Console Responses & Output File

```
C:/Personal/C997 - Data Analysis with R/population/src/r_code/
> ## WGU Performance Assessment - Population Prediction for Home State ##
> ## R script - wgu_population_predictions_ri.R ##
> ## Author - Abhishek Khatti ##
>
> ## Import required Libraries
> library(dplyr)  # To work with Data Frame type structures

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union

>
> ## Set Current Working directory
>
> setwd("C:/Personal/C997 - Data Analysis with R/population/src/r_code")
>
> ## Check data for Year 2010 for Rhode Island
> ri_2010 <- c(1052567, 1052940, 1053337)  # Data from Census, Estimates Base and 2010 Column
> ri_density <- density(ri_2010)
> plot(ri_density)  # Displays normal distribution
>
> posstdint <- mean(ri_2010) + sd(ri_2010)
> negstdint <- mean(ri_2010) - sd(ri_2010)
>
> print("1 Std. Dev Interval: ")
[1] "1 Std. Dev Interval: "
> posstdint
[1] 1053333
> negstdint
[1] 1052563
>
> ri_2010[3] - posstdint
[1] 3.937667
>
> ## Conclusion: The 2010 column entry is approx. 1 std dev away from mean.
> ## Thus we can drop April Census and Baseline Estimate features in favor of 2010 July Estimates
>
> ## Create Linear Regression Models
> trainingdata <-
+   read.csv("../../data/processed/population_ri_train.csv")  # Data saved in another subfolder
>
> ## Split Training Data set into respective Predictor and Response Vectors
> year <-
+   trainingdata[, 1] # First feature is the predictor variable Year
> population <-
+   trainingdata[, 2] # Second feature is the response variable Population
>
> ## Create Scatterplot Matrix to find if Realtionship between two variables
> pairs(cbind(population, year))
>
> ## Conclusion: Some form of Linear regression exists
>
```

```
> ## Perform Train-Test Split of the training data set
> set.seed(100)
> trainingsize <-
+   nrow(trainingdata)  # Total # of rows in training data set
> trainRowIndex <- sample(trainingsize, 0.8 * trainingsize)
>
> split_train <- trainingdata[trainRowIndex, ]
> split_test <- trainingdata[-trainRowIndex, ]
>
> split_train <- split_train[order(split_train$Year),]  ## Sort in Ascending format
>
> split_train
  Year Population
1 2010    1053337
2 2011    1052451
3 2012    1052901
6 2015    1055607
7 2016    1056426
> split_test
  Year Population
4 2013    1053033
5 2014    1054480
>
> ## Create Linear Regression Model
> lr_model <- lm(Population ~ Year, data = split_train)
> summary(lr_model)

Call:
lm(formula = Population ~ Year, data = split_train)

Residuals:
      1       2       3       6       7
 950.16 -563.54 -741.24   81.66  272.96

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -209293.2   305404.4  -0.685   0.5424
Year            627.7      151.7   4.137   0.0256 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 785.5 on 3 degrees of freedom
Multiple R-squared:  0.8509,    Adjusted R-squared:  0.8011
F-statistic: 17.11 on 1 and 3 DF,  p-value: 0.02564
```

```
> ## Check Linear Regression Model Assumptions
>
> # Independence Test
> plot(split_train$Year,
+      residuals(lr_model),
+      xlab = "Year",
+      ylab = "Residuals for Year Feature")
> qqline(0)
> # Mean of 0 AND Constant Variance - Fitted vs Residuals
> plot(fitted(lr_model),
+      residuals(lr_model),
+      xlab = "Fitted Values",
+      ylab = "Residuals")
> qqline(0)
>
> # Check for normal distribution of training data subset
> qqnorm(residuals(lr_model))
> qqline(residuals(lr_model))
>
> ## Generate test results
> predict_test <- predict(lr_model, newdata = split_test)
> predict_test
      4       5
1054270 1054898
>
> ## Calculate the performance of the model
> ActvsPred <-
+   data.frame(cbind(Actuals = split_test$Population, Predictions = predict_test))  # Create data frame for comparison
> ActvsPred
  Actuals Predictions
4 1053033     1054270
5 1054480     1054898
>
> # Calculate Error Rates
> DMwR::regr.eval(ActvsPred$Actuals, ActvsPred$Predictions)
         mae          mse         rmse         mape
8.272910e+02 8.522230e+05 9.231592e+02 7.853548e-04
> # Min-Max Accuracy
> min_max_accuracy <-
+   mean(apply(ActvsPred, 1, min) / apply(ActvsPred, 1, max))
> min_max_accuracy
[1] 0.9992154
>
> ## Generate Final predictions for next 5 years
> testfeatures <- read.csv("../../data/processed/population_test.csv")
> testfeatures
  Year
1 2020
2 2021
3 2022
4 2023
5 2024
6 2025
>
> fiveyearpredictions <- predict(lr_model, newdata = testfeatures)
> fiveyearpredictions
      1       2       3       4       5       6
1058664 1059292 1059919 1060547 1061175 1061802
```

```
> finalpredictions <-
+   data.frame(Year = testfeatures, Population = fiveyearpredictions)
> finalpredictions
  Year Population
1 2020    1058664
2 2021    1059292
3 2022    1059919
4 2023    1060547
5 2024    1061175
6 2025    1061802
>
> ## Write the output to the file
> setwd("../../data/external")  # set the directory where you want to save final output file
> write.csv(finalpredictions, "population_predictions.csv", row.names = FALSE)
> setwd("../../src/r_code")  # Set the PWD back to original
>
```

Output File

Year	Population
2020	1058664
2021	1059292
2022	1059919
2023	1060547
2024	1061175
2025	1061802