Prof. Overbey

3/10/2025

Quantitative Reasoning II

Checkpoint 4: Code and Plots

Task 1: Visualizing Actual vs Predicted Values

Goal: Evaluate how well your model performs on the dataset that generated the model using the `predict()` function and plotting functions.

Model from Checkpoint 2

```
Original_Model <- lm(Salary ~ Education_Years + Work_Experience, data = dataset)
```

Model from Checkpoint 3

```
Adjusted_Model <- lm(Salary ~ Education_Years + Work_Experience + City_Population + Age, data = dataset)
```

Task 1: Visualizing Actual vs Predicted Values

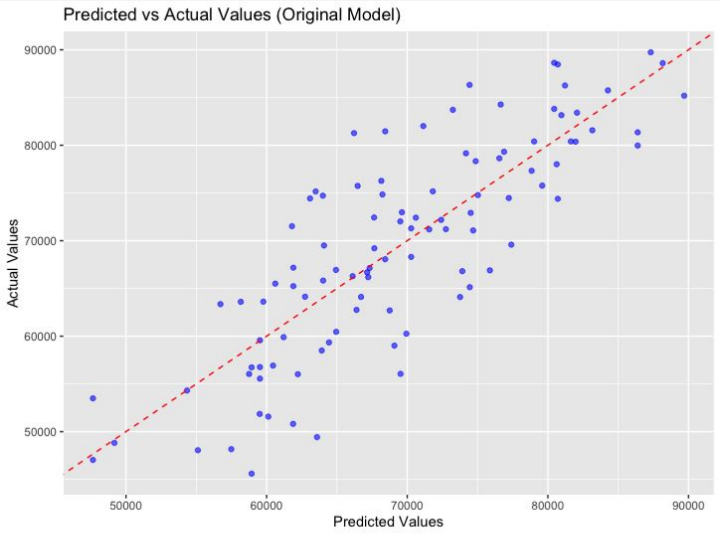
Model from Checkpoint 2

```
Original_Model <- lm(Salary ~ Education_Years + Work_Experience, data = dataset)
```

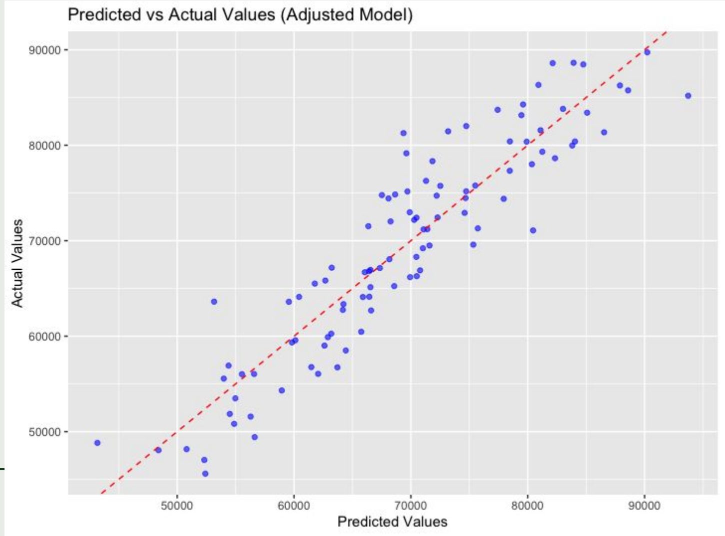
Model from Checkpoint 3

```
Adjusted_Model <- lm(Salary ~ Education_Years + Work_Experience + City_Population + Age, data = dataset)
```

Original Model



Adjusted Model

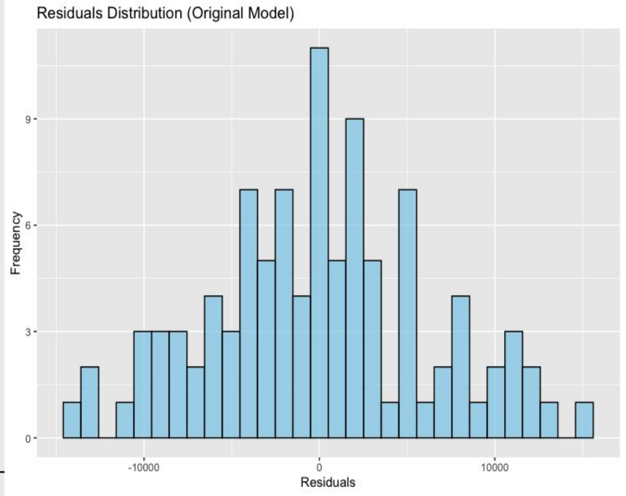


Task 2: Visualizing Residual Distributions

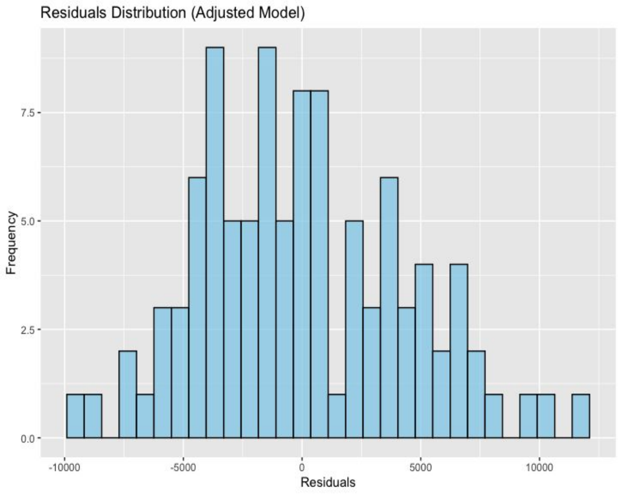
Goal: Interpret differences in residual distribution between original and adjusted models.

Residuals are the differences between the **observed values** and the **predicted values** from a statistical or regression model. They represent the part of the data that the model does **not** explain.

Original Model



Adjusted Model

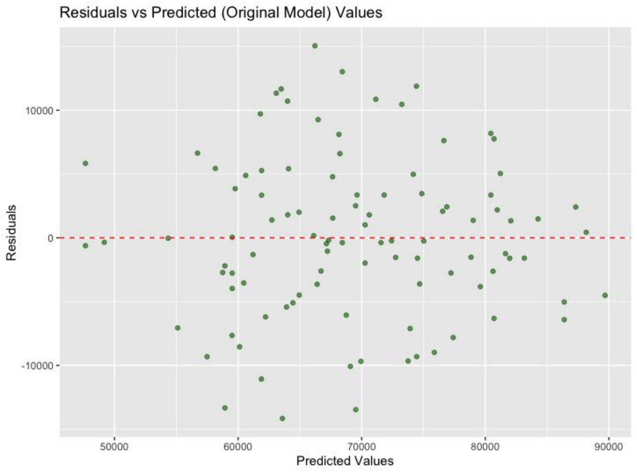


Task 3: Visualizing Residuals vs Predicted Values

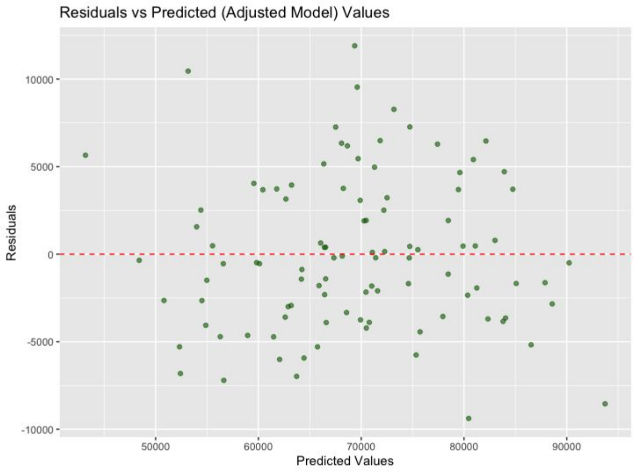
Goal: Assess the variance of residuals between your original and adjusted models.

- Identify patterns or trends in the residuals.
- Detect heteroscedasticity (unequal variance of residuals), which suggests potential violations of regression assumptions.
- Recognize influential observations or outliers that significantly affect the model.

Original Model



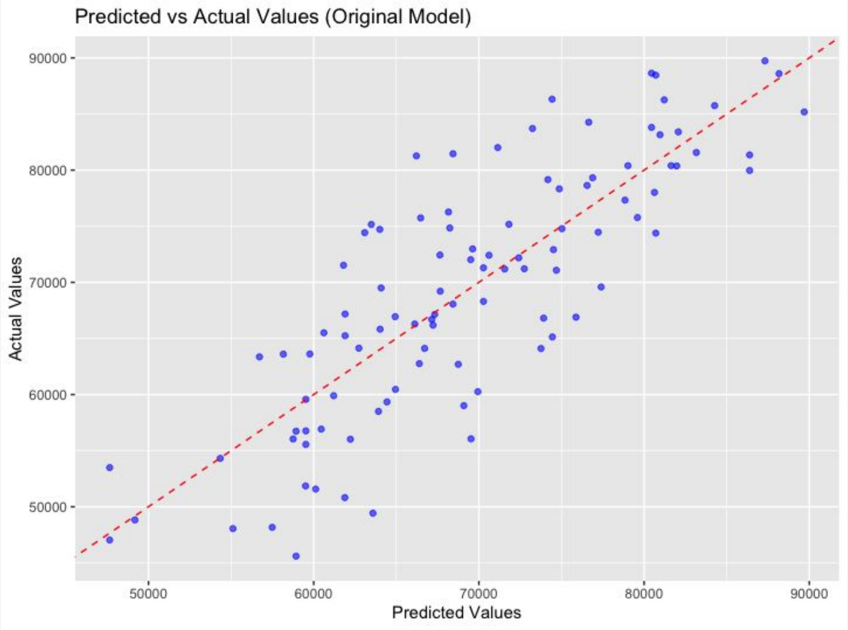
Adjusted Model



How to make plots in R

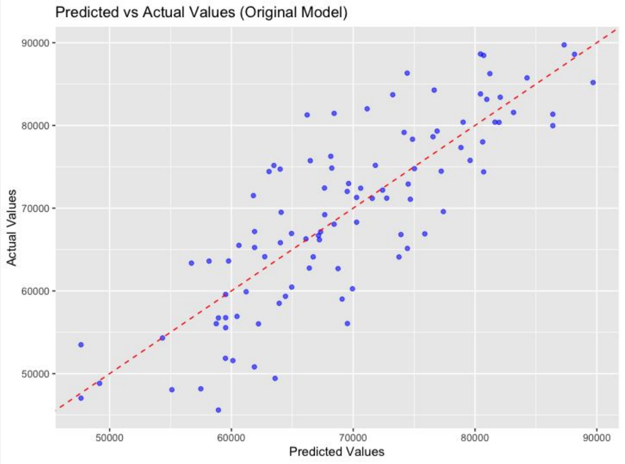
This checkpoint requires you to make multiple plots for analysis. The gold standard method of doing this in R is with a package called **`ggplot2`**.

`ggplot2` is a powerful data visualization library in R that allows users to construct graphs by layering various visual components (using the "+" operator), leading to expressive, customizable, and informative visualizations.



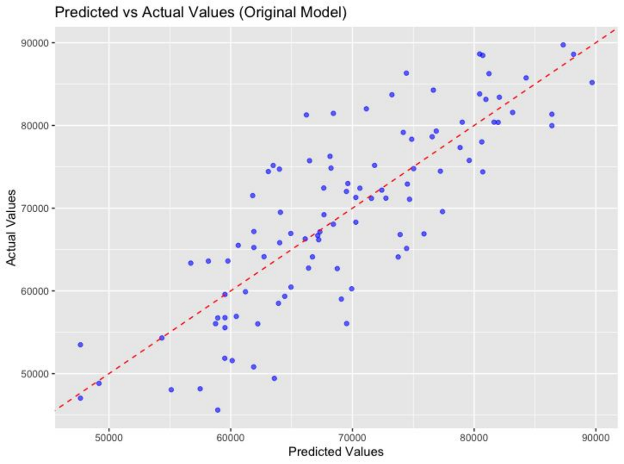
How to make plots in R

```r
# Visualization 1: Predicted vs Actual values (Original Model)

ggplot(plot_data, aes(x = Predicted_Original, y = Salary)) +

  geom_point(color = "blue", alpha = 0.6) +

  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +

  labs(title = "Predicted vs Actual Values (Original Model)",

       x = "Predicted Values",

       y = "Actual Values")
```



How to make plots in R

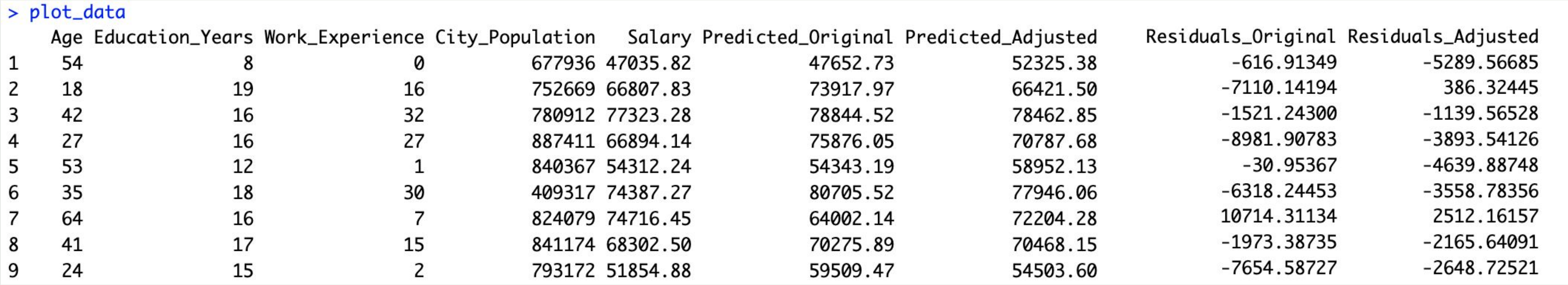
```
# Visualization 1: Predicted vs Actual values (Original Model)

ggplot(plot_data, aes(x = Predicted_Original, y = Salary)) +

  geom_point(color = "blue", alpha = 0.6) +

  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +

  labs(title = "Predicted vs Actual Values (Original Model)",

       x = "Predicted Values",

       y = "Actual Values")
```



Preprocessing Data for Visualization

```r
# Load necessary libraries
library(ggplot2)
library(dplyr)

# Load your dataset (replace path with your actual dataset path)
dataset <- read.csv("~/Downloads/simulated_salary_dataset.csv")

# Build your models for visualization purposes (adjust with your actual variable names)
Original_Model <- lm(Salary ~ Education_Years + Work_Experience, data = dataset)
Adjusted_Model <- lm(Salary ~ Education_Years + Work_Experience + City_Population + Age, data = dataset)

# Create a dataframe for visualization
plot_data <- dataset %>%
  mutate(
    Predicted_Original = predict(Original_Model),
    Predicted_Adjusted = predict(Adjusted_Model),
    Residuals_Original = resid(Original_Model),
    Residuals_Adjusted = resid(Adjusted_Model)
  )
```

Preprocessing Data for Visualization



```
> plot_data
  Age Education_Years Work_Experience City_Population   Salary Predicted_Original Predicted_Adjusted     Residuals_Original Residuals_Adjusted
1  54               8               0          677936 47035.82           47652.73           52325.38             -616.91349        -5289.56685
2  18              19              16          752669 66807.83           73917.97           66421.50            -7110.14194          386.32445
3  42              16              32          780912 77323.28           78844.52           78462.85            -1521.24300        -1139.56528
4  27              16              27          887411 66894.14           75876.05           70787.68            -8981.90783        -3893.54126
5  53              12               1          840367 54312.24           54343.19           58952.13              -30.95367        -4639.88748
6  35              18              30          409317 74387.27           80705.52           77946.06            -6318.24453        -3558.78356
7  64              16               7          824079 74716.45           64002.14           72204.28            10714.31134         2512.16157
8  41              17              15          841174 68302.50           70275.89           70468.15            -1973.38735        -2165.64091
9  24              15               2          793172 51854.88           59509.47           54503.60            -7654.58727        -2648.72521
```

Getting Started

1. Reference the starter code in the class Github in the final project folder that contains code for generating plots using your original model.

2. Load the following libraries to see if you have the required R packages installed. If R gives you an error message, first install these libraries:

```r
# Load necessary libraries
library(ggplot2)
library(dplyr)
```