49.5/50

Homework-1

Commented [DJ1]: Total Points 49.5/50

Dancun Juma

2025-01-15

TASK 1

```
head(auto_data)

##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

Dataset Description

mpg (miles per gallon): A continuous variable indicating the fuel efficiency of the car, measured in miles the car can travel per gallon of fuel.

cylinders: A numerical variable indicating the number of cylinders in the car's engine.

displacement: A continuous variable representing the engine displacement, typically measured in cubic inches. It provides an idea of the engine's size and capacity.

horsepower: A numerical variable showing the power output of the engine, measured in horsepower.

weight: A continuous variable indicating the car's weight in pounds.

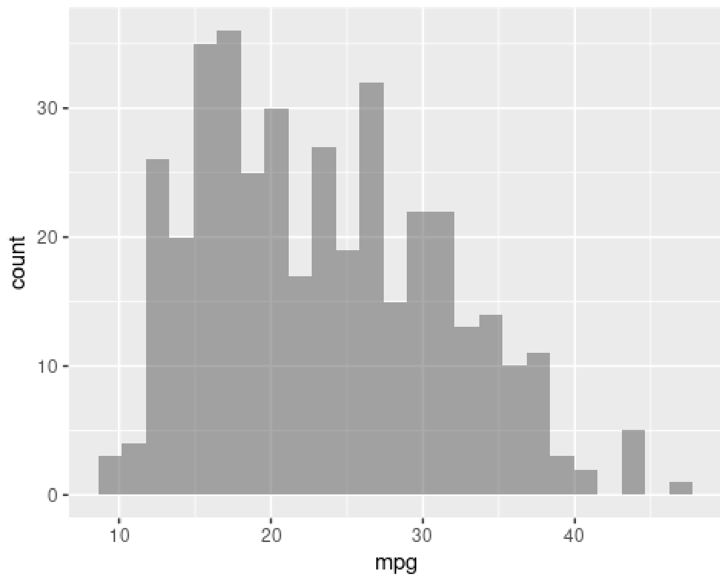
acceleration: A continuous variable representing how quickly the car can accelerate, measured in seconds required to go from 0 to 60 mph.

year: A numerical variable indicating the year of manufacture of the car. For example, "70" represents 1970.

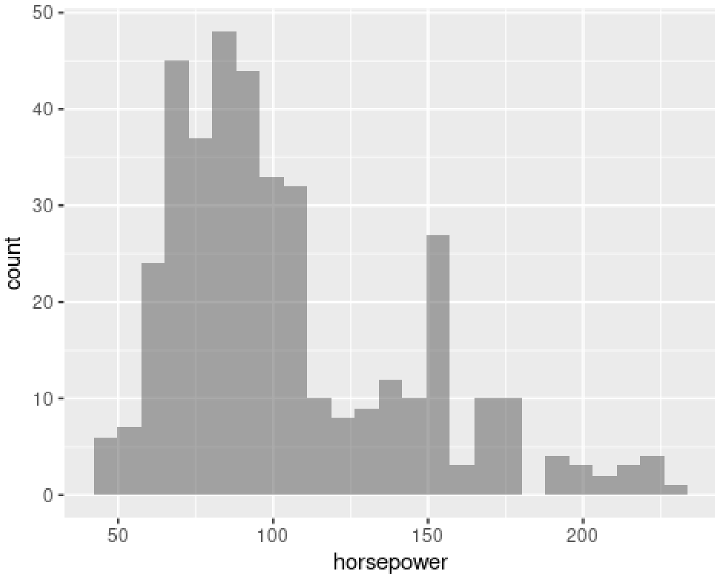
origin: A numerical variable representing the region of origin of the car.

name: A categorical (factor) variable containing the name or model of the car, including its manufacturer and specific model identifier.

```
auto_data %>%
  gf_histogram(~mpg)
```




```
auto_data %>%
  gf_histogram(~horsepower)
```



MPG

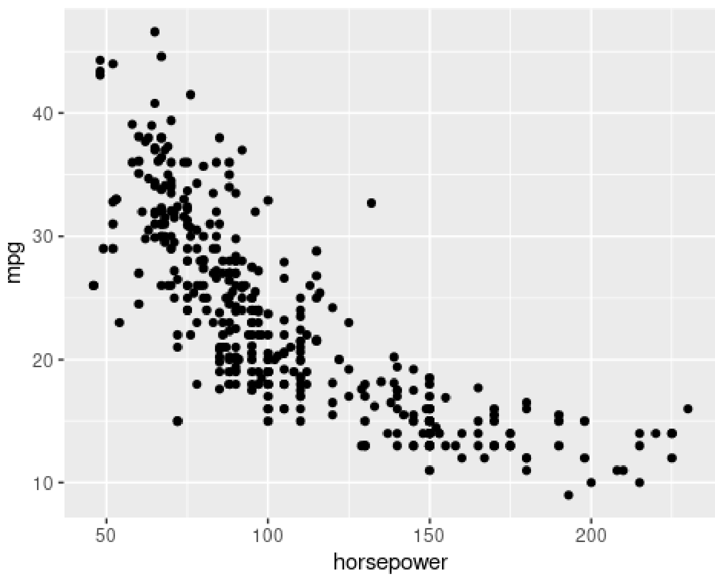
Histogram

The mpg histogram shows a slight right-skeweness. This means that most cars have fuel efficiency between 15-30 mpg, with a peak at 20-25 mpg as shown in the graph. I can say only few cars achieve higher efficiency that is above 30 mpg.

Horsepower Histogram

On the horsepower histogram which is bimodal, since peaks are at 90-110 and around 150. I believe this indicates a mix of lower-powered and higher-powered vehicles that are in this dataset. I totally oberverd that very few cars exceed 200 horsepower which shows or indicates that high-performance engines are rare. In general, I believe the data reflects diverse engine capacities, with most vehicles having average horsepower.

```
auto_data %>%
  gf_point(mpg~horsepower)
```



Descripitive statistics

```
auto_data %>%
  summarise(mean=mean(mpg),sd=sd(mpg))

##       mean       sd
## 1 23.44592 7.805007

auto_data %>%
  summarise(mean=mean(horsepower),sd=sd(horsepower))

##       mean       sd
## 1 104.4694 38.49116

with(auto_data,cor(mpg,horsepower))

## [1] -0.7784268
```

The average or mean `mpg` is 23.45 based on the results above with a standard deviation of 7.81, showing moderate variability in fuel efficiency. On the other hand, looking at the `horsepower`, the mean is 104.47, with higher variability that is (SD = 38.49). Moreover, from the results, there is a strong negative correlation (-0.78) which indicates that as horsepower increases, fuel efficiency decreases hence highlighting the trade-off between power and efficiency.

```
favstats(~mpg,data=auto_data)

##  min Q1 median Q3  max     mean       sd   n missing
##    9 17  22.75 29 46.6 23.44592 7.805007 392       0
```

```r
favstats(~horsepower,data=auto_data)

##  min Q1 median  Q3 max     mean       sd   n missing
##   46 75   93.5 126 230 104.4694 38.49116 392       0
```

MPG Summary

The summary shows that fuel efficiency (mpg) ranges from 9 to 46.6, with a median of 22.75. Most cars fall between 17 (Q1) and 29 (Q3), around the average of 23.45. The dataset includes 392 cars, with no missing values.

Horsepower Summary

The engine power (horsepower) ranges from 46 to 230, with a median of 93.5. Most vehicles lie between 75 (Q1) and 126 (Q3), near the average of 104.47.

Simple Linear Regression

a. Perform Simple Linear Regression and Analyze the Output

```r
lm_spec <- linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")

lm_spec

## Linear Regression Model Specification (regression)
##
## Computational engine: lm

# Fit the simple linear regression model
slr_mod <- lm(mpg ~ horsepower, data = auto_data)

# Display the regression model summary
tidy(slr_mod)

## # A tibble: 2 × 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)   39.9     0.717        55.7 1.22e-187
## 2 horsepower    -0.158   0.00645     -24.5 7.03e- 81
```

$$\widehat{mpg} = 39.9359 - 0.1578 \times horsepower$$

i. Is there a relationship between the predictor (horsepower) and the response (mpg)?

Yes, the p-value associated with the horsepower coefficient is < 2e-16, which is highly significant, indicating a strong relationship between horsepower and mpg.

ii. How strong is the relationship?

The multiple R-squared value is 0.6059, which means that approximately 60.6% of the variability in mpg can be explained by horsepower. This is a moderate relationship.

iii. Is the relationship between the predictor and the response positive or negative?

The coefficient for horsepower is negative (-0.157845), indicating a negative relationship between horsepower and mpg. As horsepower increases, mpg tends to decrease.

iv. What is the predicted mpg associated with a horsepower of 98?

```
predict(slr_mod, newdata = data.frame(horsepower = 98))
```

```
##        1
## 24.46708
```

To predict mpg for a horsepower of 98, we can use the regression equation:

For horsepower = 98

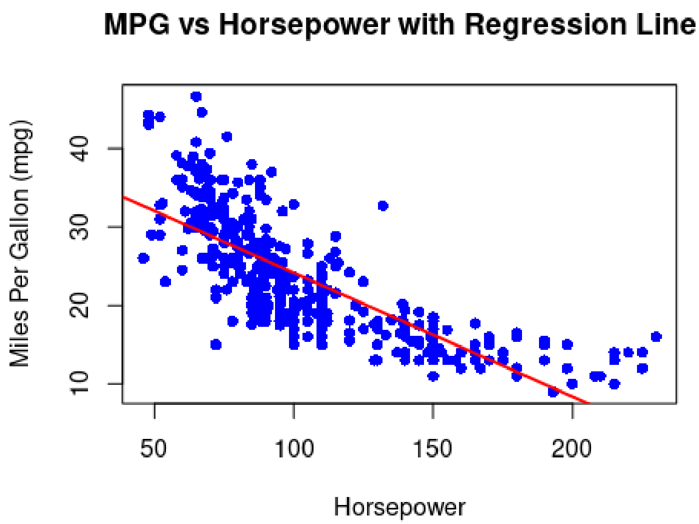
$mpg=39.935861-0.157845\times98=39.935861-15.482=24.46$

The predicted mpg is approximately 24.46.

Commented [DJ2]: 5/5 Marks. The explanations are okay

b. Plot Response vs. Predictor with Regression Line

```
# Plot mpg vs. horsepower and add regression line
plot(auto_data$horsepower, auto_data$mpg,
     xlab = "Horsepower",
     ylab = "Miles Per Gallon (mpg)",
     main = "MPG vs Horsepower with Regression Line",
     pch = 16, col = "blue")

# Add regression line
abline(slr_mod, col = "red", lwd = 2)
```

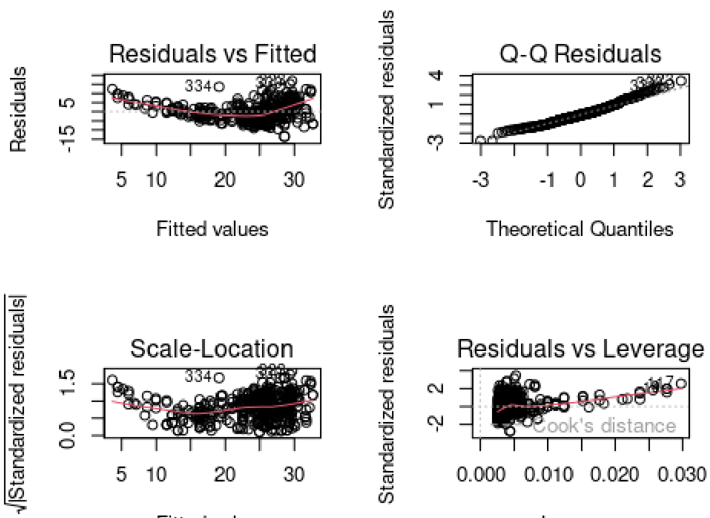


From the plot above, the negative slope of the regression line confirms an inverse relationship that I can say as horsepower increases, MPG decreases on the other side. This shows that the cars with higher engine power generally have lower fuel efficiency. The data points on the other hand show variability around the line, suggesting some cars deviate from this trend

Commented [DJ3]: For part b) 5/5 Marks. The plot response vs predictor plot looks good and interpreted correctly

c. Diagnostic Plots for Linear Regression Fit

```r
# Generate diagnostic plots
par(mfrow = c(2, 2))
plot(slr_mod)
```



The Residuals vs Fitted plot shows a curved pattern, suggesting the model maybe does not fully capture the non-linear relationship between variables.

On the second plot that is the Q-Q plot indicates residuals are mostly normal, but slight deviations at the tails suggest some non-normality.

The Scale-Location plot reveals increasing residual spread, indicating heteroscedasticity (non-constant error variance), which may affect prediction accuracy.

The Residuals vs Leverage plot highlights influential points that could overly impact the model.

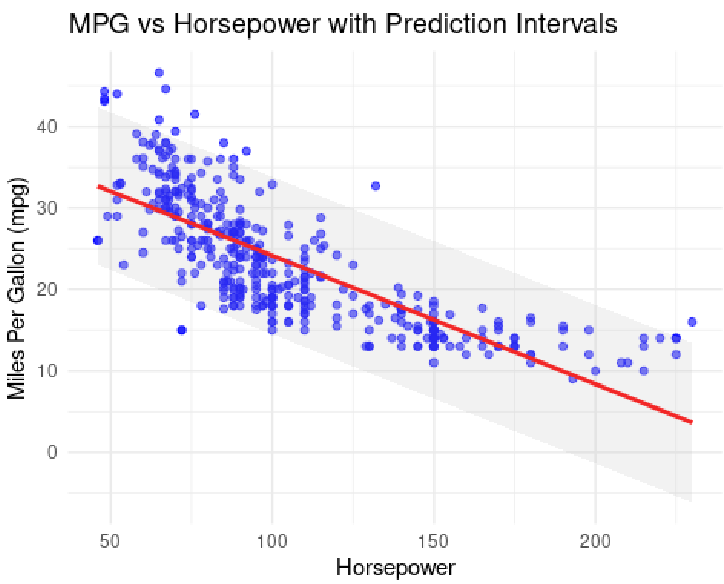
Issues include non-linearity, heteroscedasticity, and influential points, suggesting the model could benefit from adjustments, like transformations or alternative regression methods, for better accuracy.

Commented [DJ4]: For part c) 5/5 Marks. I did a lot of plots here but the ones that counts is in the last page of Task 1 Part with all the interpretations reasonably done

d. Generate a ggplot with Raw Data, Prediction Line, and Confidence/Prediction Intervals

```
# Add predictions and intervals to the dataset
predictions <- predict(slr_mod, auto_data, interval = "prediction")
auto_data <- auto_data %>%
  mutate(.fitted = predictions[, "fit"],
         .pred_lower = predictions[, "lwr"],
         .pred_upper = predictions[, "upr"])

# Generate ggplot
ggplot(auto_data, aes(x = horsepower, y = mpg)) +
  geom_point(alpha = 0.6, color = "blue") +
```

```r
  geom_line(aes(y = .fitted), color = "red", size = 1) +
  geom_ribbon(aes(ymin = .pred_lower, ymax = .pred_upper), alpha = 0.2, fill
= "grey") +
  labs(
    title = "MPG vs Horsepower with Prediction Intervals",
    x = "Horsepower",
    y = "Miles Per Gallon (mpg)"
  ) +
  theme_minimal()
```



This graphic shows the relationship between horsepower and miles per gallon that is (MPG). The red line above represents the predicted MPG based on horsepower, showing a clear negative trend which can be interpreted as vehicles with higher horsepower tend to have lower MPG. Moreover, the gray shaded area represents the prediction intervals, indicating the range within which most data points are expected to fall. The blue points in the plot highlights the observed data, showing variability but generally aligning with the trend we observed. This reinforces the inverse relationship between horsepower and fuel efficiency.

```r
lm.fit <- lm(mpg~horsepower,data=auto_data)
cbind(predict(lm.fit,interval="confidence"),
      predict(lm.fit,interval="predict")) %>% as_tibble()
```

```
## # A tibble: 392 × 6
##      fit   lwr   upr    V4    V5    V6
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 19.4  18.8  20.0  19.4   9.75  29.1
##  2 13.9  13.0  14.8  13.9   4.20  23.6
```

```
##   3 16.3   15.5   17.0   16.3    6.58    25.9
##   4 16.3   15.5   17.0   16.3    6.58    25.9
##   5 17.8   17.2   18.5   17.8    8.17    27.5
##   6  8.68   7.40   9.96   8.68 -1.05    18.4
##   7  5.21   3.67   6.75   5.21 -4.56    15.0
##   8  6.00   4.52   7.48   6.00 -3.76    15.8
##   9  4.42   2.82   6.02   4.42 -5.36    14.2
##  10  9.95   8.76  11.1    9.95  0.227   19.7
## # ℹ 382 more rows
```

```
tidy(slr_mod) %>% str()
```

```
## tibble [2 × 5] (S3: tbl_df/tbl/data.frame)
##  $ term     : chr [1:2] "(Intercept)" "horsepower"
##  $ estimate : num [1:2] 39.936 -0.158
##  $ std.error: num [1:2] 0.7175 0.00645
##  $ statistic: num [1:2] 55.7 -24.5
##  $ p.value  : num [1:2] 1.22e-187 7.03e-81
```

Commented [DJ5]: For part d) 4.5/5. The only missing thing is the CI lines but the grey region replaces the blue lines in the solution doc.

Assessing

```
glance(slr_mod)
```

```
## # A tibble: 1 × 12
##   r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BI
C
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl
>
## 1     0.606         0.605  4.91      600. 7.03e-81     1 -1179. 2363. 2375
.
## # ℹ 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Assess with test/train

```
# Set seed before random split
set.seed(15)

# Put 80% of the data into the training set
auto_split <- initial_split(auto_data, prop = 0.80)

# Assign the two splits to data frames - with descriptive names
auto_train <- training(auto_split)
auto_test <- testing(auto_split)

# Check the splits
head(auto_train)
```

```
##      mpg cylinders displacement horsepower weight acceleration year origin
## 38  18.0         6          232        100   3288         15.5   71      1
## 367 17.6         6          225         85   3465         16.6   81      1
## 164 18.0         6          225         95   3785         19.0   75      1
```

```
## 296 35.7         4           98         80   1915         14.4   79      1
## 179 23.0         4          120         88   2957         17.0   75      2
## 263 19.2         8          305        145   3425         13.2   78      1
##                             name  .fitted .pred_lower .pred_upper
## 38                   amc matador 24.15139   14.493888    33.80889
## 367        chrysler lebaron salon 26.51906   16.858575    36.17954
## 164                plymouth fury 24.94061   15.282533    34.59869
## 296  dodge colt hatchback custom 27.30828   17.645972    36.97059
## 179                  peugeot 504 26.04552   16.385936    35.70511
## 263 chevrolet monte carlo landau 17.04837    7.377393    26.71936
```

```r
head(auto_test)
```

```
##    mpg cylinders displacement horsepower weight acceleration year origin
## 3   18         8          318        150   3436         11.0   70      1
## 4   16         8          304        150   3433         12.0   70      1
## 5   17         8          302        140   3449         10.5   70      1
## 16  22         6          198         95   2833         15.5   70      1
## 17  18         6          199         97   2774         15.5   70      1
## 22  24         4          107         90   2430         14.5   70      2
##                  name  .fitted .pred_lower .pred_upper
## 3  plymouth satellite 16.25915    6.584598    25.93370
## 4       amc rebel sst 16.25915    6.584598    25.93370
## 5         ford torino 17.83760    8.169775    27.50542
## 16    plymouth duster 24.94061   15.282533    34.59869
## 17         amc hornet 24.62492   14.967125    34.28272
## 22        audi 100 ls 25.72984   16.070761    35.38891
```

Train and fit linear model

```r
# Fit the model using the training data
slr_train <- lm(mpg ~ horsepower, data = auto_train)

# Summarize the model fit
tidy(slr_train)
```

```
## # A tibble: 2 × 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)   39.8     0.797        49.9 1.56e-150
## 2 horsepower    -0.157   0.00712     -22.1 1.24e- 65
```

The regression analysis from the training subset above reveals a strong negative relationship between horsepower and MPG. The intercept (39.78) suggests that a vehicle with zero horsepower would theoretically have an MPG of 39.78. The coefficient for horsepower (-0.157) indicates that each additional unit of horsepower decreases MPG by approximately 0.157 units.

Model Evaluation on Test Data

```r
# Augment the model to include predictions on the test data
test_aug <- augment(slr_train, new_data = auto_test)
```

```r
# Check the augmented results
head(test_aug)
```

```
## # A tibble: 6 × 9
##   .rownames   mpg horsepower .fitted .resid    .hat .sigma  .cooksd .std.r
esid
##   <chr>     <dbl>      <dbl>   <dbl>  <dbl>   <dbl>  <dbl>    <dbl>      <
dbl>
## 1 38         18          100    24.1  -6.05 0.00325   4.81 0.00259      -1
.26
## 2 367        17.6         85    26.4  -8.81 0.00408   4.79 0.00690      -1
.83
## 3 164        18           95    24.8  -6.84 0.00342   4.80 0.00348      -1
.42
## 4 296        35.7         80    27.2   8.50 0.00458   4.80 0.00721       1
.77
## 5 179        23           88    25.9  -2.94 0.00384   4.82 0.000721     -0
.612
## 6 263        19.2        145    17.0   2.23 0.00668   4.82 0.000724      0
.464
```

R-Squared for the Test Data

```r
glance(slr_train)
```

```
## # A tibble: 1 × 12
##   r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BI
C
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl
>
## 1     0.610         0.609  4.81      487. 1.24e-65     1  -935. 1876. 1887
.
## # ℹ 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Based on the R-squared which is 0.610, about 61% of the variability in MPG is explained by horsepower, suggesting a moderately strong relationship.

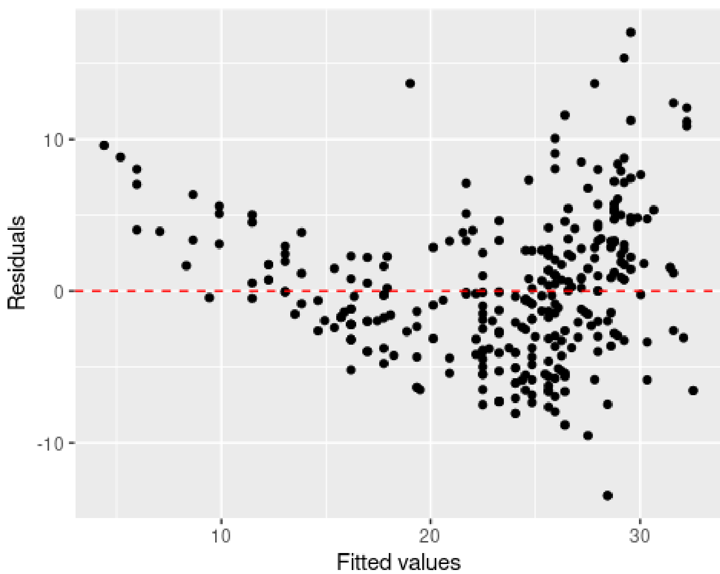
Check for Linearity (Residuals vs. Fitted)

```r
train_aug <- augment(slr_train, new_data = auto_train)
head(train_aug)
```

```
## # A tibble: 6 × 9
##   .rownames   mpg horsepower .fitted .resid    .hat .sigma  .cooksd .std.r
esid
##   <chr>     <dbl>      <dbl>   <dbl>  <dbl>   <dbl>  <dbl>    <dbl>      <
dbl>
## 1 38         18          100    24.1  -6.05 0.00325   4.81 0.00259      -1
.26
## 2 367        17.6         85    26.4  -8.81 0.00408   4.79 0.00690      -1
.83
## 3 164        18           95    24.8  -6.84 0.00342   4.80 0.00348      -1
```

```
.42
## 4 296         35.7         80     27.2    8.50 0.00458    4.80 0.00721       1
.77
## 5 179         23           88     25.9   -2.94 0.00384    4.82 0.000721     -0
.612
## 6 263         19.2        145     17.0    2.23 0.00668    4.82 0.000724      0
.464
```

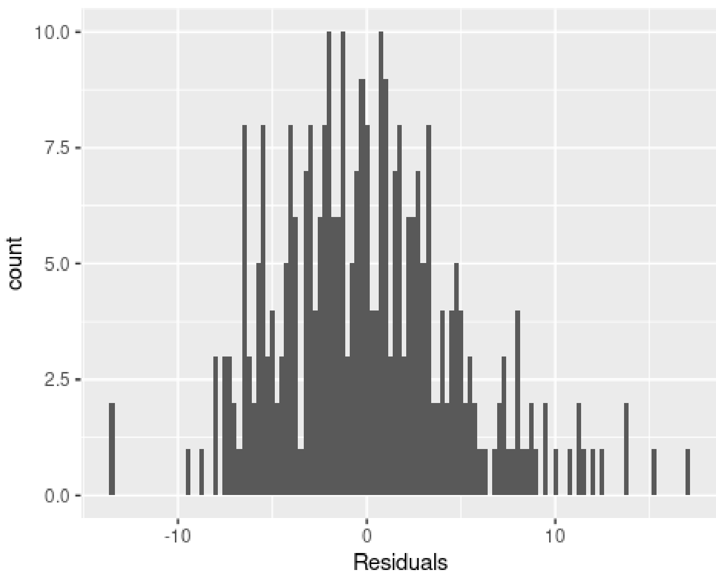
```r
ggplot(data = train_aug, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  xlab("Fitted values") +
  ylab("Residuals")
```



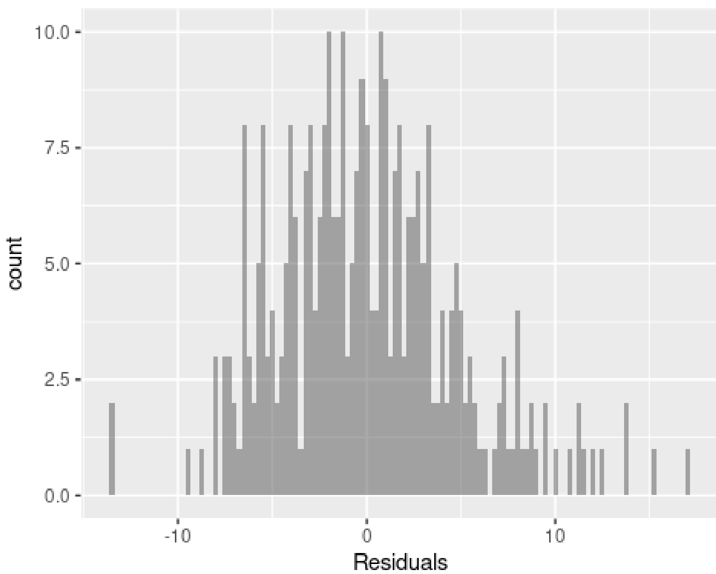
While most residuals cluster around zero, suggesting a reasonable fit, the spread increases for higher fitted values, indicating heteroscedasticity. The pattern above suggests that the model's variance is not somehow constant, violating a key linear regression assumption. Adjustments, such as transformation or alternative models, may be necessary to improve accuracy and address this issue.

Plotting residuals

```r
ggplot(data = train_aug, aes(x = .resid)) +
  geom_histogram(binwidth = 0.25) +
  xlab("Residuals")
```

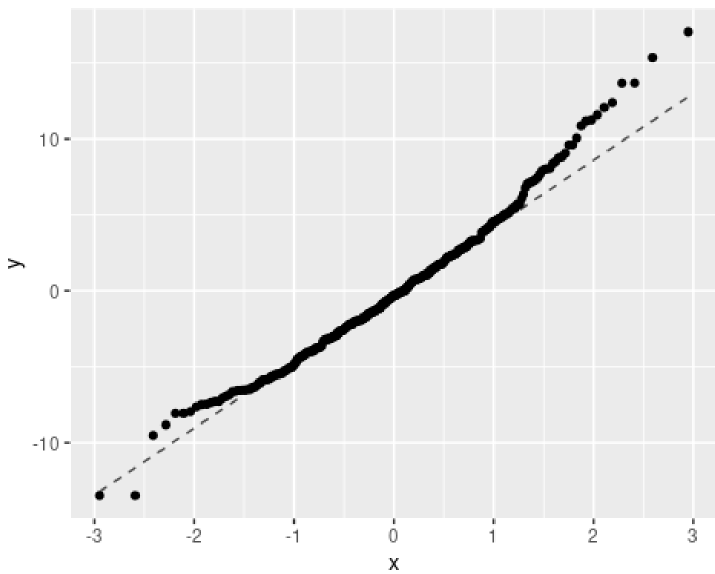



```
train_aug %>%
  gf_histogram(~.resid,binwidth=.25) %>%
  gf_labs(x="Residuals")
```



The residuals shows slight skewness hence we can say there is normality.

```
train_aug %>%
  gf_qq(~.resid) %>%
  gf_qqline()
```



I can say the points closely follow the diagonal line in the central region, indicating normality for most residuals, deviations likely occur at both tails as shown in my plot above. The points in the upper and lower extremes diverge from the line as can be observed too, suggesting that the residuals may not be perfectly normally distributed

Inference: Confidence Intervals

```
tidy(slr_mod,conf.int=TRUE)

## # A tibble: 2 × 7
##   term        estimate std.error statistic   p.value conf.low conf.high
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>    <dbl>     <dbl>
## 1 (Intercept)   39.9     0.717        55.7 1.22e-187   38.5      41.3
## 2 horsepower    -0.158   0.00645     -24.5 7.03e- 81   -0.171    -0.145
```

Confidence Intervals for Model Coefficients

```
# Get confidence intervals for the coefficients
tidy(slr_train, conf.int = TRUE)

## # A tibble: 2 × 7
##   term        estimate std.error statistic   p.value conf.low conf.high
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>    <dbl>     <dbl>
## 1 (Intercept)   39.8     0.797        49.9 1.56e-150   38.2      41.3
## 2 horsepower    -0.157   0.00712     -22.1 1.24e- 65   -0.171    -0.143

# Predictions with confidence intervals and prediction intervals
cbind(
  predict(slr_train, new_data = auto_data, interval = "confidence"),
```

```r
  predict(slr_train, new_data = auto_data, interval = "predict")
)
```

```
##           fit       lwr       upr       fit        lwr      upr
## 38  24.051691 23.511609 24.591773 24.051691 14.5663784 33.53700
## 367 26.410776 25.805680 27.015872 26.410776 16.9215394 35.90001
## 164 24.838053 24.284264 25.391842 24.838053 15.3519497 34.32416
## 296 27.197138 26.556352 27.837924 27.197138 17.7055584 36.68872
## 179 25.938959 25.352293 26.525625 25.938959 16.4508798 35.42704
## 263 16.974437 16.200595 17.748278 16.974437  7.4729472 26.47593
## 219 30.657129 29.806855 31.507403 30.657129 21.1491092 40.16515
## 206 27.983499 27.301666 28.665333 27.983499 18.4890605 37.47794
## 368 25.938959 25.352293 26.525625 25.938959 16.4508798 35.42704
## 85  25.938959 25.352293 26.525625 25.938959 16.4508798 35.42704
## 232  9.897182  8.592682 11.201682  9.897182  0.3378311 19.45653
## 195 25.624414 25.048654 26.200175 25.624414 16.1370033 35.11183
## 142 26.725321 26.106654 27.343988 26.725321 17.2352090 36.21543
## 247 31.600763 30.683613 32.517912 31.600763 22.0865295 41.11500
## 397 26.882593 26.256780 27.508406 26.882593 17.3920128 36.37317
## 217 29.084406 28.337809 29.831002 29.084406 19.5850962 38.58371
## 283 25.938959 25.352293 26.525625 25.938959 16.4508798 35.42704
## 220 24.680781 24.130419 25.231142 24.680781 15.1948768 34.16668
## 215 19.333522 18.694747 19.972296 19.333522  9.8420778 28.82497
## 183 26.253504 25.654816 26.852191 26.253504 16.7646736 35.74233
## 360 27.197138 26.556352 27.837924 27.197138 17.7055584 36.68872
## 19  25.938959 25.352293 26.525625 25.938959 16.4508798 35.42704
## 330 29.241678 28.485243 29.998113 29.241678 19.7415903 38.74177
## 108 24.051691 23.511609 24.591773 24.051691 14.5663784 33.53700
## 288 18.075343 17.369093 18.781593 18.075343  8.5791195 27.57157
## 21  26.096231 25.503690 26.688773 26.096231 16.6077870 35.58468
## 317 25.624414 25.048654 26.200175 25.624414 16.1370033 35.11183
## 167 19.490794 18.859559 20.122028 19.490794  9.9998546 28.98173
## 80  28.927133 28.190240 29.664027 28.927133 19.4285816 38.42568
## 285 22.478968 21.939363 23.018573 22.478968 12.9936823 31.96425
## 87  16.188075 15.362195 17.013955 16.188075  6.6822059 25.69394
## 2   13.828990 12.833619 14.824361 13.828990  4.3068983 23.35108
## 144 27.511682 26.855067 28.168298 27.511682 18.0190212 37.00434
## 188 17.760798 17.035949 18.485648 17.760798  8.2631734 27.25842
## 10   9.897182  8.592682 11.201682  9.897182  0.3378311 19.45653
## 203 24.838053 24.284264 25.391842 24.838053 15.3519497 34.32416
## 175 24.523508 23.976237 25.070779 24.523508 15.0377833 34.00923
## 211 22.793513 22.256732 23.330293 22.793513 13.3083872 32.27864
## 191 15.873530 15.026107 16.720954 15.873530  6.3657652 25.38130
## 291 17.446254 16.702212 18.190295 17.446254  7.9471448 26.94536
## 297 27.197138 26.556352 27.837924 27.197138 17.7055584 36.68872
## 190 20.906245 20.331827 21.480662 20.906245 11.4189150 30.39357
## 393 26.253504 25.654816 26.852191 26.253504 16.7646736 35.74233
## 352 29.556223 28.779725 30.332720 29.556223 20.0545164 39.05793
## 6    8.639003  7.231489 10.046518  8.639003 -0.9349492 18.21296
## 99  24.051691 23.511609 24.591773 24.051691 14.5663784 33.53700
```

```
## 281 21.692606 21.139748 22.245465 21.692606 12.2065575 31.17866
## 124 20.591700 20.006527 21.176873 20.591700 11.1037130 30.07969
## 307 21.692606 21.139748 22.245465 21.692606 12.2065575 31.17866
## 76  16.188075 15.362195 17.013955 16.188075  6.6822059 25.69394
## 50  26.253504 25.654816 26.852191 26.253504 16.7646736 35.74233
## 186 27.354410 26.705812 28.003008 27.354410 17.8623001 36.84652
## 323 29.556223 28.779725 30.332720 29.556223 20.0545164 39.05793
## 40  12.256267 11.140169 13.372365 12.256267  2.7207992 21.79173
## 89  18.232615 17.535429 18.929802 18.232615  8.7370616 27.72817
## 82  25.309870 24.743836 25.875903 25.309870 15.8230440 34.79670
## 213 11.469905 10.291828 12.647982 11.469905  1.9269846 21.01283
## 20  32.544397 31.557732 33.531062 32.544397 23.0232109 42.06558
## 65  16.188075 15.362195 17.013955 16.188075  6.6822059 25.69394
## 45  12.256267 11.140169 13.372365 12.256267  2.7207992 21.79173
## 342 22.478968 21.939363 23.018573 22.478968 12.9936823 31.96425
## 242 24.523508 23.976237 25.070779 24.523508 15.0377833 34.00923
## 252 17.918071 17.202598 18.633543 17.918071  8.4211568 27.41498
## 372 26.568048 25.956290 27.179807 26.568048 17.0783846 36.05771
## 138 16.188075 15.362195 17.013955 16.188075  6.6822059 25.69394
## 298 27.668955 27.004122 28.333787 27.668955 18.1757217 37.16219
## 286 19.333522 18.694747 19.972296 19.333522  9.8420778 28.82497
## 303 28.769861 28.042530 29.497192 28.769861 19.2720464 38.26768
## 362 21.535334 20.978803 22.091866 21.535334 12.0490704 31.02160
## 71   9.897182  8.592682 11.201682  9.897182  0.3378311 19.45653
## 181 21.692606 21.139748 22.245465 21.692606 12.2065575 31.17866
## 305 28.927133 28.190240 29.664027 28.927133 19.4285816 38.42568
## 264 13.828990 12.833619 14.824361 13.828990  4.3068983 23.35108
## 359 28.140772 27.450168 28.831375 28.140772 18.6456990 37.63584
## 146 30.185312 29.367283 31.003341 30.185312 20.6801215 39.69050
## 11  13.042629 11.987495 14.097762 13.042629  3.5141041 22.57115
## 170 24.051691 23.511609 24.591773 24.051691 14.5663784 33.53700
## 329 29.241678 28.485243 29.998113 29.241678 19.7415903 38.74177
## 7    5.179012  3.482232  6.875793  5.179012 -4.4417223 14.79975
## 340 26.568048 25.956290 27.179807 26.568048 17.0783846 36.05771
## 78  27.826227 27.152987 28.499467 27.826227 18.3324014 37.32005
## 91   8.639003  7.231489 10.046518  8.639003 -0.9349492 18.21296
## 149 27.983499 27.301666 28.665333 27.983499 18.4890605 37.47794
## 364 22.478968 21.939363 23.018573 22.478968 12.9936823 31.96425
## 135 22.478968 21.939363 23.018573 22.478968 12.9936823 31.96425
## 13  16.188075 15.362195 17.013955 16.188075  6.6822059 25.69394
## 205 28.769861 28.042530 29.497192 28.769861 19.2720464 38.26768
## 293 16.188075 15.362195 17.013955 16.188075  6.6822059 25.69394
## 25  25.624414 25.048654 26.200175 25.624414 16.1370033 35.11183
## 240 29.241678 28.485243 29.998113 29.241678 19.7415903 38.74177
## 55  28.927133 28.190240 29.664027 28.927133 19.4285816 38.42568
## 105 13.514446 12.495327 14.533564 13.514446  3.9898420 23.03905
## 395 26.568048 25.956290 27.179807 26.568048 17.0783846 36.05771
## 347 29.241678 28.485243 29.998113 29.241678 19.7415903 38.74177
## 326 32.229852 31.266617 33.193087 32.229852 22.7110658 41.74864
## 88  16.974437 16.200595 17.748278 16.974437  7.4729472 26.47593
```

```
## 200 24.051691 23.511609 24.591773 24.051691 14.5663784 33.53700
## 365 23.265330 22.730055 23.800605 23.265330 13.7802893 32.75037
## 74  19.333522 18.694747 19.972296 19.333522  9.8420778 28.82497
## 102 24.838053 24.284264 25.391842 24.838053 15.3519497 34.32416
## 81  26.253504 25.654816 26.852191 26.253504 16.7646736 35.74233
## 306 25.624414 25.048654 26.200175 25.624414 16.1370033 35.11183
## 261 22.478968 21.939363 23.018573 22.478968 12.9936823 31.96425
## 260 26.410776 25.805680 27.015872 26.410776 16.9215394 35.90001
## 177 25.624414 25.048654 26.200175 25.624414 16.1370033 35.11183
## 139 16.188075 15.362195 17.013955 16.188075  6.6822059 25.69394
## 101 25.938959 25.352293 26.525625 25.938959 16.4508798 35.42704
## 184 27.039865 26.406677 27.673054 27.039865 17.5487959 36.53093
## 369 25.938959 25.352293 26.525625 25.938959 16.4508798 35.42704
## 388 25.309870 24.743836 25.875903 25.309870 15.8230440 34.79670
## 148 27.983499 27.301666 28.665333 27.983499 18.4890605 37.47794
## 84  27.197138 26.556352 27.837924 27.197138 17.7055584 36.68872
## 30  25.938959 25.352293 26.525625 25.938959 16.4508798 35.42704
## 292 20.119883 19.516491 20.723275 20.119883 10.6307550 29.60901
## 268 24.838053 24.284264 25.391842 24.838053 15.3519497 34.32416
## 258 25.624414 25.048654 26.200175 25.624414 16.1370033 35.11183
## 97  12.256267 11.140169 13.372365 12.256267  2.7207992 21.79173
## 312 28.769861 28.042530 29.497192 28.769861 19.2720464 38.26768
## 8    5.965374  4.334951  7.595797  5.965374 -3.6438794 15.57463
## 202 22.478968 21.939363 23.018573 22.478968 12.9936823 31.96425
## 157 13.042629 11.987495 14.097762 13.042629  3.5141041 22.57115
## 147 27.983499 27.301666 28.665333 27.983499 18.4890605 37.47794
## 357 27.983499 27.301666 28.665333 27.983499 18.4890605 37.47794
## 118 32.072580 31.120967 33.024193 32.072580 22.5549625 41.59020
## 185 25.309870 24.743836 25.875903 25.309870 15.8230440 34.79670
## 308 21.692606 21.139748 22.245465 21.692606 12.2065575 31.17866
## 140 17.760798 17.035949 18.485648 17.760798  8.2631734 27.25842
## 18  26.410776 25.805680 27.015872 26.410776 16.9215394 35.90001
## 128 24.051691 23.511609 24.591773 24.051691 14.5663784 33.53700
## 231 13.042629 11.987495 14.097762 13.042629  3.5141041 22.57115
## 95   5.965374  4.334951  7.595797  5.965374 -3.6438794 15.57463
## 251 17.760798 17.035949 18.485648 17.760798  8.2631734 27.25842
## 24  22.007151 21.460633 22.553669 22.007151 12.5214695 31.49283
## 114 22.950785 22.414872 23.486697 22.950785 13.4657086 32.43586
## 245 32.229852 31.266617 33.193087 32.229852 22.7110658 41.74864
## 212 20.906245 20.331827 21.480662 20.906245 11.4189150 30.39357
## 56  30.342584 29.513904 31.171264 30.342584 20.8364713 39.84870
## 381 27.983499 27.301666 28.665333 27.983499 18.4890605 37.47794
## 158 16.974437 16.200595 17.748278 16.974437  7.4729472 26.47593
## 62  26.253504 25.654816 26.852191 26.253504 16.7646736 35.74233
## 318 27.511682 26.855067 28.168298 27.511682 18.0190212 37.00434
## 314 25.624414 25.048654 26.200175 25.624414 16.1370033 35.11183
## 229 24.366236 23.821711 24.910761 24.366236 14.8806690 33.85180
## 384 29.241678 28.485243 29.998113 29.241678 19.7415903 38.74177
## 77  22.164423 21.620561 22.708286 22.164423 12.6788945 31.64995
## 204 28.612589 27.894674 29.330503 28.612589 19.1154905 38.10969
```

```
## 131 27.197138 26.556352 27.837924 27.197138 17.7055584 36.68872
## 36  24.051691 23.511609 24.591773 24.051691 14.5663784 33.53700
## 363 20.906245 20.331827 21.480662 20.906245 11.4189150 30.39357
## 93  14.929896 14.015726 15.844067 14.929896  5.4159499 24.44384
## 254 24.838053 24.284264 25.391842 24.838053 15.3519497 34.32416
## 120 25.467142 24.896397 26.037887 25.467142 15.9800340 34.95425
## 223 22.478968 21.939363 23.018573 22.478968 12.9936823 31.96425
## 390 24.680781 24.130419 25.231142 24.680781 15.1948768 34.16668
## 187 26.725321 26.106654 27.343988 26.725321 17.2352090 36.21543
## 214 16.974437 16.200595 17.748278 16.974437  7.4729472 26.47593
## 279 28.612589 27.894674 29.330503 28.612589 19.1154905 38.10969
## 169 26.725321 26.106654 27.343988 26.725321 17.2352090 36.21543
## 315 25.938959 25.352293 26.525625 25.938959 16.4508798 35.42704
## 178 24.838053 24.284264 25.391842 24.838053 15.3519497 34.32416
## 226 22.478968 21.939363 23.018573 22.478968 12.9936823 31.96425
## 152 29.241678 28.485243 29.998113 29.241678 19.7415903 38.74177
## 174 24.523508 23.976237 25.070779 24.523508 15.0377833 34.00923
## 163 22.478968 21.939363 23.018573 22.478968 12.9936823 31.96425
## 32  24.838053 24.284264 25.391842 24.838053 15.3519497 34.32416
## 156 28.455316 27.746665 29.163967 28.455316 18.9589140 37.95172
## 380 25.938959 25.352293 26.525625 25.938959 16.4508798 35.42704
## 70  14.615352 13.678312 15.552392 14.615352  5.0991806 24.13152
## 136 23.265330 22.730055 23.800605 23.265330 13.7802893 32.75037
## 166 22.478968 21.939363 23.018573 22.478968 12.9936823 31.96425
## 392 25.624414 25.048654 26.200175 25.624414 16.1370033 35.11183
## 194 27.039865 26.406677 27.673054 27.039865 17.5487959 36.53093
## 41  15.716258 14.857922 16.574594 15.716258  6.2075141 25.22500
## 336 25.938959 25.352293 26.525625 25.938959 16.4508798 35.42704
## 111 24.995325 24.437777 25.552873 24.995325 15.5090018 34.48165
## 290 15.401713 14.521288 16.282139 15.401713  5.8909501 24.91248
## 356 27.983499 27.301666 28.665333 27.983499 18.4890605 37.47794
## 235 25.938959 25.352293 26.525625 25.938959 16.4508798 35.42704
## 379 28.769861 28.042530 29.497192 28.769861 19.2720464 38.26768
## 68   7.066280  5.528201  8.604359  7.066280 -2.5277365 16.66030
## 57  28.769861 28.042530 29.497192 28.769861 19.2720464 38.26768
## 192 24.051691 23.511609 24.591773 24.051691 14.5663784 33.53700
## 116 16.974437 16.200595 17.748278 16.974437  7.4729472 26.47593
## 44  13.042629 11.987495 14.097762 13.042629  3.5141041 22.57115
## 256 25.938959 25.352293 26.525625 25.938959 16.4508798 35.42704
## 137 17.760798 17.035949 18.485648 17.760798  8.2631734 27.25842
## 259 23.265330 22.730055 23.800605 23.265330 13.7802893 32.75037
## 153 24.838053 24.284264 25.391842 24.838053 15.3519497 34.32416
## 324 23.265330 22.730055 23.800605 23.265330 13.7802893 32.75037
## 382 28.769861 28.042530 29.497192 28.769861 19.2720464 38.26768
## 265 17.918071 17.202598 18.633543 17.918071  8.4211568 27.41498
## 46  22.478968 21.939363 23.018573 22.478968 12.9936823 31.96425
## 106 13.042629 11.987495 14.097762 13.042629  3.5141041 22.57115
## 201 27.511682 26.855067 28.168298 27.511682 18.0190212 37.00434
## 269 24.523508 23.976237 25.070779 24.523508 15.0377833 34.00923
## 9    4.392651  2.629224  6.156077  4.392651 -5.2400614 14.02536
```

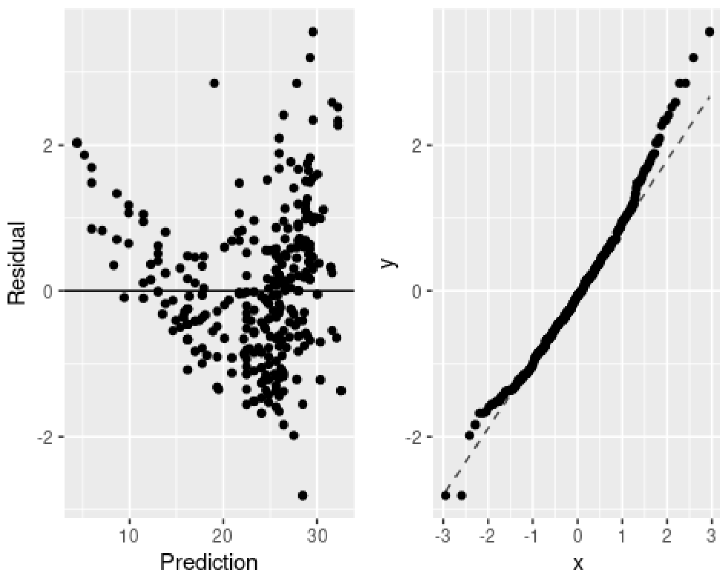
```
## 110 28.455316 27.746665 29.163967 28.455316 18.9589140 37.95172
## 199 31.443490 30.537686 32.349295 31.443490 21.9303441 40.95664
## 133 27.983499 27.301666 28.665333 27.983499 18.4890605 37.47794
## 277 21.692606 21.139748 22.245465 21.692606 12.2065575 31.17866
## 319 25.624414 25.048654 26.200175 25.624414 16.1370033 35.11183
## 233 16.345347 15.530092 17.160603 16.345347  6.8403953 25.85030
## 320 27.983499 27.301666 28.665333 27.983499 18.4890605 37.47794
## 121 22.164423 21.620561 22.708286 22.164423 12.6788945 31.64995
## 1   19.333522 18.694747 19.972296 19.333522  9.8420778 28.82497
## 334 19.018977 18.364481 19.673472 19.018977  9.5264622 28.51149
## 389 22.164423 21.620561 22.708286 22.164423 12.6788945 31.64995
## 54  29.556223 28.779725 30.332720 29.556223 20.0545164 39.05793
## 37  25.938959 25.352293 26.525625 25.938959 16.4508798 35.42704
## 376 29.084406 28.337809 29.831002 29.084406 19.5850962 38.58371
## 193 23.265330 22.730055 23.800605 23.265330 13.7802893 32.75037
## 343 26.568048 25.956290 27.179807 26.568048 17.0783846 36.05771
## 216 16.188075 15.362195 17.013955 16.188075  6.6822059 25.69394
## 348 29.556223 28.779725 30.332720 29.556223 20.0545164 39.05793
## 386 22.478968 21.939363 23.018573 22.478968 12.9936823 31.96425
## 130 29.241678 28.485243 29.998113 29.241678 19.7415903 38.74177
## 12  14.615352 13.678312 15.552392 14.615352  5.0991806 24.13152
## 302 28.769861 28.042530 29.497192 28.769861 19.2720464 38.26768
## 371 26.410776 25.805680 27.015872 26.410776 16.9215394 35.90001
## 301 25.624414 25.048654 26.200175 25.624414 16.1370033 35.11183
## 150 24.523508 23.976237 25.070779 24.523508 15.0377833 34.00923
## 218 27.197138 26.556352 27.837924 27.197138 17.7055584 36.68872
## 274 24.523508 23.976237 25.070779 24.523508 15.0377833 34.00923
## 234 27.511682 26.855067 28.168298 27.511682 18.0190212 37.00434
## 52  28.769861 28.042530 29.497192 28.769861 19.2720464 38.26768
## 333 30.028040 29.220558 30.835521 30.028040 20.5237511 39.53233
## 162 23.265330 22.730055 23.800605 23.265330 13.7802893 32.75037
## 272 23.265330 22.730055 23.800605 23.265330 13.7802893 32.75037
## 253 23.265330 22.730055 23.800605 23.265330 13.7802893 32.75037
## 366 25.938959 25.352293 26.525625 25.938959 16.4508798 35.42704
## 349 30.028040 29.220558 30.835521 30.028040 20.5237511 39.53233
## 374 25.309870 24.743836 25.875903 25.309870 15.8230440 34.79670
## 141 16.188075 15.362195 17.013955 16.188075  6.6822059 25.69394
## 294 28.612589 27.894674 29.330503 28.612589 19.1154905 38.10969
## 327 32.229852 31.266617 33.193087 32.229852 22.7110658 41.74864
## 276 20.119883 19.516491 20.723275 20.119883 10.6307550 29.60901
## 361 27.826227 27.152987 28.499467 27.826227 18.3324014 37.32005
## 208 23.737147 23.200079 24.274215 23.737147 14.2520049 33.22229
## 61  25.624414 25.048654 26.200175 25.624414 16.1370033 35.11183
## 58  24.838053 24.284264 25.391842 24.838053 15.3519497 34.32416
## 31  25.624414 25.048654 26.200175 25.624414 16.1370033 35.11183
## 377 29.084406 28.337809 29.831002 29.084406 19.5850962 38.58371
## 29   9.425365  8.082406 10.768325  9.425365 -0.1393101 18.99004
## 75  17.760798 17.035949 18.485648 17.760798  8.2631734 27.25842
## 222 16.974437 16.200595 17.748278 16.974437  7.4729472 26.47593
## 383 29.241678 28.485243 29.998113 29.241678 19.7415903 38.74177
```

```
## 335 24.051691 23.511609 24.591773 24.051691 14.5663784 33.53700
## 394 31.600763 30.683613 32.517912 31.600763 22.0865295 41.11500
## 387 26.410776 25.805680 27.015872 26.410776 16.9215394 35.90001
## 278 18.861705 18.199044 19.524365 18.861705  9.3686234 28.35479
## 227 23.265330 22.730055 23.800605 23.265330 13.7802893 32.75037
## 15  24.838053 24.284264 25.391842 24.838053 15.3519497 34.32416
## 196 31.600763 30.683613 32.517912 31.600763 22.0865295 41.11500
## 310 27.826227 27.152987 28.499467 27.826227 18.3324014 37.32005
## 207 28.455316 27.746665 29.163967 28.455316 18.9589140 37.95172
## 346 30.342584 29.513904 31.171264 30.342584 20.8364713 39.84870
## 53  27.826227 27.152987 28.499467 27.826227 18.3324014 37.32005
## 66  15.716258 14.857922 16.574594 15.716258  6.2075141 25.22500
## 271 24.838053 24.284264 25.391842 24.838053 15.3519497 34.32416
## 64  12.256267 11.140169 13.372365 12.256267  2.7207992 21.79173
## 92  16.188075 15.362195 17.013955 16.188075  6.6822059 25.69394
## 230 11.469905 10.291828 12.647982 11.469905  1.9269846 21.01283
## 165 22.478968 21.939363 23.018573 22.478968 12.9936823 31.96425
## 168 27.983499 27.301666 28.665333 27.983499 18.4890605 37.47794
## 27   8.324459  6.890976  9.757941  8.324459 -1.2533459 17.90226
## 197 30.342584 29.513904 31.171264 30.342584 20.8364713 39.84870
## 299 20.119883 19.516491 20.723275 20.119883 10.6307550 29.60901
## 266 17.760798 17.035949 18.485648 17.760798  8.2631734 27.25842
## 72  24.523508 23.976237 25.070779 24.523508 15.0377833 34.00923
## 103 32.544397 31.557732 33.531062 32.544397 23.0232109 42.06558
## 322 27.983499 27.301666 28.665333 27.983499 18.4890605 37.47794
## 267 29.084406 28.337809 29.831002 29.084406 19.5850962 38.58371
## 26   5.965374  4.334951  7.595797  5.965374 -3.6438794 15.57463
## 350 29.084406 28.337809 29.831002 29.084406 19.5850962 38.58371
## 180 24.366236 23.821711 24.910761 24.366236 14.8806690 33.85180
## 134 24.051691 23.511609 24.591773 24.051691 14.5663784 33.53700
## 67  16.188075 15.362195 17.013955 16.188075  6.6822059 25.69394
## 143 29.241678 28.485243 29.998113 29.241678 19.7415903 38.74177
## 42  16.188075 15.362195 17.013955 16.188075  6.6822059 25.69394
## 35  23.265330 22.730055 23.800605 23.265330 13.7802893 32.75037
## 273 26.410776 25.805680 27.015872 26.410776 16.9215394 35.90001
## 107 11.469905 10.291828 12.647982 11.469905  1.9269846 21.01283
## 282 26.410776 25.805680 27.015872 26.410776 16.9215394 35.90001
## 351 29.870767 29.073726 30.667808 29.870767 20.3673602 39.37417
## 161 22.478968 21.939363 23.018573 22.478968 12.9936823 31.96425
## 243 22.478968 21.939363 23.018573 22.478968 12.9936823 31.96425
## 370 25.938959 25.352293 26.525625 25.938959 16.4508798 35.42704
## 14   4.392651  2.629224  6.156077  4.392651 -5.2400614 14.02536
## 237 25.781687 25.200617 26.362757 25.781687 16.2939519 35.26942
## 316 25.624414 25.048654 26.200175 25.624414 16.1370033 35.11183
## 126 24.838053 24.284264 25.391842 24.838053 15.3519497 34.32416
## 69  15.401713 14.521288 16.282139 15.401713  5.8909501 24.91248
## 79  26.096231 25.503690 26.688773 26.096231 16.6077870 35.58468
## 122 16.188075 15.362195 17.013955 16.188075  6.6822059 25.69394
## 155 28.455316 27.746665 29.163967 28.455316 18.9589140 37.95172
## 373 25.624414 25.048654 26.200175 25.624414 16.1370033 35.11183
```

```
## 115 25.624414 25.048654 26.200175 25.624414 16.1370033 35.11183
## 182 31.443490 30.537686 32.349295 31.443490 21.9303441 40.95664
## 63  13.828990 12.833619 14.824361 13.828990  4.3068983 23.35108
## 225 19.333522 18.694747 19.972296 19.333522  9.8420778 28.82497
## 172 24.680781 24.130419 25.231142 24.680781 15.1948768 34.16668
## 221 28.769861 28.042530 29.497192 28.769861 19.2720464 38.26768
## 304 29.556223 28.779725 30.332720 29.556223 20.0545164 39.05793
## 125 11.469905 10.291828 12.647982 11.469905  1.9269846 21.01283
## 325 29.556223 28.779725 30.332720 29.556223 20.0545164 39.05793
## 339 26.568048 25.956290 27.179807 26.568048 17.0783846 36.05771
## 112 25.624414 25.048654 26.200175 25.624414 16.1370033 35.11183
## 328 29.241678 28.485243 29.998113 29.241678 19.7415903 38.74177
## 295 29.556223 28.779725 30.332720 29.556223 20.0545164 39.05793
## 228 24.051691 23.511609 24.591773 24.051691 14.5663784 33.53700
## 104 16.188075 15.362195 17.013955 16.188075  6.6822059 25.69394
## 300 28.612589 27.894674 29.330503 28.612589 19.1154905 38.10969
## 173 28.612589 27.894674 29.330503 28.612589 19.1154905 38.10969
```

Residuals Analysis Function

```r
# Define a function to simplify residuals analysis
residualAnalysis <- function(model = NULL) {
  if (!require(gridExtra)) stop('Install the gridExtra package.')
  if (!require(ggformula)) stop('Install the ggformula package.')

  df <- data.frame(Prediction = predict(model),
                   Residual = rstandard(model))

  p1 <- gf_point(Residual ~ Prediction, data = df) %>% gf_hline(yintercept =
0)
  p2 <- gf_qq(~Residual, data = df) %>% gf_qqline()

  grid.arrange(p1, p2, ncol = 2)
}

# Apply the residual analysis function
residualAnalysis(slr_train)
```



The residual plot on the left shows residuals scattered randomly around zero, indicating homoscedasticity and no clear pattern somehow, which supports model validity. The Q-Q plot on the right shows residuals aligning closely with the diagonal, suggesting that residuals are approximately normally distributed. Deviations at the tails may indicate slight departures from normality in extreme values.

TASK 2

Subset the Dataset

```
# Subset Diamonds Dataset
data("diamonds")
set.seed(1995)
diamond_subset <- diamonds %>%
  sample_n(1000) %>%
  select(price, carat)

head(diamond_subset)

## # A tibble: 6 × 2
##   price carat
##   <int> <dbl>
## 1  2697  0.7
## 2  6093  1.35
## 3  1917  0.81
## 4  1056  0.43
```

```
## 5   776  0.3
## 6   866  0.38
```

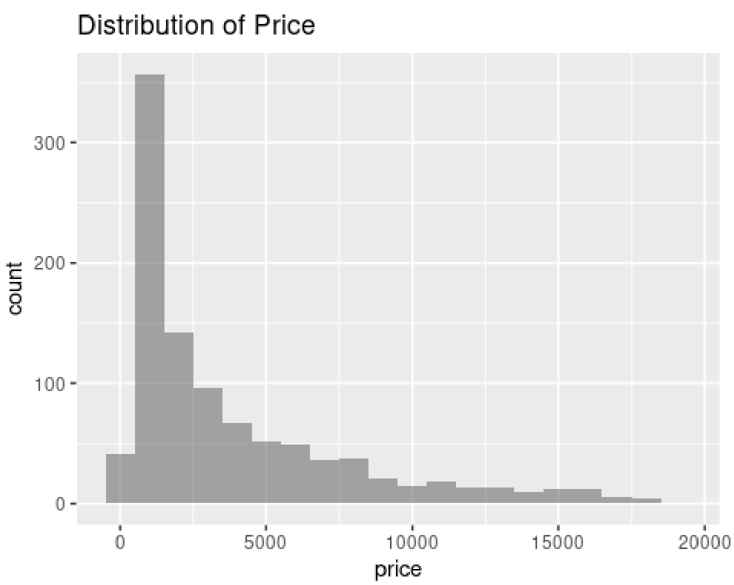
price (integer) which represents the price of a diamond in U.S. dollars. Prices in the dataset vary widely based on characteristics such as carat, cut, color, and clarity.

carat (double) which refers to the weight of the diamond in carats. Carat is a key determinant of a diamond's size and often has a strong relationship with its price.

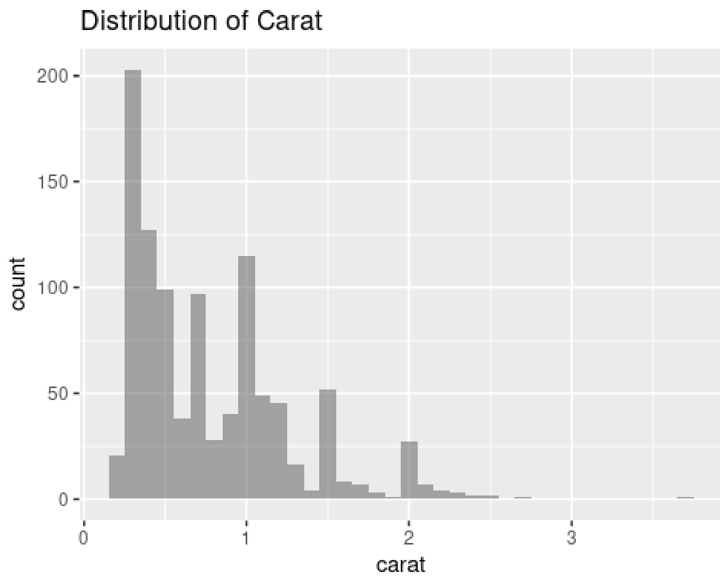
Commented [DJ6]: For task 2; 5/5 for subsetting the data I wanted to use

Exploratory Data Analysis (EDA)

```
# Histogram of Price
diamond_subset %>%
  gf_histogram(~price, binwidth = 1000) %>%
  gf_labs(title = "Distribution of Price")
```



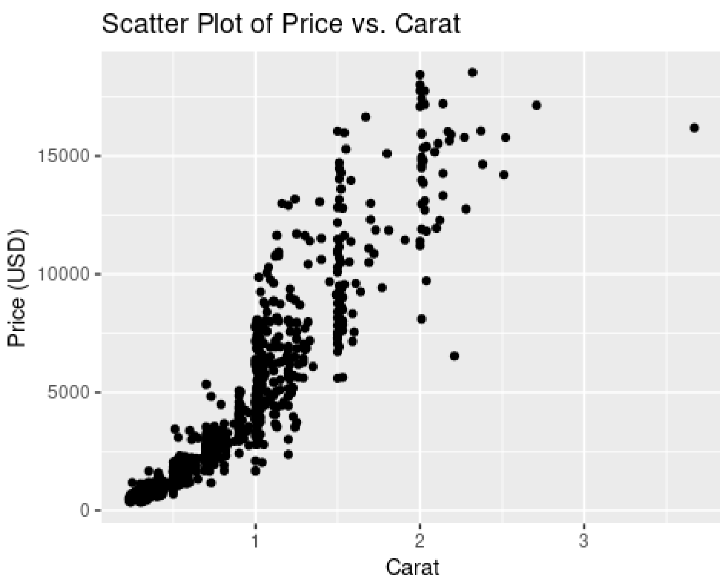

```
# Histogram of Carat
diamond_subset %>%
  gf_histogram(~carat, binwidth = 0.1) %>%
  gf_labs(title = "Distribution of Carat")
```



The histograms of price shows a right-skewed distribution, with most diamonds priced under $5,000, indicating many smaller or lower-quality diamonds. Moreover, the histogram of carat shows a similar right-skewed pattern, with the majority of diamonds weighing under 1 carat. Larger carats (>2) are less common, as shown by their lower frequency. Both distributions most likely suggest that smaller, more affordable diamonds dominate the data set, while larger, expensive diamonds are rare. This highlights the typical market trend in diamond pricing and size.

Scatter Plot for Price vs. Carat

```
# Scatter Plot: Price vs Carat
diamond_subset %>%
  gf_point(price ~ carat) %>%
  gf_labs(
    title = "Scatter Plot of Price vs. Carat",
    x = "Carat",
    y = "Price (USD)"
  )
```



Summary Statistics and Correlation

```
# Summary statistics for Price
favstats(~price, data = diamond_subset)

##  min     Q1 median   Q3   max     mean       sd    n missing
##  357 898.75 2191.5 5306 18532 3810.505 3967.046 1000       0

# Summary statistics for Carat
favstats(~carat, data = diamond_subset)

##   min   Q1 median   Q3  max    mean       sd    n missing
##  0.23 0.38    0.7 1.03 3.67 0.78496 0.488944 1000       0

# Correlation between Price and Carat
cor(diamond_subset$price, diamond_subset$carat)

## [1] 0.9302809
```

Commented [DJ7]: For the remaining parts of task 2; I have reasonably done a lot like; SLR,Model predictions for specific values, Diagnostic plots, Visualizing the regression line, Confidence and Prediction intervals,Assessing using training sets and testing sets, evaluating model on test set, residual analysis in test datam and confidence interval for coefficients. This aligns with the rubrics since I did not do the transformation and Prof said it was't mentioned hence everything looks greta for this part since there was no particular approach to this analysis.

Simple Linear Regression Model

```
# Specify and Fit the Linear Model
lm_model <- lm(price ~ carat, data = diamond_subset)

# Display Regression Coefficients
summary(lm_model)

##
## Call:
## lm(formula = price ~ carat, data = diamond_subset)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9393.3  -739.3   -27.4   481.4  6840.5
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2114.24      87.12  -24.27   <2e-16 ***
## carat        7547.83      94.22   80.11   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1456 on 998 degrees of freedom
## Multiple R-squared:  0.8654, Adjusted R-squared:  0.8653
## F-statistic:  6418 on 1 and 998 DF,  p-value: < 2.2e-16
```

The linear regression model highlights a strong and significant relationship between diamond price and carat weight. For every 1-carat increase in weight, the price of a diamond increases by an average of $7,547.83. The intercept, at -2,114.24 dollars, represents the predicted price when the carat weight is zero. While this value isn't realistic in practice, it's part of the model's mathematical structure.

The model performs well, explaining 86.54% of the variation in diamond prices, meaning carat weight is a strong predictor of price. However, the residuals (differences between actual and predicted prices) suggest that other factors, such as cut, color, or clarity, may also play an important role in determining price. Residuals range widely, with some predictions deviating significantly, showing that the relationship, while strong, is not perfect. Overall, the model is highly statistically significant and demonstrates a clear, positive trend between weight and price.

$$\widehat{price} = -2458.19 + 8028 \times carat$$

Model Predictions for Specific Values

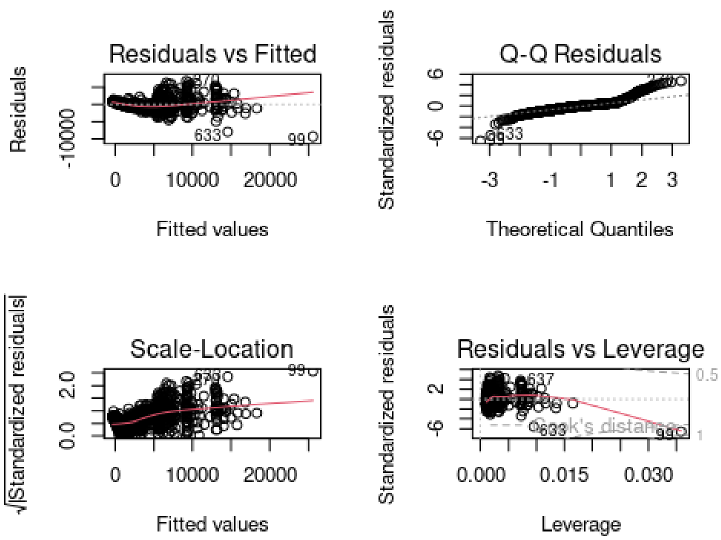
```r
# Predict Price for Carat = 1.0
predict(lm_model, newdata = data.frame(carat = 1.0))
```

```
##        1
## 5433.591
```

The predicted price is 5433.59 for 1 carat

Diagnostic Plots for Regression Model

```r
# Diagnostic Plots
par(mfrow = c(2, 2))
plot(lm_model)
```



Residuals vs. Fitted Plot

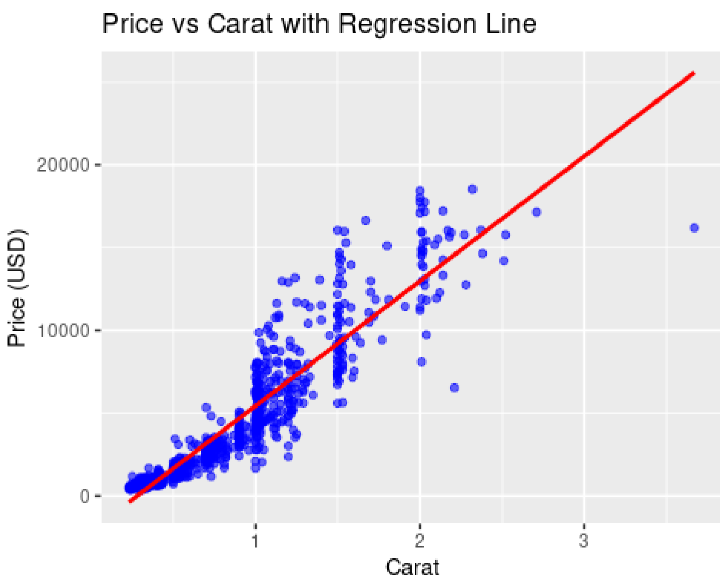
The residuals display a noticeable curved pattern instead of random scatter around the zero line. This suggests a potential non-linear relationship between the predictor and the response variable. Additionally, there is evidence of heteroscedasticity, where the variance of residuals increases as fitted values increase, violating the assumption of constant variance.

Q-Q Plot

The residuals deviate significantly from the diagonal line, especially at the extremes (tails). This indicates that the residuals are not normally distributed, which can impact the reliability of hypothesis tests and confidence intervals in the model.

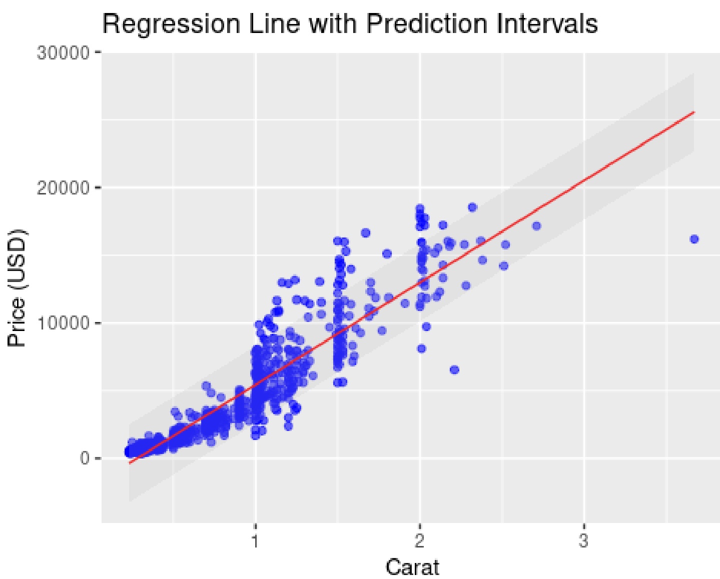
Visualize the Regression Line

```
# Scatter Plot with Regression Line
ggplot(diamond_subset, aes(x = carat, y = price)) +
  geom_point(alpha = 0.6, color = "blue") +
  geom_smooth(method = "lm", color = "red", se = FALSE) +
  labs(
    title = "Price vs Carat with Regression Line",
    x = "Carat",
    y = "Price (USD)"
  )
```



Confidence and Prediction Intervals

```
# Add Predictions with Intervals
diamond_subset <- diamond_subset %>%
  mutate(
    .fitted = predict(lm_model),
    .pred_lower = predict(lm_model, interval = "prediction")[, "lwr"],
    .pred_upper = predict(lm_model, interval = "prediction")[, "upr"]
  )

# Plot with Confidence and Prediction Intervals
ggplot(diamond_subset, aes(x = carat, y = price)) +
  geom_point(alpha = 0.6, color = "blue") +
  geom_line(aes(y = .fitted), color = "red") +
  geom_ribbon(aes(ymin = .pred_lower, ymax = .pred_upper), alpha = 0.2, fill
= "grey") +
  labs(
    title = "Regression Line with Prediction Intervals",
    x = "Carat",
    y = "Price (USD)"
  )
```



The plot above shows the relationship between price and carat. The red line above represents the predicted price based on carat, showing a clear positive trend. Moreover, the gray shaded area represents the prediction intervals, indicating the range within which most data points are expected to fall. The blue points in the plot highlights the observed data, showing variability but generally aligning with the trend we observed. This reinforces the inverse relationship between price and carat.

Assessing

Split the Data into Training and Testing Sets

```
# Split into Train and Test Sets
set.seed(1995)
diamond_split <- initial_split(diamond_subset, prop = 0.8)
diamond_train <- training(diamond_split)
diamond_test <- testing(diamond_split)

head(diamond_train)

## # A tibble: 6 × 5
##   price carat .fitted .pred_lower .pred_upper
##   <int> <dbl>   <dbl>       <dbl>       <dbl>
## 1  6503  1.24   7245.        4385.      10105.
## 2   863  0.34    452.       -2408.       3312.
## 3  1269  0.41    980.       -1879.       3840.
## 4 16036  2.17  14265.       11394.      17135.
```

```
## 5  1868  0.57   2188.       -671.       5047.
## 6   790  0.35    528.      -2332.       3387.
```

```r
head(diamond_test)
```

```
## # A tibble: 6 × 5
##   price carat .fitted .pred_lower .pred_upper
##   <int> <dbl>   <dbl>       <dbl>       <dbl>
## 1  4876  1.01   5509.        2650.       8368.
## 2  1911  0.58   2264.        -595.       5122.
## 3  7859  1.53   9434.        6572.      12296.
## 4  1668  0.52   1811.       -1048.       4670.
## 5  1675  0.55   2037.        -822.       4896.
## 6  2167  0.56   2113.        -746.       4972.
```

Train Linear Regression Model on Training Data

```r
# Train the Model on Training Data
train_model <- lm(price ~ carat, data = diamond_train)

# Summary of the Model
summary(train_model)
```

```
##
## Call:
## lm(formula = price ~ carat, data = diamond_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9451.8  -755.3   -19.4   480.7  6816.3
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2113.8      100.7  -20.99   <2e-16 ***
## carat         7563.7      107.5   70.35   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1512 on 798 degrees of freedom
## Multiple R-squared:  0.8611, Adjusted R-squared:  0.861
## F-statistic:  4949 on 1 and 798 DF,  p-value: < 2.2e-16
```

The regression results for assessment indicate that **carat** is a highly significant predictor of **price** since p-value < 2.2e-16. The estimated intercept is -2113.8, suggesting that for a carat weight of zero, the price would be negative, which is not meaningful and indicates the intercept has limited practical value. The slope estimate of 7563.7 implies that for every additional carat, the price increases by approximately $7563.7.

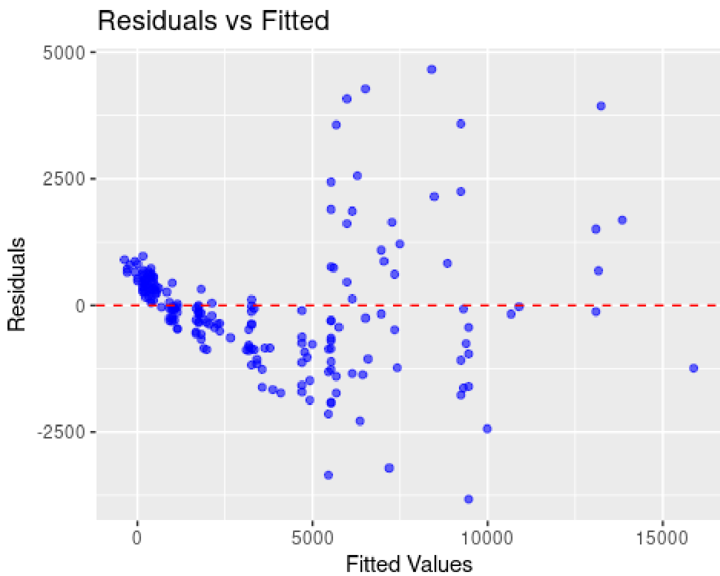
Moreover, my model explains 86.1% of the variation in price (R^2 = 0.8611), demonstrating a strong linear relationship. However, the residual standard error (1512) suggests some unexplained variability, which could relate to omitted predictors or heteroscedasticity as seen in the diagnostics.

Evaluate Model on Test Data

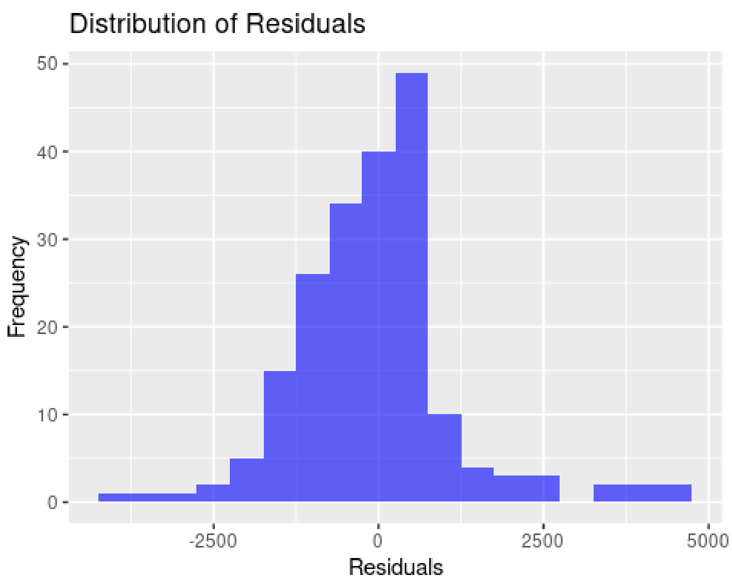
```
# Predict on Test Data
test_predictions <- predict(train_model, newdata = diamond_test)

# Combine Predictions with Actual Data
test_results <- diamond_test %>%
  mutate(
    predicted_price = test_predictions,
    residuals = price - test_predictions
  )

# View Results
head(test_results)

## # A tibble: 6 × 7
##   price carat .fitted .pred_lower .pred_upper predicted_price residuals
##   <int> <dbl>   <dbl>       <dbl>       <dbl>           <dbl>     <dbl>
## 1  4876  1.01   5509.       2650.       8368.           5525.    -649.
## 2  1911  0.58   2264.       -595.       5122.           2273.    -362.
## 3  7859  1.53   9434.       6572.      12296.           9459.   -1600.
## 4  1668  0.52   1811.      -1048.       4670.           1819.    -151.
## 5  1675  0.55   2037.       -822.       4896.           2046.    -371.
## 6  2167  0.56   2113.       -746.       4972.           2122.      45.2
```

Residual Analysis on Test Data

```
# Plot Residuals vs Fitted
ggplot(test_results, aes(x = predicted_price, y = residuals)) +
  geom_point(alpha = 0.6, color = "blue") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(
    title = "Residuals vs Fitted",
    x = "Fitted Values",
    y = "Residuals"
  )
```




```r
# Histogram of Residuals
ggplot(test_results, aes(x = residuals)) +
  geom_histogram(binwidth = 500, fill = "blue", alpha = 0.6) +
  labs(
    title = "Distribution of Residuals",
    x = "Residuals",
    y = "Frequency"
  )
```



There is skweness observed above hence normality should be considered maybe by transformations or any other methods.

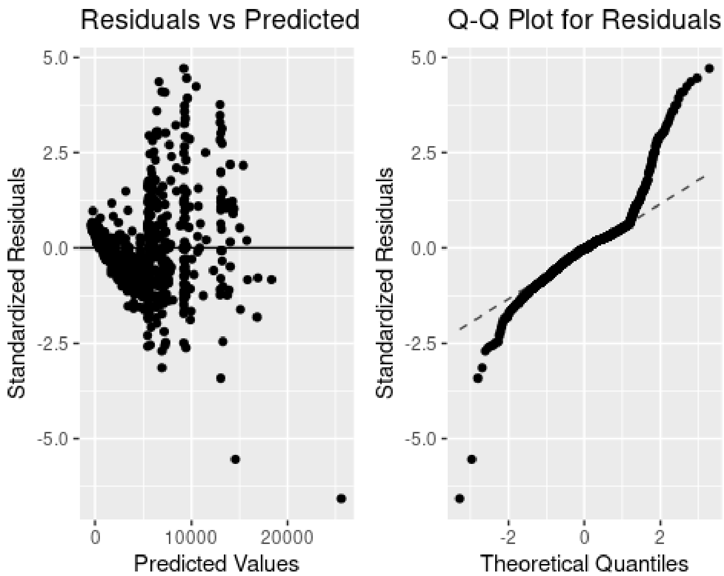
Confidence Intervals for Coefficients

```
# Confidence Intervals
confint(train_model)

##                 2.5 %    97.5 %
## (Intercept) -2311.444 -1916.156
## carat        7352.602  7774.714
```

Here, the 95% confidence intervals confirm carat's strong influence on price (7352.602 to 7774.714), while the intercept (-2311.444 to -1916.156) lacks practical meaning. The narrow interval for carat highlights the precision of the estimated price increase per additional carat.

```
# Define a function to simplify residuals analysis
residualAnalysis <- function(model = NULL) {
  if (!require(gridExtra)) stop('Install the gridExtra package.')
  if (!require(ggformula)) stop('Install the ggformula package.')

  # Create a dataframe with predictions and standardized residuals
  df <- data.frame(
    Prediction = predict(model),
    Residual = rstandard(model)
  )
```

```r
  # Plot residuals vs predictions
  p1 <- gf_point(Residual ~ Prediction, data = df) %>%
    gf_hline(yintercept = 0) %>%
    gf_labs(
      title = "Residuals vs Predicted Values",
      x = "Predicted Values",
      y = "Standardized Residuals"
    )

  # Q-Q Plot for residuals
  p2 <- gf_qq(~Residual, data = df) %>%
    gf_qqline() %>%
    gf_labs(
      title = "Q-Q Plot for Residuals",
      x = "Theoretical Quantiles",
      y = "Standardized Residuals"
    )

  # Combine the plots in a grid
  grid.arrange(p1, p2, ncol = 2)
}

# Apply the residual analysis function on the trained model
residualAnalysis(lm_model)
```



The values are not around 0 line for the residuals vs predicted values plot showing there is discrepancies in the data used or the scale. This is confirmed by the QQ plot which shows many points were not in the line.