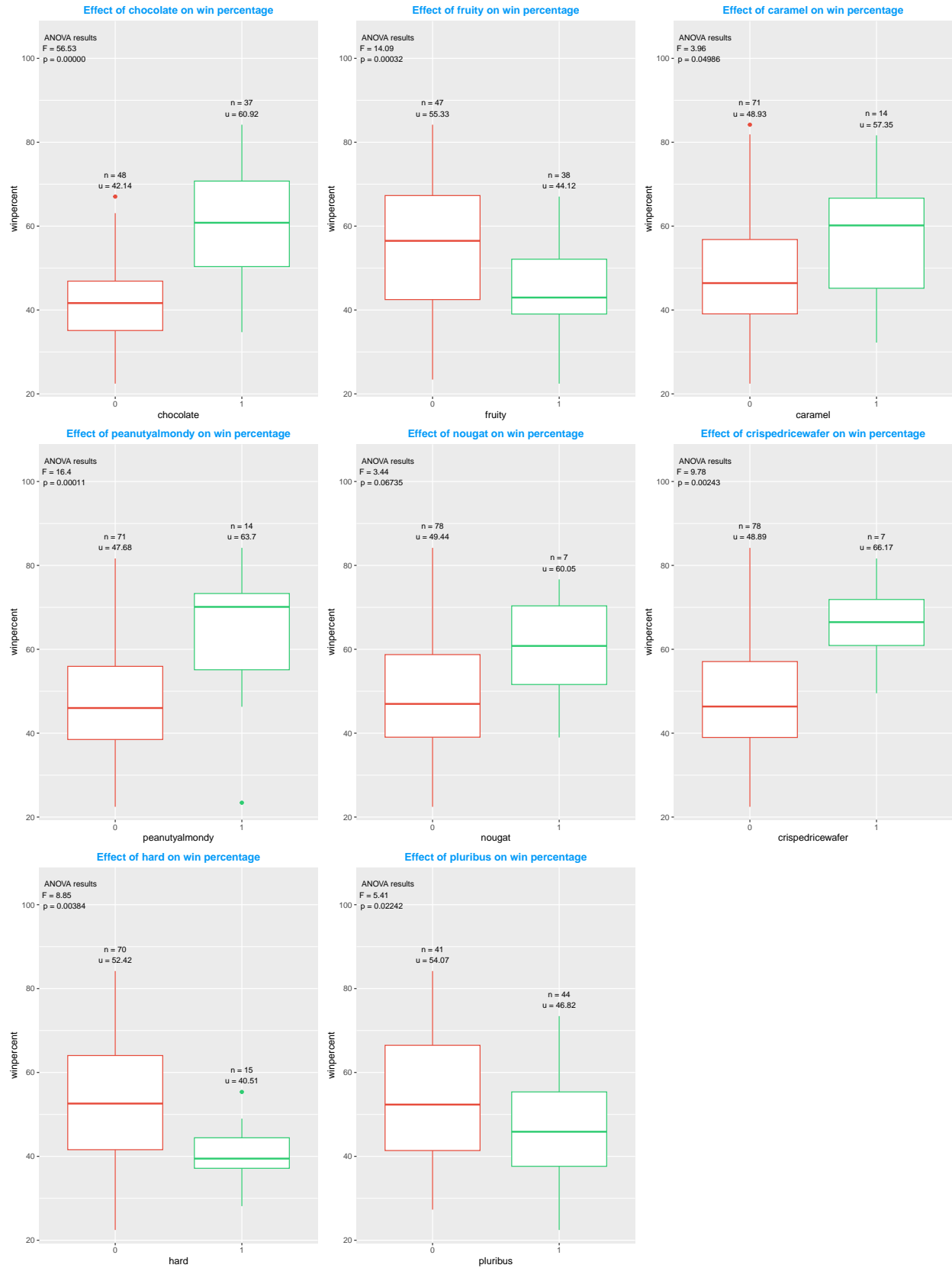
  return(plot)
}

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, crispedricewafer_plot, hard_plot, pluribus_plot)
}

plot_all_categorical_boxplots_improved(candy_data)

```



1.1.2 Insights

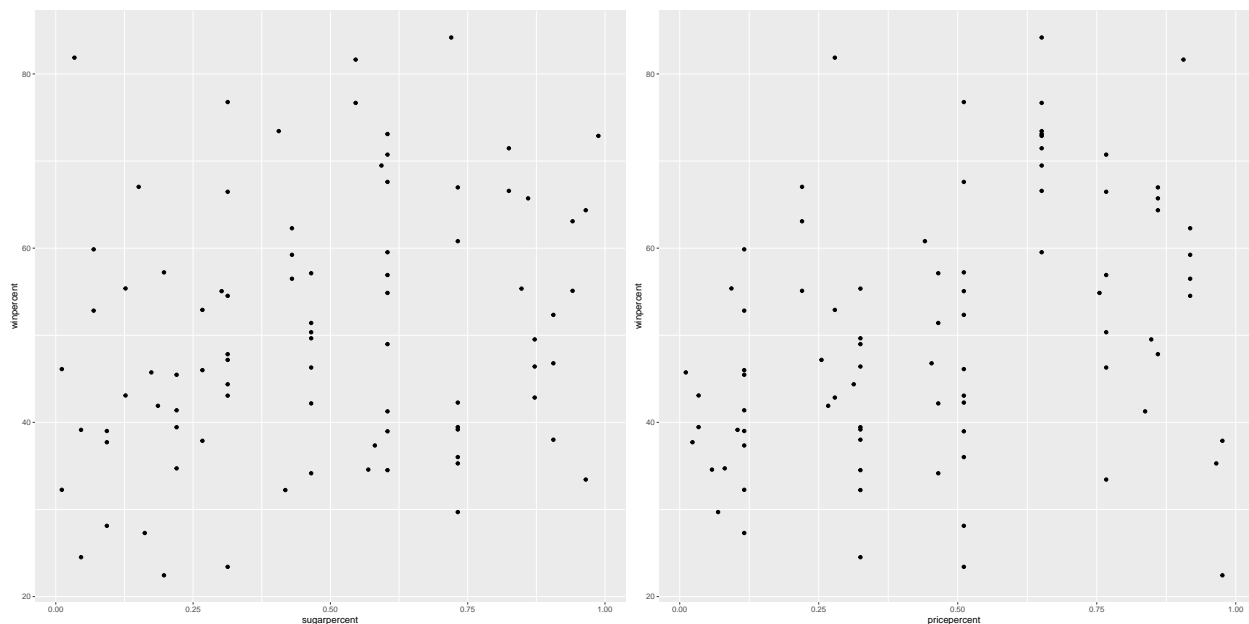
Here we can see that the `chocolate` variable has the highest effect on `winpercent` (highest f-value), and it has a very low p-value as well, indicating that it is most likely to be having an effect. On the other hand, `nougat` has a p-value > 0.05 , (as well as a low f-value) which indicates that its 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 (coefficient of determination in the chart) to determine whether 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"
  )

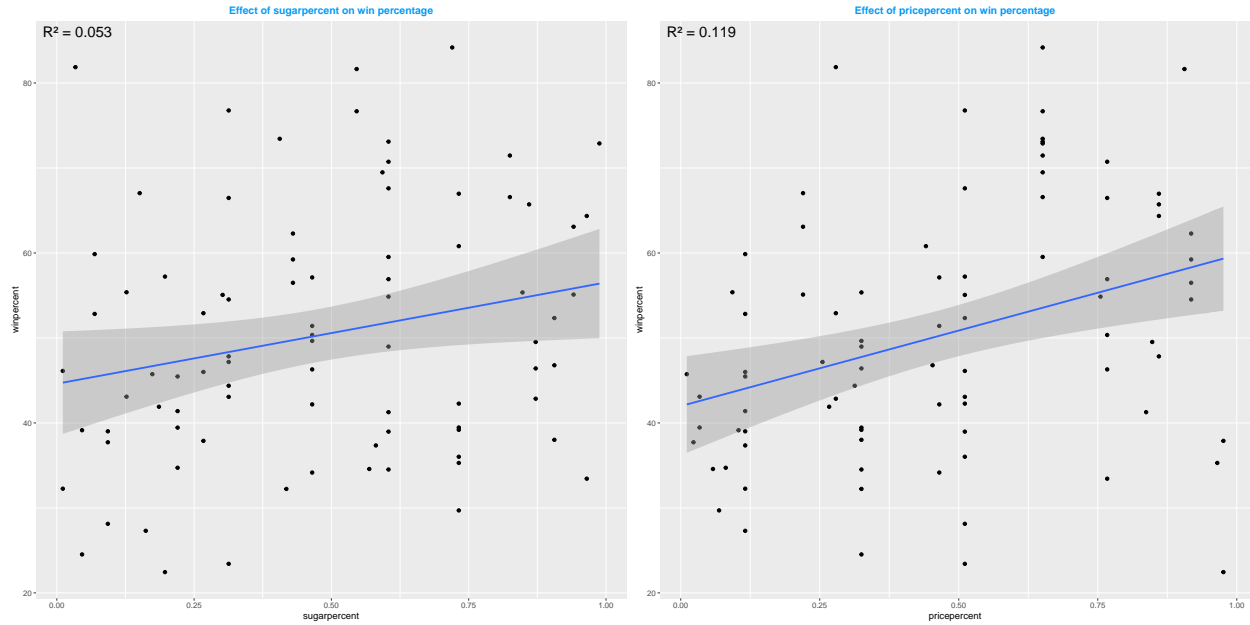
  return (plot)
}

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 there is no significant correlation present.

TASK 2: Bank Churn

2.1 Exploratory Data Analysis

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

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

```
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
## 3rd Qu.:15753234
## Max.   :15815690
##      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.

```
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 = 3)}")

  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)
}

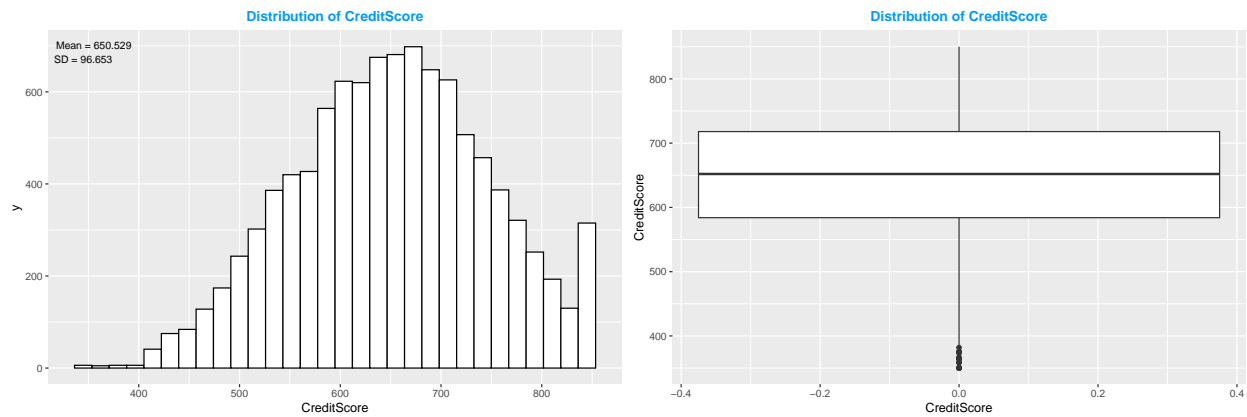
plot_numerical <- function(data, variable) {
  histogram <- suppressMessages(plot_histogram(data, variable))
  boxplot <- plot_boxplot(data, variable)

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

plot_numerical(bank_churn_data, "CreditScore")

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

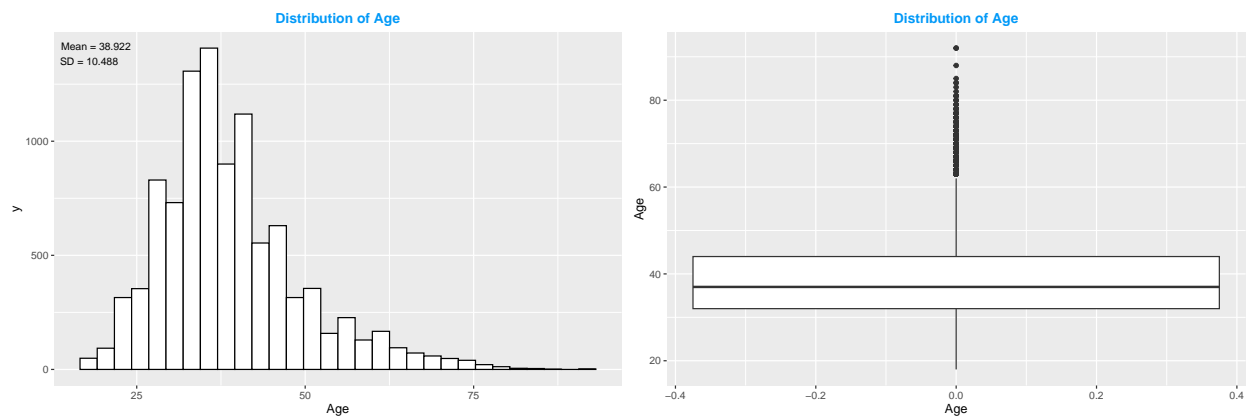
```



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

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

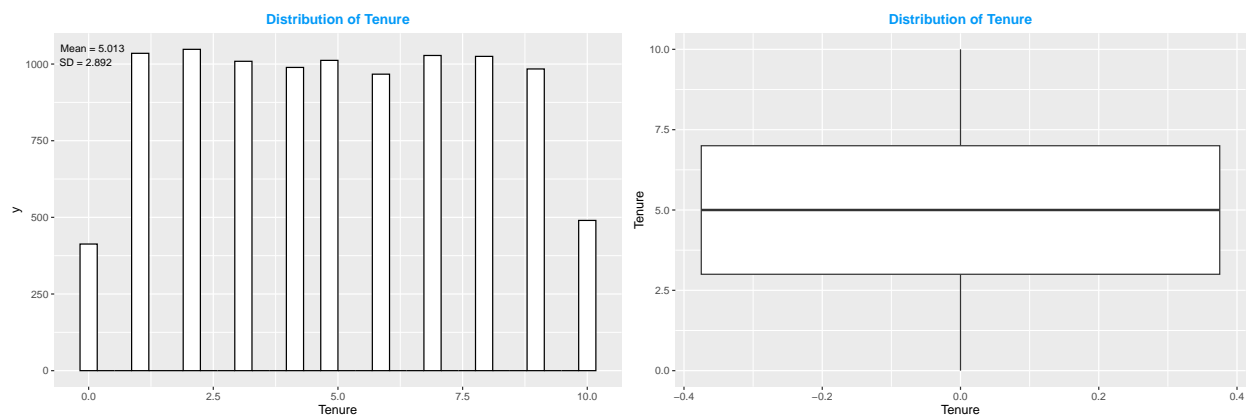
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



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

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

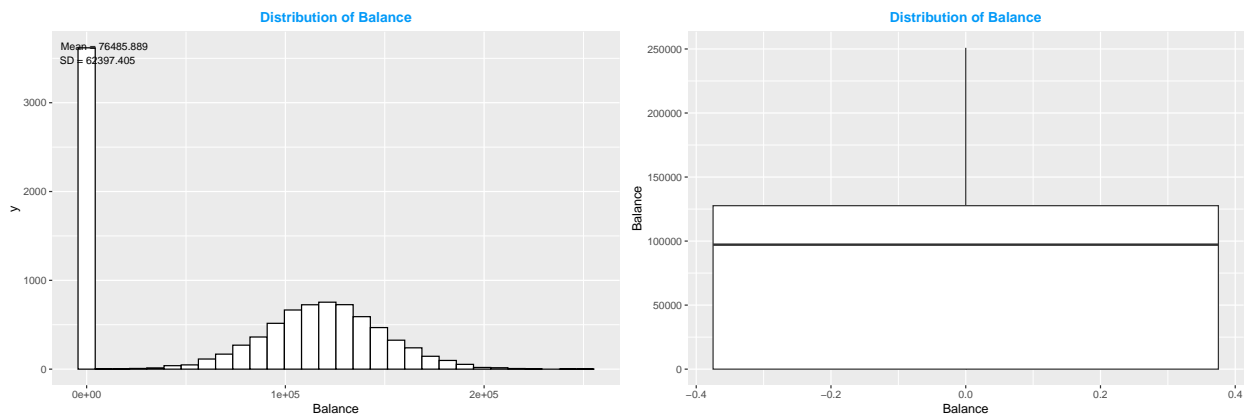
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



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

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

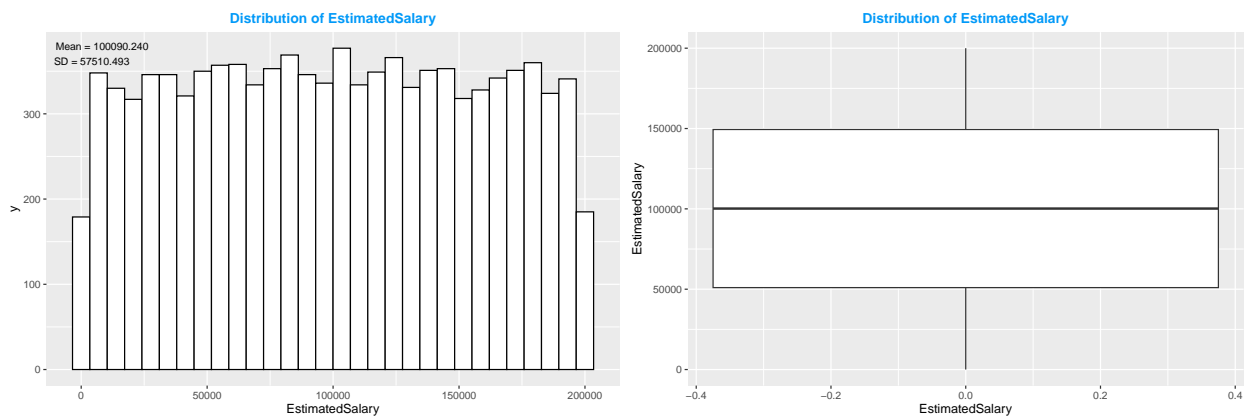
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



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

```
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.

```
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.

```
plot_bar_chart <- function(data, variable) {
  plot <- 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),
  )
return (plot)
}

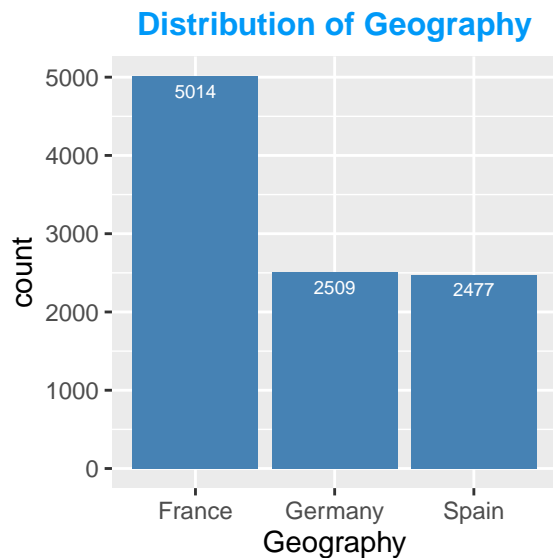
```

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

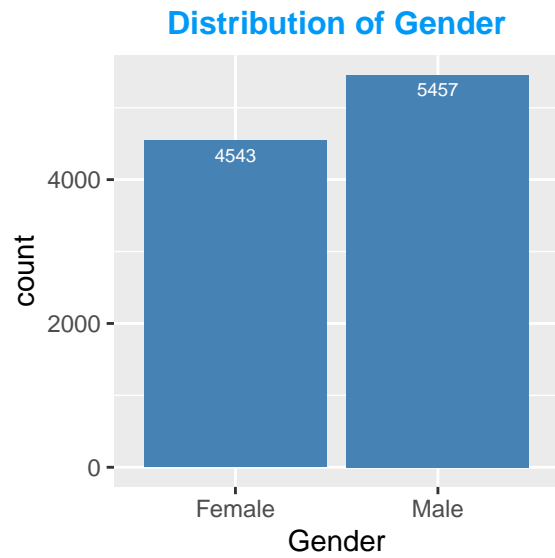
```

## 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.

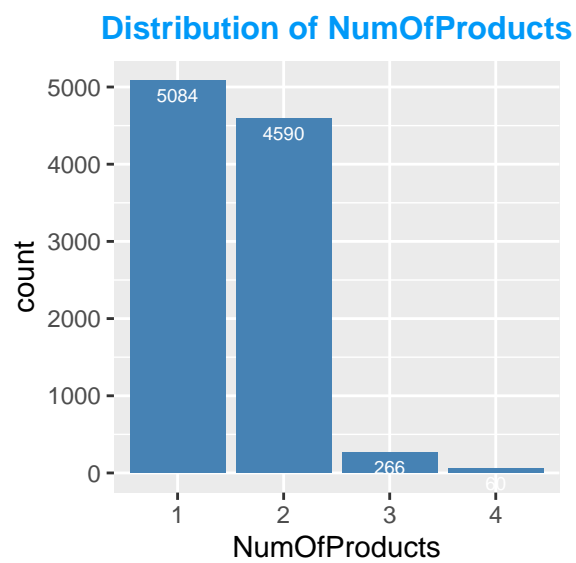
```



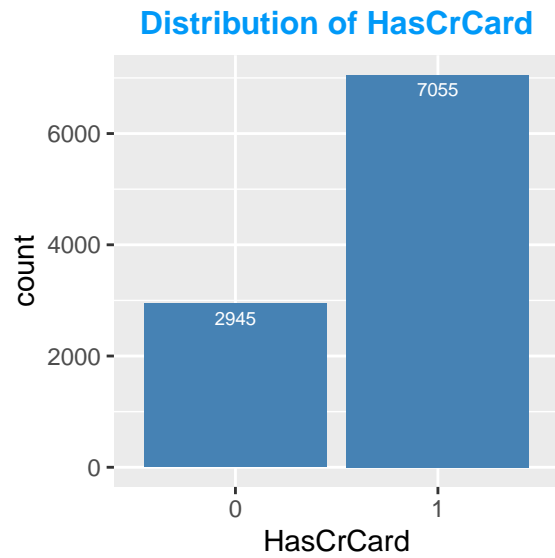
```
plot_bar_chart(bank_churn_data, "Gender")
```



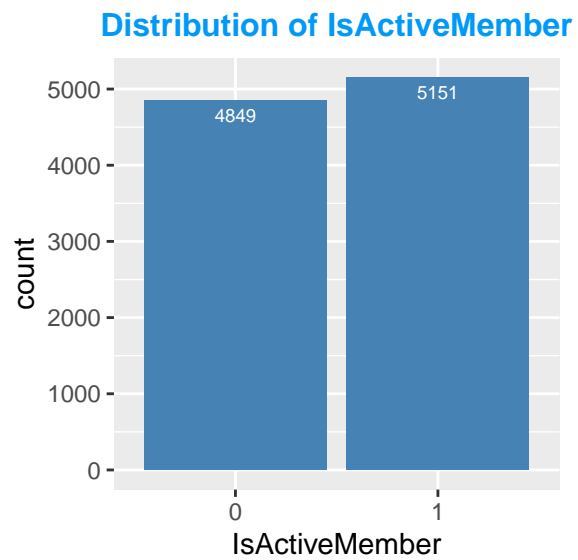
```
plot_bar_chart(bank_churn_data, "NumOfProducts")
```



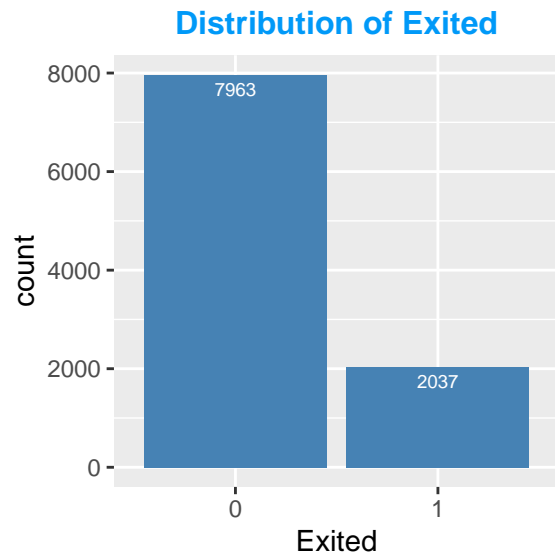
```
plot_bar_chart(bank_churn_data, "HasCrCard")
```



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



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

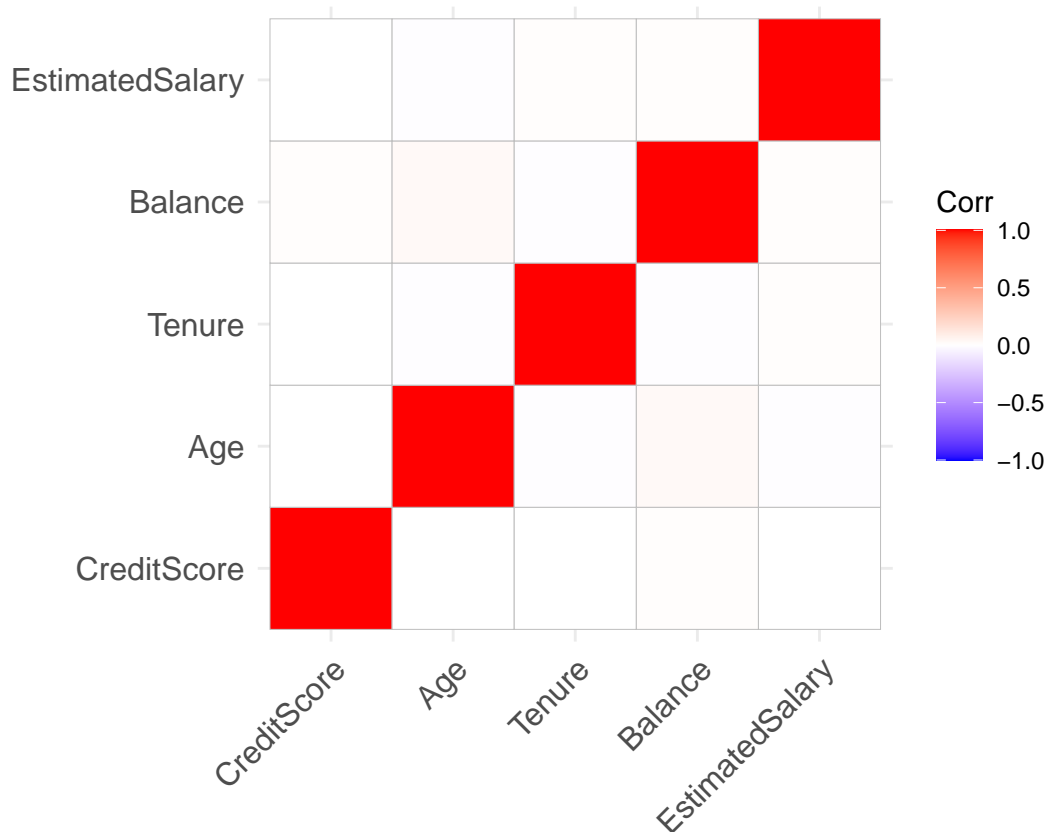
2.1.3 Identifying Numeric-Numeric Correlations

We can calculate the correlations between numeric variables.

```
numeric_only <- subset(bank_churn_data, select=c("CreditScore", "Age", "Tenure", "Balance", "EstimatedSalary"))
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_columns[j]])
    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
  }
}

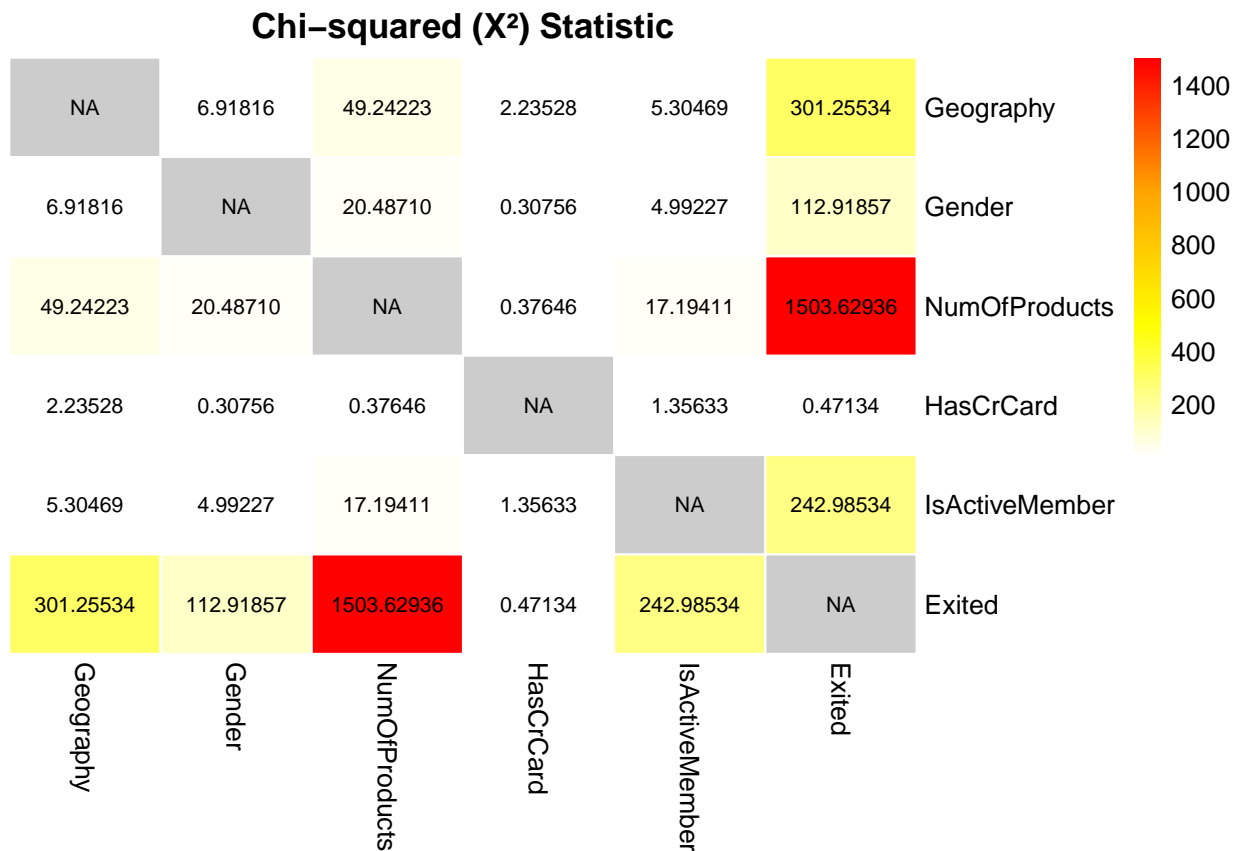
```

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.

```

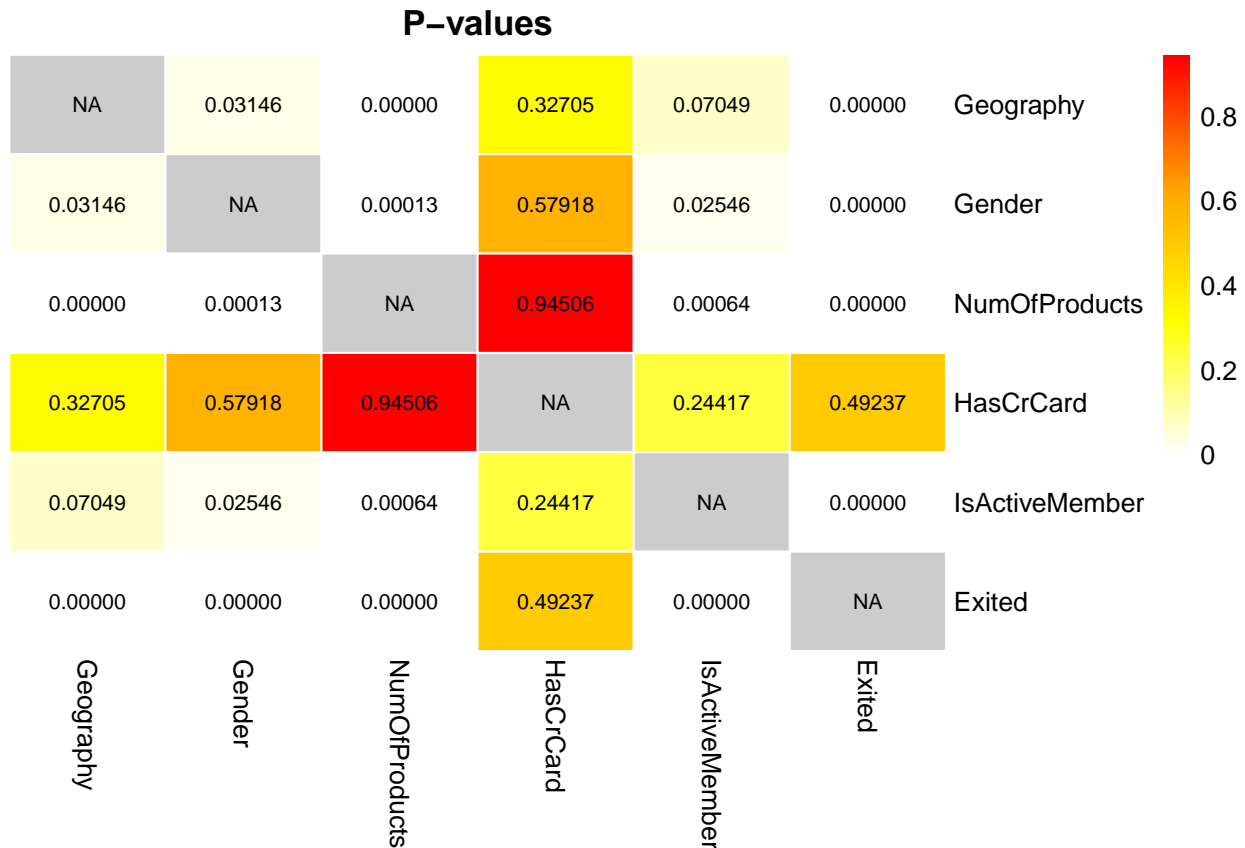
pheatmap(pairwise_p_vals,
  color = colorRampPalette(rev(c("red", "orange", "yellow", "white")))(100),
  display_numbers = TRUE,
  number_format = "%.5f",
  number_color = "black",
  na_col = "grey80",
  main = "P-values",

```

```

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
}
}

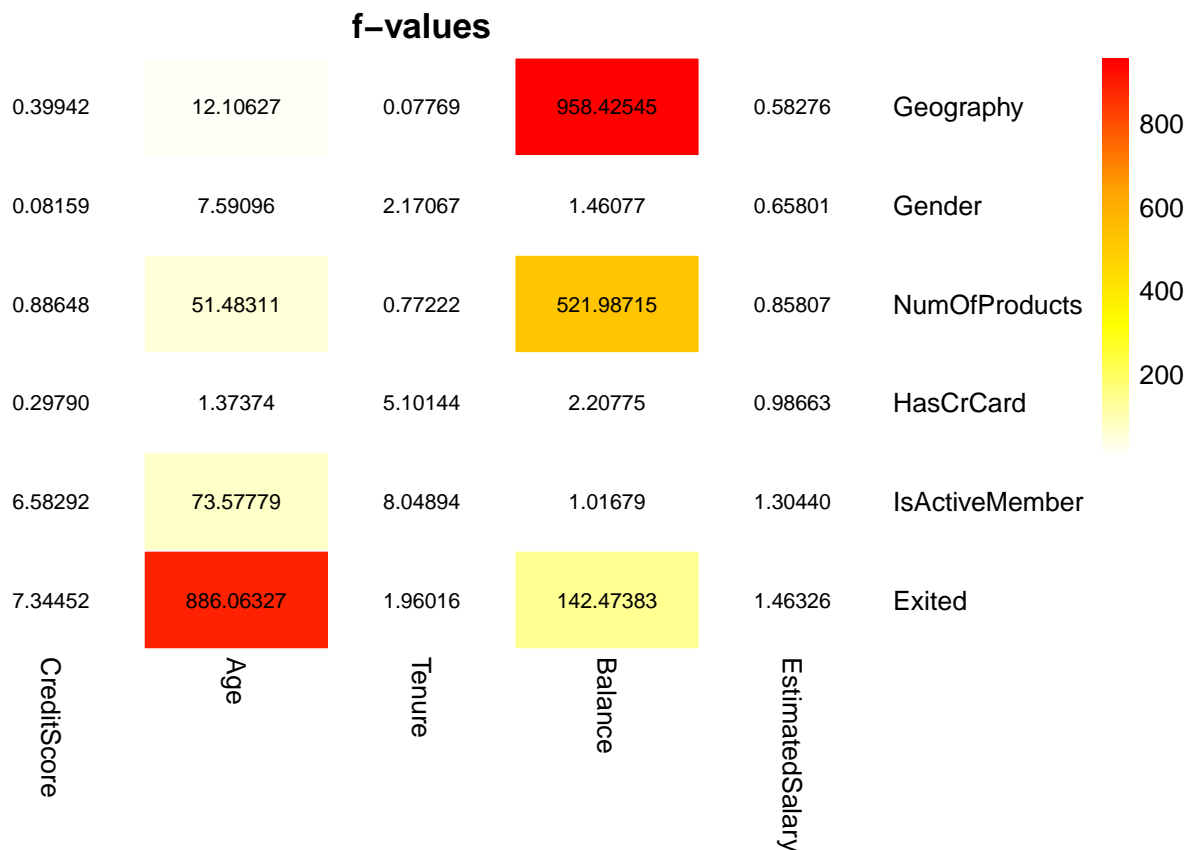
```

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

```

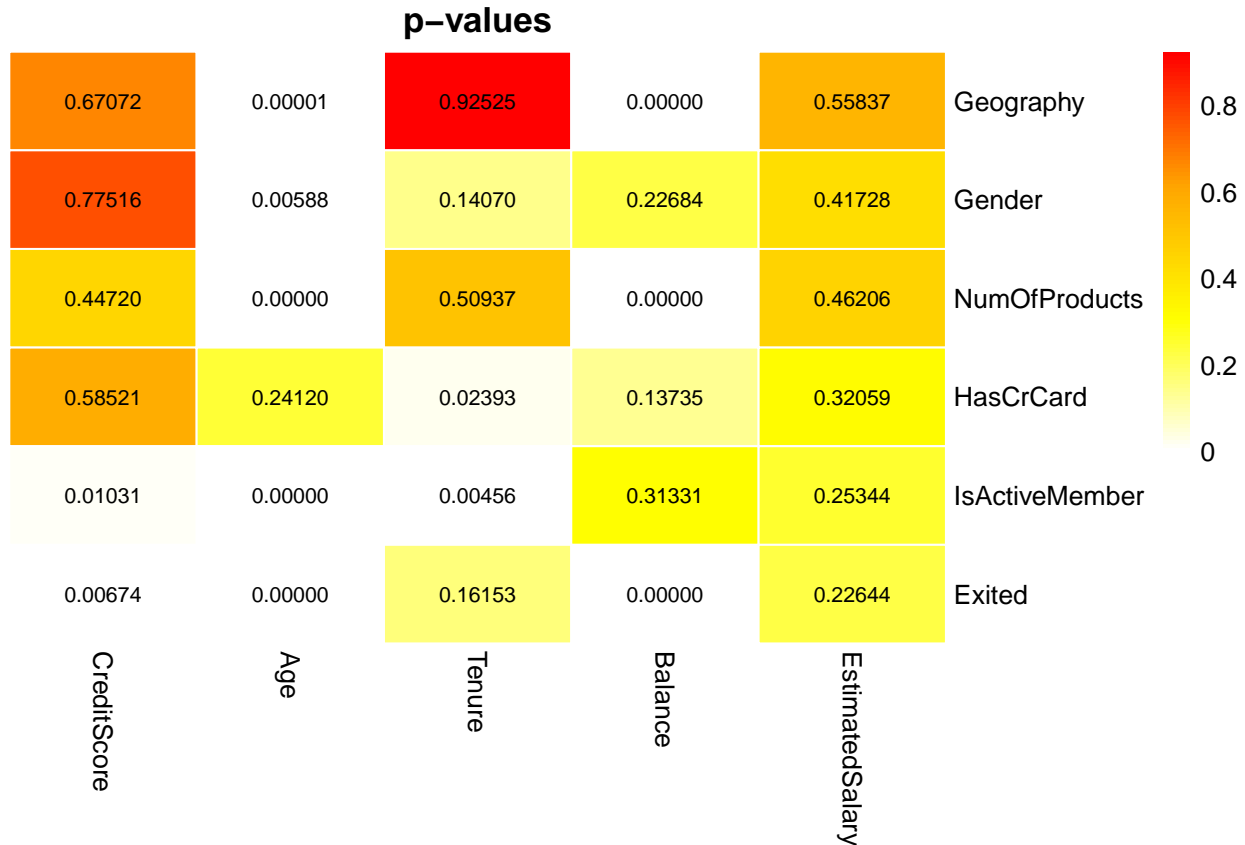
pheatmap(pairwise_p_vals,
  color = colorRampPalette(rev(c("red", "orange", "yellow", "white")))(100),
  display_numbers = TRUE,
  number_format = "%.5f",
  number_color = "black",
  na_col = "grey80",

```

```

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

```



2.2 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_data)

```

We can view the summary of the model.

```
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)   -3.387e+00  1.502e-01 -22.552  < 2e-16 ***
## Balance       -3.997e-07  6.364e-07  -0.628    0.530
## NumOfProducts2 -1.549e+00  7.973e-02 -19.427  < 2e-16 ***
## NumOfProducts3  2.492e+00  1.948e-01  12.793  < 2e-16 ***
## NumOfProducts4  1.639e+01  1.870e+02   0.088    0.930
## GeographyGermany 9.203e-01  8.175e-02  11.258  < 2e-16 ***
## GeographySpain   3.963e-02  8.502e-02   0.466    0.641
## GenderMale      -4.770e-01  6.606e-02  -7.220 5.18e-13 ***
## Age              7.203e-02  3.088e-03  23.326  < 2e-16 ***
## IsActiveMember1 -1.207e+00  7.061e-02 -17.094  < 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: 5913.4  on 7990  degrees of freedom
## AIC: 5933.4
##
## Number of Fisher Scoring iterations: 14
```

2.3 Getting predictions for test dataset

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

```
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.

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

```
##      Predicted
## Actual    0    1
##      0 1541   52
##      1  263  144
```

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

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

```

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

```
## Accuracy: 0.8425
```

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

```
## Precision: 0.73469387755102
```

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

```
## Sensitivity: 0.353808353808354
```

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

```
## Specificity: 0.967357187696171
```

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

```
## F1 score: 0.477611940298508
```

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.

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

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

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

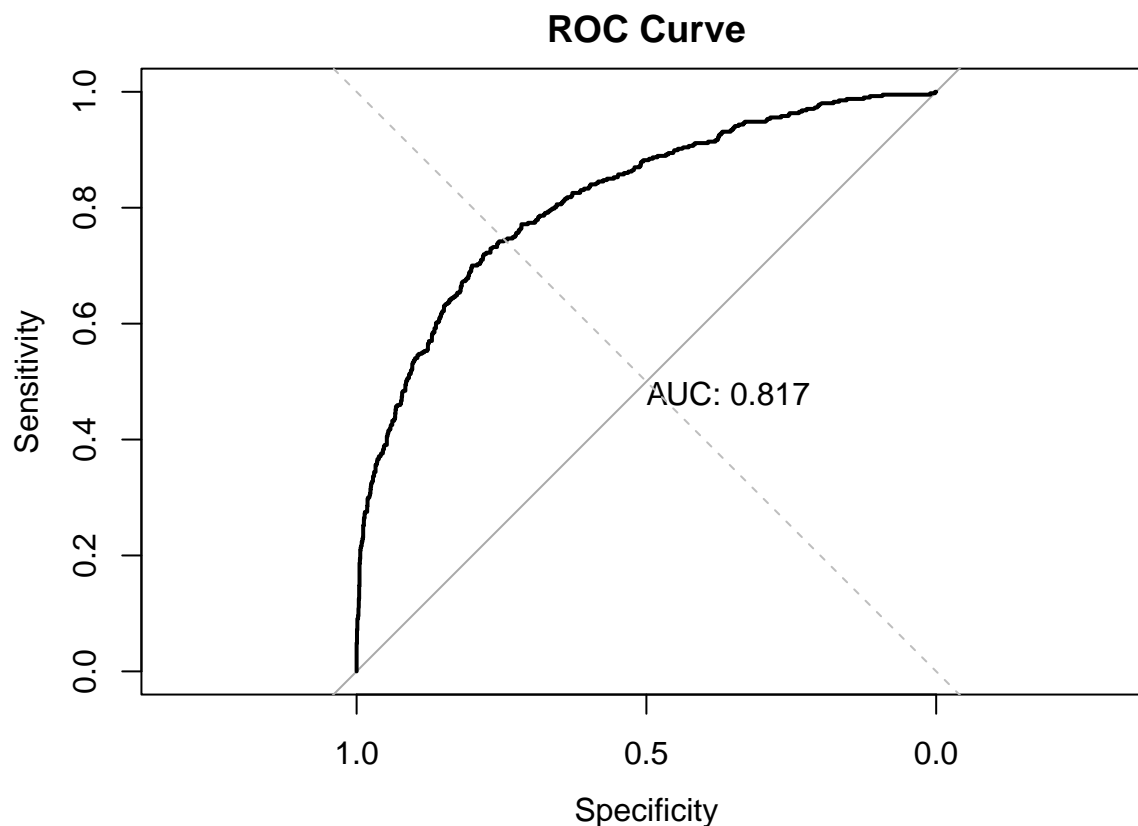
```
auc_value <- auc(roc_curve)
```

```
print(glue("AUC: {round(auc_value, 4)}"))
```

```
## AUC: 0.8172
```

```
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 correlation 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
## -5.2931 -2.2276 -0.0218  2.6361  5.3447
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.200e+00  2.740e-01  18.979  <2e-16 ***
## CreditScore   -3.034e-04  3.332e-04  -0.910   0.3626
## GeographyGermany 2.768e-02  8.652e-02   0.320   0.7491
## GeographySpain  4.069e-02  7.909e-02   0.514   0.6069
## GenderMale      8.568e-02  6.498e-02   1.319   0.1873
## Age            -2.298e-03  3.104e-03  -0.740   0.4592
## Balance        -4.729e-07  6.137e-07  -0.771   0.4409
```

```
## NumOfProducts2      5.374e-02  7.176e-02   0.749   0.4539
## NumOfProducts3     -9.793e-02  2.025e-01  -0.484   0.6287
## NumOfProducts4      2.514e-01  4.036e-01   0.623   0.5333
## HasCrCard1          9.872e-02  7.093e-02   1.392   0.1640
## IsActiveMember1    -1.358e-01  6.499e-02  -2.090   0.0367 *
## EstimatedSalary     5.409e-07  5.601e-07   0.966   0.3342
## ---
## 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:  0.001688,    Adjusted R-squared:  0.0001882
## F-statistic: 1.125 on 12 and 7987 DF,  p-value: 0.3336
```

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 = 2.91163013155856
```

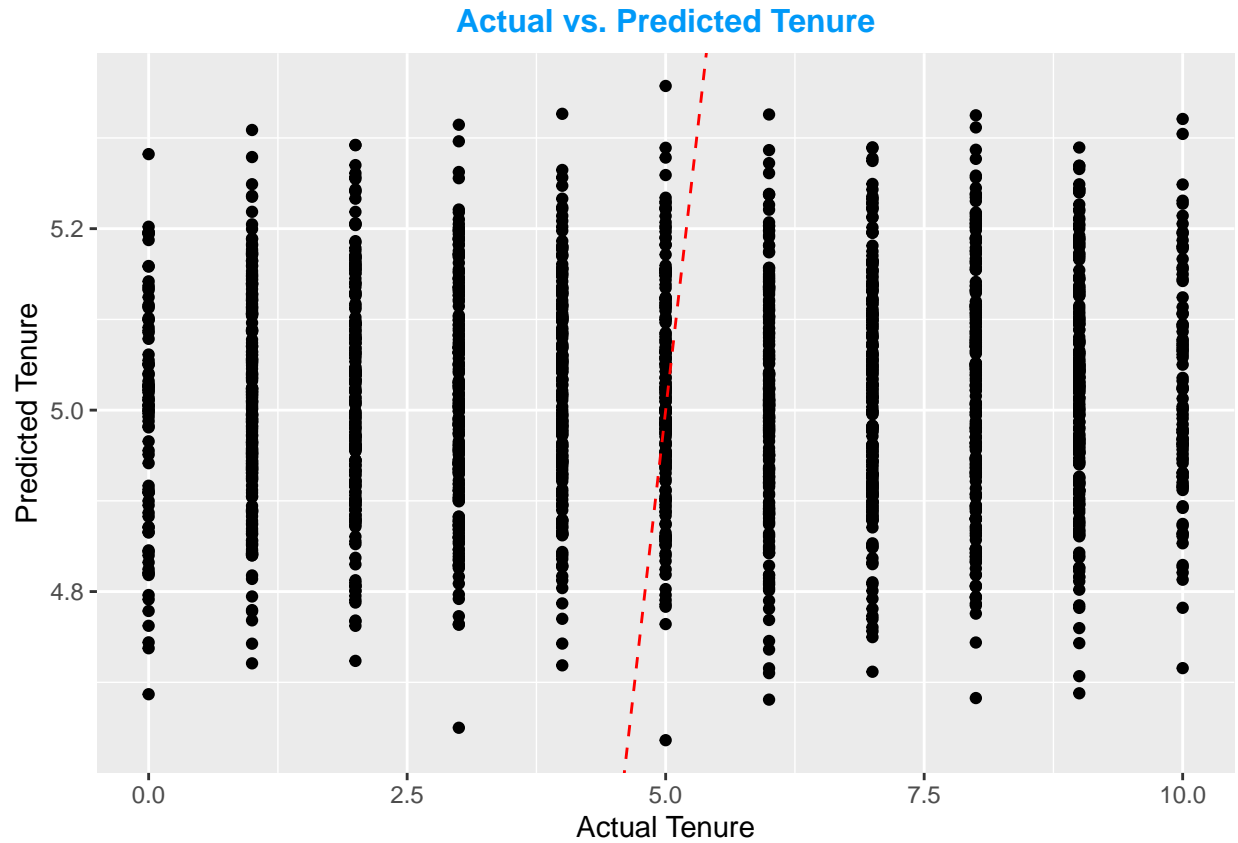
The root mean squared error we obtain is 2.9, which is not great considering that this would cover over 50% of the values in Tenure.

```
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 = 0.0017
```

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) {  
  quantitative <- c()  
  qualitative <- c()  
  
  for (column_name in names(data)){  
    column_type = class(data[,column_name])  
  
    if (column_type == "integer" || column_type == "numeric" ) {  
      quantitative <- c(quantitative, column_name)  
    } else if (column_type == "factor" || column_type == "logical") {  
      qualitative <- c(qualitative, column_name)  
    }  
  }  
  
  return (list(quantitative=quantitative, qualitative=qualitative))  
}  
  
quantitative_or_qualitative(bank_churn_data)
```

```
## $quantitative  
## [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, 999)
```

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)
```

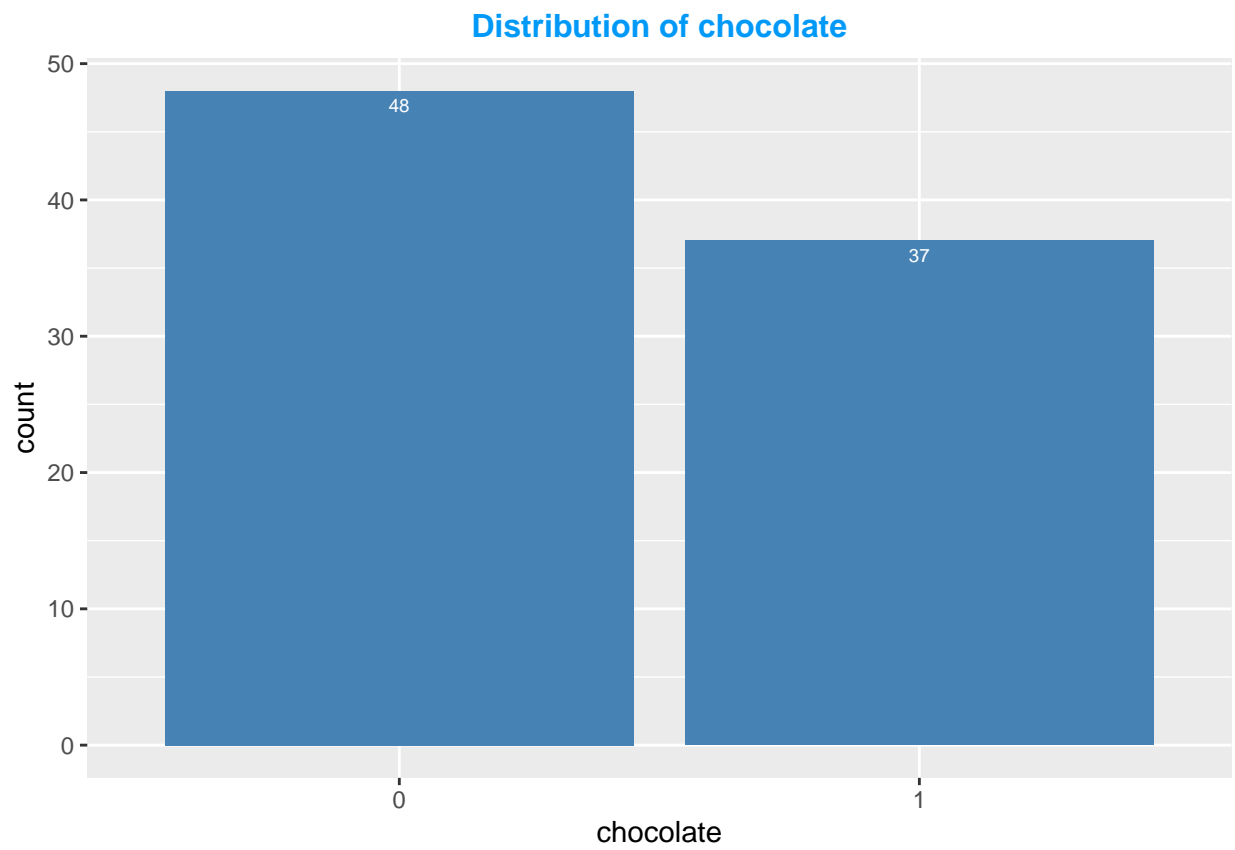
3.4 Visualize

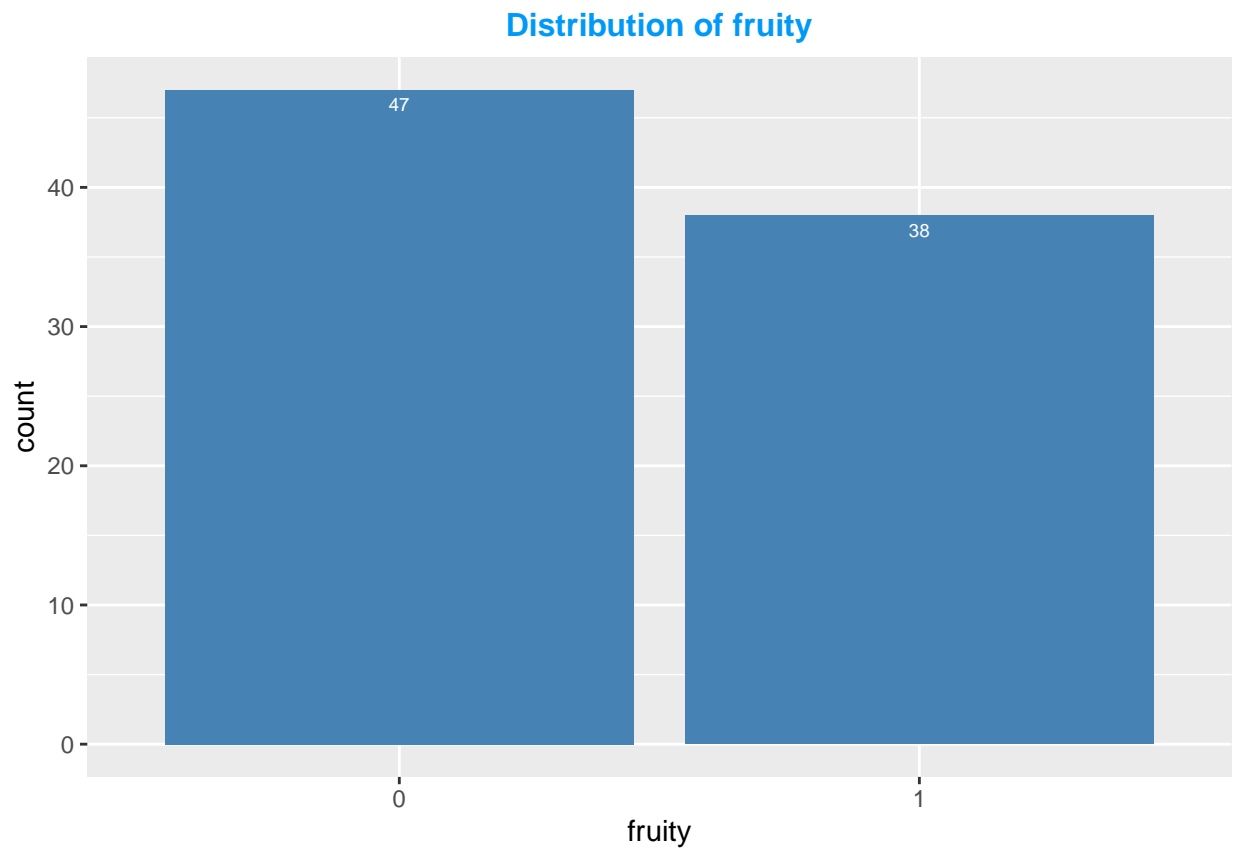
In order to visualize the data, we re-use two methods used in task 2. We have two separate types of visualizations for qualitative data and quantitative data.

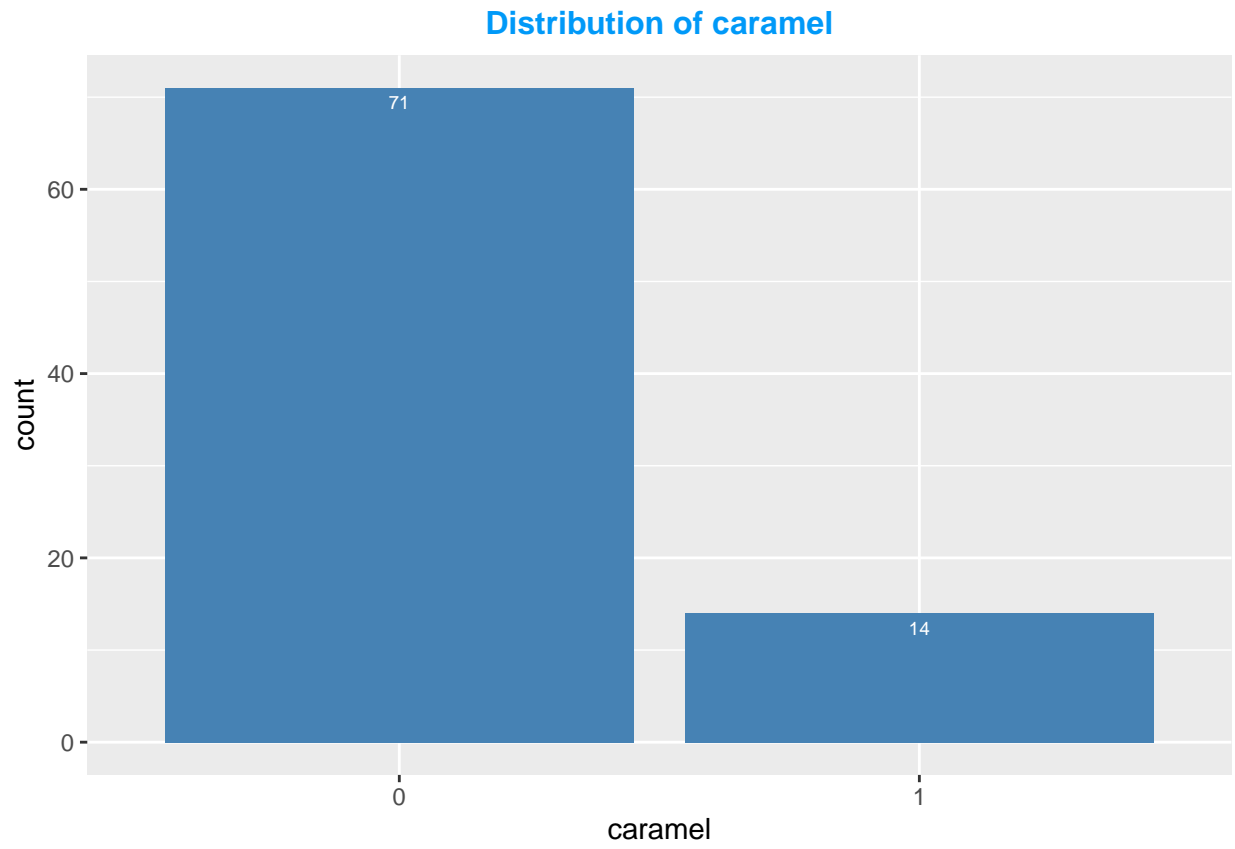
```
visualize <- function(data) {
  for (column_name in names(data)){
    column_type = class(data[,column_name])

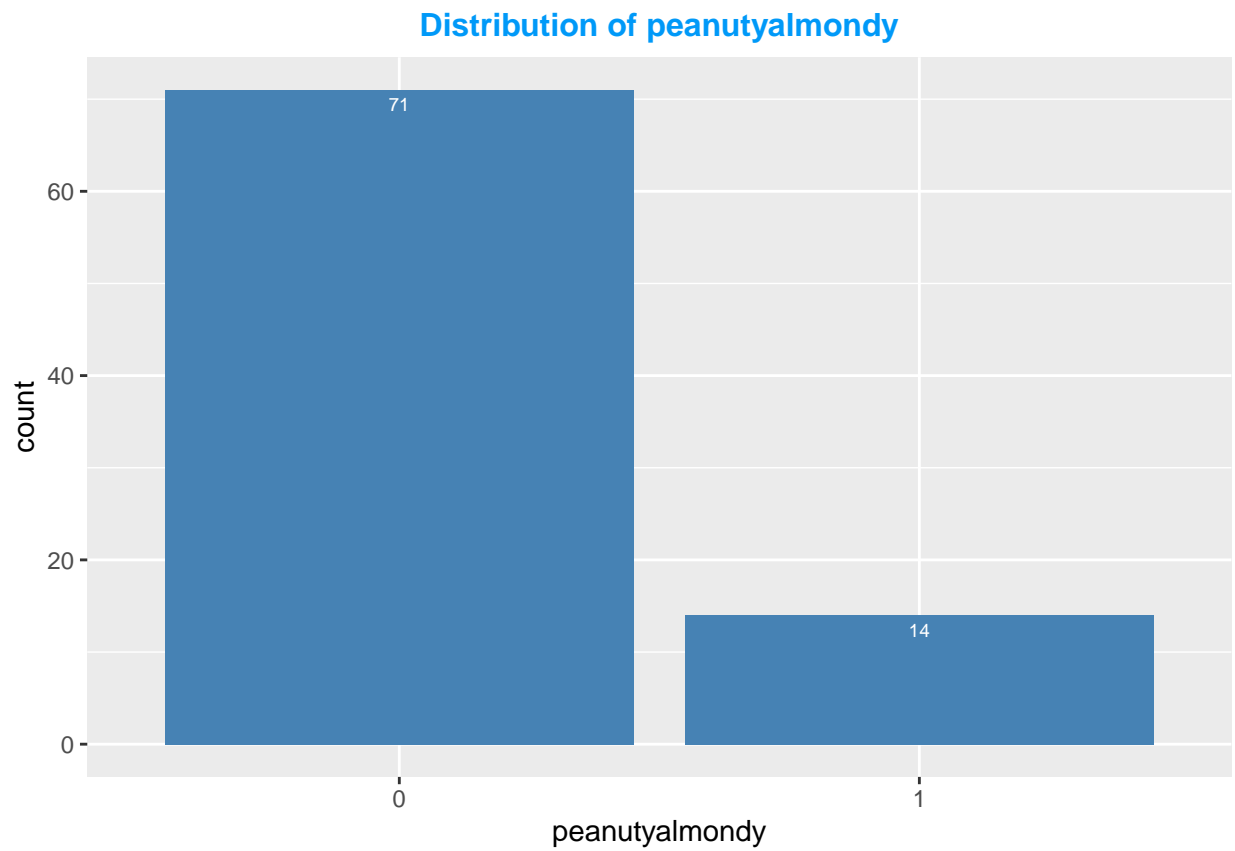
    if (column_type == "integer" || column_type == "numeric") {
      #+ fig.height = 3, fig.width = 6
      plot_numerical(data, column_name)
    } else if (column_type == "factor" || column_type == "logical") {
      #+ fig.height = 3, fig.width = 3
      print(plot_bar_chart(data, column_name))
    }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

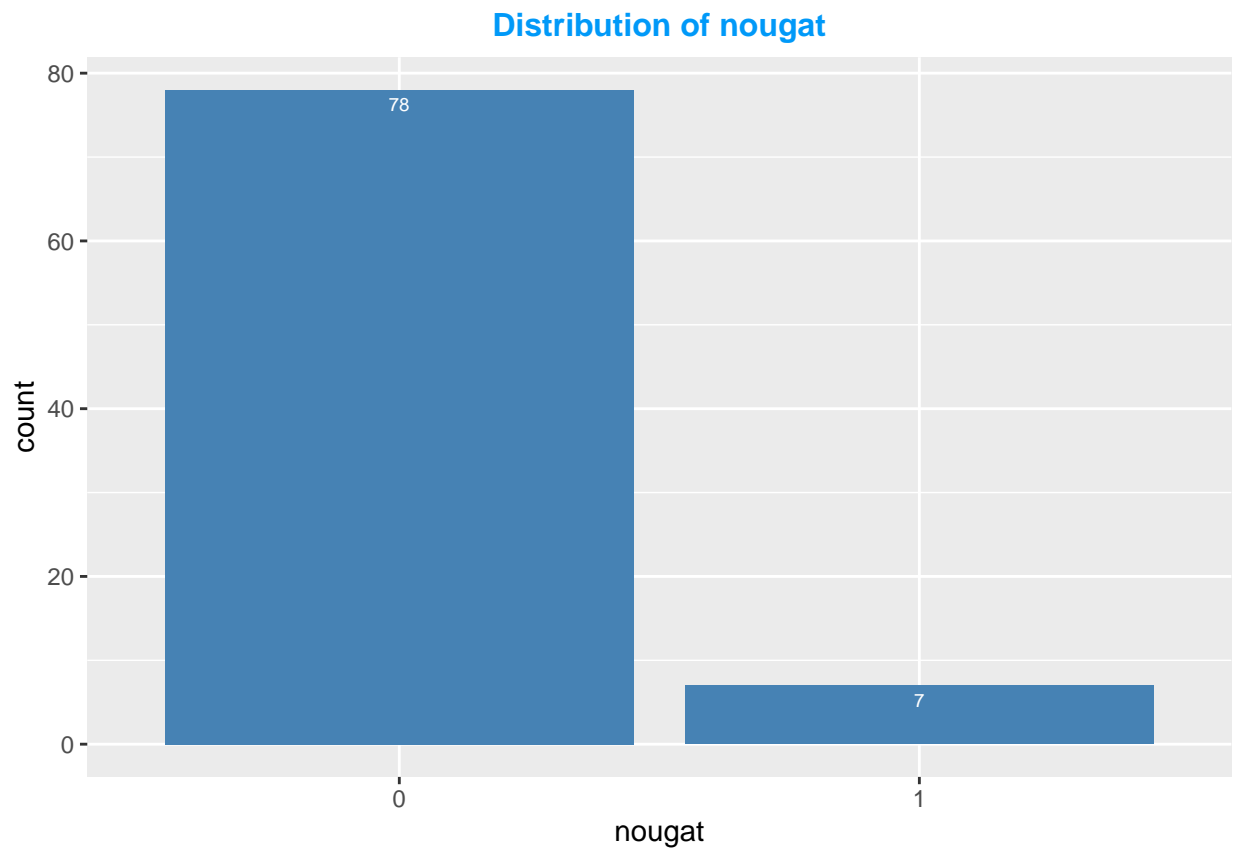
```
visualize(candy_data)
```

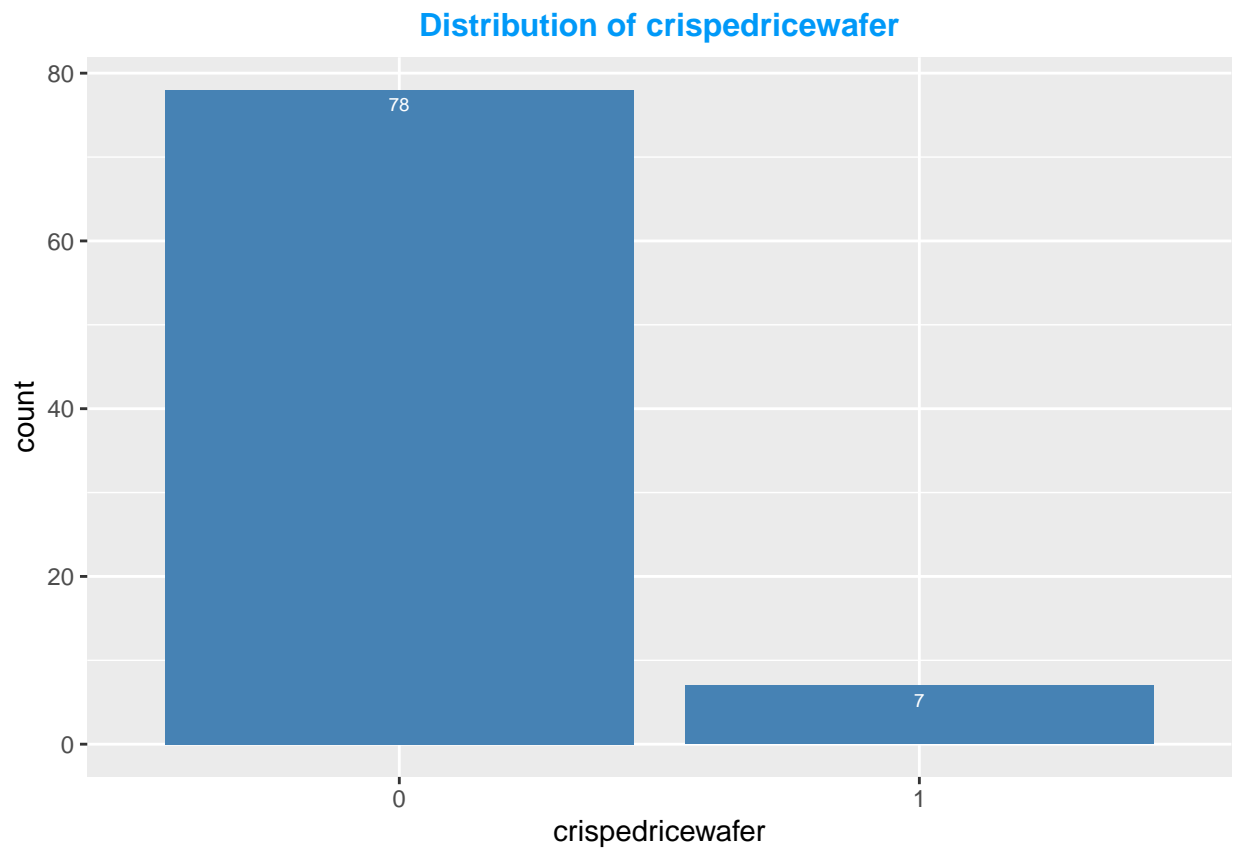


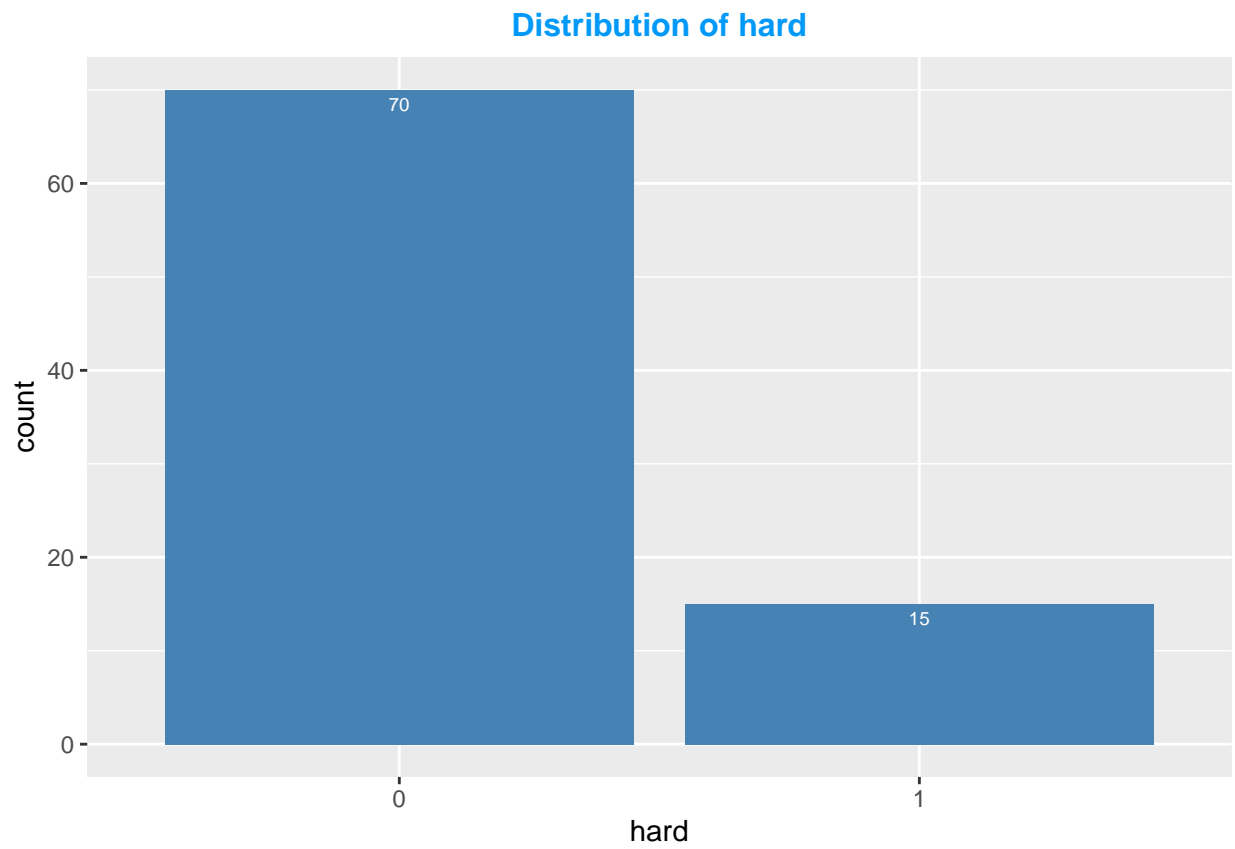


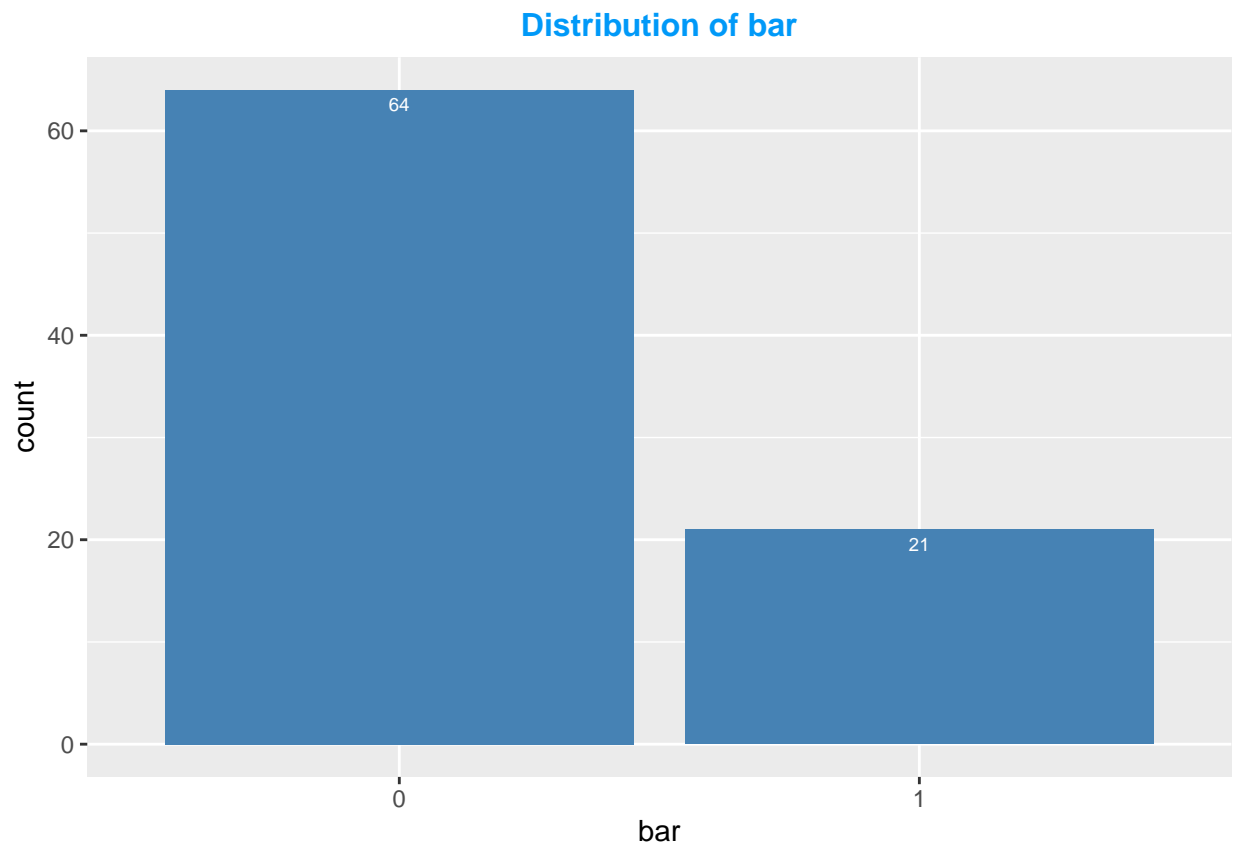


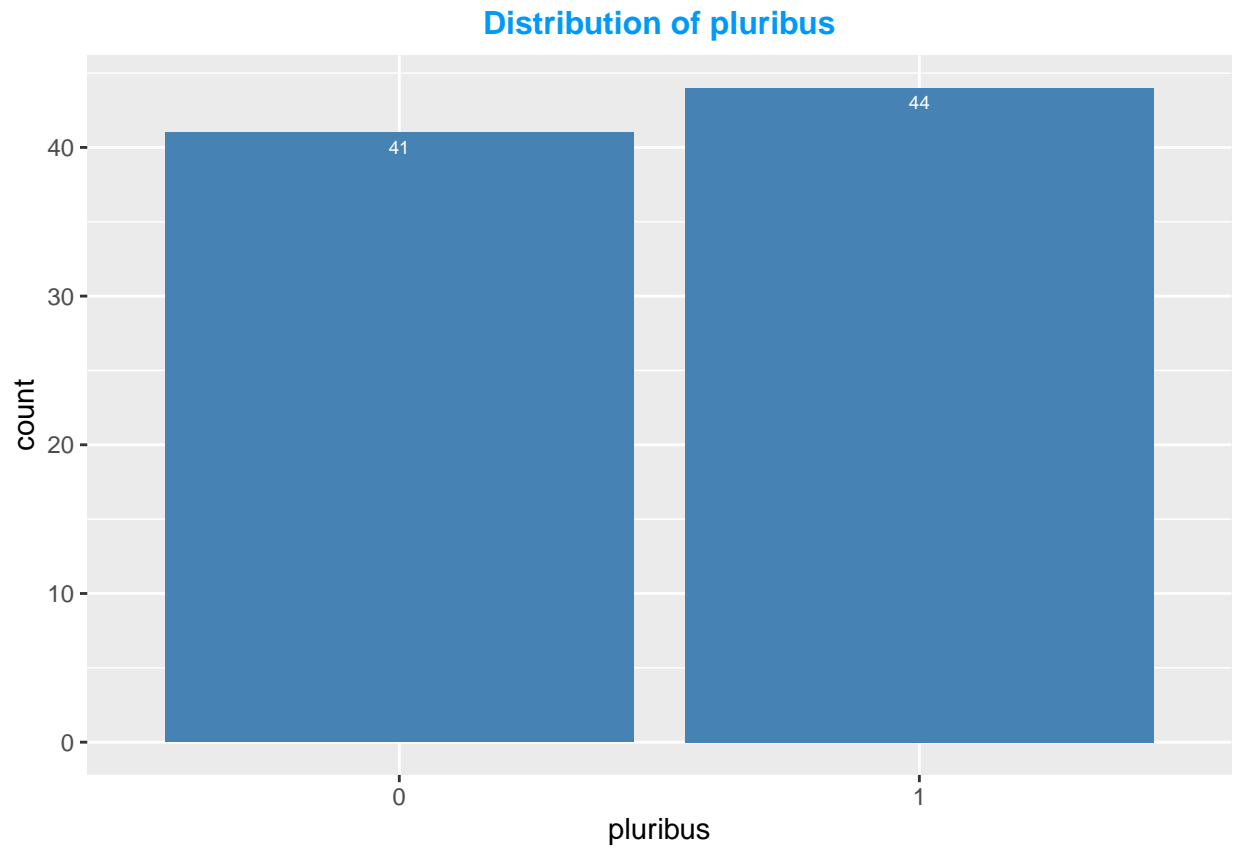




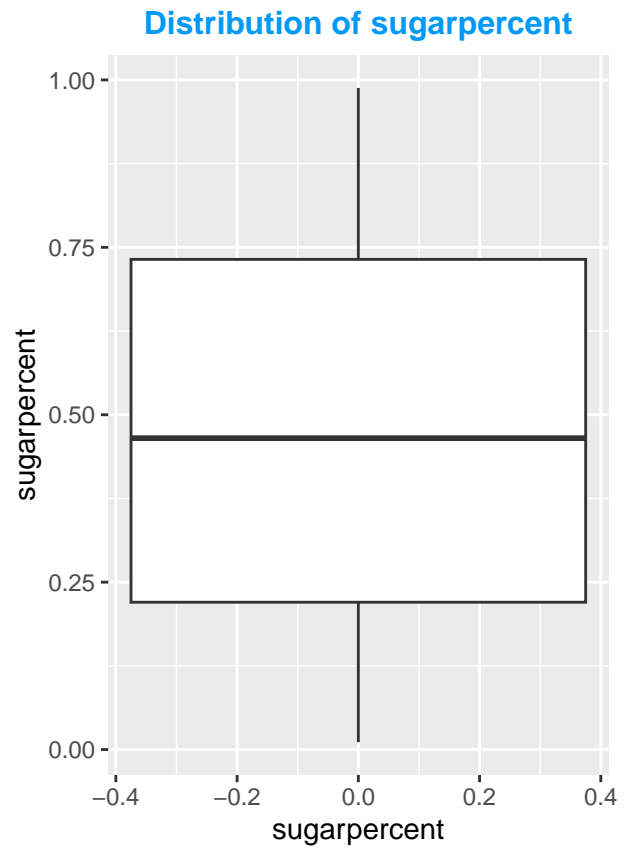
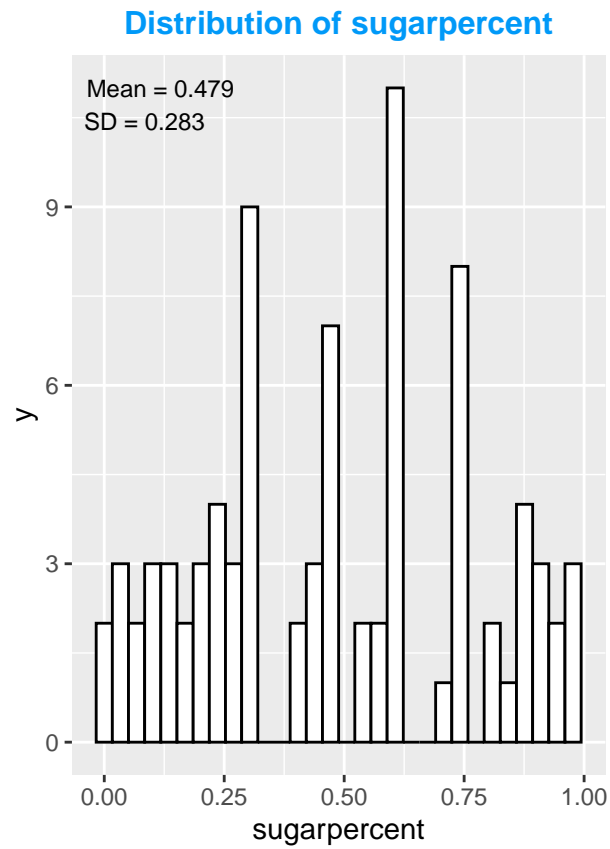




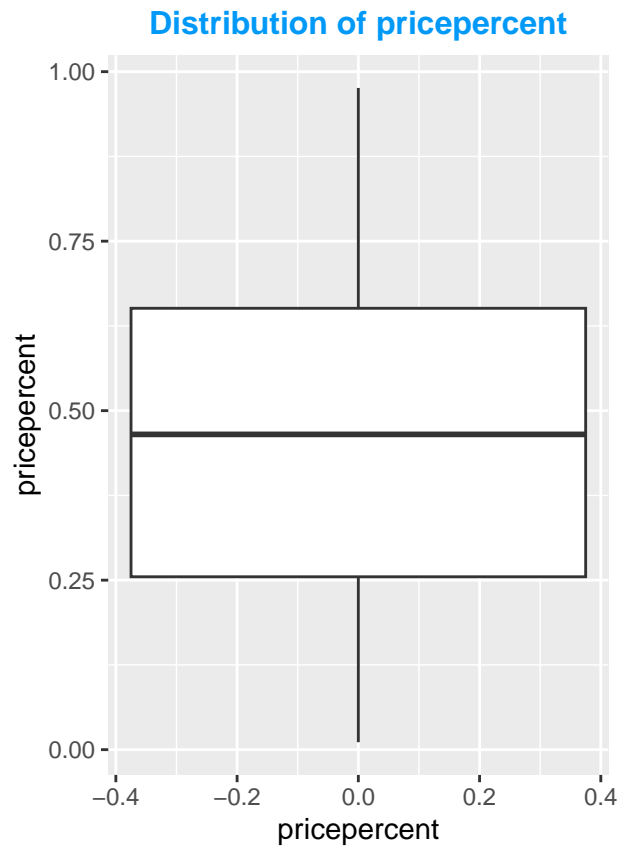
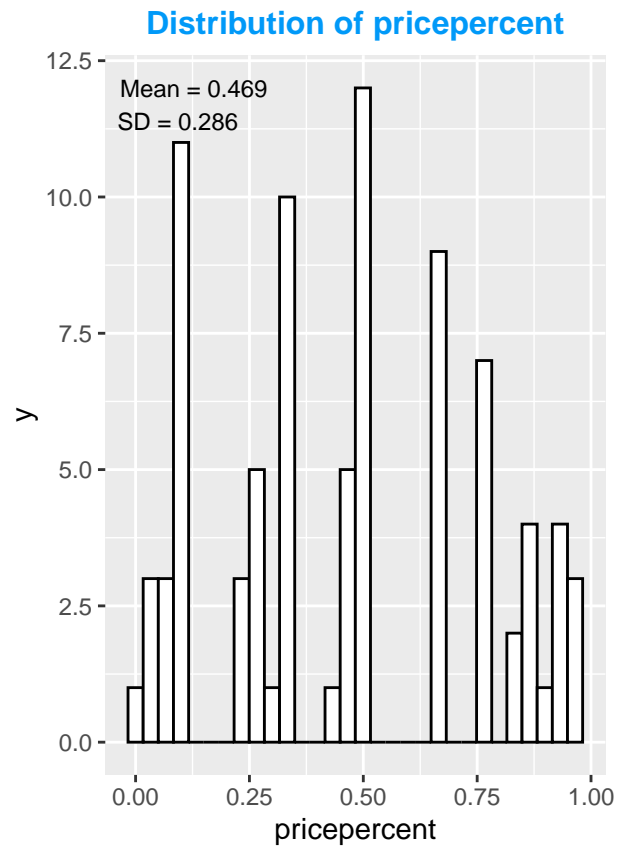




```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

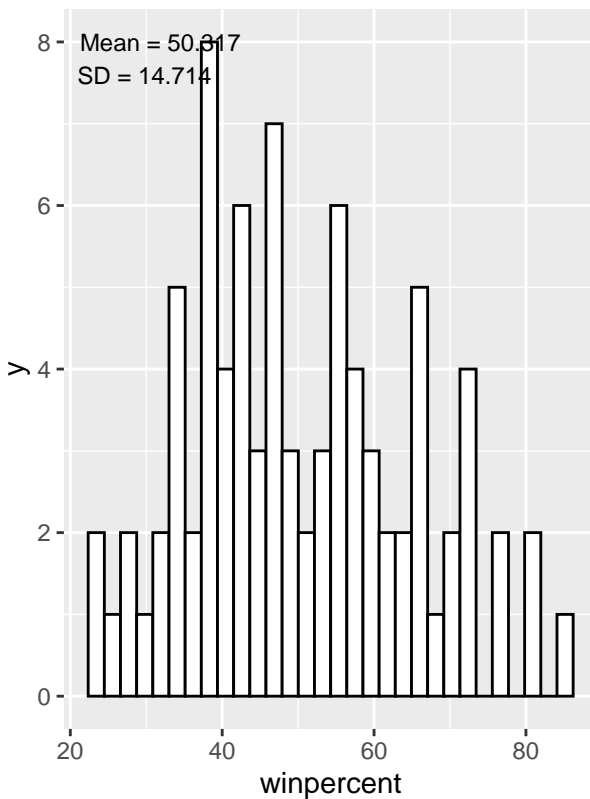



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

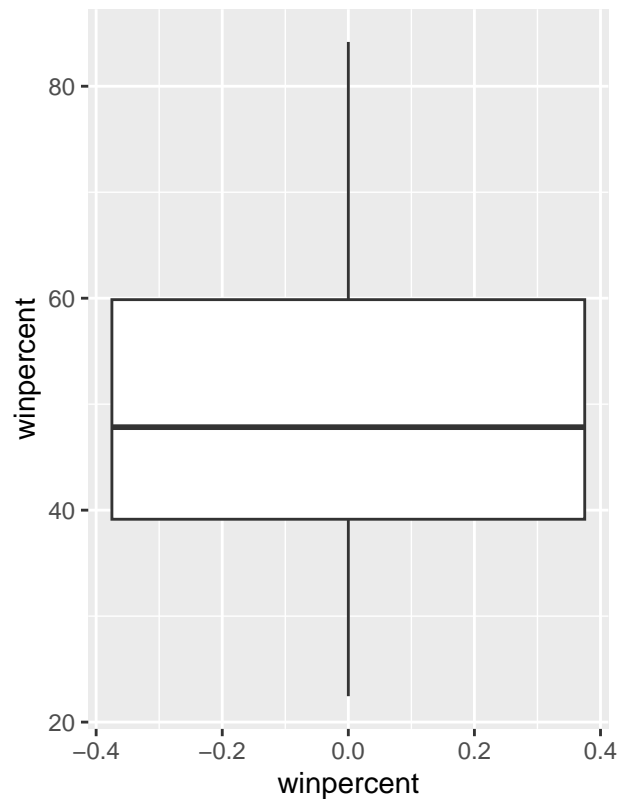


`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Distribution of winpercent



Distribution of winpercent



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)
```

```

split <- sample.split(data[,variable], 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)^2)
  tss <- sum((actual - mean(actual))^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)

```

```

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

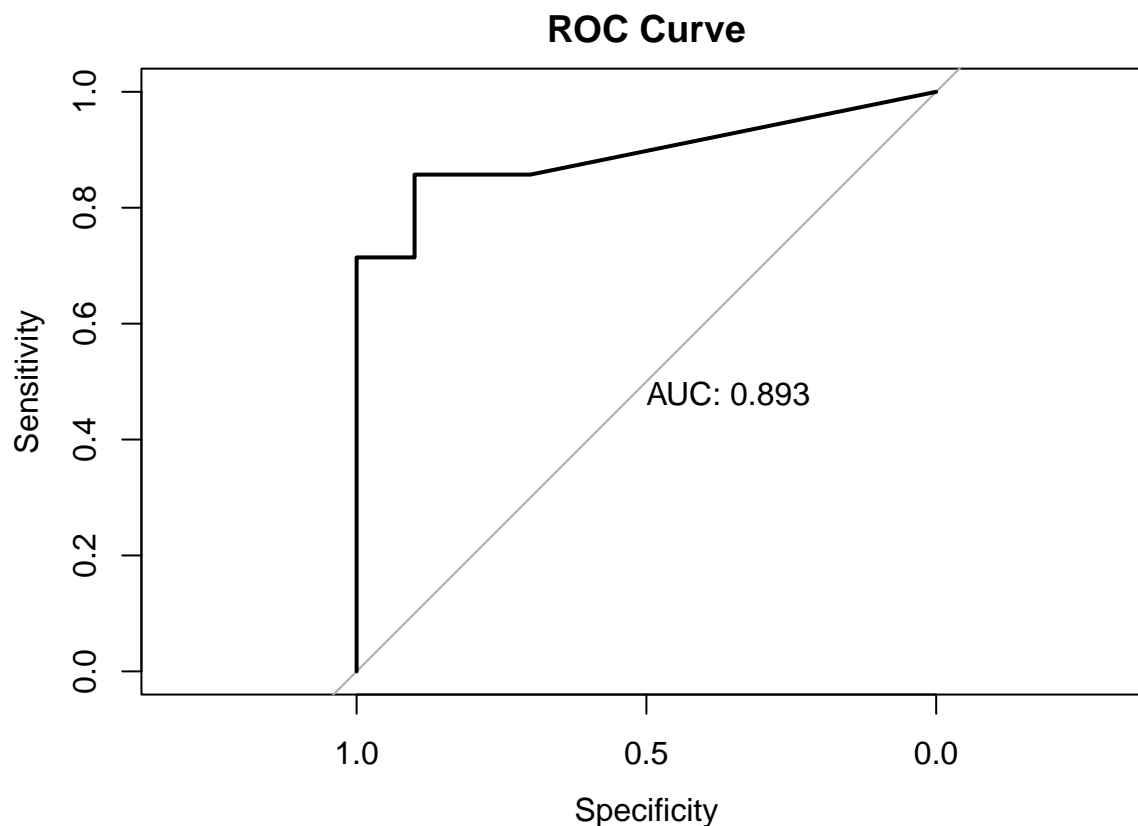
- Summary of the model
- Confusion matrix
- Accuracy, precision, sensitivity, specificity, f1 score
- ROC curve
- AUC value

```
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
```



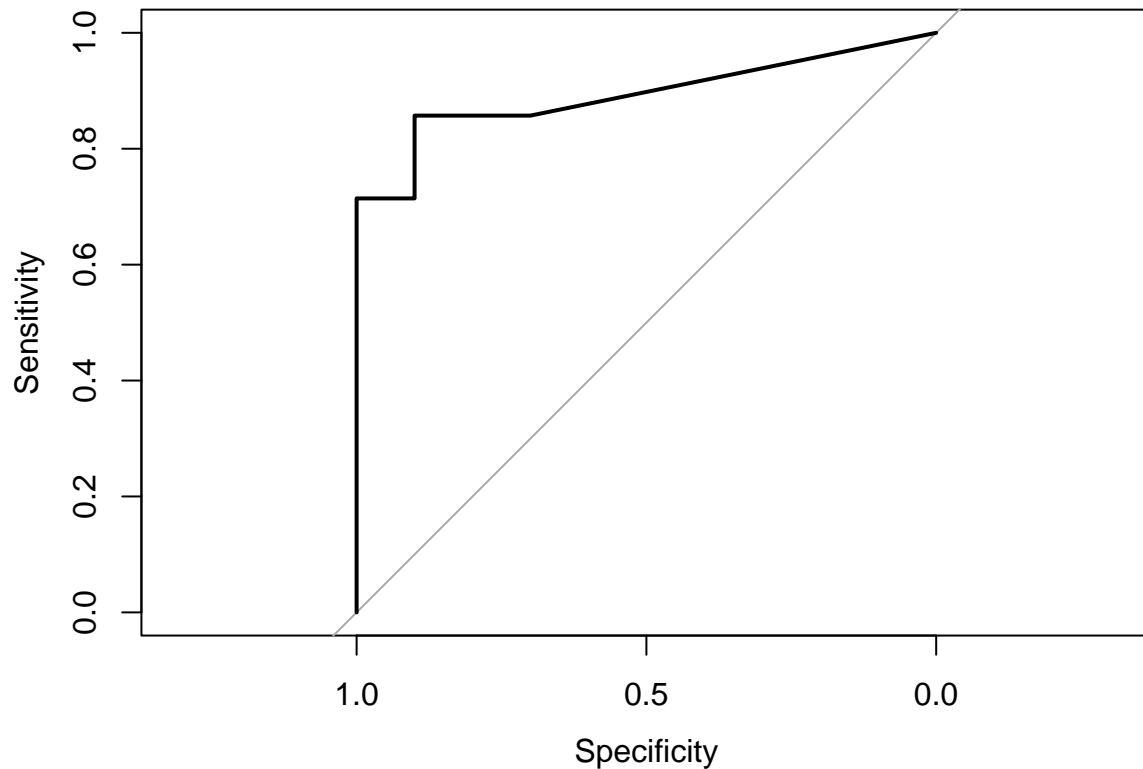
```
print(results)
```

```
## $summary
##
## Call:
## glm(formula = formula, family = binomial(link = "logit"), data = train_data)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -19.3701    10.3498  -1.872   0.0613 .
## fruity1       -35.0914   9561.9314  -0.004   0.9971
## caramell1      -3.0478    20.6937  -0.147   0.8829
## peanutyalmondy1 -11.7477     7.8514  -1.496   0.1346
## nougat1       -21.5492  15887.1136  -0.001   0.9989
## crispedricewafer1 15.1830  22891.8614   0.001   0.9995
## hard1        -19.5816  17761.2462  -0.001   0.9991
## bar1          25.2017  15887.1140   0.002   0.9987
## pluribus1      0.9238     3.1946   0.289   0.7724
## sugarpercent   -4.5125     4.7680  -0.946   0.3439
## pricepercent    3.6845     5.1833   0.711   0.4772
## winpercent      0.5079     0.2623   1.936   0.0528 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```

##      Null deviance: 93.3247  on 67  degrees of freedom
## Residual deviance:  6.4613  on 56  degrees of freedom
## AIC: 30.461
##
## Number of Fisher Scoring iterations: 22
##
##
## $conf_matrix
##      Predicted
## Actual 0 1
##      0 9 1
##      1 2 5
##
## $accuracy
## [1] 0.8235294
##
## $precision
## [1] 0.8333333
##
## $sensitivity
## [1] 0.7142857
##
## $specificity
## [1] 0.9
##
## $f1_score
## [1] 0.7692308
##
## $plot
##
## Call:
## roc.default(response = actual, predictor = test_probabilities)
##
## Data: test_probabilities in 10 controls (actual 0) < 7 cases (actual 1).
## Area under the curve: 0.8929
plot(results$plot)

```



For quantitative data, we print the

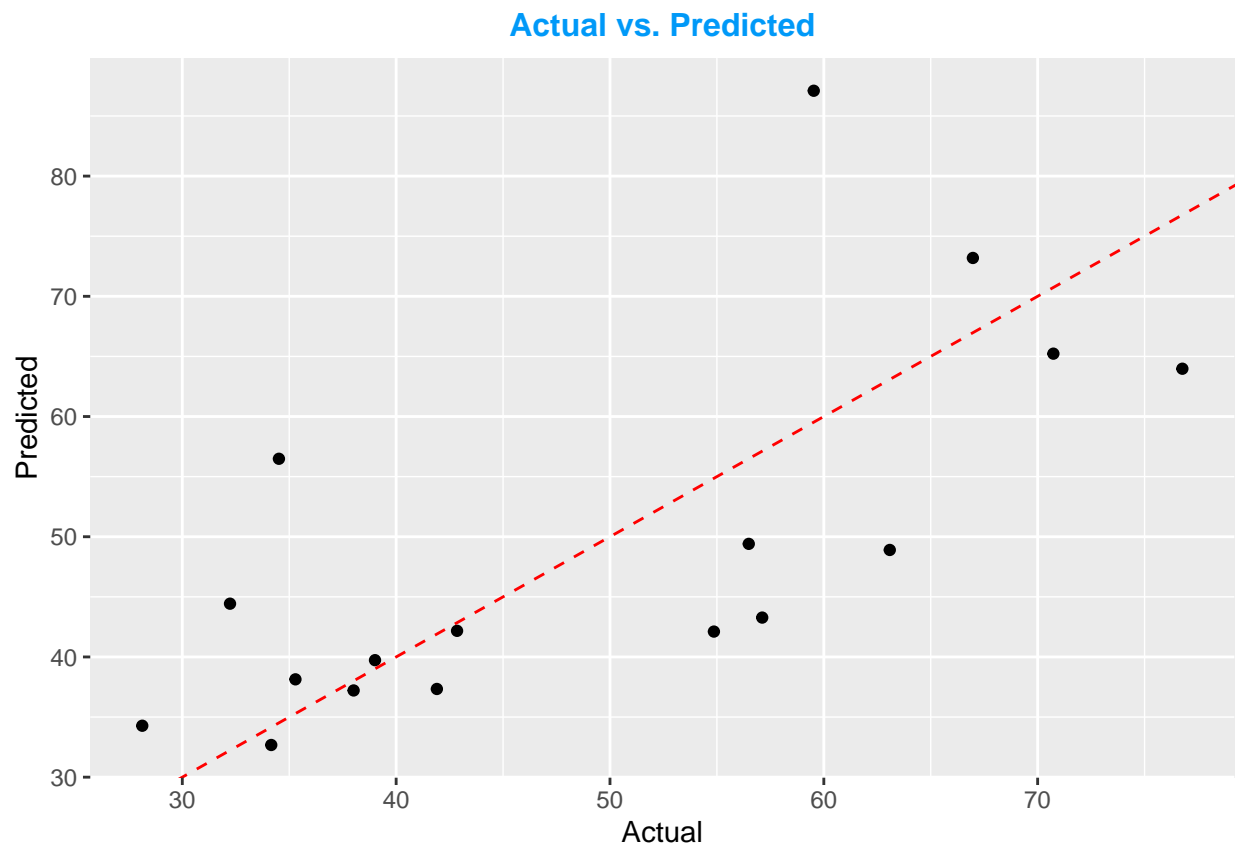
- Summary of the model
- RMSE
- R^2 value
- Scatter plot of predicted vs actual values

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

```
## $summary
##
## Call:
## lm(formula = formula, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.001  -7.019   0.118   6.474  24.315
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    35.0380     5.2302   6.699 1.09e-08 ***
## chocolate1     19.6143     4.4710   4.387 5.13e-05 ***
## fruity1        10.5985     4.6290   2.290 0.02583 *
## caramell        8.6470     4.8102   1.798 0.07763 .
## peanutyalmondy 13.2872     4.1829   3.177 0.00243 **
## nougat1        -0.8325     6.3594  -0.131 0.89632
## crispedricewafer1 12.3231     6.6333   1.858 0.06846 .
```



```
## hard1          -5.7303      4.3397  -1.320  0.19207
## bar1           -1.4800      5.7919  -0.256  0.79925
## pluribus1     -2.3867      3.4536  -0.691  0.49237
## sugarpercent   7.8206      5.5523   1.409  0.16450
## pricepercent  -7.7640      6.6860  -1.161  0.25048
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.89 on 56 degrees of freedom
## Multiple R-squared:  0.5412, Adjusted R-squared:  0.451
## F-statistic: 6.004 on 11 and 56 DF,  p-value: 2.292e-06
##
##
## $rmse
## [1] 11.6323
##
## $rsq_test
## [1] 0.3741696
##
## $plot
```



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](#).

```

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(
      "none", "remove", "replace", "smooth"
    )),
    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
          }
        }
      },
      error = function(e) {}
    )

    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$quantitative, 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))
})

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)

```

You can view a screenshot of the application [here](#).


```
[1] "Exited"
$summary

Call:
glm(formula = formula, family = binomial(link = "logit"), data = train_data)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.708e+00  6.600e+00  -0.562  0.5742
CustomerId    1.379e-08  4.205e-07   0.033  0.9738
CreditScore   -5.346e-04  3.130e-04  -1.708  0.0877 .
GeographyGermany 7.750e-01  7.543e-02  10.274 < 2e-16 ***
GeographySpain  5.420e-02  7.861e-02   0.690  0.4905
GenderMale     -5.022e-01  6.068e-02  -8.276 < 2e-16 ***
Age            7.161e-02  2.872e-03  24.936 < 2e-16 ***
Tenure        -1.303e-02  1.040e-02  -1.253  0.2101
Balance        2.596e-06  5.738e-07  4.525 6.04e-06 ***
NumOfProducts -7.693e-02  5.204e-02  -1.478  0.1394
HasCrCard1     -3.021e-02  6.620e-02  -0.456  0.6482
IsActiveMember1 -1.034e+00  6.391e-02 -16.172 < 2e-16 ***
EstimatedSalary 2.020e-07  5.285e-07   0.382  0.7023
---
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: 6898.6  on 7987  degrees of freedom
AIC: 6924.6

Number of Fisher Scoring iterations: 5

$conf_matrix
      Predicted
Actual    0    1
      0 1547  46
      1  318  89

$accuracy
[1] 0.818

$precision
[1] 0.6592593

$sensitivity
[1] 0.2186732

$specificity
[1] 0.9711237

$f1_score
[1] 0.3284133

$plot

Call:
roc.default(response = actual, predictor = test_probabilities)

Data: test_probabilities in 1593 controls (actual 0) < 407 cases (actual 1).
Area under the curve: 0.7868
```

