Assignment 2

PART 1: Data Exploration

a)

```r
# Load the data
data <- read.csv('./Honda_Sales.csv')

# Splitting the data into training and testing sets
train_data <- data %>% filter(Year <= 12)
test_data <- data %>% filter(Year >= 13)
```

a)(i)

```r
# Calculating the percentages
total_size <- nrow(data)
train_size <- nrow(train_data)
test_size <- nrow(test_data)

train_per <- (train_size / total_size) * 100
test_per <- (test_size / total_size) * 100

# Printing the percentages
print(paste("Training data percentage:", train_per, "%"))

## [1] "Training data percentage: 72 %"

print(paste("Testing data percentage:", test_per, "%"))

## [1] "Testing data percentage: 28 %"
```

a)(ii)

```r
# Saving the training data to a CSV file
write.csv(train_data, 'Honda_Sales_Training.csv', row.names = FALSE)
```

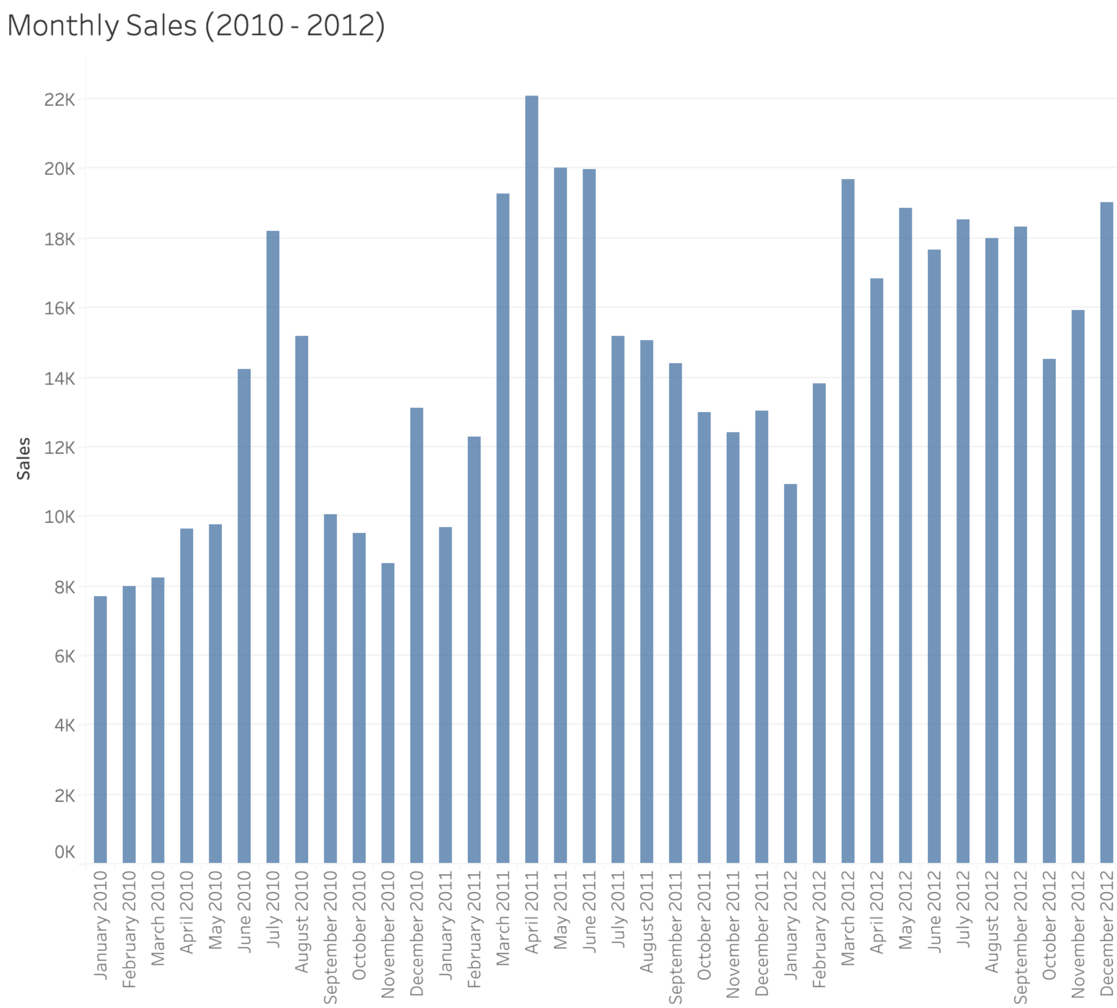

Figure 1: Monthly Sales (2010 - 2012)

a)(iii)

After observation from the graph illustrated above, one thing that stands out is that the sales tend to spike during the spring season every year, another thing that can be observed with the naked eye is that the average sales for each year also follows an upward trend. Moreover, the month with least sales from 2010 to 2012 was January 2010 with 7,690 sales, and the month with the most sales was April 2011 with 22,100 sales.

PART 2: Building the Model

a)

```
#Building model1
model1 <- lm(Sales ~ Unemployment + CPI_All + CPI_Energy + Queries, data = train_data)
summary(model1)

##
## Call:
## lm(formula = Sales ~ Unemployment + CPI_All + CPI_Energy + Queries,
##     data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6785.2 -2101.8  -562.5  2901.7  7021.0
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   95385.36  170663.81   0.559    0.580
## Unemployment  -3179.90    3610.26  -0.881    0.385
## CPI_All        -297.65     704.84  -0.422    0.676
## CPI_Energy       38.51     109.60   0.351    0.728
## Queries          19.03      11.26   1.690    0.101
##
## Residual standard error: 3295 on 31 degrees of freedom
## Multiple R-squared:  0.4282, Adjusted R-squared:  0.3544
## F-statistic: 5.803 on 4 and 31 DF,  p-value: 0.00132
```

a)(i)

```
# model1 R-squared value
model1_r2 <- summary(model1)$r.squared
model1_r2

## [1] 0.4281568

# Getting significant variables
model1_sig_vars <- summary(model1)$coefficients[,4] < 0.10  # 90% confidence interval
model1_sig_vars

##  (Intercept) Unemployment      CPI_All   CPI_Energy      Queries
##        FALSE        FALSE        FALSE        FALSE        FALSE
```

So, none of the variables in model1 are significant assuming 90% confidence interval.

b)

```r
# Building model2
model2 <- lm(Sales ~ Unemployment + CPI_All + CPI_Energy + Queries + Year + Month, data =
train_data)
summary(model2)

##
## Call:
## lm(formula = Sales ~ Unemployment + CPI_All + CPI_Energy + Queries +
##     Year + Month, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6158.4 -1823.9  -271.2  2258.7  6773.2
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   278126.66  224662.04   1.238    0.226
## Unemployment   -1285.26    4694.00  -0.274    0.786
## CPI_All        -1694.06    1299.96  -1.303    0.203
## CPI_Energy       139.66     135.60   1.030    0.312
## Queries           18.06      12.22   1.478    0.150
## Year            7837.84    6808.27   1.151    0.259
## Month            599.97     465.80   1.288    0.208
##
## Residual standard error: 3313 on 29 degrees of freedom
## Multiple R-squared:  0.4592, Adjusted R-squared:  0.3473
## F-statistic: 4.103 on 6 and 29 DF,  p-value: 0.004224
```

b)(i)

```r
# model2 R-squared value
model2_r2 <- summary(model2)$r.squared
model2_r2

## [1] 0.459161

# Getting significant variables
model2_sig_vars <- summary(model2)$coefficients[,4] < 0.10  # 90% confidence interval
model2_sig_vars

##  (Intercept) Unemployment      CPI_All   CPI_Energy      Queries         Year
##        FALSE        FALSE        FALSE        FALSE        FALSE        FALSE
##        Month
##        FALSE
```

Again, none of the variables in model2 are significant assuming 90% confidence interval.

b)(ii)

Even though the R-squared value in model2 was 45.92% in comparison to 42.82% in model1, our primary focus in order to evaluate which model performs better should be the adjusted R-squared values as we introduced two new variables (Year and Month) in model2. It can be deduced from the summary of the models that model1 adj. R-squared = 35.44%, whereas model2 adj. R-squared = 34.73%. This means, that the addition of the new variables did not increase the model significantly, instead it multicollinearity, meaning they might be correlated with other variables in the model, which ended up not improving the model's ability to explain the variability in sales substantially. This indicates that the addition of new variables in model2 only ended up overfitting the model, increasing the complexity. Hence, model2 does not perform better than model1.

c)

As months are categorical, they shouldn't be viewed as having a linear, numerical connection, so we must convert Month to a factor variable. Sales can be impacted by a variety of factors, including seasonality, holidays, marketing campaigns, and other events, in each given month. The model may accommodate non -linear and non-sequential fluctuations by appropriately accounting for the distinct influence of each month on sales through the use of a factor variable.

c)(i)

```r
# Converting Month to a factor variable
train_data$Month_Factor <- as.factor(train_data$Month)
test_data$Month_Factor <- as.factor(test_data$Month)
```

c)(ii)

```r
# Building model3
model3 <- lm(Sales ~ Unemployment + CPI_All + CPI_Energy + Queries + Year + Month_Factor,
data = train_data)
summary(model3)

##
## Call:
## lm(formula = Sales ~ Unemployment + CPI_All + CPI_Energy + Queries +
##     Year + Month_Factor, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3610.9 -1178.1  -171.9  1214.1  3219.4
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   403022.570 157480.157   2.559 0.019181 *
## Unemployment   -5139.177   3527.190  -1.457 0.161441
## CPI_All        -2241.673    899.379  -2.492 0.022088 *
## CPI_Energy       363.623    111.971   3.247 0.004238 **
## Queries           -9.457     13.206  -0.716 0.482640
```

```
## Year              6328.512   4827.015   1.311 0.205462
## Month_Factor2     2278.855   1909.325   1.194 0.247349
## Month_Factor3     7102.867   1981.167   3.585 0.001974 **
## Month_Factor4     8293.571   2079.757   3.988 0.000788 ***
## Month_Factor5     8853.806   2206.056   4.013 0.000743 ***
## Month_Factor6    10942.647   2361.694   4.633 0.000181 ***
## Month_Factor7    12259.004   2827.625   4.335 0.000356 ***
## Month_Factor8    10463.700   2796.742   3.741 0.001383 **
## Month_Factor9     8067.546   3054.487   2.641 0.016102 *
## Month_Factor10    5874.455   3288.845   1.786 0.090040 .
## Month_Factor11    6569.576   3464.253   1.896 0.073219 .
## Month_Factor12    9245.553   3725.286   2.482 0.022588 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2266 on 19 degrees of freedom
## Multiple R-squared:  0.8343, Adjusted R-squared:  0.6947
## F-statistic: 5.977 on 16 and 19 DF,  p-value: 0.0001859
```

c)(iii)

```
# model3 R-squared value
model3_r2 <- summary(model3)$r.squared
model3_r2

## [1] 0.8342584
```

c)(iv)

Indeed, the model performs noticeably better than both models 1 and 2. This is explained by the high r-squared value of 83.43%, which indicates that 83% of the variability in the response variable is explained by the model. Furthermore, the new model's efficacy is demonstrated by its high adjusted value of 69.43%

d)

```
# Applying step-wise elimination to model3
best_model <- step(model3)

## Start:  AIC=567.23
## Sales ~ Unemployment + CPI_All + CPI_Energy + Queries + Year +
##     Month_Factor
##
##                Df Sum of Sq       RSS    AIC
## - Queries       1   2632038 100153591 566.19
## <none>                       97521552 567.23
## - Year          1   8822525 106344077 568.35
## - Unemployment  1  10896233 108417785 569.05
## - CPI_All       1  31886454 129408006 575.42
## - CPI_Energy    1  54129928 151651481 581.13
## - Month_Factor 11 238910505 336432058 589.81
##
## Step:  AIC=566.19
## Sales ~ Unemployment + CPI_All + CPI_Energy + Year + Month_Factor
```

```
##
##                    Df  Sum of Sq       RSS    AIC
## <none>                          100153591 566.19
## - Year              1    6911316 107064906 566.60
## - Unemployment      1   16291755 116445346 569.62
## - CPI_All           1   30284528 130438119 573.70
## - CPI_Energy        1   64673567 164827158 582.13
## - Month_Factor     11  263805102 363958692 590.65
```

```r
summary(best_model)
```

```
##
## Call:
## lm(formula = Sales ~ Unemployment + CPI_All + CPI_Energy + Year +
##     Month_Factor, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3650.2 -1259.5  -191.5  1158.4  3432.0
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    413900.69  154824.83   2.673 0.014604 *
## Unemployment    -5950.79    3299.20  -1.804 0.086363 .
## CPI_All         -2171.71     883.10  -2.459 0.023161 *
## CPI_Energy        314.63      87.55   3.594 0.001815 **
## Year             5391.56    4589.36   1.175 0.253875
## Month_Factor2    2562.10    1845.01   1.389 0.180208
## Month_Factor3    7380.28    1919.11   3.846 0.001009 **
## Month_Factor4    8443.35    2043.85   4.131 0.000518 ***
## Month_Factor5    8686.44    2166.76   4.009 0.000689 ***
## Month_Factor6   10700.42    2308.70   4.635 0.000160 ***
## Month_Factor7   11313.12    2469.51   4.581 0.000181 ***
## Month_Factor8   10053.29    2703.84   3.718 0.001359 **
## Month_Factor9    7727.56    2980.38   2.593 0.017395 *
## Month_Factor10   5787.00    3246.30   1.783 0.089831 .
## Month_Factor11   6860.28    3398.22   2.019 0.057119 .
## Month_Factor12   9302.94    3678.78   2.529 0.019964 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2238 on 20 degrees of freedom
## Multiple R-squared:  0.8298, Adjusted R-squared:  0.7021
## F-statistic:   6.5 on 15 and 20 DF,  p-value: 8.678e-05
```

d)(i)

```r
# best_model R-squared value
best_model_r2 <- summary(model3)$r.squared
best_model_r2
```

```
## [1] 0.8342584
```

```r
# Finding removed variables
removed_vars <- setdiff(names(coef(model3)), names(coef(best_model)))
removed_vars
```

```
## [1] "Queries"
```

d)(ii)

```r
# Regression equation for best model
best_model_eq <- formula(best_model)
best_model_eq

## Sales ~ Unemployment + CPI_All + CPI_Energy + Year + Month_Factor
```

$Sales = 413900.69 - 5950.79 * (Unemployment) - 2171.71 * (CPIAll) + 314.63 * (CPI\_Energy) + 5391.56 * (Year) + 2562.10 * (MonthFactor2) + 7380.28 * (MonthFactor3) + *(MonthFactor4) + 8686.44 * (MonthFactor5) + 10700.42 * (MonthFactor6) + 11313.12 * (MonthFactor7) + 10053.29 * (MonthFactor8) + 7727.56 * (MonthFactor9) + 5787.00 * (MonthFactor10) + 6860.28 * (MonthFactor11) + 6860.28 * (MonthFactor12)$

d)(iii)

Coeff. of the Year variable = 5391.559

This means that for each additional year, the monthly sales of Honda Civic are expected to increase by 5391.56 units, while keeping all other variables constant.

d)(iv)

```r
# making predictions
predict_train <- predict(best_model, newdata = train_data)
train_data$PredictTrain <- predict_train
```

d)(v)

```r
library(Metrics)

# Calculating RMSE, MAE, and MAPE for the training dataset
rmse_value <- rmse(train_data$Sales, train_data$PredictTrain)
print( paste("RMSE value:", rmse_value))

## [1] "RMSE value: 1667.94609840513"

mae_value <- mae(train_data$Sales, train_data$PredictTrain)
print( paste("MAE value:", mae_value))

## [1] "MAE value: 1383.16018661506"

mape_value <- mape(train_data$Sales, train_data$PredictTrain)
print( paste("MAPE value:", mape_value))

## [1] "MAPE value: 0.104403548443748"
```

d)(vi)

```r
write.csv(predict_train, 'predict_train.csv', row.names = TRUE)
```

Figure 2: Actual v/s Predicted Sales

Part 3: Model's Out-of-Sample Performance

a)

```
# Making predictions on testing data
predict_test <- predict(best_model, newdata = test_data)
test_data$PredictTest <- predict_test
```

b)

```
# Saving testing data in 'testing_data.csv'
write.csv(test_data, "testing_data.csv")

# Saving predicted data
write.csv(predict_test, 'predict_test.csv', row.names = TRUE)
```

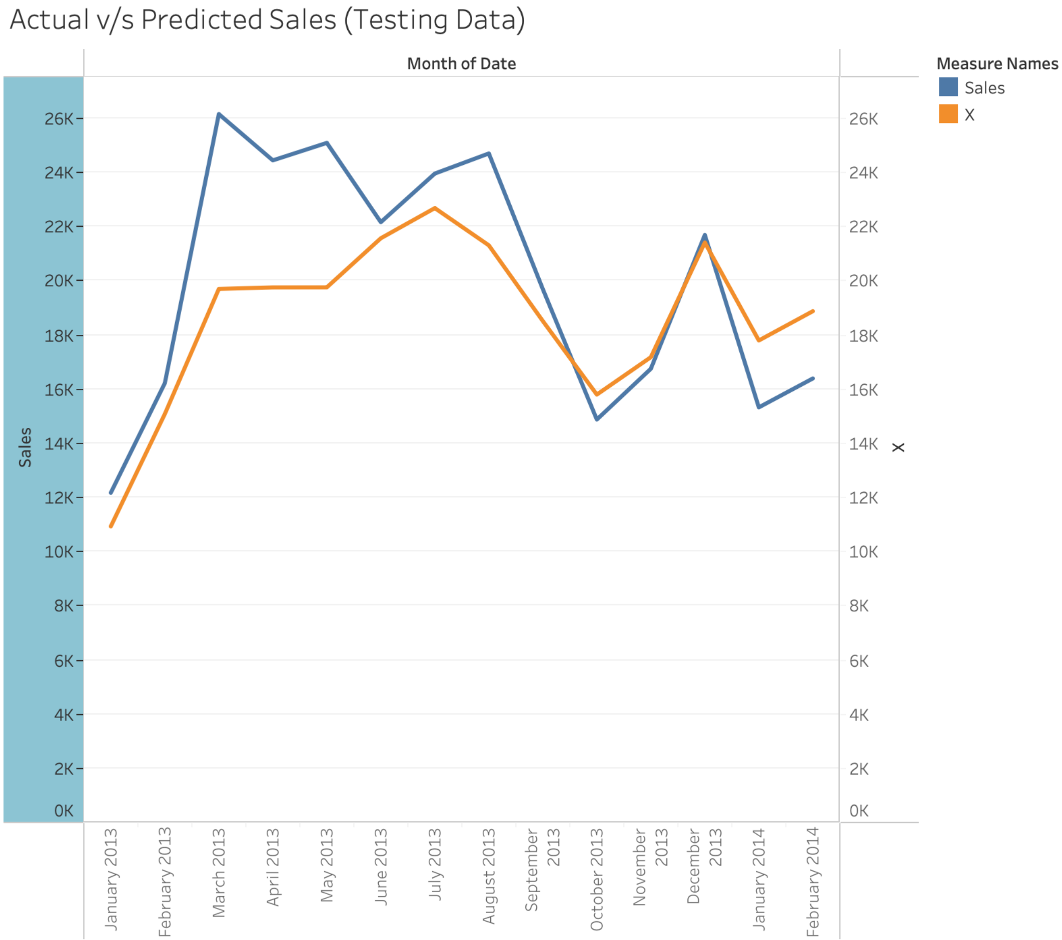

Figure 2: Actual v/s Predicted Sales (Testing Data)

c)

```
# Calculating RMSE, MAE, and MAPE for the testing dataset
rmse_value <- rmse(test_data$Sales, predict_test)
print(paste("RMSE value:", rmse_value))

## [1] "RMSE value: 2966.29744160816"

mae_value <- mae(test_data$Sales, predict_test)
print(paste("MAE value:", mae_value))

## [1] "MAE value: 2276.94691846237"

mape_value <- mape(test_data$Sales, predict_test)
print(paste("MAPE value:", mape_value))

## [1] "MAPE value: 0.108102185571369"
```

d)

The performance scores of both the datasets were as follows- TRAINING dataset: RMSE_train: 1667.946 MAE_train: 1383.16 MAPE_train: 10.44035%

TESTING dataset: RMSE_test: 2966.297 MAE_test: 2276.947 MAPE_test: 10.81022%

Observations:

1. Compared to the training dataset, the testing dataset's RMSE is noticeably greater. The prediction errors standard deviation (residuals) is measured by RMSE. The model may be overfitting the training data if the RMSE on the testing data is larger, indicating that the model has more prediction errors on unknown data.

2. Compared to the training dataset, the MAE on the testing dataset is substantially greater. Without taking into account the direction of the errors, MAE calculates the average magnitude of the errors in a series of forecasts.

3. Compared to the training dataset, the MAPE on the testing dataset is marginally greater. The MAPE calculates the model's accuracy as a percentage. On the testing dataset, a little decrease in prediction accuracy is indicated by the slight increase in MAPE.

In conclusion, when compared to the training dataset, the model performs worse on the testing dataset. The testing dataset shows a small rise in MAPE along with considerable increases in RMSE and MAE. The testing data's larger errors raise the possibility that the model is overfitting the training set, collecting noise instead of the underlying patterns.