

Final

Megan Steele

2025-12-08

Data Pre-processing

Load in the dataset.

```
car = read.csv("MPG.csv")
```

A first look at the data:

```
head(car)
```

```
##   mpg cylinders displacement horsepower weight acceleration model.year
## 1 13.0         8          360        175    3821       11.0        73
## 2 15.0         8          390        190    3850        8.5        70
## 3 17.0         8          304        150    3672       11.5        72
## 4 19.4         6          232         90    3210       17.2        78
## 5 24.3         4          151         90    3003       20.1        80
## 6 18.1         6          258        120    3410       15.1        78
##
##           car.name
## 1 amc ambassador brougham
## 2      amc ambassador dpl
## 3      amc ambassador sst
## 4            amc concord
## 5            amc concord
## 6      amc concord d/l
```

Here's a quick summary of the variables:

- cylinders: The number of cylinders in the vehicle's engine. Vehicles with more cylinders typically have larger, more powerful engines, which tend to consume more fuel.
- displacement: The total volume of all cylinders in the engine. Larger displacement generally corresponds to a more powerful engine and is often associated with lower fuel efficiency.
- horsepower: A measure of engine power. Vehicles with higher horsepower usually consume more fuel, as they are designed more for performance than for efficiency.
- weight: The total weight of the vehicle. Heavier cars require more energy to move, which typically reduces mpg.
- acceleration: The time it takes for the vehicle to reach a certain speed. Faster acceleration often indicates a performance-oriented vehicle, which can impact fuel efficiency.
- model year: The year the vehicle model was produced. This variable can capture technological improvements over time, as newer cars often benefit from advances in fuel efficiency.
- car name: The vehicle's name, usually including make and model. As unique identifiers, these names do not have a direct numerical impact on fuel efficiency and will thus not be used in predictive modeling. However, they can be useful for reference or exploration, for example, to identify specific cars in the dataset.

The response variable in this dataset is `mpg` (miles per gallon), which indicates how many miles a vehicle can travel on one gallon of fuel. Our goal in this project is to use the other variables to predict `mpg`. Therefore,

our primary focus is on accurate prediction rather than on interpreting the effects of individual predictors.

First, we check for duplicated data and missing values.

```
# --- Check for duplicates ---
# View rows with duplicated data
car[duplicated(car), ]

# --- Check for missing values ---
# View rows with any missing values
car[!complete.cases(car), ]
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model.year	car.name
## [1]	23.0	4	151	NA	3035	20.5	82	amc concord dl
## [2]	NA	8	360	175	3850	11.0	70	amc rebel sst (sw)
## [3]	NA	8	350	165	4142	11.5	70	chevrolet chevelle concours (sw)
## [4]	NA	4	133	115	3090	17.5	70	citroen ds-21 pallas
## [5]	21.0	6	200	NA	2875	17.0	74	ford maverick
## [6]	NA	8	302	140	3353	8.0	70	ford mustang boss 302
## [7]	23.6	4	140	NA	2905	14.3	80	ford mustang cobra
## [8]	NA	8	351	153	4034	11.0	70	ford pinto
## [9]	NA	8	383	175	4166	10.5	70	ford torino (sw)
## [10]	34.5	4	100	NA	2320	15.8	81	plymouth satellite (sw)
## [11]	40.9	4	85	NA	1835	17.3	80	renault 18i
## [12]	NA	4	121	110	2800	15.4	81	renault lecar deluxe
## [13]	NA	4	97	48	1978	20.0	71	saab 900s
## [14]	NA	4	117					volkswagen super beetle

We observe that there are no duplicate row entries in the data, but there are 14 rows with missing data.

To assess whether missing values are missing completely at random (MCAR), we use Little's MCAR Test with a significance level of $\alpha = 0.05$. This statistical test evaluates whether the missingness in a dataset is entirely random. If the data are truly MCAR, the probability of a value being missing does not depend on any observed or unobserved variables.

The hypotheses can be written as:

$$H_0 : \text{The missing data are MCAR}$$
$$H_A : \text{The missing data are not MCAR}$$

```
naniar::mcar_test(car)

## # A tibble: 1 x 4
##   statistic    df p.value missing.patterns
##       <dbl>  <dbl>    <dbl>          <int>
## 1      18.0     14    0.208            3
```

Since the p -value is greater than $\alpha = 0.05$, we fail to reject the null hypothesis and conclude that the missing data are plausibly MCAR. In other words, at the 0.05 significance level, there is no strong evidence that the missingness depends on any observed or unobserved variables.

Handling missing values early is important. We will postpone any decision to impute or remove missing entries until after completing our initial exploratory data analysis.

Next, we examine the variable data types:

```
str(car)
```

```
## 'data.frame': 406 obs. of 8 variables:
## $ mpg : num 13 15 17 19.4 24.3 ...
## $ cylinders : int 8 8 8 6 4 6 4 6 6 ...
## $ displacement: num 360 390 304 232 151 ...
## $ horsepower : int 175 190 150 90 90 120 NA ...
## $ weight : int 3821 3850 3672 3210 3003 ...
## $ acceleration: num 11 8.5 11.5 17.2 20.1 ...
## $ model.year : int 73 70 72 78 80 78 82 79 ...
## $ car.name : chr "amc ambassador brougham" "amc ambassador dpl" "amc ambassador sst" "amc concorde" ...
```

The variables `mpg`, `displacement`, `horsepower`, `weight`, and `acceleration` should all be treated as continuous numerical variables. Currently, RStudio is interpreting `horsepower` and `weight` as integers, which will need to be corrected.

The variables `cylinders` and `model.year` are multi-valued discrete variables. For `cylinders`, we must decide whether to treat it as a quantitative integer or a categorical variable. Treating it as numeric implies a linear effect on `mpg`, which may not be appropriate. Treating `cylinders` as categorical variable allows each cylinder count to represent its own group without assuming linearity. Similarly, `model.year` could be treated as a numeric variable, which implies a continuous trend in `mpg` over time. Or we can treat `model.year` as a categorical variable to capture year-specific effects. Again, these decisions will be held to the end after we are given an opportunity to perform our initial exploratory data analysis.

We must also address the variable `car.name`.

```
length(unique(car$car.name))
```

```
## [1] 312
```

We will not treat `car.name` as a categorical variable because doing could potentially create 312 separate indicator variables. Since each level would consume a degree of freedom, this would drastically reduce the model's power to detect effects from other predictors and likely lead to severe overfitting. Leaving the variable as a character avoids creating hundreds of meaningless factor levels and keeps the model focused on variables that actually help explain `mpg`.

Exploratory Data Analysis

We now proceed with graphical diagnostics for the response variable (`mpg`) and for the predictor variables to assess their distribution and identify any potential extreme values that could influence the regression analysis.

```
p1 = ggplot(car, aes(x = mpg)) +
  geom_histogram(binwidth = 2 * IQR(car$mpg, na.rm = TRUE) / nrow(car)^{(1/3)}, # Freedman-Diaconis Rule
```

```

            fill = "steelblue", color = "black") +
ggtitle("Miles Per Gallon") +
theme_bw() +
theme(plot.title = element_text(hjust = 0.5))

p2 = ggplot(car, aes(x = cylinders)) +
geom_bar(fill = "steelblue", color = "black") + # categorical variable
ggtitle("Number of Cylinders") +
theme_bw() +
theme(plot.title = element_text(hjust = 0.5))

p3 = ggplot(car, aes(x = displacement)) +
geom_histogram(binwidth = 2 * IQR(car$displacement, na.rm = TRUE) / nrow(car)^(1/3), # Freedman-Diaconis Rule
               fill = "steelblue", color = "black") +
ggtitle("Displacement") +
theme_bw() +
theme(plot.title = element_text(hjust = 0.5))

p4 = ggplot(car, aes(x = horsepower)) +
geom_histogram(binwidth = 2 * IQR(car$horsepower, na.rm = TRUE) / nrow(car)^(1/3), # Freedman-Diaconis Rule
               fill = "steelblue", color = "black") +
ggtitle("Horsepower") +
theme_bw() +
theme(plot.title = element_text(hjust = 0.5))

p5 = ggplot(car, aes(x = weight)) +
geom_histogram(binwidth = 2 * IQR(car$weight, na.rm = TRUE) / nrow(car)^(1/3), # Freedman-Diaconis Rule
               fill = "steelblue", color = "black") +
ggtitle("Weight") +
theme_bw() +
theme(plot.title = element_text(hjust = 0.5))

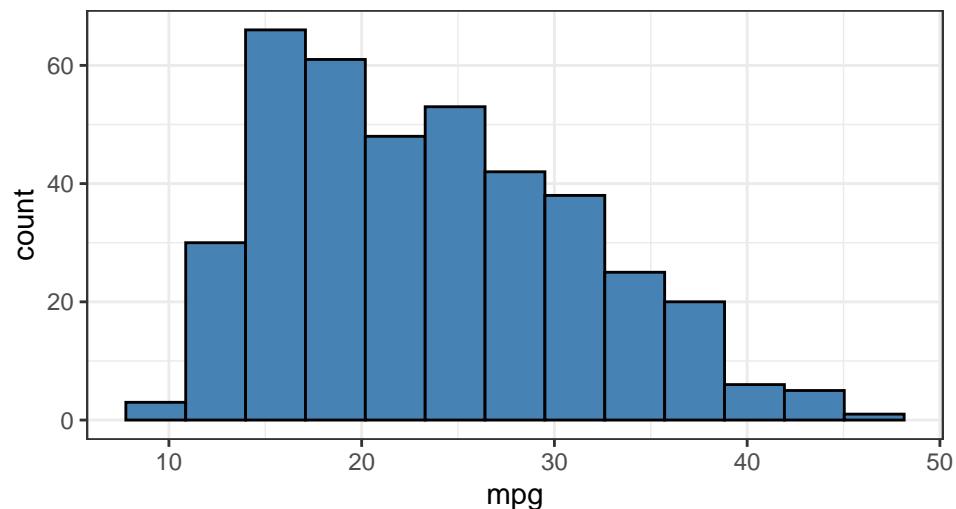
p6 = ggplot(car, aes(x = acceleration)) +
geom_histogram(binwidth = 2 * IQR(car$acceleration, na.rm = TRUE) / nrow(car)^(1/3), # Freedman-Diaconis Rule
               fill = "steelblue", color = "black") +
ggtitle("Acceleration") +
theme_bw() +
theme(plot.title = element_text(hjust = 0.5))

p7 = ggplot(car, aes(x = model.year)) +
geom_bar(fill = "steelblue", color = "black") +
ggtitle("Model Year Group") +
theme_bw() +
theme(plot.title = element_text(hjust = 0.5))

p1

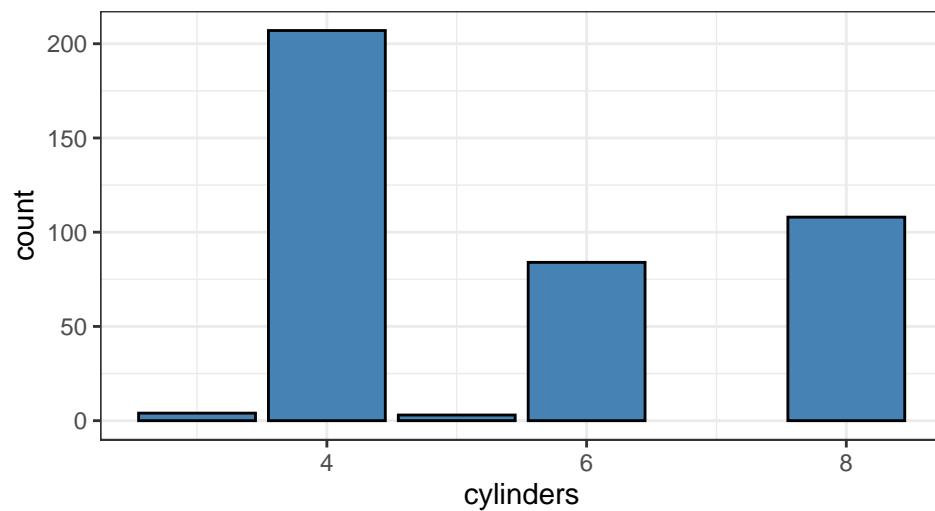
```

Miles Per Gallon



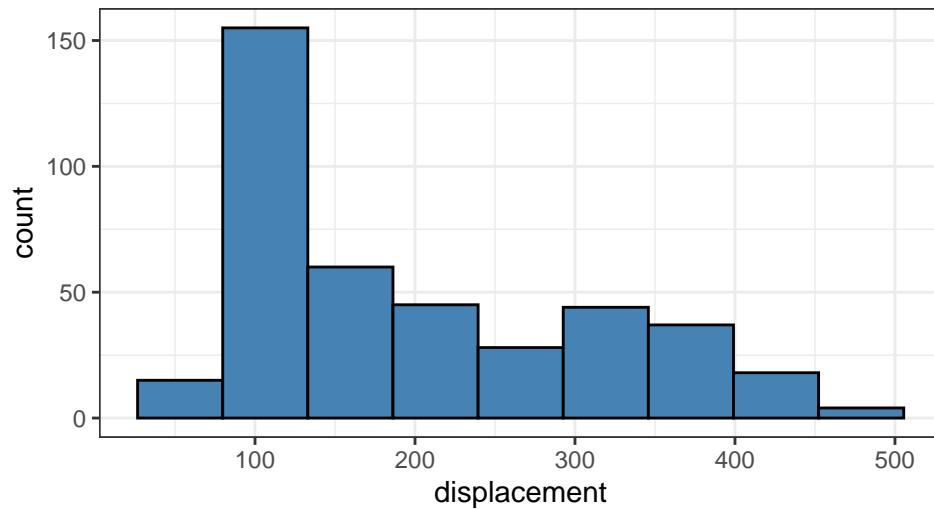
p2

Number of Cylinders



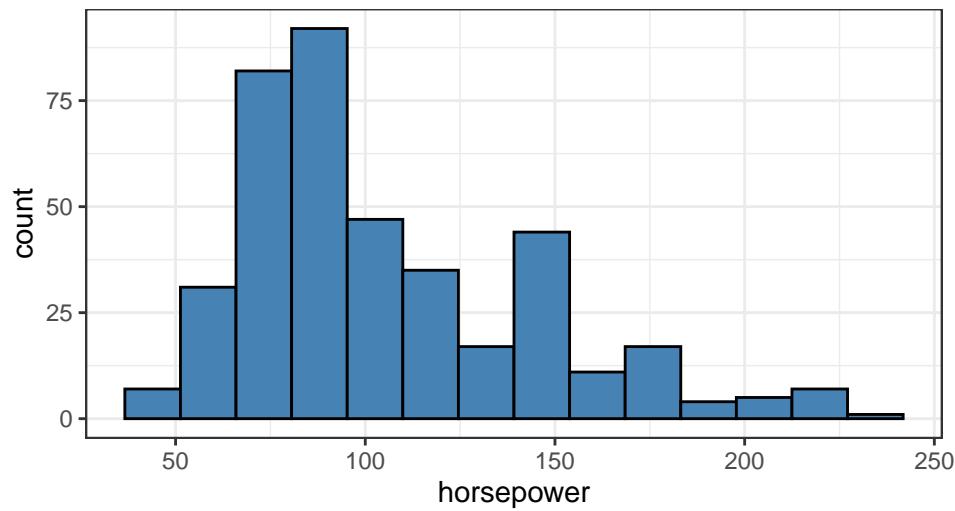
p3

Displacement



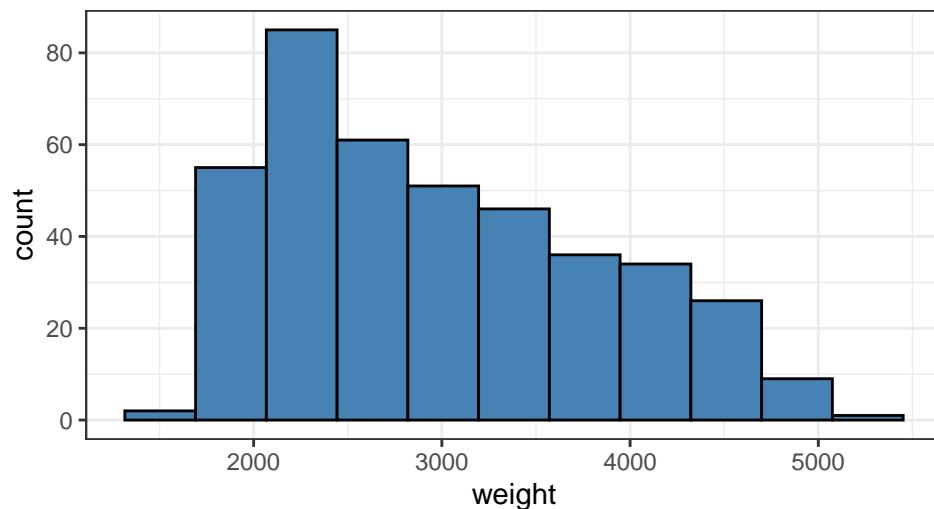
p4

Horsepower



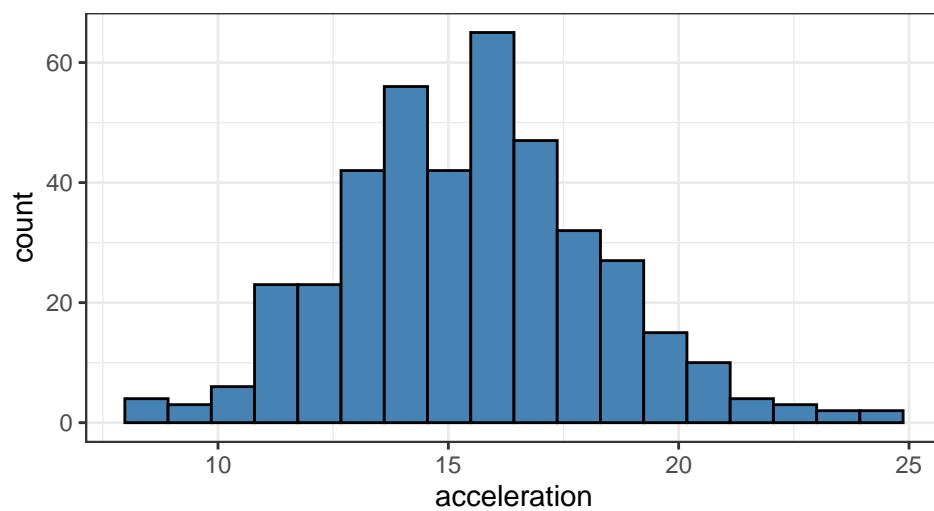
p5

Weight



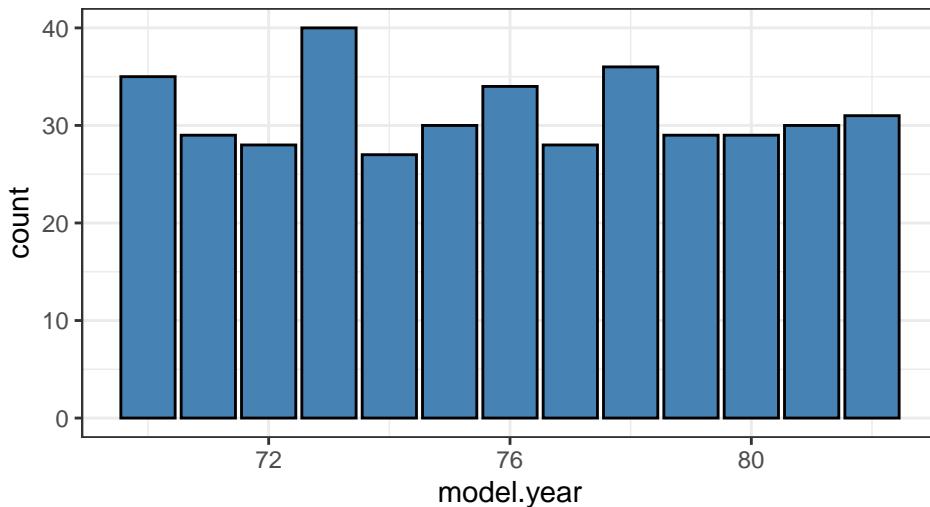
p6

Acceleration



p7

Model Year Group



```
plots = list(p1, p2, p3, p4, p5, p6, p7)

for (i in seq_along(plots)) {
  ggsave(filename = sprintf("figures/car_plot_%02d.png", i),
         plot = plots[[i]],
         width = 5,
         height = 3,
         units = "in",
         dpi = 300)
}
```

The plots above have been saved to the `figures` folder.

The response variable (`mpg`) is right-skewed. Similarly, the predictor variables `displacement`, `horsepower`, and `weight` are all right-skewed. This indicates that most vehicles are relatively light but there are a few much heavier vehicles. Since lighter vehicles tend to have smaller engines, this means that these vehicles tend to have smaller displacements and consequently lower horsepower. This could account for the skew we observe. Additionally, there are missing values for `mpg` and `horsepower` that will need to be addressed.

For the most part, the distribution of `acceleration` appears to be symmetrical about a mean value of approximately 16. The skewed distributions of the predictor variables indicate that we should carefully check the standard assumptions of linear regression (linearity, constant variance of the errors, independence of observations, and approximate normality of residuals).

The bar plot for `cylinders` shows that cars with an odd number of cylinders are rare. This aligns with engineering practices, as most engines are designed with an even number of cylinders for operational balance. Given the small number of observations for odd-cylinder cars, we will consider potentially removing these entries from the dataset.

```
b1 = ggplot(car, aes(y = mpg)) +
  geom_boxplot(fill = "steelblue", color = "black", na.rm = TRUE) +
  ggtitle("Miles Per Gallon") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

b2 = ggplot(car, aes(y = displacement)) +
  geom_boxplot(fill = "steelblue", color = "black", na.rm = TRUE) +
  ggtitle("Displacement") +
```

```

theme_bw() +
theme(plot.title = element_text(hjust = 0.5))

b3 = ggplot(car, aes(y = horsepower)) +
geom_boxplot(fill = "steelblue", color = "black", na.rm = TRUE) +
ggtitle("Horsepower") +
theme_bw() +
theme(plot.title = element_text(hjust = 0.5))

b4 = ggplot(car, aes(y = weight)) +
geom_boxplot(fill = "steelblue", color = "black", na.rm = TRUE) +
ggtitle("Weight") +
theme_bw() +
theme(plot.title = element_text(hjust = 0.5))

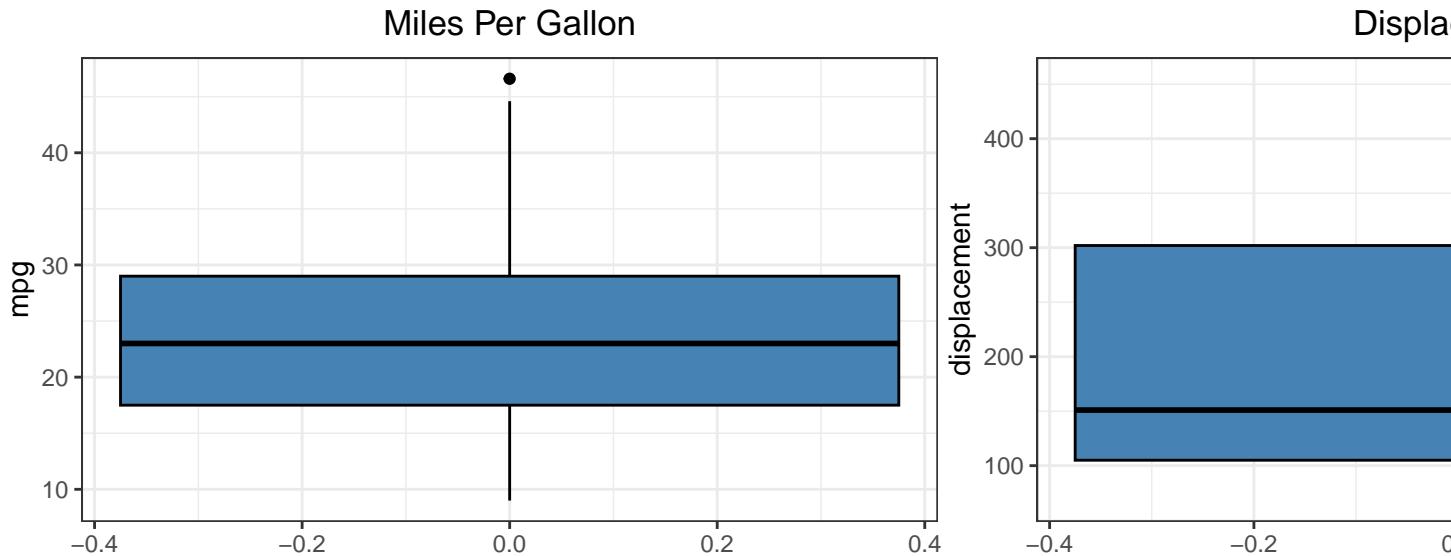
b5 = ggplot(car, aes(y = acceleration)) +
geom_boxplot(fill = "steelblue", color = "black", na.rm = TRUE) +
ggtitle("Acceleration") +
theme_bw() +
theme(plot.title = element_text(hjust = 0.5))

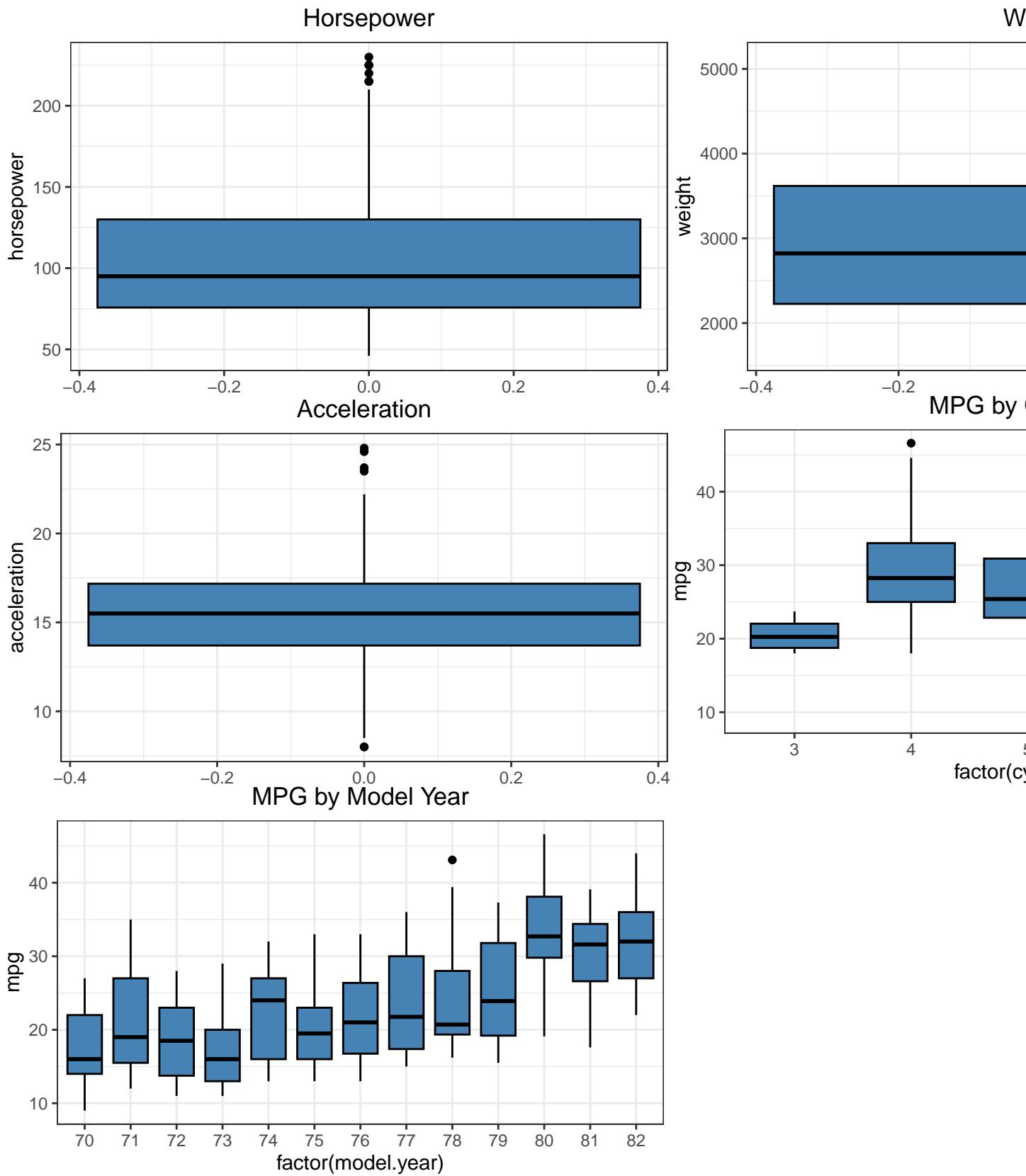
# Boxplot for categorical variables (cylinders and model.year)
b6 = ggplot(car, aes(x = factor(cylinders), y = mpg)) +
geom_boxplot(fill = "steelblue", color = "black", na.rm = TRUE) +
ggtitle("MPG by Cylinders") +
theme_bw() +
theme(plot.title = element_text(hjust = 0.5))

b7 = ggplot(car, aes(x = factor(model.year), y = mpg)) +
geom_boxplot(fill = "steelblue", color = "black", na.rm = TRUE) +
ggtitle("MPG by Model Year") +
theme_bw() +
theme(plot.title = element_text(hjust = 0.5))

b1; b2; b3; b4; b5; b6; b7

```





```
# Save plots to files
boxplots = list(b1, b2, b3, b4, b5, b6, b7)
```

```

for (i in seq_along(boxplots)) {
  ggsave(filename = sprintf("figures/car_boxplot_%02d.png", i),
         plot = boxplots[[i]],
         width = 5,
         height = 3,
         units = "in",
         dpi = 300)
}

```

Based on the box plots above, we observe potential outliers in `mpg`, `horsepower`, `acceleration`, `cylinders`, and `model.year`. These extreme values likely reflect unusual vehicle types, manufacturing years, or specific design priorities, and we will investigate them further to understand their impact on the dataset and subsequent modeling.

Below we look at these extreme values.

```

# Find rows where mpg is an outlier
mpg_outliers = car[car$mpg %in% boxplot.stats(car$mpg)$out, ]
mpg_outliers

##      mpg cylinders displacement horsepower weight acceleration model.year
## 244 46.6          4           86          65     2110        17.9         80
##   car.name
## 244 mazda glc

```

For `mpg`, the Mazda GLC from 1980 is flagged as an extreme high value. This is a small, lightweight, fuel-efficient car, which explains its exceptionally high miles per gallon. Its characteristics explain the extremely high `mpg`.

```

# Find rows where horsepower is an outlier
mpg_outliers = car[car$horsepower %in% boxplot.stats(car$horsepower)$out, ]
mpg_outliers

```

```

##      mpg cylinders displacement horsepower weight acceleration model.year
## 45    12          8           455          225    4951        11.0         73
## 46    14          8           455          225    3086        10.0         70
## 82    14          8           454          220    4354         9.0         70
## 111   13          8           440          215    4735        11.0         73
## 179   10          8           360          215    4615        14.0         70
## 299   14          8           440          215    4312         8.5         70
## 322   14          8           455          225    4425        10.0         70
## 327   16          8           400          230    4278         9.5         73
##   car.name
## 45      buick electra 225 custom
## 46      buick estate wagon (sw)
## 82      chevrolet impala
## 111    chrysler new yorker brougham
## 179      ford f250
## 299      plymouth fury iii
## 322      pontiac catalina
## 327      pontiac grand prix

```

For `horsepower`, the flagged cars are primarily muscle cars or large luxury vehicles from the 1970s. These vehicles were designed for power and performance, so their high horsepower values are reasonable within their context.

```

# Find rows where acceleration is an outlier
mpg_outliers = car[car$acceleration %in% boxplot.stats(car$acceleration)$out, ]
mpg_outliers

##          mpg cylinders displacement horsepower weight acceleration model.year
## 208      NA          8           302        140    3353         8.0       70
## 285  27.2          4           141        71    3190        24.8       79
## 292 14.0          8           340       160    3609         8.0       70
## 394 23.0          4           97        54    2254        23.5       72
## 401 43.4          4           90        48    2335        23.7       80
## 402 44.0          4           97        52    2130        24.6       82
##              car.name
## 208   ford mustang boss 302
## 285      peugeot 504
## 292   plymouth 'cuda 340
## 394   volkswagen type 3
## 401   vw dasher (diesel)
## 402      vw pickup

```

For `acceleration`, the extreme values occur at both ends of the spectrum. Cars like the Ford Mustang Boss 302 and Plymouth 'Cuda 340 have acceleration around 8 seconds, which is much faster than most cars in the dataset. These are high-performance muscle cars with powerful engines, explaining their unusually low acceleration times. On the other end, vehicles like the VW Dasher (diesel), VW Pickup, Peugeot 504, and Volkswagen Type 3 have acceleration around 23–44 seconds, which is extremely slow compared to typical passenger cars. These are lightweight, low-power, or diesel vehicles, where acceleration is not a design priority.

Overall, these extreme values are not necessarily errors, but they can have high leverage in regression models. It will be important to consider their influence when building predictive models for `mpg`.

We will compute studentized deleted residuals only after finalizing our regression model, including the selection of predictors and the functional form.

Summary statistics are below.

```
inspect(car)
```

```

##
## categorical variables:
##      name      class levels n missing
## 1 car.name character     312 406      0
##                                         distribution
## 1 ford pinto (1.5%) ...
##
## quantitative variables:
##      name      class min      Q1 median      Q3 max      mean
## 1     mpg numeric   9  17.50    23.0  29.000   46.6 23.514572864
## 2 cylinders integer   3    4.00    4.0   8.000    8.0  5.475369458
## 3 displacement numeric  68 105.00  151.0 302.000  455.0 194.779556650
## 4 horsepower integer   46  75.75   95.0 130.000  230.0 105.082500000
## 5      weight integer 1613 2226.50 2822.5 3618.250 5140.0 2979.413793103
## 6 acceleration numeric   8   13.70   15.5  17.175   24.8 15.519704433
## 7 model.year integer   70   73.00   76.0  79.000   82.0 75.921182266
##      sd n missing
## 1 7.815984313 398      8
## 2 1.712159632 406      0
## 3 104.922458379 406      0

```

```

## 4 38.768779183 400      6
## 5 847.004328239 406      0
## 6   2.803358816 406      0
## 7   3.748737345 406      0

```

The mean value for `mpg` is 23.514573. We also note that 50% of the vehicles in our sample have an `mpg` less than 23.0.

A second useful diagnostic for the response and predictor variables is a sequence plot. With 406 observations, plotting each point and connecting them with a line produces a cluttered figure, so as an alternative we also produce a sequence plot with an LOESS smooth line overlaid on top to summarize the overall trend. Both sequence plots can also be viewed below for each variable.

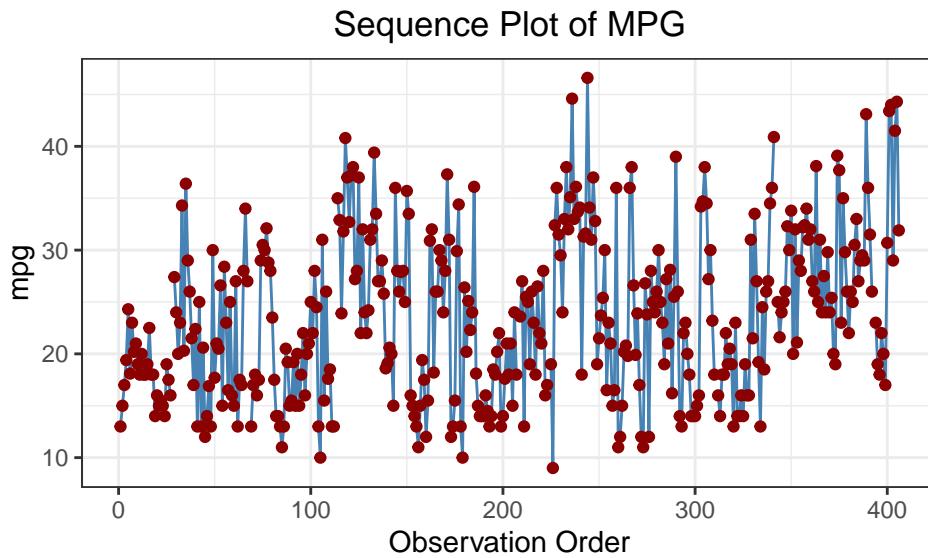
First we plot everything and then discuss afterwards. Plots are saved to the folder `figures`.

```

mpg_seq = ggplot(car, aes(x = 1:nrow(car), y = mpg)) +
  geom_line(color = "steelblue") +
  geom_point(color = "darkred", size = 1.5) +
  labs(title = "Sequence Plot of MPG",
       x = "Observation Order",
       y = "mpg") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

mpg_seq

```



```

# Save the plot as a PNG
ggsave(filename = "figures/mpg_seq.png",
       plot = mpg_seq,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)

```

```

mpg_seq2 = ggplot(car, aes(x = 1:nrow(car), y = mpg)) +
  geom_point(alpha = 0.4, size = 1, color = "darkred") +
  geom_smooth(method = "loess", color = "steelblue", se = FALSE, size = 1) +
  labs(title = "Sequence Plot of MPG",
       x = "Observation Order",
       y = "mpg")

```

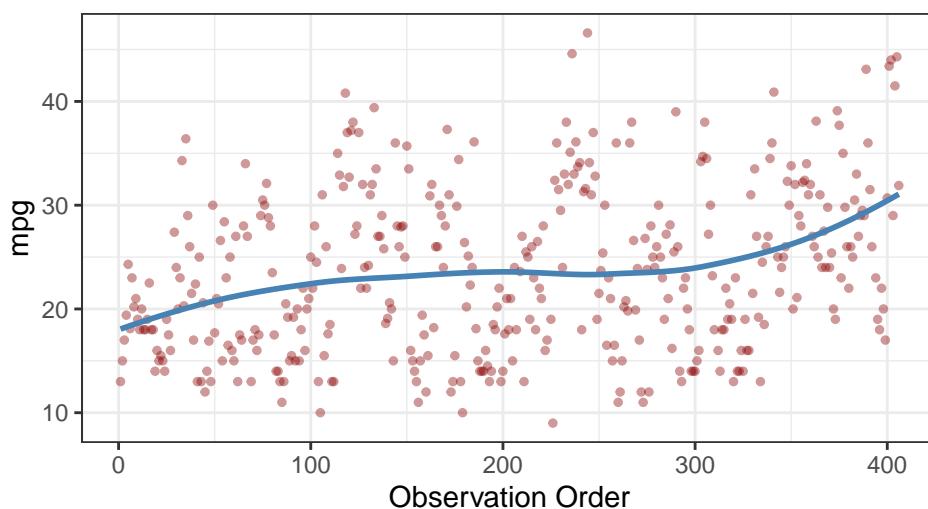
```

y = "mpg") +
theme_bw() +
theme(plot.title = element_text(hjust = 0.5))

mpg_seq2

```

Sequence Plot of MPG



```

# Save the plot as a PNG
ggsave(filename = "figures/mpg_seq2.png",
plot = mpg_seq2,
width = 5,
height = 3,
units = "in",
dpi = 300)

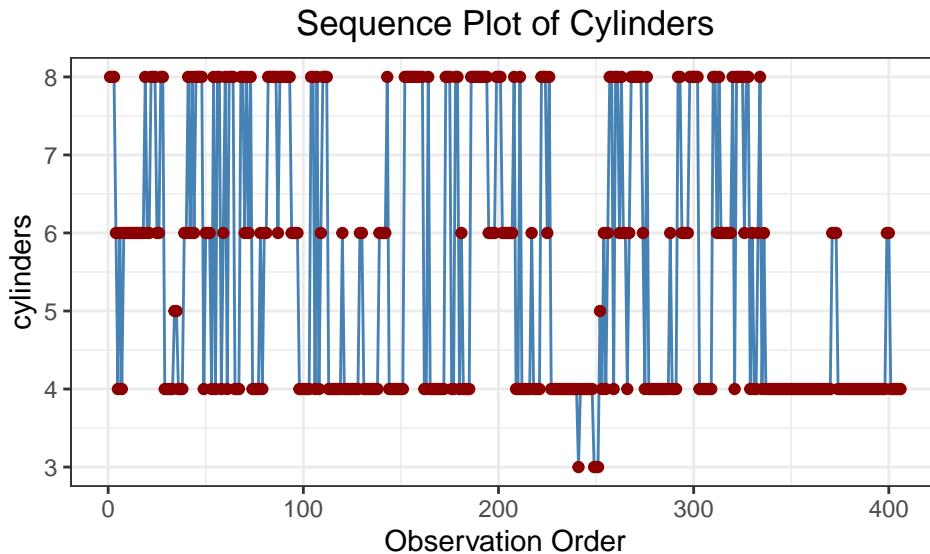
```

```

cyl_seq = ggplot(car, aes(x = 1:nrow(car), y = cylinders)) +
geom_line(color = "steelblue") +
geom_point(color = "darkred", size = 1.5) +
labs(title = "Sequence Plot of Cylinders",
x = "Observation Order",
y = "cylinders") +
theme_bw() +
theme(plot.title = element_text(hjust = 0.5))

cyl_seq

```

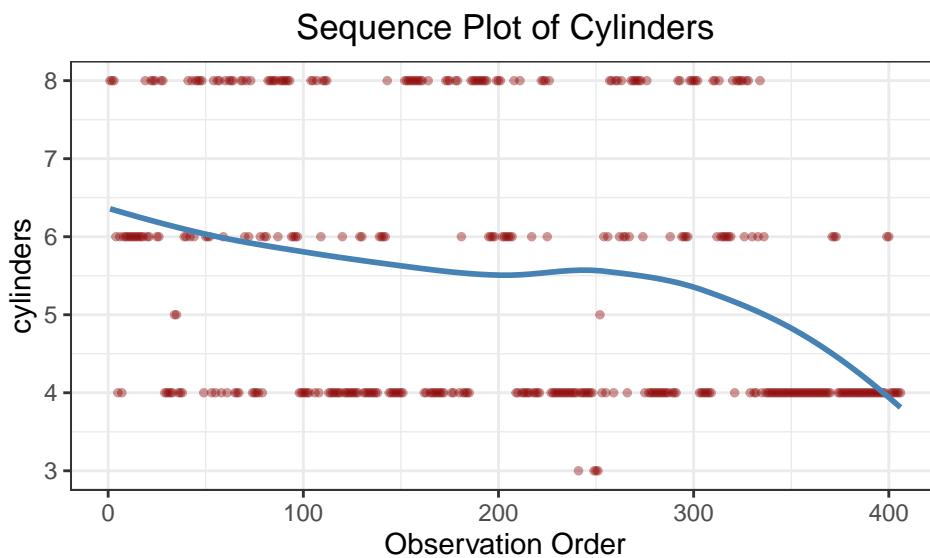


```
# Save the plot as a PNG
```

```
ggsave(filename = "figures/cyl_seq.png",
plot = cyl_seq,
width = 5,
height = 3,
units = "in",
dpi = 300)
```

```
cyl_seq2 = ggplot(car, aes(x = 1:nrow(car), y = cylinders)) +
  geom_point(alpha = 0.4, size = 1, color = "darkred") +
  geom_smooth(method = "loess", color = "steelblue", se = FALSE, size = 1) +
  labs(title = "Sequence Plot of Cylinders",
       x = "Observation Order",
       y = "cylinders") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```

```
cyl_seq2
```



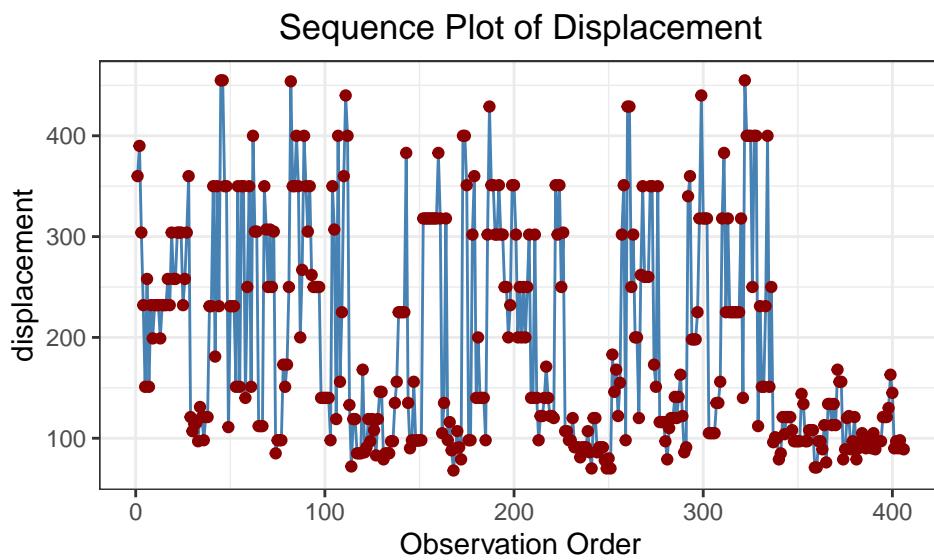
```

# Save the plot as a PNG
ggsave(filename = "figures/cyl_seq2.png",
       plot = cyl_seq2,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)

dis_seq = ggplot(car, aes(x = 1:nrow(car), y = displacement)) +
  geom_line(color = "steelblue") +
  geom_point(color = "darkred", size = 1.5) +
  labs(title = "Sequence Plot of Displacement",
       x = "Observation Order",
       y = "displacement") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

dis_seq

```



```

# Save the plot as a PNG
ggsave(filename = "figures/dis_seq.png",
       plot = dis_seq,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)

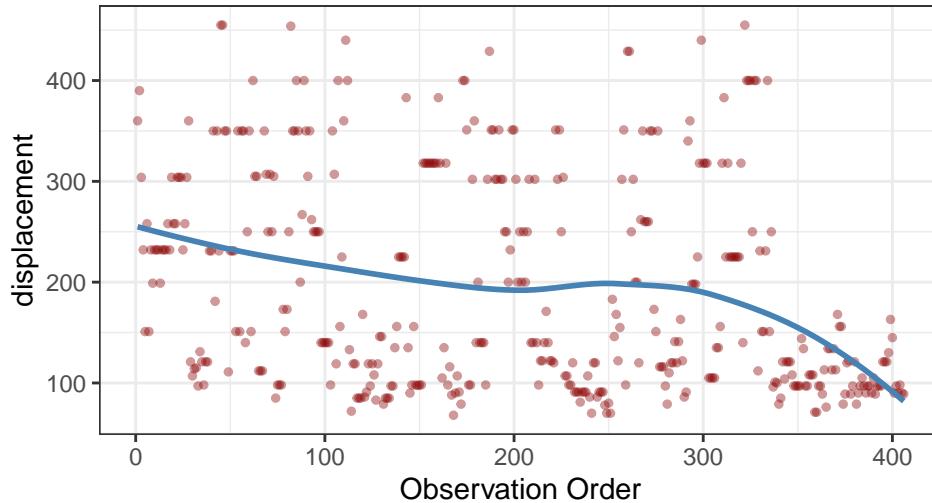
dis_seq2 = ggplot(car, aes(x = 1:nrow(car), y = displacement)) +
  geom_point(alpha = 0.4, size = 1, color = "darkred") +
  geom_smooth(method = "loess", color = "steelblue", se = FALSE, size = 1) +
  labs(title = "Sequence Plot of Displacement",
       x = "Observation Order",
       y = "displacement") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

dis_seq2

```

```
dis_seq2
```

Sequence Plot of Displacement

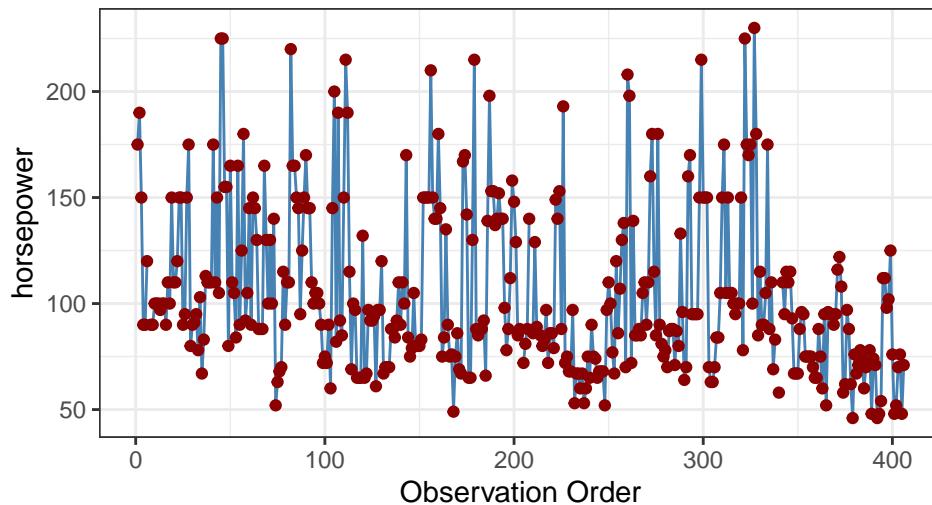


```
# Save the plot as a PNG
ggsave(filename = "figures/dis_seq2.png",
       plot = dis_seq2,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)

horse_seq = ggplot(car, aes(x = 1:nrow(car), y = horsepower)) +
  geom_line(color = "steelblue") +
  geom_point(color = "darkred", size = 1.5) +
  labs(title = "Sequence Plot of Horsepower",
       x = "Observation Order",
       y = "horsepower") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

horse_seq
```

Sequence Plot of Horsepower



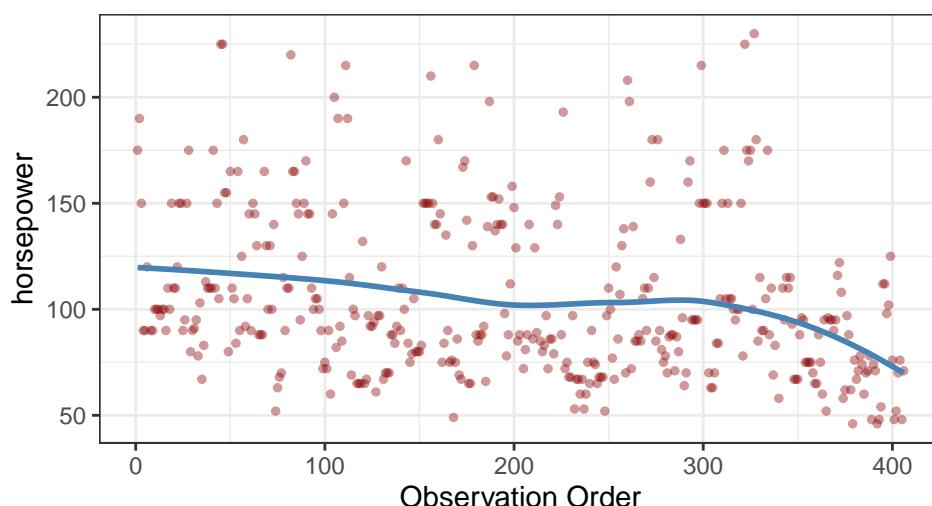
```
# Save the plot as a PNG
```

```
ggsave(filename = "figures/horse_seq.png",
       plot = horse_seq,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)
```

```
horse_seq2 = ggplot(car, aes(x = 1:nrow(car), y = horsepower)) +
  geom_point(alpha = 0.4, size = 1, color = "darkred") +
  geom_smooth(method = "loess", color = "steelblue", se = FALSE, size = 1) +
  labs(title = "Sequence Plot of Horsepower",
       x = "Observation Order",
       y = "horsepower") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))
```

```
horse_seq2
```

Sequence Plot of Horsepower



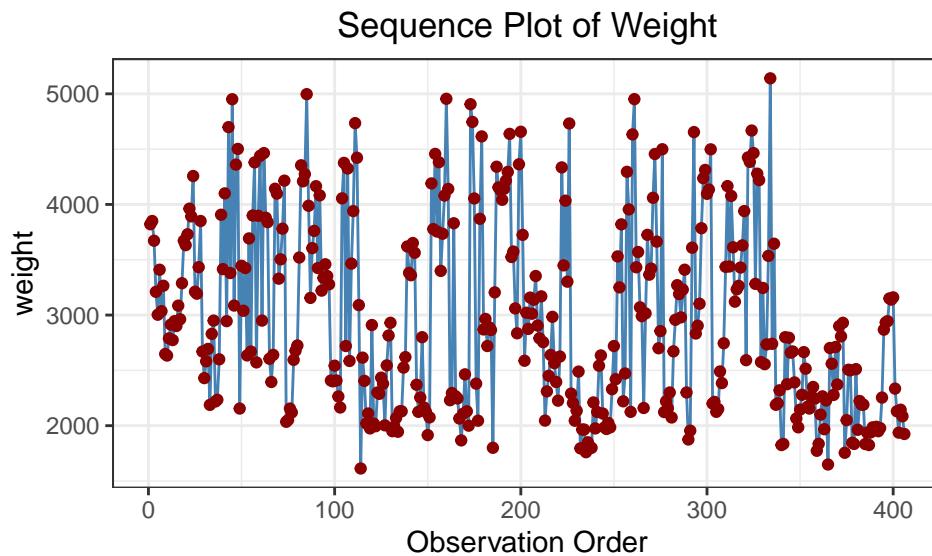
```

# Save the plot as a PNG
ggsave(filename = "figures/horse_seq2.png",
       plot = horse_seq2,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)

weight_seq = ggplot(car, aes(x = 1:nrow(car), y = weight)) +
  geom_line(color = "steelblue") +
  geom_point(color = "darkred", size = 1.5) +
  labs(title = "Sequence Plot of Weight",
       x = "Observation Order",
       y = "weight") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

weight_seq

```



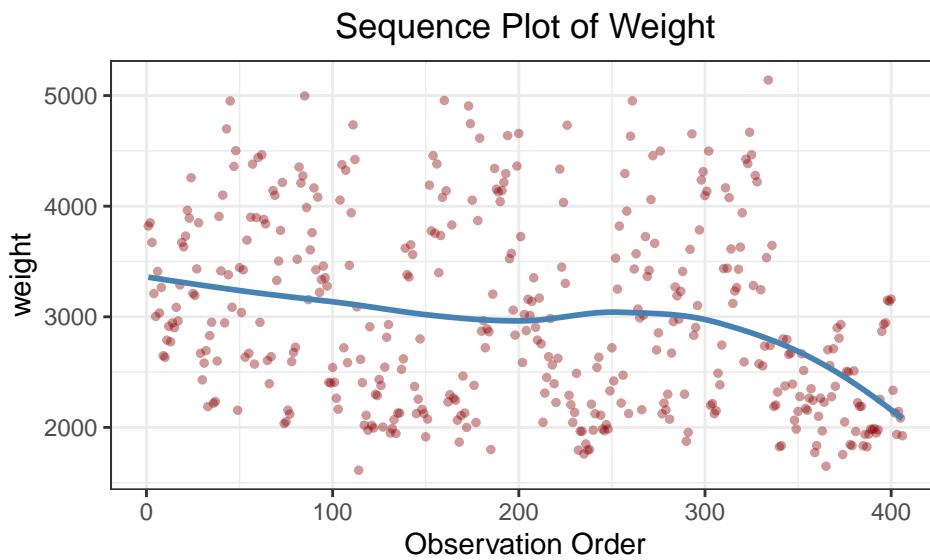
```

# Save the plot as a PNG
ggsave(filename = "figures/weight_seq.png",
       plot = weight_seq,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)

weight_seq2 = ggplot(car, aes(x = 1:nrow(car), y = weight)) +
  geom_point(alpha = 0.4, size = 1, color = "darkred") +
  geom_smooth(method = "loess", color = "steelblue", se = FALSE, size = 1) +
  labs(title = "Sequence Plot of Weight",
       x = "Observation Order",
       y = "weight") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

weight_seq2

```

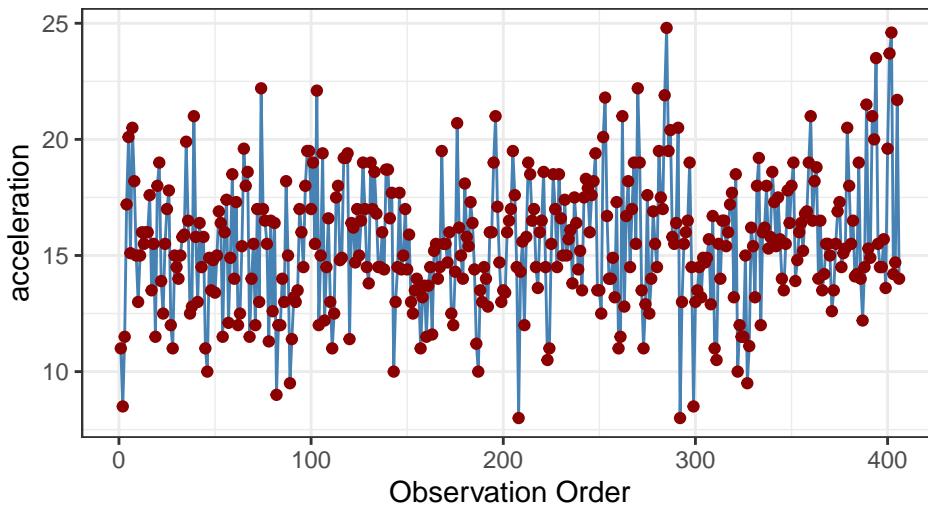


```
# Save the plot as a PNG
ggsave(filename = "figures/weight_seq2.png",
       plot = weight_seq2,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)

acc_seq = ggplot(car, aes(x = 1:nrow(car), y = acceleration)) +
  geom_line(color = "steelblue") +
  geom_point(color = "darkred", size = 1.5) +
  labs(title = "Sequence Plot of Acceleration",
       x = "Observation Order",
       y = "acceleration") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

acc_seq
```

Sequence Plot of Acceleration

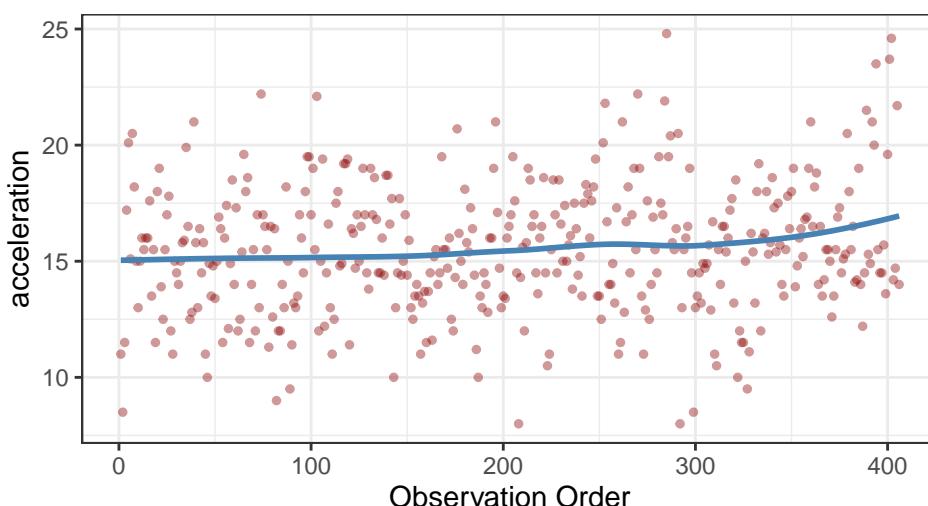


```
# Save the plot as a PNG
ggsave(filename = "figures/acc_seq.png",
       plot = acc_seq,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)

acc_seq2 = ggplot(car, aes(x = 1:nrow(car), y = acceleration)) +
  geom_point(alpha = 0.4, size = 1, color = "darkred") +
  geom_smooth(method = "loess", color = "steelblue", se = FALSE, size = 1) +
  labs(title = "Sequence Plot of Acceleration",
       x = "Observation Order",
       y = "acceleration") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

acc_seq2
```

Sequence Plot of Acceleration



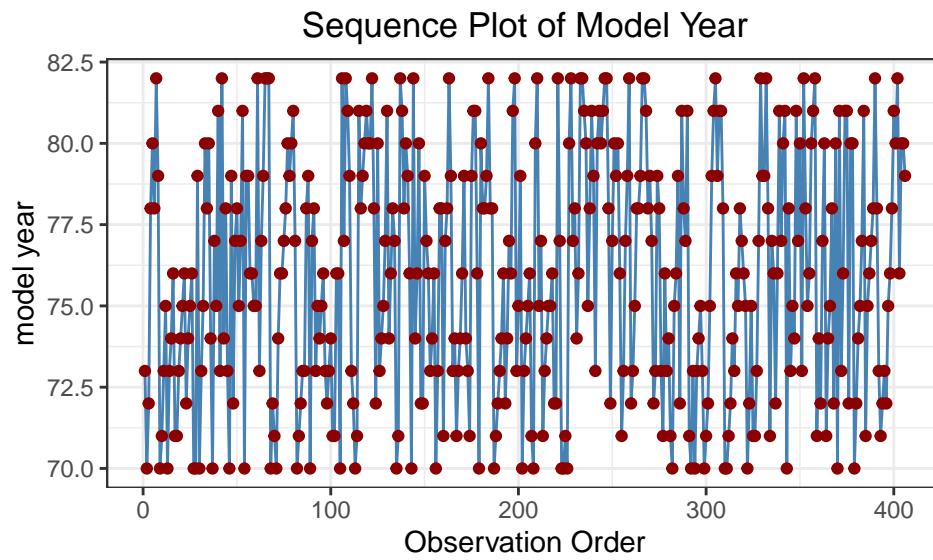
```

# Save the plot as a PNG
ggsave(filename = "figures/acc_seq2.png",
       plot = acc_seq2,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)

year_seq = ggplot(car, aes(x = 1:nrow(car), y = model.year)) +
  geom_line(color = "steelblue") +
  geom_point(color = "darkred", size = 1.5) +
  labs(title = "Sequence Plot of Model Year",
       x = "Observation Order",
       y = "model year") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

year_seq

```



```

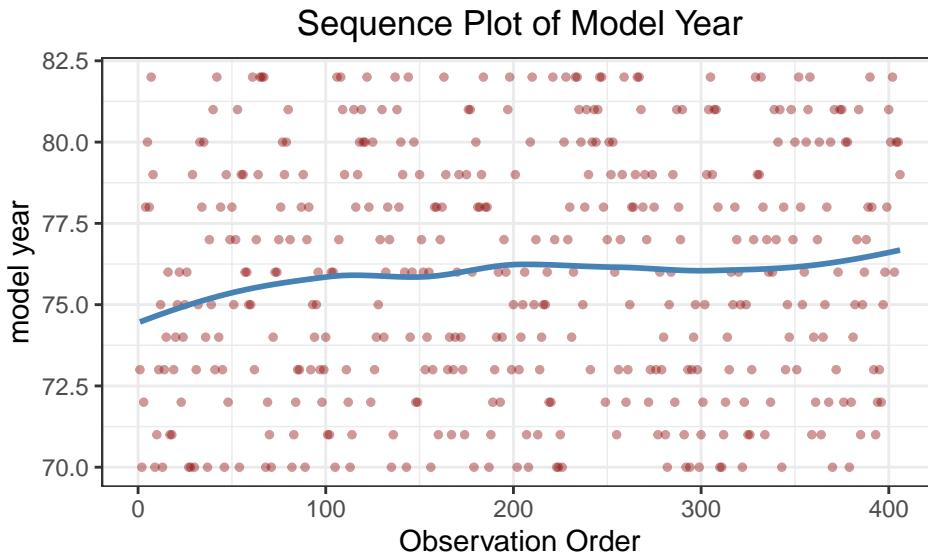
# Save the plot as a PNG
ggsave(filename = "figures/year_seq.png",
       plot = year_seq,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)

year_seq2 = ggplot(car, aes(x = 1:nrow(car), y = model.year)) +
  geom_point(alpha = 0.4, size = 1, color = "darkred") +
  geom_smooth(method = "loess", color = "steelblue", se = FALSE, size = 1) +
  labs(title = "Sequence Plot of Model Year",
       x = "Observation Order",
       y = "model year") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

year_seq2

```

year_seq2



```
# Save the plot as a PNG
ggsave(filename = "figures/year_seq2.png",
       plot = year_seq2,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)
```

The sequence plot for `mpg` indicates that the last portion of the dataset contains higher-mpg cars. The sequence plots for the predictor variables also reveal a noticeable pattern. Observations toward the tail end of the dataset tend to have smaller displacement, lower horsepower, and lower weight. This pattern does not automatically violate regression assumptions, but it signals a potential problem with independence of the residuals if left unaccounted for.

A quick inspection of the data suggests that this pattern is likely driven by smaller engines in later-model vehicles. This is consistent with expectations, as the latter part of the dataset corresponds to the late 1970s. This is, this data takes place during the oil crisis, when fuel efficiency became a greater design priority.

```
# View the last 10 rows of the dataset
tail(car, 10)
```

```
##      mpg cylinders displacement horsepower weight acceleration model.year
## 397 22.0          4           121         98   2945        14.5       75
## 398 20.0          4           130        102   3150        15.7       76
## 399 17.0          6           163        125   3140        13.6       78
## 400 30.7          6           145         76   3160        19.6       81
## 401 43.4          4           90          48   2335        23.7       80
## 402 44.0          4           97          52   2130        24.6       82
## 403 29.0          4           90          70   1937        14.2       76
## 404 41.5          4           98          76   2144        14.7       80
## 405 44.3          4           90          48   2085        21.7       80
## 406 31.9          4           89          71   1925        14.0       79
##
##      car.name
## 397      volvo 244dl
## 398      volvo 245
## 399      volvo 264gl
```

```

## 400      volvo diesel
## 401  vw dasher (diesel)
## 402          vw pickup
## 403          vw rabbit
## 404          vw rabbit
## 405 vw rabbit c (diesel)
## 406    vw rabbit custom

```

Our findings suggests that the regression model should include `displacement`, `horsepower`, `weight`, and/or `model.year` to account for the observed structure. Additionally, the relationships between the response variable and some predictors may not be strictly linear. If these patterns are not accounted for, they could introduce confounding or serial correlation in the residuals, potentially affecting model assumptions and interpretation.

Now we pivot to using scatter plots and the correlation matrix to explore possible interactions and non-linear relationships.

```

custom_smooth = function(data, mapping, ...) {
  ggplot(data = data, mapping = mapping) +
    geom_point(color = "darkred", alpha = 0.4, size = 1) +
    geom_smooth(method = "loess", color = "steelblue", se = FALSE, ...)
}

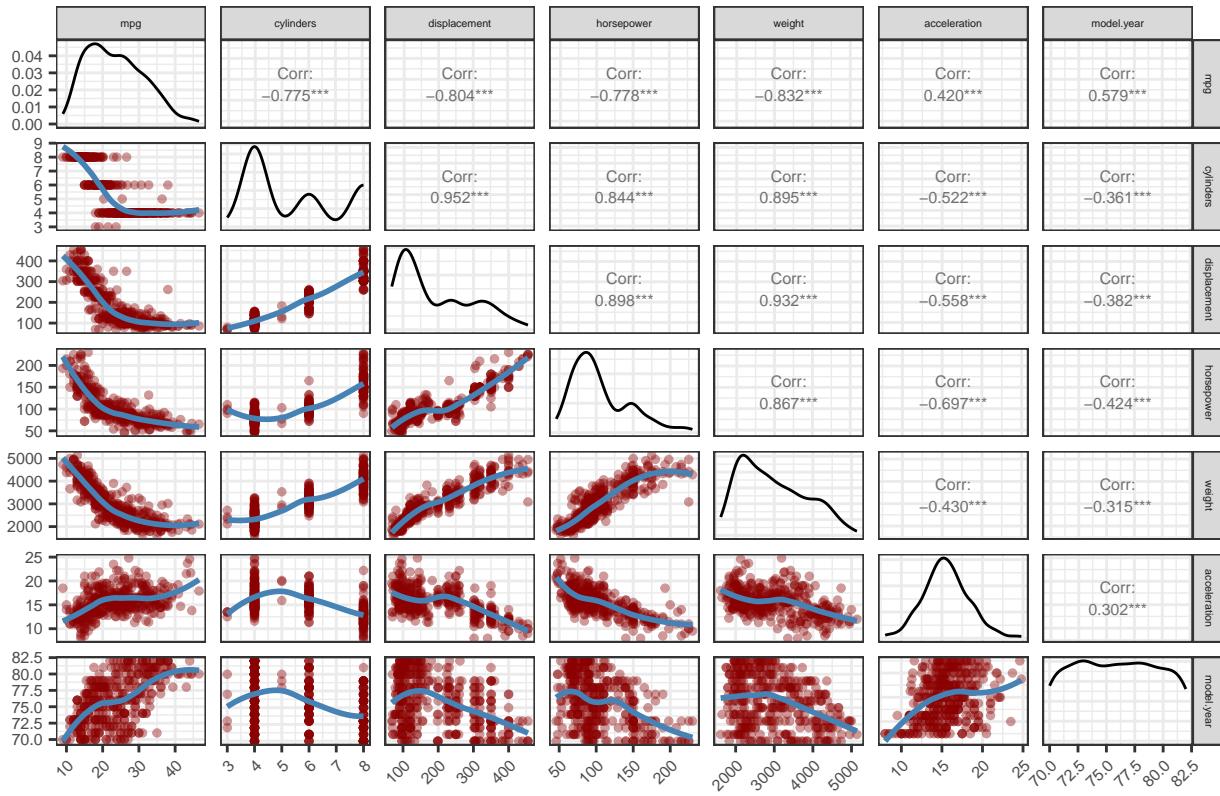
car_small = car %>%
  dplyr::select(-car.name) # remove categorical variables

scatter_mat = ggpairs(car_small,
                      upper = list(continuous = wrap("cor", size = 2)),
                      lower = list(continuous = custom_smooth),
                      diag = list(continuous = "densityDiag")) +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 6),
        axis.text.y = element_text(angle = 0, size = 6),
        strip.text = element_text(size = 4),
        plot.title = element_text(hjust = 0.5)) +
  labs(title = "Scatter Plot Matrix with Correlations")

scatter_mat

```

Scatter Plot Matrix with Correlations



```
# Save the plot as a PNG
ggsave(filename = "figures/scatter_mat.png",
       plot = scatter_mat,
       width = 6,
       height = 4,
       units = "in",
       dpi = 300)
```

We removed the categorical variable (`car.name`) for this analysis because correlation and scatter plots are meaningful only for numeric predictors.

Based on the correlation matrix, `mpg` is highly negatively linearly associated with `cylinders`, `displacement`, `horsepower`, and `weight`, which have correlations of -0.775, -0.804, -0.778, and -0.832, respectively. The relationship between `mpg` and `model.year` is more moderate, with a correlation of 0.579. The associated scatter plots also reveal that a linear fit may not fully capture the relationship with the response variable. In other words, non-linear relationships should also be considered. In particular, we should explore polynomial relationships between `mpg` and `displacement`, `horsepower`, `weight`, and possibly even `acceleration`. This is consistent with our observations from the sequence plots.

The correlations among the predictors also reveal notable patterns. In particular, `displacement` and `horsepower` are highly correlated (0.898), `displacement` and `weight` are highly correlated (0.932), and `weight` and `horsepower` are also highly correlated (0.867). These strong correlations indicate that multicollinearity may be an issue. One approach to mitigate this is to center the predictor variables before fitting the regression model.

Since multiple variables are highly correlated, we must also investigate whether variable pairs might contribute redundant information. Removing one of these variables can simplify the model without sacrificing accuracy.

Data Preprocessing

Now we will leverage the insights gathered so far to pre-process the data, address missing values, and manage potential multicollinearity. We will also consider appropriate transformations, evaluate whether any outliers should be removed, and decide which variables should be treated as categorical before building our predictive model for `mpg`. In addition, we will split the dataset into a training set and a validation set to properly assess model performance.

Given that only 14 rows out of 406 are missing values (approximately 3.5% of the dataset) and that the missingness appears random, removing these rows will have minimal impact on the analysis and is unlikely to introduce bias. Therefore, it is reasonable to delete the rows with missing values in this dataset.

We now remove rows with missing values. The dataset name is changed to preserve the original data, should we need it in the future.

```
# Remove rows with any missing values
new_car = car %>% drop_na()
```

Next, we must ensure that all variables are stored with the correct data types.

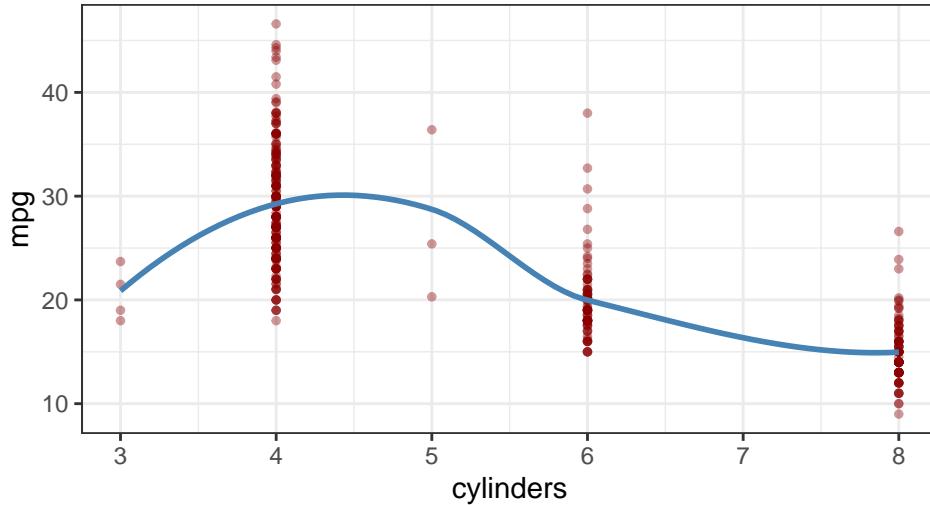
Based on our earlier findings, treating `cylinders` as a numeric variable would be inappropriate because it would impose the assumption that each additional cylinder has the same linear effect on `mpg`. The scatterplot of `mpg` versus `cylinders` suggests that this assumption does not hold. Moreover, the median values from the box plots for each cylinder group show that the `mpg` for 3-, 4-, 5-, 6-, and 8-cylinder cars does not follow a strictly linear trend. In particular, the `mpg` for 3-cylinder vehicles is much lower than that of both 4- and 5-cylinder cars, further violating the linearity assumption.

Instead, we treat `cylinders` as a categorical variable so that each cylinder group can have its own mean effect without enforcing linearity. This approach also provides greater flexibility when incorporating interaction terms, as each cylinder level is allowed to interact differently with other predictors, capturing more nuanced relationships in the data.

```
mpg_vs_cyl = ggplot(new_car, aes(x = cylinders, y = mpg)) +
  geom_point(alpha = 0.4, size = 1, color = "darkred") +
  geom_smooth(method = "loess", color = "steelblue", se = FALSE, size = 1) +
  labs(x = "cylinders", y = "mpg",
       title = "MPG by Number of Cylinders") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

mpg_vs_cyl
```

MPG by Number of Cylinders



```
# Save the plot as a PNG
ggsave(filename = "figures/mpg_vs_cyl.png",
       plot = mpg_vs_cyl,
       width = 6,
       height = 4,
       units = "in",
       dpi = 300)
```

Note that we still must address the issue of sparse levels (aka, cylinder levels 3 and 5).

```
new_car %>%
  group_by(cylinders) %>%
  summarise(n = n())

## # A tibble: 5 x 2
##   cylinders     n
##       <int> <int>
## 1         3     4
## 2         4    199
## 3         5     3
## 4         6    83
## 5         8   103
```

Cylinder levels 3 and 5 have extremely few observations (4 and 3, respectively), which is too small to produce stable coefficient estimates in a regression model. In contrast, the 4-, 6-, and 8-cylinder levels have sufficient observations. Retaining the 3- and 5-cylinder vehicles as separate levels would likely result in highly noisy and unreliable estimates.

We must still check with SK but this conversation might be mute as a whole, because there is a strong argument to drop `cylinders` as a predictor all together in the regression model. We will continue talking about this later, but essentially, `cylinders` is highly correlated with `weight` and thus they both capture the same information.

If SK does not like this there are alternatives. Keep the argument below, so that we don't have to re-create our talking points just in case we need to pivot.

Check with SK: We are considering options to address the issue of the very small number of odd-cylinder cars (3- and 5-cylinder) in our dataset. The options are:

1. Remove the odd-cylinder cars from the dataset. This is the simplest and most reliable approach. We can note in our report that these cars were removed due to insufficient sample size, which is a standard and justified practice. Removing them ensures stable coefficient estimates, reliable predictions, and meaningful train/validation splits.
2. Combine the 3- and 5-cylinder cars with another group. While this is mechanically possible, it is not meaningful from a modeling perspective, as 3- and 5-cylinder engines are not comparable to 4-cylinder cars in terms of expected mpg. Combining them could distort the relationship between cylinders and mpg.
3. Keep the 3- and 5-cylinder cars as a separate factor level. Given that there are only 7 observations, this is likely unwise. Predictions for this group would be highly unreliable, and the main goal of the project is accurate prediction. This approach risks introducing noise without meaningful benefit.

Unless SK advises otherwise, the plan below proceeds under the assumption that option 1 is approved.

Another variable we must discuss is `model.year`. In our dataset, `model.year` represents the year in which each car was manufactured. Although technically numeric, treating it as a continuous variable would imply a strictly linear effect of year on `mpg`. The scatterplot of `mpg` versus `model.year` suggests that this assumption may not hold, a finding further supported by the boxplots for each year group.

Fuel efficiency in the 1970s was influenced by tightening federal regulations, which led to discrete improvements in `mpg` rather than a smooth, linear increase. For example, the Corporate Average Fuel Economy (CAFE) standards introduced in 1975 began affecting vehicle design in the subsequent years, pushing manufacturers toward incremental gains in fuel efficiency. Additionally, this real-world context supports modeling `model.year` as a categorical predictor.

We can also look at the summary table below. Looking at average and median mpg across the years, there is a general upward trend but the progression is uneven. For example, mpg jumps from 17.1 in 1973 to 22.7 in 1974, then drops slightly in 1975 before increasing again. Similarly, the median mpg doesn't increase smoothly each year.

```
car_summary = new_car %>%
  group_by(model.year) %>%
  summarise(mean_mpg = mean(mpg, na.rm = TRUE),
            median_mpg = median(mpg, na.rm = TRUE),
            sd_mpg = sd(mpg, na.rm = TRUE),
            n = n()) %>%
  arrange(model.year)

car_summary

## # A tibble: 13 x 5
##   model.year  mean_mpg median_mpg  sd_mpg     n
##       <int>     <dbl>      <dbl>    <dbl> <int>
## 1        1      70.0      17.7     16.0    5.34    29
## 2        2      71.0      21.1     19.0    6.68    27
## 3        3      72.0      18.7     18.5    5.44    28
```

```

##  4      73    17.1      16    4.70     40
##  5      74    22.8      24.5   6.54     26
##  6      75    20.3      19.5   4.94     30
##  7      76    21.6      21     5.89     34
##  8      77    23.4      21.8   6.68     28
##  9      78    24.1      20.7   6.90     36
## 10     79    25.1      23.9   6.79     29
## 11     80    33.8      32.7   6.89     27
## 12     81    30.2      31.2   5.64     28
## 13     82     32       32     5.23     30

```

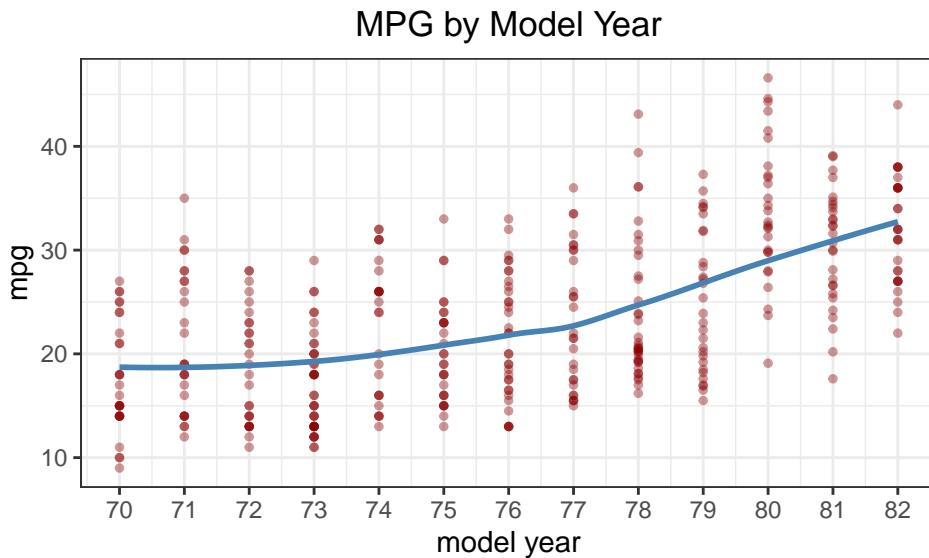
Below we provide an alternate view of the mpg against model.year plot.

```

mpg_vs_year = ggplot(new_car, aes(x = model.year, y = mpg)) +
  geom_point(alpha = 0.4, size = 1, color = "darkred") +
  geom_smooth(method = "loess", color = "steelblue", se = FALSE, size = 1) +
  scale_x_continuous(breaks = seq(min(new_car$model.year),
                                  max(new_car$model.year),
                                  by = 1)) +
  labs(x = "model year", y = "mpg", title = "MPG by Model Year") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

```

mpg_vs_year



```

# Save the plot as a PNG
ggsave(filename = "figures/mpg_vs_year.png",
       plot = mpg_vs_year,
       width = 6,
       height = 4,
       units = "in",
       dpi = 300)

```

```

# Convert variables to appropriate data types
new_car = new_car %>%
  mutate(cylinders = as.factor(cylinders),
        horsepower = as.numeric(horsepower),
        weight = as.numeric(weight),

```

```

model.year = as.factor(model.year)

# Only use this code to remove 3- and 5-cylinder cars
# Depending on the result of our talk with SK
#new_car = new_car %>%
#  #filter(!cylinders %in% c(3, 5))

# Only use this code to Re-code cylinders as a factor (4, 6, 8)
# if we end up removing odd numbered cylinders
#new_car$cylinders = factor(new_car$cylinders)

# Check the result
str(new_car)

## 'data.frame':    392 obs. of  8 variables:
##   $ mpg        : num  13 15 17 19.4 24.3 ...
##   $ cylinders  : Factor w/ 5 levels "3","4","5","6",...
##   $ displacement: num  360 390 304 232 151 ...
##   $ horsepower : num  175 190 150 90 90 ...
##   $ weight     : num  3821 3850 3672 3210 3003 ...
##   $ acceleration: num  11 8.5 11.5 17.2 20.1 ...
##   $ model.year : Factor w/ 13 levels "70","71","72",...
##   $ car.name   : chr  "amc ambassador brougham" "amc ambassador dpl" "amc ambassador sst" "amc concorde"

```

Before fitting a regression model, it is important to examine the number of observations in each level of the categorical variable. This ensures that each level has enough data to produce stable and reliable estimates.

We take a look at `cylinders`.

```

new_car %>%
  group_by(cylinders) %>%
  summarise(n = n())

## # A tibble: 5 x 2
##   cylinders     n
##   <fct>     <int>
## 1 3             4
## 2 4            199
## 3 5             3
## 4 6             83
## 5 8            103

```

Depending on whether we keep odd numbered cylinders in the dataset, we either do not have sufficient observations for each level or we do.

We now take a look at `model.year`.

```

new_car %>%
  group_by(model.year) %>%
  summarise(n = n())

## # A tibble: 13 x 2
##   model.year     n
##   <fct>     <int>
## 1 70          29
## 2 71          27
## 3 72          28

```

```

## 4 73      40
## 5 74      26
## 6 75      30
## 7 76      34
## 8 77      28
## 9 78      36
## 10 79     29
## 11 80     27
## 12 81     28
## 13 82     30

```

There are sufficient observations for each level.

To mitigate potential issues with multicollinearity, we now center the predictor variables.

```

# Center continuous predictor variables
new_car_centered = new_car %>%
  mutate(displacement_c = displacement - mean(displacement),
        horsepower_c = horsepower - mean(horsepower),
        weight_c = weight - mean(weight),
        acceleration_c = acceleration - mean(acceleration))

```

Next, we will split the data into a training set and a validation set. We aim for a 50/50 split while maintaining the same proportion of cylinder and model year levels in both sets to ensure balanced representation of these categorical variables.

With a total of 392 observations in the dataset, splitting it evenly into a training set and a validation set leaves approximately 196 observations for training. To balance model complexity with predictive accuracy, we aim to drop certain variables from consideration in the regression model. Specifically, we propose removing `cylinders`, `acceleration` and `displacement`. These features are highly correlated with other predictors, such as `horsepower` and `weight`, and therefore provide overlapping information. By eliminating these redundant variables, we enhance the interpretability of the model while reducing unnecessary complexity.

Since we are removing `cylinders`, `acceleration` and `displacement` from the model, it is also practical to exclude their squared terms as well as any interaction terms involving these variables.

The full model under consideration thus includes `horsepower` and its square, `weight` and its square, the interaction between `weight` and `horsepower`, and `model.year` as a factor with 13 levels.

To assess whether we have enough observations for reliable estimation, we can calculate the number of observations per predictor degree of freedom:

$$\frac{\text{Number of training observations}}{\text{Number of predictor degrees of freedom}} = \frac{196}{17} \approx 11.5$$

This indicates that there are roughly 10 observations available for each predictor df, satisfying the commonly cited rule of thumb in linear regression. Meeting this guideline suggests that we have sufficient data to estimate model coefficients reliably and reduces the risk of overfitting. Consequently, we can proceed with confidence in both coefficient estimation and predictive performance.

Calculating the degrees of freedom for each predictor term:

Horsepower (continuous) : 1 df
Horsepower² (continuous) : 1 df
Weight (continuous) : 1 df
Weight² (continuous) : 1 df
Weight × Horsepower (interaction) : 1 df
Model Year (13 levels) : $13 - 1 = 12$ df

Total predictor df = $1 + 1 + 1 + 1 + 1 + 12 = 17$ df
Training observations : $n = 196$
Residual df = $n - \text{predictor df} - 1 = 196 - 17 - 1 = 178$

```
set.seed(333)

# Stratified split by model.year only
train_index = createDataPartition(new_car_centered$model.year, p = 0.5, list = FALSE)

train_data = new_car_centered[train_index, ]
valid_data = new_car_centered[-train_index, ]

cat("Proportions in full dataset:\n")

## Proportions in full dataset:
prop.table(table(new_car_centered$model.year))

##
##          70         71         72         73         74
## 0.07397959184 0.06887755102 0.07142857143 0.10204081633 0.06632653061
##          75         76         77         78         79
## 0.07653061224 0.08673469388 0.07142857143 0.09183673469 0.07397959184
##          80         81         82
## 0.06887755102 0.07142857143 0.07653061224

# Proportions in training dataset
cat("\nProportions in training dataset:\n")

##
## Proportions in training dataset:
prop.table(table(train_data$model.year))

##
##          70         71         72         73         74
## 0.07575757576 0.07070707071 0.07070707071 0.10101010101 0.06565656566
##          75         76         77         78         79
## 0.07575757576 0.08585858586 0.07070707071 0.09090909091 0.07575757576
##          80         81         82
## 0.07070707071 0.07070707071 0.07575757576

# Proportions in validation dataset
cat("\nProportions in validation dataset:\n")

##
## Proportions in validation dataset:
```

```

prop.table(table(valid_data$model.year))

##
##      70          71          72          73          74
## 0.07216494845 0.06701030928 0.07216494845 0.10309278351 0.06701030928
##      75          76          77          78          79
## 0.07731958763 0.08762886598 0.07216494845 0.09278350515 0.07216494845
##      80          81          82
## 0.06701030928 0.07216494845 0.07731958763

```

Now we attempt to fit a regression model based on our proposed full model and the insights we have uncovered so far.

```

# Fit a polynomial regression model
# Include: squared terms for each continuous predictor and interaction terms that make sense
# Interaction terms: weight*horsepower
poly_model = lm(mpg ~ horsepower_c + I(horsepower_c^2) +
                 weight_c + I(weight_c^2) +
                 weight_c:horsepower_c +
                 model.year,
                 data = train_data)

summary(poly_model)

##
## Call:
## lm(formula = mpg ~ horsepower_c + I(horsepower_c^2) + weight_c +
##     I(weight_c^2) + weight_c:horsepower_c + model.year, data = train_data)
##
## Residuals:
##      Min        1Q    Median        3Q       Max
## -8.8948244 -1.3592776 -0.0305249  1.1978614 10.3435512
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.916975e+01 7.558687e-01 25.36122 < 2.22e-16 ***
## horsepower_c -3.934733e-02 1.397946e-02 -2.81465 0.00542670 **
## I(horsepower_c^2) 3.951015e-05 1.906689e-04 0.20722 0.83607334
## weight_c -5.378537e-03 5.267581e-04 -10.21064 < 2.22e-16 ***
## I(weight_c^2) 9.901549e-07 6.214335e-07 1.59334 0.11283838
## model.year71 -4.994325e-01 1.010334e+00 -0.49432 0.62167985
## model.year72 -7.297705e-01 9.744886e-01 -0.74888 0.45490982
## model.year73 -1.719868e+00 8.740397e-01 -1.96772 0.05063602 .
## model.year74 8.556215e-01 1.059003e+00 0.80795 0.42018596
## model.year75 1.329697e+00 1.015735e+00 1.30910 0.19216980
## model.year76 2.154185e+00 9.668530e-01 2.22804 0.02711646 *
## model.year77 2.184989e+00 9.686977e-01 2.25559 0.02529982 *
## model.year78 3.136587e+00 9.212701e-01 3.40463 0.00081683 ***
## model.year79 5.405369e+00 9.953055e-01 5.43086 1.7986e-07 ***
## model.year80 9.169699e+00 1.008298e+00 9.09423 < 2.22e-16 ***
## model.year81 6.827216e+00 1.005914e+00 6.78708 1.5991e-10 ***
## model.year82 8.542272e+00 9.982826e-01 8.55697 4.9761e-15 ***
## horsepower_c:weight_c 2.601423e-05 1.867648e-05 1.39289 0.16537183
##
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```

## 
## Residual standard error: 2.381232 on 180 degrees of freedom
## Multiple R-squared:  0.9137447, Adjusted R-squared:  0.9055984 
## F-statistic: 112.1664 on 17 and 180 DF,  p-value: < 2.2204e-16

```

Observe that $R_a^2 = 0.9056$. We WIN! However, this model is estimating 17 different parameter values. While this is impressive, we should still follow the formal procedure using a best subset selection algorithm.

Reduction of Explanatory Variables

It's time to get serious and find a model that balances being parsimonious with adequately capturing the key relationships in the data.

Practically speaking, variables must be chosen carefully for analysis. Using keen insight and domain specific knowledge, we have already dropped two variables from the model (`cylinders`, `acceleration` and `displacement`). We seek to reduce our model further.

In particular, we aim to determine whether adding quadratic terms for the `horsepower`, and `weight` predictors meaningfully improves the model's ability to explain variation in the response. We will use the partial F test to determine this.

For this particular partial F test, our full model takes the following form:

$$\begin{aligned}
Y_i = & \beta_0 + \beta_1 x_{i,\text{hp}} + \beta_2 x_{i,\text{hp}}^2 + \beta_3 x_{i,\text{wt}} + \beta_4 x_{i,\text{wt}}^2 + \beta_5 (x_{i,\text{hp}} x_{i,\text{wt}}) \\
& + \beta_6 X_{i,71} + \beta_7 X_{i,72} + \beta_8 X_{i,73} + \beta_9 X_{i,74} + \beta_{10} X_{i,75} \\
& + \beta_{11} X_{i,76} + \beta_{12} X_{i,77} + \beta_{13} X_{i,78} + \beta_{14} X_{i,79} \\
& + \beta_{15} X_{i,80} + \beta_{16} X_{i,81} + \beta_{17} X_{i,82} + \varepsilon_i,
\end{aligned}$$

and the reduced model takes the following form:

$$\begin{aligned}
Y_i = & \beta_0 + \beta_1 x_{i,\text{hp}} + \beta_3 x_{i,\text{wt}} + \beta_5 (x_{i,\text{hp}} x_{i,\text{wt}}) \\
& + \beta_6 X_{i,71} + \beta_7 X_{i,72} + \beta_8 X_{i,73} + \beta_9 X_{i,74} + \beta_{10} X_{i,75} \\
& + \beta_{11} X_{i,76} + \beta_{12} X_{i,77} + \beta_{13} X_{i,78} + \beta_{14} X_{i,79} \\
& + \beta_{15} X_{i,80} + \beta_{16} X_{i,81} + \beta_{17} X_{i,82} + \varepsilon_i.
\end{aligned}$$

where

$$\begin{aligned}
Y_i &= \text{mpg} \\
x_{i,\text{hp}} &= \text{centered horsepower} \\
x_{i,\text{wt}} &= \text{centered weight} \\
X_{i,71}, X_{i,72}, \dots, X_{i,82} &= \begin{cases} 1, & \text{if model year } = j \\ 0, & \text{otherwise} \end{cases} \quad (j = 71, \dots, 82)
\end{aligned}$$

We now test whether the quadratic terms $x_{i,\text{hp}}^2$ and $x_{i,\text{wt}}^2$ may be removed from the regression model, given that all other variables are retained. This test is conducted at the $\alpha = 0.05$ significance level.

Under the null hypothesis, after accounting for all other variables in the model, the quadratic terms do not improve prediction of Y_i (mpg). Hence, they may be removed without loss of explanatory power. The alternative hypothesis states that at least one of these terms contributes additional predictive information and therefore should be retained.

The hypotheses are

$$H_0 : \beta_2 = \beta_4 = 0$$

$$H_A : \text{not both } \beta_2 \text{ and } \beta_4 \text{ equal zero}$$

The general linear test statistic is

$$F^* = \frac{SSE(\text{reduced}) - SSE(\text{full})}{df_{\text{reduced}} - df_{\text{full}}} / \frac{SSE(\text{full})}{df_{\text{full}}}$$

$$= \frac{SSR(x_{i2}^2, x_{i3}^2 | \text{other predictors})}{df_{\text{reduced}} - df_{\text{full}}} / MSE_{\text{full}}.$$

```
# Fit full model
full_model = lm(mpg ~ horsepower_c + I(horsepower_c^2) +
                 weight_c + I(weight_c^2) +
                 weight_c:horsepower_c +
                 model.year,
                 data = train_data)

# Fit reduced model
reduced_model = lm(mpg ~ horsepower_c +
                 weight_c +
                 weight_c:horsepower_c +
                 model.year,
                 data = train_data)

anova(reduced_model, full_model)

## Analysis of Variance Table
##
## Model 1: mpg ~ horsepower_c + weight_c + weight_c:horsepower_c + model.year
## Model 2: mpg ~ horsepower_c + I(horsepower_c^2) + weight_c + I(weight_c^2) +
##           weight_c:horsepower_c + model.year
##   Res.Df   RSS Df Sum of Sq    F Pr(>F)
## 1     182 1037.6612
## 2     180 1020.6475  2 17.013692 1.50026 0.22585
```

The p -value for this test is $p = 0.22585$, which is larger than $\alpha = 0.05$. Therefore, at the 5% significance level, we fail to reject the null hypothesis. This indicates that there is insufficient evidence to conclude that the quadratic terms for `horsepower` and `weight` provide additional explanatory value for predicting `mpg`.

Let's check the adjusted R^2 value for the model we have thus far.

```
full_model = lm(mpg ~ horsepower_c +
                 weight_c +
                 weight_c:horsepower_c +
                 model.year,
                 data = train_data)
summary(full_model)

##
## Call:
## lm(formula = mpg ~ horsepower_c + weight_c + weight_c:horsepower_c +
##       model.year, data = train_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -24.5000  -4.4286  -1.3333  -0.7143  42.8571
```

```

## -8.6211881 -1.3420670 -0.0076507  1.1480973 10.3015546
##
## Coefficients:
##                               Estimate     Std. Error    t value   Pr(>|t|) 
## (Intercept)           1.919723e+01  7.298325e-01  26.30361 < 2.22e-16 ***
## horsepower_c          -4.751788e-02  1.113558e-02 -4.26721  3.1793e-05 ***
## weight_c              -5.005924e-03  4.355184e-04 -11.49417 < 2.22e-16 ***
## model.year71          -2.119549e-01  9.898950e-01 -0.21412  0.83069431
## model.year72          -4.494062e-01  9.436474e-01 -0.47624  0.63447192
## model.year73          -1.585017e+00  8.638656e-01 -1.83480  0.06816849 .
## model.year74          1.231482e+00  1.036656e+00  1.18794  0.23640618
## model.year75          1.627548e+00  1.002613e+00  1.62331  0.10625460
## model.year76          2.317983e+00  9.573141e-01  2.42134  0.01644511 *
## model.year77          2.403656e+00  9.507941e-01  2.52805  0.01232014 *
## model.year78          3.251563e+00  9.078983e-01  3.58142  0.00043841 ***
## model.year79          5.537247e+00  9.911126e-01  5.58690  8.3194e-08 ***
## model.year80          9.267466e+00  1.001881e+00  9.25007 < 2.22e-16 ***
## model.year81          6.943356e+00  9.997641e-01  6.94499  6.4786e-11 ***
## model.year82          8.594765e+00  9.934256e-01  8.65165  2.6242e-15 ***
## horsepower_c:weight_c 4.716841e-05  5.694362e-06  8.28335  2.5326e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.387768 on 182 degrees of freedom
## Multiple R-squared:  0.9123069, Adjusted R-squared:  0.9050794
## F-statistic: 126.2279 on 15 and 182 DF, p-value: < 2.2204e-16

```

Observe that $R_a^2 = 0.9051$. Not bad! We still have 15 parameter values that we are estimating.

Note that there is no coefficient for `model.year70` in the summary. This means that all other year coefficients (`model.year71`, `model.year72`, ...) are interpreted relative to year 70. For example, `model.year71 = -0.2119` means that, holding other predictors constant, the expected mpg for year 71 is 0.2119 units lower than year 70 (although this is not statistically significant here).

To see if we can simplify our model further, we'll try the automatic best subset approach using the R_a^2 Criterion.

```

# Fit all possible subsets
best_subsets = regsubsets(mpg ~ horsepower_c +
                           weight_c +
                           weight_c:horsepower_c +
                           model.year,
                           data = train_data,
                           nbest = 15, # takes the top 15 models from each model size
                           nvmax = 8, # limits the number of predictors in the model
                           method = "exhaustive")

summary_best = summary(best_subsets)

# Create a truth matrix
all_models = data.frame(Adjusted_R2 = summary_best$adjr2, summary_best$which) %>%
  arrange(desc(Adjusted_R2))

all_models[1:5, ]

##      Adjusted_R2 X.Intercept. horsepower_c weight_c model.year71 model.year72

```

```

## X8    0.8919131988      TRUE      TRUE      TRUE      FALSE      FALSE
## X8.1   0.8890171966      TRUE      TRUE      TRUE      FALSE      FALSE
## X8.2   0.8830549021      TRUE      TRUE      TRUE      FALSE      FALSE
## X8.3   0.8826399966      TRUE      TRUE      TRUE      FALSE      FALSE
## X8.4   0.8825360239      TRUE      TRUE      TRUE      FALSE      TRUE
##      model.year73 model.year74 model.year75 model.year76 model.year77
## X8       TRUE      FALSE      FALSE      FALSE      FALSE
## X8.1    FALSE      FALSE      FALSE      FALSE      FALSE
## X8.2    FALSE      FALSE      FALSE      FALSE      TRUE
## X8.3    FALSE      FALSE      FALSE      TRUE      FALSE
## X8.4    FALSE      FALSE      FALSE      FALSE      FALSE
##      model.year78 model.year79 model.year80 model.year81 model.year82
## X8       FALSE      TRUE      TRUE      TRUE      TRUE
## X8.1    TRUE      TRUE      TRUE      TRUE      TRUE
## X8.2    FALSE      TRUE      TRUE      TRUE      TRUE
## X8.3    FALSE      TRUE      TRUE      TRUE      TRUE
## X8.4    FALSE      TRUE      TRUE      TRUE      TRUE
##      horsepower_c.weight_c
## X8       TRUE
## X8.1    TRUE
## X8.2    TRUE
## X8.3    TRUE
## X8.4    TRUE

```

The five best subsets of predictor variables for a regression model, based on the R_a^2 criterion, are:

- $(x_{i, \text{hp}}, x_{i, \text{wt}}, X_{i, 73}, X_{i, 79}, X_{i, 80}, X_{i, 81}, X_{i, 82}, x_{i, \text{hp}} x_{i, \text{wt}})$, - $(x_{i, \text{hp}}, x_{i, \text{wt}}, X_{i, 78}, X_{i, 79}, X_{i, 80}, X_{i, 81}, X_{i, 82}, x_{i, \text{hp}} x_{i, \text{wt}})$,
- $(x_{i, \text{hp}}, x_{i, \text{wt}}, X_{i, 77}, X_{i, 79}, X_{i, 80}, X_{i, 81}, X_{i, 82}, x_{i, \text{hp}} x_{i, \text{wt}})$, - $(x_{i, \text{hp}}, x_{i, \text{wt}}, X_{i, 76}, X_{i, 79}, X_{i, 80}, X_{i, 81}, X_{i, 82}, x_{i, \text{hp}} x_{i, \text{wt}})$,
and - $(x_{i, \text{hp}}, x_{i, \text{wt}}, X_{i, 72}, X_{i, 79}, X_{i, 80}, X_{i, 81}, X_{i, 82}, x_{i, \text{hp}} x_{i, \text{wt}})$.

These results indicate that the model consistently selects **horsepower**, **weight**, their interaction, and the most recent model years (79-82). This supports retaining these key predictors while potentially dropping less informative year levels.

In performing the best subsets analysis with the R_a^2 criterion, the choice of reference level for the **model.year** factor does not affect which predictors are selected, because R_a^2 depends on the overall fit of the model rather than the interpretation of individual coefficients. However, setting a consistent reference level (e.g., year 70) can improve interpretability of the coefficient estimates without altering the predictive performance or subset selection results.

The corresponding R_a^2 values for the top models are relatively close, suggesting that this metric alone may not clearly distinguish the best model. Therefore, additional model selection criteria are useful. Other commonly used criteria include Mallows' C_p , the Akaike Information Criterion (AIC_p), Schwarz' Bayesian Criterion (SBC_p), and the Prediction Sum of Squares Criterion ($PRESS_p$).

Mallows' C_p criterion evaluates both bias and variance, with models having C_p values close to the number of predictors p preferred. AIC and BIC penalize model complexity while rewarding good in-sample fit, favoring lower values. The PRESS criterion is particularly useful here because it directly measures predictive accuracy. It estimates how well a model will predict unseen data. Smaller PRESS values indicate better out-of-sample prediction, which aligns closely with our primary goal of building a predictive model.

By using these criteria in combination with R_a^2 , we can identify models that are both parsimonious and likely to perform well on new data. In particular, PRESS will serve as our key metric for predictive reliability.

We first create a function to calculate PRESS. The function extracts the model matrix, computes the hat matrix, and then obtains the diagonal elements. It calculates ordinary residuals, the sum of squared errors (SSE) for the training data, and PRESS as the sum of squared predicted residuals. The SSE reflects in-sample fit, while PRESS estimates predictive performance.

```

calc_PRESS = function(model) {
  X = model.matrix(model)          # model matrix
  H = X %*% solve(t(X) %*% X) %*% t(X)  # hat matrix
  h = diag(H)
  res = residuals(model)          # ordinary residuals
  SSE = sum(res^2)
  PRESS = sum((res / (1 - h))^2)
  return(list(SSE = SSE, PRESS = PRESS))
}

```

The best subsets analysis using the R_a^2 criterion suggested that including certain levels of `model.year` provides the strongest predictive performance. Since `model.year` is a categorical variable, we cannot drop individual levels arbitrarily without changing the meaning of the factor. To address this, we create a new factor, `model_year_sub`, which explicitly includes only the levels of interest identified from the subset analysis. We then calculate the corresponding PRESS values with model year 70 held as our reference year.

```

# Only the first years change in the best subsets algorithm
# Check them all
first_years = 70:78

# Initialize a data frame to store results
press_results = data.frame(first_year = integer(),
                           SSE = numeric(),
                           PRESS = numeric(),
                           percent_diff = numeric())

for (yr in first_years) {

  # Create a factor with only the selected years (first_years + 79:82)
  # All other years are implicitly treated as zeros in the model matrix
  year_levels = unique(c(yr, 79, 80, 81, 82))

  train_data_sub = train_data %>%
    mutate(model_year_sub = ifelse(model.year %in% year_levels, as.character(model.year), "other"))

  # Make 70 the reference level
  train_data_sub$model_year_sub = factor(train_data_sub$model_year_sub,
                                         #This prepends "70" to the beginning of the level list
                                         levels = c("70", sort(setdiff(unique(train_data_sub$model_year_sub), 70)))

  # Fit the reduced model
  mod = lm(mpg ~ horsepower_c +
            weight_c +
            weight_c:horsepower_c +
            model_year_sub,
            data = train_data_sub)

  # Calculate SSE and PRESS
  press_vals = calc_PRESS(mod)

  # Compute relative difference as a percentage
  percent_diff = 100 * (press_vals$PRESS - press_vals$SSE) / press_vals$SSE
}

```

```

# Store results
press_results = rbind(press_results, data.frame(first_year = yr,
                                                SSE = press_vals$SSE,
                                                PRESS = press_vals$PRESS,
                                                percent_diff = percent_diff))
}

press_results_sorted = press_results %>%
  arrange(percent_diff)

press_results_sorted

##   first_year      SSE      PRESS percent_diff
## 1        74 1359.724967 1514.623822  11.39192550
## 2        77 1327.601044 1479.138736  11.41439983
## 3        72 1333.491529 1489.297657  11.68407332
## 4        71 1339.320274 1496.584226  11.74207210
## 5        76 1332.311195 1489.263284  11.78043762
## 6        75 1355.478303 1516.127295  11.85183062
## 7        70 1355.135124 1516.242794  11.88867939
## 8        78 1259.915024 1413.195786  12.16596036
## 9        73 1227.038608 1382.810895  12.69497844

```

NOTE: Setting year 70 as the reference level for `model.year` affects only coefficient interpretation and does not change predictions or PRESS values. I just thought it would be nice to keep a consistent base reference year throughout the analysis. This also helps with interpretation questions we may be asked during the presentation.

The analysis shows that using $X_{i,74}$ as the first-year indicator yields the lowest percent difference between PRESS and SSE (11.39%), but we observe that all other years, such as 77 (11.41%) and 72 (11.68%), perform nearly equivalently. Considering both predictive reliability and explanatory power (adjusted R^2), selecting $X_{i,77}$ as the first-year indicator provides a reasonable balance.

However, I want to WIN! A model with $X_{i,73}$ produces a slightly better R_a^2 value and the PRESS criterion indicates that the model still predicts reasonably well. Megan feel free to overrule me on this.

===== Below is a quick sanity check, since this is all getting very technical and crazy...

The code below is intended to fit the best subsets regression model we identified above (aka, a model with `horsepower`, `weight`, their interaction term, and model years 73, 79, 80, 81, and 82 with model year 70 treated as the reference category) to the training dataset. If all goes well the R_a^2 value should match that of the `regsubsets` procedure above.

```

# Create a factor for just the years of interest, with 70 as reference
train_data_sub = train_data %>%
  mutate(model_year_sub = ifelse(model.year %in% c(73, 79, 80, 81, 82),
                                 as.character(model.year), "70")) # 70 = reference

train_data_sub$model_year_sub = factor(train_data_sub$model_year_sub,
                                       levels = c("70", "73", "79", "80", "81", "82"))

# Fit the best subset model
best_subset_model = lm(mpg ~ horsepower_c +
                       weight_c +
                       weight_c:horsepower_c +
                       model_year_sub,

```

```

        data = train_data_sub)

summary(best_subset_model)

##
## Call:
## lm(formula = mpg ~ horsepower_c + weight_c + weight_c:horsepower_c +
##     model_year_sub, data = train_data_sub)
##
## Residuals:
##      Min        1Q    Median        3Q       Max
## -8.6339015 -1.4163709 -0.0064834  1.4871195 10.2540030
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)            2.070620e+01 2.910243e-01 71.14939 < 2.22e-16 ***
## horsepower_c          -5.410952e-02 9.914820e-03 -5.45744 1.5031e-07 ***
## weight_c              -4.934639e-03 4.050762e-04 -12.18200 < 2.22e-16 ***
## model_year_sub73      -2.815537e+00 6.220675e-01 -4.52610 1.0608e-05 ***
## model_year_sub79      4.027438e+00 7.063027e-01  5.70214 4.4937e-08 ***
## model_year_sub80      7.696002e+00 7.442248e-01 10.34096 < 2.22e-16 ***
## model_year_sub81      5.376953e+00 7.378301e-01  7.28752 8.3482e-12 ***
## model_year_sub82      7.009622e+00 7.204151e-01  9.72998 < 2.22e-16 ***
## horsepower_c:weight_c 4.459048e-05 5.938927e-06  7.50817 2.3028e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.547993 on 189 degrees of freedom
## Multiple R-squared:  0.8963025, Adjusted R-squared:  0.8919132
## F-statistic: 204.2012 on 8 and 189 DF,  p-value: < 2.2204e-16

```

It matches! YAY! And all predictor variables are all significant!!!

Diagnositics of the Fitted Regression

I wanna be done... But we must check the regression assumptions now.

A lot of this is copy and paste from my Midterm, so please check my work.

The standard assumptions of simple linear regression are: - linearity, - constant error variance, - independence of observations, and - approximate normality of residuals.

```

# Fit the best subset model
best_subset_model = lm(mpg ~ horsepower_c +
                      weight_c +
                      weight_c:horsepower_c +
                      model_year_sub,
                      data = train_data_sub)

# Create a data frame with residuals and observation sequence
resid_df = train_data_sub %>%
  mutate(obs_seq = 1:length(best_subset_model$residuals),
         residuals = best_subset_model$residuals,
         fitted = fitted(best_subset_model),

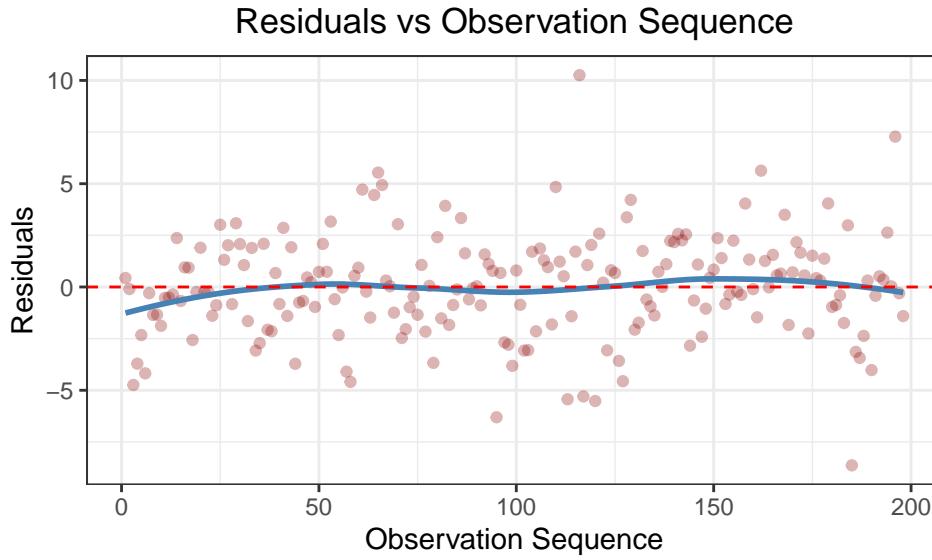
```

```
stud_resid = rstudent(best_subset_model))
```

Independence, is by far the most important assumption to check.

```
# Plot residuals vs observation sequence
resid_vs_seq = ggplot(resid_df, aes(x = obs_seq, y = residuals)) +
  geom_point(alpha = 0.3, color = "darkred") +
  geom_smooth(method = "loess", color = "steelblue", se = FALSE, size = 1) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Residuals vs Observation Sequence",
       x = "Observation Sequence",
       y = "Residuals") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

resid_vs_seq
```



```
# Save the plot as a PNG
ggsave(filename = "figures/resid_vs_seq.png",
       plot = resid_vs_seq,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)
```

This plot allows us to check for independence of errors. The residuals appear to be randomly scattered without visible trends or cycles, indicating that first-order autocorrelation is unlikely to be present. In other words, there is no visual evidence of time-based correlation or dependence in the errors. This supports the assumption of independence of residuals, which is a key requirement for valid statistical inference in linear regression.

To formally evaluate the independence of the errors, we conduct a Durbin–Watson test for first-order autocorrelation. The test is performed at the $\alpha = 0.05$ significance level. The null hypothesis assumes that the residuals are uncorrelated ($\rho = 0$), while the alternative hypothesis allows for the presence of first-order autocorrelation ($\rho \neq 0$). Formally, we have

$$H_0 : \rho = 0$$

$$H_A : \rho \neq 0$$

```

dw_result = dwtest(best_subset_model, alternative = "two.sided")

dw_result

## 
## Durbin-Watson test
##
## data: best_subset_model
## DW = 1.9119258, p-value = 0.4952534
## alternative hypothesis: true autocorrelation is not 0

```

Based on the resulting p -value, there is insufficient evidence to reject the null hypothesis at the $\alpha = 0.05$ level. This indicates that there is no significant first-order autocorrelation in the residuals, which is consistent with the residuals versus observation sequence plot. Overall, this supports the assumption that the errors in our simple linear regression model are independent, and no temporal or cyclical structure remains unaccounted for in the data.

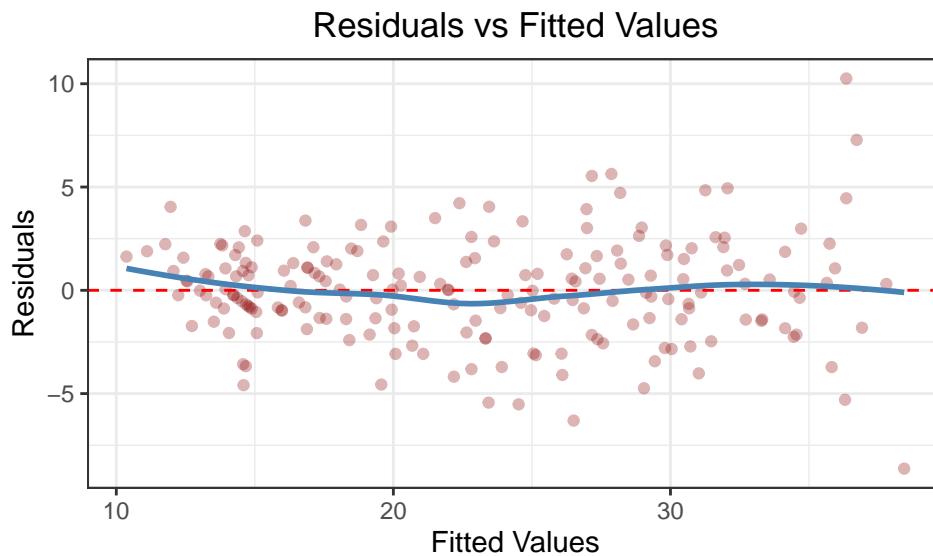
Next, we examine the residuals versus fitted values plot.

```

# Residuals vs Predictor
resid_vs_mpg = ggplot(resid_df, aes(x = fitted, y = residuals)) +
  geom_point(alpha = 0.3, color = "darkred") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  geom_smooth(method = "loess", se = FALSE, color = "steelblue") +
  labs(title = "Residuals vs Fitted Values",
       x = "Fitted Values",
       y = "Residuals") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

resid_vs_mpg

```



```

# Save the plot as a PNG
ggsave(filename = "figures/resid_vs_mpg.png",
       plot = resid_vs_mpg,
       width = 5,
       height = 3,
       units = "in",

```

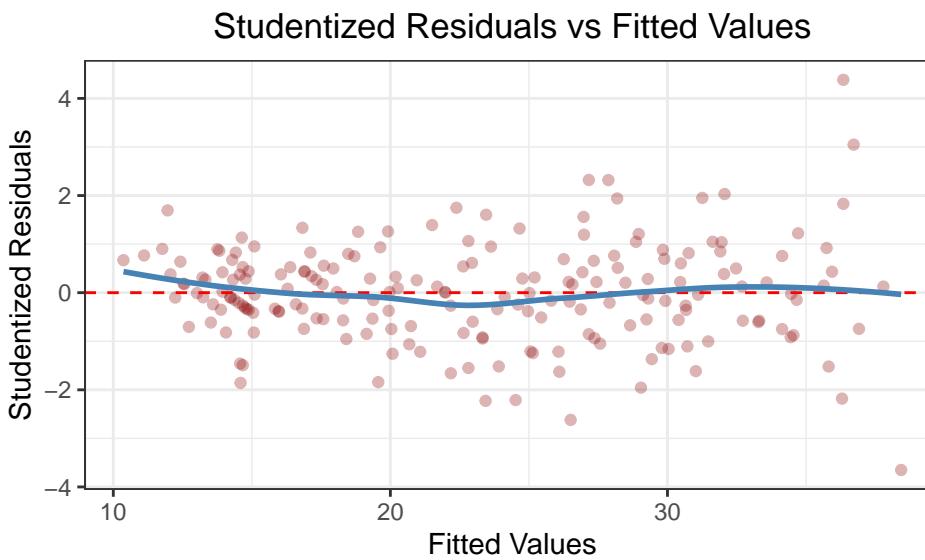
```

dip = 300)

# Studentized residuals vs Fitted Values
stud_resid_plot = ggplot(resid_df, aes(x = fitted, y = stud_resid)) +
  geom_point(alpha = 0.3, color = "darkred") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  geom_smooth(method = "loess", se = FALSE, color = "steelblue") +
  labs(title = "Studentized Residuals vs Fitted Values",
       x = "Fitted Values",
       y = "Studentized Residuals") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

stud_resid_plot

```



```

# Save the plot as a PNG
ggsave(filename = "figures/stud_resid_vs_fitted.png",
       plot = stud_resid_plot,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)

```

The residuals versus fitted values plot shows residuals randomly scattered around zero, with no obvious systematic patterns or curvature. The relatively flat LOESS curve supports the assumption of linearity. There is a slight hint of a funnel shape in the residuals, suggesting some heteroscedasticity. Overall, the plot indicates that the linear model is reasonably appropriate for the data. However, the potential heteroscedasticity motivates considering a transformation of the response variable, `mpg`, to better satisfy the constant variance assumption.

A formal F test for lack of fit is typically used to assess whether a linear regression model adequately captures the relationship between a predictor and the response variable. The test requires replicated observations at one or more levels of the predictor, which allow the total residual variation to be partitioned into two components:

- Pure error: variation among observations at the same predictor value, and
- Lack-of-fit: variation due to the model failing to capture the true mean relationship.

However, in our dataset, we have multiple continuous predictors and interaction terms, so the traditional lack-of-fit F test is not generally applicable.

There also appears to be few high-mpg observations, which could be influential points. Therefore, it is necessary to perform outlier diagnostics (e.g., studentized deleted residuals) to identify these potential extreme points.

To further assess the constant variance (homoscedasticity) assumption of the regression model, we employ the Brown–Forsythe test, a robust modification of Levene’s test that uses deviations from the median rather than the mean and does not rely on the normality of the residuals. The test evaluates whether the variability of the residuals differs across groups. Significant differences in residual variability suggest that the assumption of constant variance may be violated.

In our dataset, observations are naturally grouped by model.year. If the residual variance changes across these year groups, the test will detect it as a statistically significant difference.

Using a significance level of $\alpha = 0.05$, the hypotheses for the Brown–Forsythe test are:

$$H_0 : \sigma_{73}^2 = \sigma_{79}^2 = \sigma_{80}^2 = \sigma_{81}^2 = \sigma_{82}^2 \quad \text{Residual variance is equal between groups (homoscedasticity)}$$

$$H_A : \sigma_i^2 \neq \sigma_j^2 \quad \text{for some } i \neq j \quad \text{Residual variance differs between groups (heteroscedasticity)}$$

```
# Brown-Forsythe test (Levene test using median)
bf_test = leveneTest(residuals(best_subset_model) ~ model_year_sub,
                     data = train_data_sub,
                     center = median)

bf_test

## Levene's Test for Homogeneity of Variance (center = median)
##          Df F value    Pr(>F)
## group      5 3.66961 0.0034057 **
##           192
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

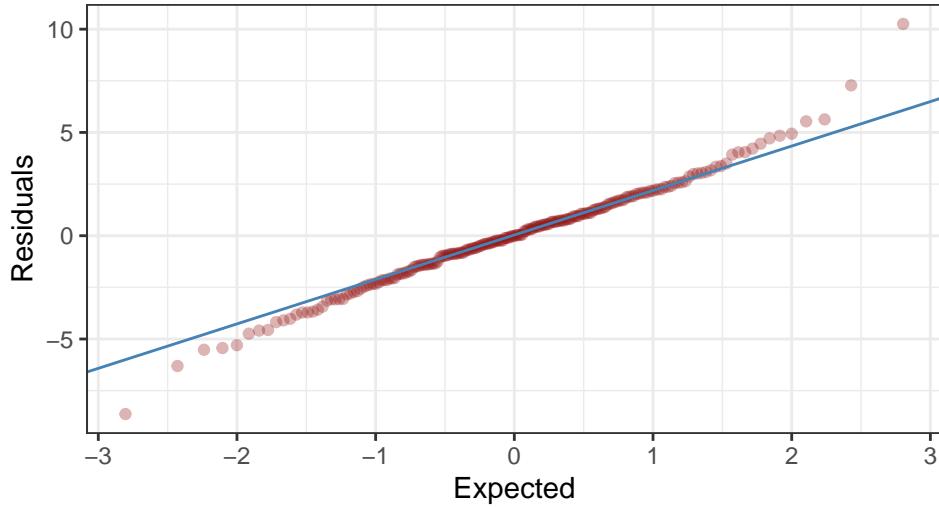
The Brown–Forsythe test returned a p -value smaller than 0.05. Therefore, we reject the null hypothesis, and there is statistical evidence to suggest that the error variance differs between year groups. This supports the our move to transform the response variable, mpg.

Finally, we examine the Normal QQ plot.

```
# Normal Q-Q Plot of Residuals
resid_qq = ggplot(resid_df, aes(sample = residuals)) +
  stat_qq(alpha = 0.3, color = "darkred") +
  stat_qq_line(color = "steelblue") +
  labs(title = "Normal Q-Q Plot of Residuals",
       x = "Expected",
       y = "Residuals") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

resid_qq
```

Normal Q–Q Plot of Residuals



```
# Save the plot as a PNG
ggsave(filename = "figures/resid_qq.png",
       plot = resid_qq,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)
```

If the residuals were approximately normally distributed, we would expect the points to lie close to the reference line. However, we note that the points in both the upper and lower tails bend away from the line slightly, indicating heavier tails than those expected under a normal distribution. This suggests that extreme residual values occur more frequently than the normal model assumes. While the central portion of the distribution roughly follows the reference line, the pronounced tail spread implies that the assumption of normally distributed errors may not be entirely appropriate for this model. This observation does not necessarily invalidate the regression, but it does caution against inference procedures that strongly depend on strict normality of residuals.

To formally assess the normality of the residuals, we now apply the Shapiro-Wilk test. We conduct the test at the significance level $\alpha = 0.05$. The Shapiro-Wilk test formally evaluates whether the residuals are normally distributed. The hypotheses for the test are:

$$H_0 : \text{The residuals are normally distributed}$$

$$H_A : \text{The residuals are not normally distributed}$$

```
# Shapiro-Wilk test for normality
shapiro_test = shapiro.test(resid_df$residuals)

# Print results
shapiro_test

## 
##  Shapiro-Wilk normality test
## 
## data: resid_df$residuals
## W = 0.98522562, p-value = 0.0361365
```

Since the p -value is smaller than 0.05, we reject the null hypothesis. This indicates that the residuals deviate from normality, but only by the smallest of margins.

Given this, it may be appropriate to consider a transformation of the response variable, such as a Box-Cox power transformation, to better satisfy the assumptions of linear regression. This could improve the approximation of normality in the residuals and enhance the reliability of inference from the model.

Testing for Outliers

We identify potentially outlying response values Y_i as those observations whose studentized deleted residuals are large in absolute value. To formally test whether the observation with the largest absolute studentized deleted residual is an outlier, we can use the Bonferroni procedure, which adjusts for the fact that we are simultaneously testing all n observations.

If the regression model is appropriate and the residuals satisfy the assumptions, each studentized deleted residual follows a t distribution with $n - p - 1$ degrees of freedom, where p is the number of predictors (excluding the intercept). Since we do not know in advance which case will have the largest residual, we account for the family-wise error by considering all n tests. The critical value for testing at significance level α is $t_{\frac{1-\alpha}{2n}; n-p-1}$.

An observation whose absolute studentized deleted residual exceeds this threshold can be considered an outlier at the α significance level, after adjusting for multiple comparisons.

We shall use the Bonferroni simultaneous test procedure with a family significance level of $\alpha = 0.05$.

```
# Number of observations and predictors
n = nrow(resid_df)
p = length(coef(best_subset_model)) - 1 # exclude intercept

# Compute studentized deleted residuals
stud_del_resid = rstudent(best_subset_model)

# Bonferroni critical value
alpha = 0.05
t_crit = qt(1 - alpha / (2 * n), df = n - p - 1)

# Identify potential outliers
outliers = which(abs(stud_del_resid) > t_crit)

# Display results
data.frame(Observation = outliers,
           StudDeletedResid = stud_del_resid[outliers],
           BonferroniThreshold = t_crit)
```

```
##      Observation StudDeletedResid BonferroniThreshold
## 235          116     4.382885691       3.730565574
```

In our analysis, observation 116 has a studentized deleted residual of 4.38, which exceeds the Bonferroni threshold of 3.73. Therefore, it is identified as an outlier.

```
resid_df[116, ]

##      mpg cylinders displacement horsepower weight acceleration model.year
## 235 46.6          4           86          65    2110        17.9         80
##      car.name displacement_c horsepower_c      weight_c acceleration_c
## 235 mazda glc   -108.4119898 -39.46938776 -867.5841837    2.358673469
##      model_year_sub obs_seq  residuals      fitted  stud_resid
## 235            80      116 10.25400298 36.34599702 4.382885691
```

We can use the plot below to verify that this is the only outlier.

```

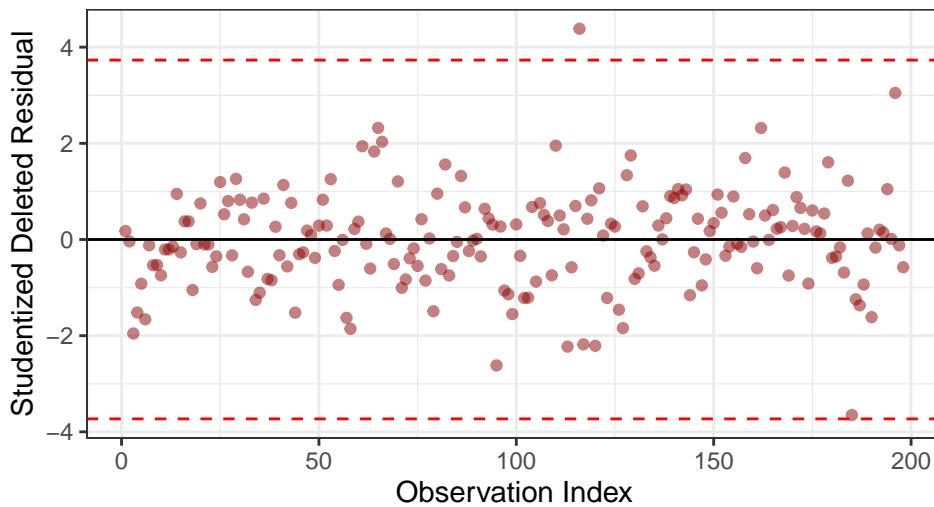
# Number of observations and predictors
n = nrow(resid_df)
p = length(coef(best_subset_model)) - 1 # exclude intercept

# Bonferroni critical value
alpha = 0.05
t_crit = qt(1 - alpha / (2 * n), df = n - p - 1)

# Plot studentized deleted residuals with Bonferroni threshold
ggplot(resid_df, aes(x = 1:nrow(resid_df), y = stud_del_resid)) +
  geom_point(alpha = 0.5, color = "darkred") +
  geom_hline(yintercept = c(-t_crit, 0, t_crit),
              linetype = c("dashed", "solid", "dashed"),
              color = c("red", "black", "red")) +
  labs(title = "Studentized Deleted Residuals with Bonferroni Threshold",
       x = "Observation Index",
       y = "Studentized Deleted Residual") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

```

Studentized Deleted Residuals with Bonferroni Threshold



We now remove the point from the dataset.

```

# Remove outliers from the training dataset
train_data_clean = train_data_sub[-outliers, ]

```

Re-fit the best subset model to the clean data.

```

# Fit the best subset model
best_subset_model = lm(mpg ~ horsepower_c +
                       weight_c +
                       weight_c:horsepower_c +
                       model_year_sub,
                       data = train_data_clean)

summary(best_subset_model)

##
## Call:
## lm(formula = mpg ~ horsepower_c + weight_c + weight_c:horsepower_c + model_year_sub, data = train_data_clean)

```

```

## lm(formula = mpg ~ horsepower_c + weight_c + weight_c:horsepower_c +
##     model_year_sub, data = train_data_clean)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -7.7716789 -1.4069895  0.0106084  1.3802111  7.3700503
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)            2.074078e+01 2.780545e-01 74.59248 < 2.22e-16 ***
## horsepower_c          -5.224972e-02 9.478646e-03 -5.51236 1.1564e-07 ***
## weight_c              -4.949096e-03 3.868819e-04 -12.79227 < 2.22e-16 ***
## model_year_sub73      -2.814281e+00 5.941053e-01 -4.73701 4.2644e-06 ***
## model_year_sub79      4.038215e+00 6.745585e-01  5.98646 1.0664e-08 ***
## model_year_sub80      6.934058e+00 7.317229e-01  9.47634 < 2.22e-16 ***
## model_year_sub81      5.399909e+00 7.046837e-01  7.66288 9.3890e-13 ***
## model_year_sub82      7.036646e+00 6.880597e-01 10.22680 < 2.22e-16 ***
## horsepower_c:weight_c 4.308851e-05 5.682312e-06  7.58292 1.5072e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.433459 on 188 degrees of freedom
## Multiple R-squared:  0.901252, Adjusted R-squared:  0.8970499
## F-statistic: 214.4795 on 8 and 188 DF, p-value: < 2.2204e-16

# Create a data frame with residuals and observation sequence
train_data_clean = train_data_clean %>%
  mutate(obs_seq = 1:length(best_subset_model$residuals),
        residuals = best_subset_model$residuals,
        fitted = fitted(best_subset_model))

```

A Discussion on Model Deviations

Based on our diagnostic analyses, the assumptions of linearity and independence of observations appear to be reasonably satisfied for the regression model. Additionally, the Durbin-Watson test indicates no evidence of first-order autocorrelation.

However, two assumptions are not fully satisfied: constant error variance (homoscedasticity) and normality of the residuals.

Let's look at transforming the response variable, `mpg`.

```

# Define transformations
transformations = list("none" = function(x) x,
                      "sqrt" = sqrt,
                      "log10" = function(x) log10(x),
                      "inverse" = function(x) 1/x)

# Initialize results list
transformed_models = list()
resid_diagnostics = list()

for(name in names(transformations)) {

  # Apply transformation to response in train_data_clean
  train_data_clean$mpg_trans = transformations[[name]](train_data_clean$mpg)
}

```

```

# Fit model with transformed response
mod = lm(mpg_trans ~ horsepower_c +
          weight_c +
          weight_c:horsepower_c +
          model_year_sub,
          data = train_data_clean)

transformed_models[[name]] = mod

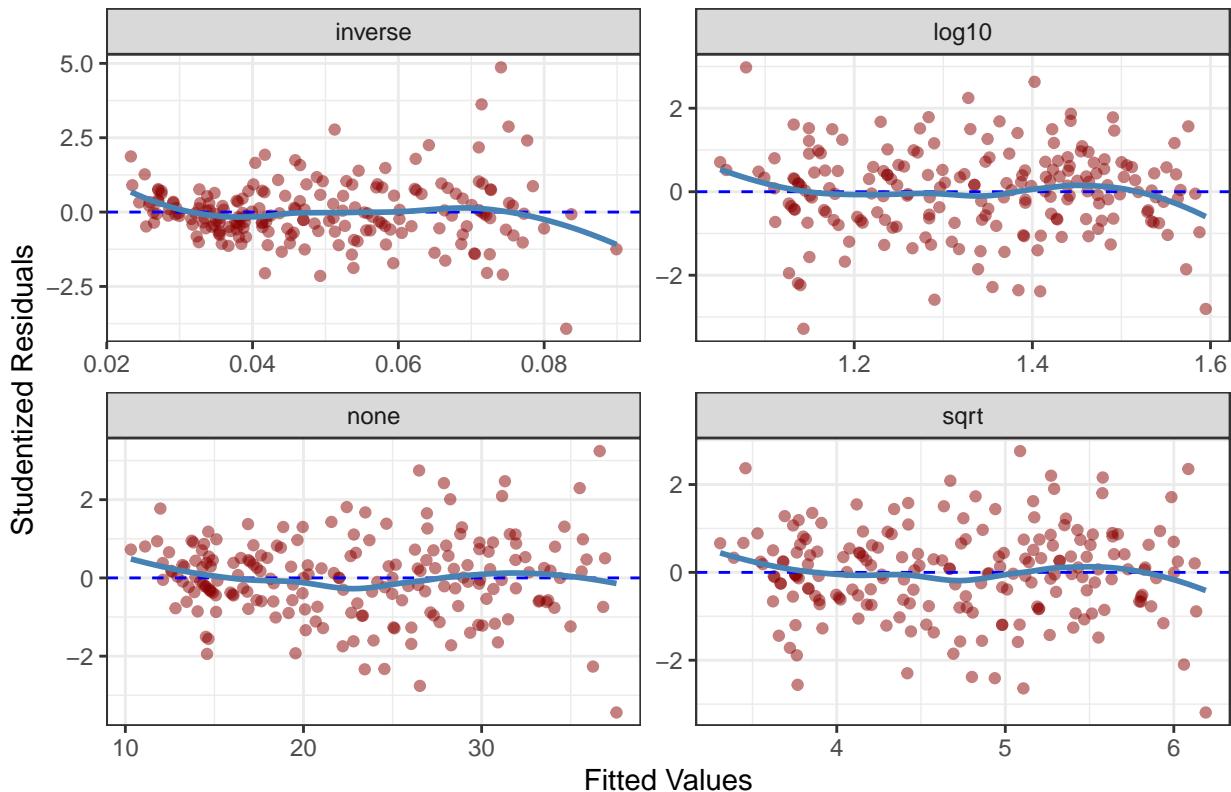
# Store studentized residuals and fitted values
resid_diagnostics[[name]] = data.frame(transformation = name,
                                       stud_resid = rstudent(mod),
                                       fitted = fitted(mod))
}

# Combine diagnostics for plotting
resid_plot_df = bind_rows(resid_diagnostics)

# Studentized residuals vs Fitted plot
ggplot(resid_plot_df, aes(x = fitted, y = stud_resid)) +
  geom_point(alpha = 0.5, color = "darkred") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "blue") +
  geom_smooth(method = "loess", se = FALSE, color = "steelblue") +
  facet_wrap(~transformation, scales = "free") +
  labs(title = "Studentized Residuals vs Fitted Values for Different Transformations",
       x = "Fitted Values",
       y = "Studentized Residuals") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

```

Studentized Residuals vs Fitted Values for Different Transformations



Both the log-base-10 and square-root transformations improve model behavior. The square-root transformation appears to perform slightly better at the low-mpg end of the data. Since it also tends to be more interpretable than the log-base-10 transformation, we will proceed with the square-root transformation for the response variable.

The Final Model

```
# Apply square-root transformation to mpg
train_data_clean$mpg_sqrt = sqrt(train_data_clean$mpg)

# Refit the best-subsets model using the transformed response
best_subset_model_sqrt = lm(mpg_sqrt ~ horsepower_c +
                           weight_c +
                           weight_c:horsepower_c +
                           model_year_sub,
                           data = train_data_clean)

# View summary
summary(best_subset_model_sqrt)

## 
## Call:
## lm(formula = mpg_sqrt ~ horsepower_c + weight_c + weight_c:horsepower_c +
##     model_year_sub, data = train_data_clean)
## 
## Residuals:
##       Min        1Q    Median        3Q       Max 
## -3.000000 -0.500000  0.000000  0.500000  3.000000 
```

```

## -0.73074139 -0.15461073  0.01146181  0.15118016  0.63037225
##
## Coefficients:
##                               Estimate     Std. Error    t value   Pr(>|t|)    
## (Intercept)            4.540230e+00  2.807580e-02 161.71332 < 2.22e-16 ***
## horsepower_c          -5.226556e-03  9.570805e-04 -5.46094  1.4859e-07 ***
## weight_c               -5.276850e-04  3.906434e-05 -13.50810 < 2.22e-16 ***
## model_year_sub73      -3.072424e-01  5.998816e-02 -5.12172  7.4635e-07 ***
## model_year_sub79       4.119404e-01  6.811171e-02  6.04801  7.7543e-09 ***
## model_year_sub80       6.624893e-01  7.388372e-02  8.96665  3.0621e-16 ***
## model_year_sub81       5.293750e-01  7.115352e-02  7.43990  3.4924e-12 ***
## model_year_sub82       6.696970e-01  6.947495e-02  9.63940 < 2.22e-16 ***
## horsepower_c:weight_c  3.479057e-06  5.737560e-07  6.06365  7.1489e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2457119 on 188 degrees of freedom
## Multiple R-squared:  0.908029, Adjusted R-squared:  0.9041153 
## F-statistic: 232.0152 on 8 and 188 DF, p-value: < 2.2204e-16

# Create a data frame with residuals and observation sequence
train_data_clean = train_data_clean %>%
  mutate(obs_seq = 1:length(best_subset_model_sqrt$residuals),
         residuals = best_subset_model_sqrt$residuals,
         fitted = fitted(best_subset_model_sqrt),
         stud_resid = rstudent(best_subset_model_sqrt))

```

All diagnostic plots for the re-fitted regression model, following the removal of an outlier and the transformation of mpg is provided below.

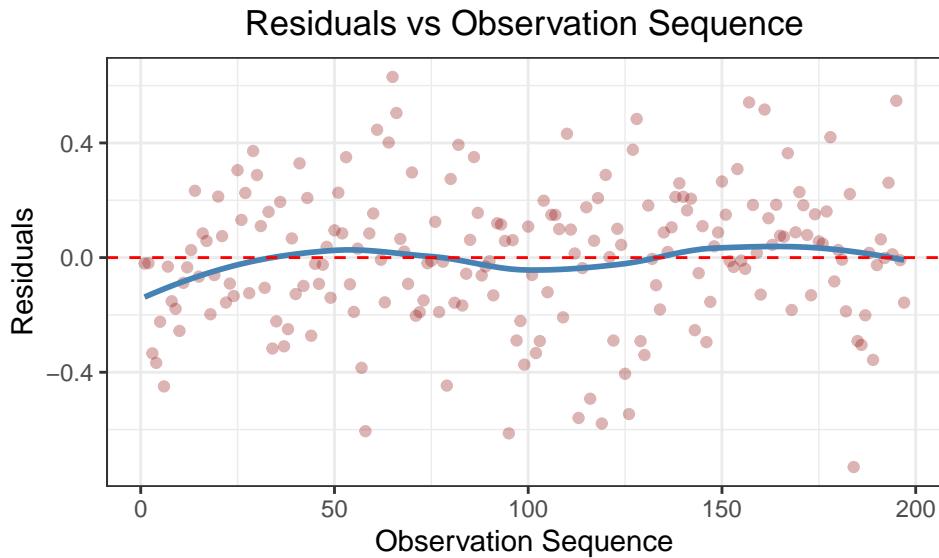
Independence, is by far the most important assumption to check.

```

# Plot residuals vs observation sequence
resid_vs_seq_final = ggplot(train_data_clean, aes(x = obs_seq, y = residuals)) +
  geom_point(alpha = 0.3, color = "darkred") +
  geom_smooth(method = "loess", color = "steelblue", se = FALSE, size = 1) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Residuals vs Observation Sequence",
       x = "Observation Sequence",
       y = "Residuals") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

resid_vs_seq_final

```



```
# Save the plot as a PNG
ggsave(filename = "figures/resid_vs_seq_final.png",
       plot = resid_vs_seq_final,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)
```

This plot allows us to check for independence of errors. The residuals appear to be randomly scattered without visible trends or cycles, indicating that first-order autocorrelation is unlikely to be present. In other words, there is no visual evidence of time-based correlation or dependence in the errors. This supports the assumption of independence of residuals, which is a key requirement for valid statistical inference in linear regression.

To formally evaluate the independence of the errors, we conduct a Durbin–Watson test for first-order autocorrelation. The test is performed at the $\alpha = 0.05$ significance level. The null hypothesis assumes that the residuals are uncorrelated ($\rho = 0$), while the alternative hypothesis allows for the presence of first-order autocorrelation ($\rho \neq 0$). Formally, we have

$$H_0 : \rho = 0$$

$$H_A : \rho \neq 0$$

```
dw_result = dwtest(best_subset_model_sqrt, alternative = "two.sided")

dw_result

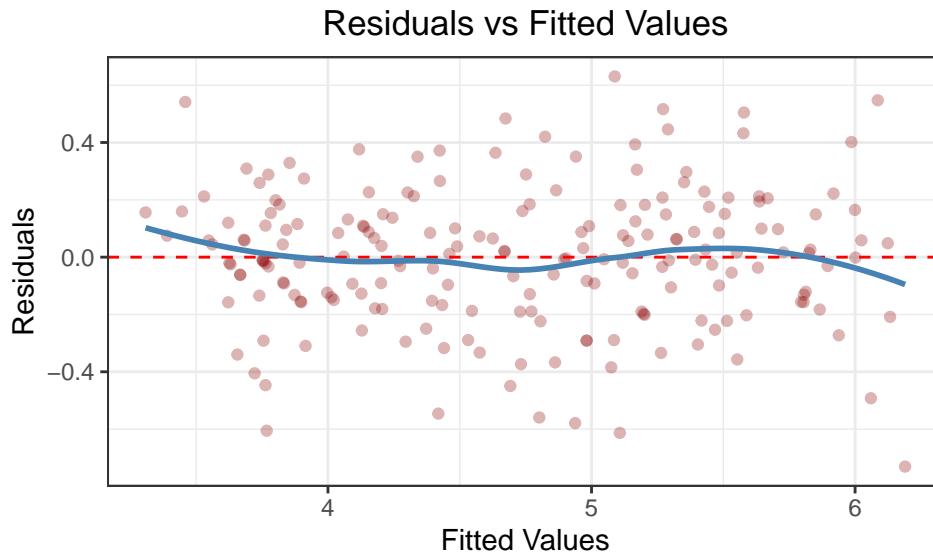
## 
##  Durbin-Watson test
## 
## data: best_subset_model_sqrt
## DW = 1.8022603, p-value = 0.1455902
## alternative hypothesis: true autocorrelation is not 0
```

Based on the resulting p -value, there is insufficient evidence to reject the null hypothesis at the $\alpha = 0.05$ level. This indicates that there is no significant first-order autocorrelation in the residuals, which is consistent with the residuals versus observation sequence plot. Overall, this supports the assumption that the errors in our simple linear regression model are independent, and no temporal or cyclical structure remains unaccounted for in the data.

Next, we examine the residuals versus fitted values plot.

```
# Residuals vs Predictor
resid_vs_mpg_final = ggplot(train_data_clean, aes(x = fitted, y = residuals)) +
  geom_point(alpha = 0.3, color = "darkred") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  geom_smooth(method = "loess", se = FALSE, color = "steelblue") +
  labs(title = "Residuals vs Fitted Values",
       x = "Fitted Values",
       y = "Residuals") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

resid_vs_mpg_final
```

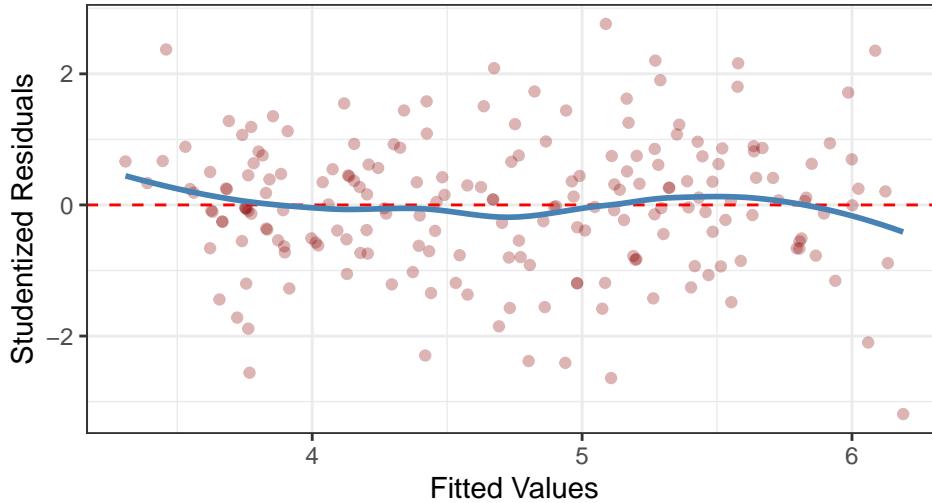


```
# Save the plot as a PNG
ggsave(filename = "figures/resid_vs_mpg_final.png",
       plot = resid_vs_mpg_final,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)

# Studentized residuals vs Fitted Values
stud_resid_plot_final = ggplot(train_data_clean, aes(x = fitted, y = stud_resid)) +
  geom_point(alpha = 0.3, color = "darkred") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  geom_smooth(method = "loess", se = FALSE, color = "steelblue") +
  labs(title = "Studentized Residuals vs Fitted Values",
       x = "Fitted Values",
       y = "Studentized Residuals") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

stud_resid_plot_final
```

Studentized Residuals vs Fitted Values



```
# Save the plot as a PNG
ggsave(filename = "figures/stud_resid_plot_final.png",
       plot = stud_resid_plot_final,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)
```

The residuals versus fitted values plot shows residuals randomly scattered around zero, with no obvious systematic patterns or curvature. The relatively flat LOESS curve supports the assumption of linearity. Additionally, there is no clear cone or funnel shape, indicating that heteroscedasticity does not appear to be a concern.

To further assess the constant variance (homoscedasticity) assumption of the regression model, we employ the Brown–Forsythe test, a robust modification of Levene’s test that uses deviations from the median rather than the mean and does not rely on the normality of the residuals. The test evaluates whether the variability of the residuals differs across groups. Significant differences in residual variability suggest that the assumption of constant variance may be violated.

In our dataset, observations are naturally grouped by model.year. If the residual variance changes across these year groups, the test will detect it as a statistically significant difference.

Using a significance level of $\alpha = 0.05$, the hypotheses for the Brown–Forsythe test are:

$$H_0 : \sigma_{73}^2 = \sigma_{79}^2 = \sigma_{80}^2 = \sigma_{81}^2 = \sigma_{82}^2 \quad \text{Residual variance is equal between groups (homoscedasticity)}$$

$$H_A : \sigma_i^2 \neq \sigma_j^2 \quad \text{for some } i \neq j \quad \text{Residual variance differs between groups (heteroscedasticity)}$$

```
# Brown-Forsythe test (Levene test using median)
bf_test = leveneTest(residuals(best_subset_model_sqrt) ~ model_year_sub,
                      data = train_data_clean,
                      center = median)

bf_test

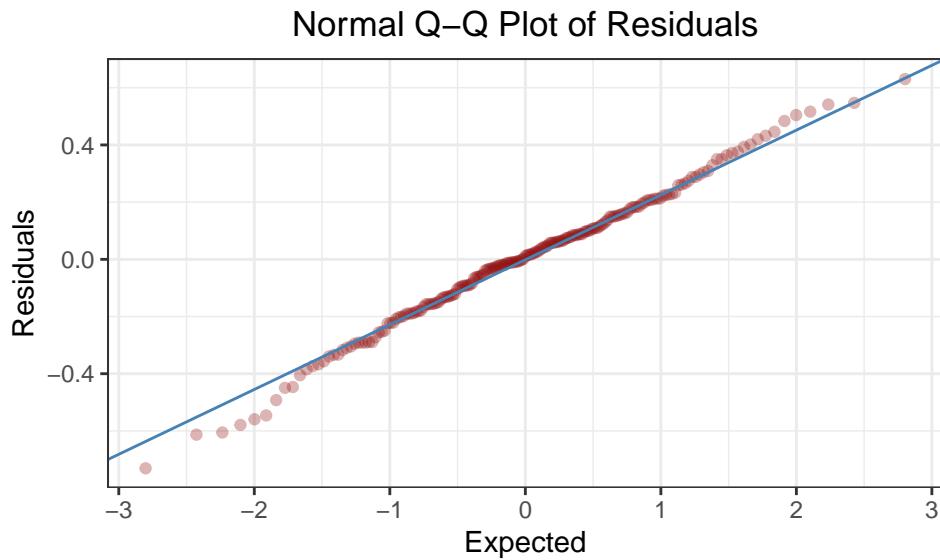
## Levene's Test for Homogeneity of Variance (center = median)
##          Df F value Pr(>F)
## group    5  0.74212 0.5928
##      191
```

The Brown-Forsythe test returned a p -value larger than 0.05. Therefore, we fail to reject the null hypothesis, and we conclude that there is no statistical evidence to suggest that the error variance differs between year groups. This supports the our move to transform the response variable, `mpg`.

Finally, we examine the Normal QQ plot.

```
# Normal Q-Q Plot of Residuals
resid_qq_final = ggplot(train_data_clean, aes(sample = residuals)) +
  stat_qq(alpha = 0.3, color = "darkred") +
  stat_qq_line(color = "steelblue") +
  labs(title = "Normal Q-Q Plot of Residuals",
       x = "Expected",
       y = "Residuals") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5))

resid_qq_final
```



```
# Save the plot as a PNG
ggsave(filename = "figures/resid_qq_final.png",
       plot = resid_qq_final,
       width = 5,
       height = 3,
       units = "in",
       dpi = 300)
```

If the residuals were approximately normally distributed, we would expect the points to lie close to the reference line. This now appears to be the case.

To formally assess the normality of the residuals, we now apply the Shapiro-Wilk test. We conduct the test at the significance level $\alpha = 0.05$. The Shapiro-Wilk test formally evaluates whether the residuals are normally distributed. The hypotheses for the test are:

$$H_0 : \text{The residuals are normally distributed}$$

$$H_A : \text{The residuals are not normally distributed}$$

```
# Shapiro-Wilk test for normality
shapiro_test = shapiro.test(train_data_clean$residuals)
```

```
# Print results
shapiro_test

##
## Shapiro-Wilk normality test
##
## data: train_data_clean$residuals
## W = 0.99335898, p-value = 0.5205822
```

Since the p -value is greater than 0.05, we fail to reject the null hypothesis. This indicates that the residuals are approximately normally distributed.

Let's now examine multicollinearity in the model.

```
# compute VIFs
vif(best_subset_model_sqrt)

##          GVIF Df GVIF^(1/(2*Df))
## horsepower_c      5.071847932  1     2.252076360
## weight_c          4.007861448  1     2.001964397
## model_year_sub    1.212700926  5     1.019472161
## horsepower_c:weight_c 1.784301626  1     1.335777536
```

The generalized VIF output shows that once degrees of freedom are accounted for, there is no meaningful collinearity concerns. The highest adjusted VIFs occurs for horsepower, but that is a value of 2.

Model Validation

Now, let's take a look at how the model performs on the validation dataset.

```
# Create a factor for just the years of interest, with 70 as reference
valid_data_sub = valid_data %>%
  mutate(model_year_sub = ifelse(model.year %in% c(73, 79, 80, 81, 82),
                                 as.character(model.year), "70")) # 70 = reference

# Make sure factor levels match training set
valid_data_sub$model_year_sub = factor(valid_data_sub$model_year_sub,
                                         levels = c("70", "73", "79", "80", "81", "82"))

# transform the response
valid_data_sub$mpg_sqrt = sqrt(valid_data_sub$mpg)

# Fit the best subset model on validation set
valid_subset_model = lm(mpg_sqrt ~ horsepower_c +
                        weight_c +
                        weight_c:horsepower_c +
                        model_year_sub,
                        data = valid_data_sub)

summary(valid_subset_model)

##
## Call:
## lm(formula = mpg_sqrt ~ horsepower_c + weight_c + weight_c:horsepower_c +
##     model_year_sub, data = valid_data_sub)
##
```

```

## Residuals:
##      Min       1Q   Median      3Q     Max
## -0.77995014 -0.17963789  0.02707485  0.17394077  0.99040246
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)          4.495799e+00 3.303706e-02 136.08351 < 2.22e-16 ***
## horsepower_c        -7.361962e-03 1.431112e-03 -5.14422 6.8125e-07 ***
## weight_c            -5.701237e-04 5.592801e-05 -10.19388 < 2.22e-16 ***
## model_year_sub73    -1.214717e-01 7.394453e-02 -1.64274 0.10214
## model_year_sub79    4.061128e-01 8.375339e-02  4.84891 2.6229e-06 ***
## model_year_sub80    7.330163e-01 8.858378e-02  8.27484 2.4867e-14 ***
## model_year_sub81    4.442831e-01 8.495550e-02  5.22960 4.5668e-07 ***
## model_year_sub82    5.594965e-01 8.184231e-02  6.83627 1.1440e-10 ***
## horsepower_c:weight_c 4.510159e-06 7.498111e-07  6.01506 9.4180e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2939819 on 185 degrees of freedom
## Multiple R-squared:  0.870554, Adjusted R-squared:  0.8649563
## F-statistic: 155.5209 on 8 and 185 DF, p-value: < 2.2204e-16

# Extract coefficients and standard errors
train_coef = coef(best_subset_model_sqrt)
train_se   = summary(best_subset_model_sqrt)$coefficients[, "Std. Error"]

val_coef   = coef(valid_subset_model)
val_se     = summary(valid_subset_model)$coefficients[, "Std. Error"]

# Ensure all coefficients are matched by name
coef_names = names(train_coef)

# Create comparison table
coef_comparison = data.frame(Coefficient = coef_names,
                               Train_Estimate = train_coef[coef_names],
                               Train_SE = train_se[coef_names],
                               Validation_Estimate = val_coef[coef_names],
                               Validation_SE = val_se[coef_names])

# Print
print(coef_comparison, row.names = FALSE)

##           Coefficient Train_Estimate Train_SE Validation_Estimate
## (Intercept) 4.540230237e+00 2.807579639e-02 4.495799308e+00
## horsepower_c -5.226556449e-03 9.570804831e-04 -7.361962208e-03
## weight_c     -5.276849786e-04 3.906434269e-05 -5.701237141e-04
## model_year_sub73 -3.072423893e-01 5.998816191e-02 -1.214716970e-01
## model_year_sub79  4.119404120e-01 6.811170846e-02  4.061128064e-01
## model_year_sub80  6.624892865e-01 7.388372304e-02  7.330162707e-01
## model_year_sub81  5.293750321e-01 7.115351575e-02  4.442830959e-01
## model_year_sub82  6.696970187e-01 6.947495361e-02  5.594964697e-01
## horsepower_c:weight_c 3.479056736e-06 5.737559855e-07 4.510159195e-06
## Validation_SE
## 3.303706169e-02
## 1.431112311e-03

```

```
## 5.592801424e-05
## 7.394452690e-02
## 8.375339271e-02
## 8.858378201e-02
## 8.495549508e-02
## 8.184230957e-02
## 7.498110910e-07
```

The model fitted to the validation dataset produces regression coefficient estimates that are very consistent with those from the training set. All coefficients maintain the same sign and are similar in magnitude, indicating that the relationships identified by the model generalize well to new data.

Overall, the model appears stable for both the main continuous predictors (horsepower and weight) and the model year indicators, as well as the interaction term, suggesting reliable performance across both the training and validation datasets.

Additionally, the MSE values for the two datasets are close in magnitude (0.0604 for the model-building data and 0.0864 for the validation sample). Since the validation MSE is only slightly larger this indicates that the model's predictive accuracy does not deteriorate substantially when applied to an independent sample. The coefficients of multiple determination are also comparable across samples.

Together, these comparisons suggest that the regression model is stable, shows no evidence of overfitting, and demonstrates good external predictive ability when applied to new applicants.