A Modern Approach to Regression and Statistical Inference

AUTHOR
Mutwiri, Ian

PUBLISHED
December 10, 2023

Introduction

The R programming language is really powerful in performing statistical analysis. From outset, it was purposely created for analysis and visualization of data. I am of the opinion that `Nothing beats R in data wrangling and analysis. Give it to Python for Machine Learning`. However, doing statistical analysis the traditional way, i.e. using base R functions, can be quite cumbersome and verbose at the same time. With the help of new, opinionated packages, one can streamline this process and make is as easy as it can get. In this guide I will cover hypothesis testing and regression analysis where the main goal is to test for significance of coefficients and fitted models. I will also demonstrate how to perform statistical tests quickly and efficiently with the `infer` package. While the results from this exercise might not be meaningful or generalizable, this guide will illustrate how to better perform statistical analysis in R and how one can generalize this approach to other settings.

Linear Regression

In linear regression, one seeks to understand the relationship between two or more variables; the independent variable(s), and some other variable of interest called the dependent variable. We do `Simple Regression` if there is one independent variable and do `Multiple Regression` if there are two or more independent variables. In both cases we want to understand how changes in the independent variable(s) affect the dependent variables. Linear regression, as implemented in R, applies the `maximum likelihood` algorithm to extract these factors. These factors are referred to as the coefficients of our variables. They show the magnitude and the direction of the effect of changes in independent variables on the dependent variables.

One should however note that these coefficients are not the true coefficients because in most cases we do not have the exact population on which we are regressing on. In almost all cases, researchers obtain samples from the data and use them to extract estimates which they generalize to the entire population. One should be careful when making claims based on these estimates. This is where `significance testing` comes into play. Significance testing involves performing checks on these estimates to ensure that the claims we make about the population are credible and based on evidence from the data. This is because in sampling, there is a lot of randomness. It could be that whatever results we obtain on a given sample would be quite different from results we would have obtained from another random sample from the same population. Significance testing helps researchers avoid this uncertainty and make conclusions from studies and other related experiments.

Data

This guide will based on the `Car` dataset obtained from Kaggle. This data contains variables that are relevant in explaining the selling price of a car. It also suitable for this setting as it contains continuous and categorical features which are perfect for hypothesis testing.

Some Exploratory Analysis

It is helpful to put the data in the right format and perform some exploratory analysis to get a sense of how the data looks like. We can load the data first and then do some manipulation to put it into the right format.

```r
library(tidyverse)

cars <- read_csv('cars.csv')

cars <- cars %>%
  janitor::clean_names() %>% #set column names lower case
  #select the relevant columns in this setting
  select(
    price, year, kilometer, fuel_type, transmission, engine,drivetrain,
    seating_capacity, fuel_tank_capacity
  ) %>%
  rename(
    'sitting_capacity' = seating_capacity
  ) %>%
  #create column for years since the car was manufactured(age)
  mutate(
    age = year(today()) - year,
    engine = str_replace_all(engine, " cc",''), #clean the columns
    engine = as.numeric(engine), #transform to numeric
    price = price/1e3
  ) %>%
  select(-year) %>% #year is nolonger relevant and has been replaced by age
  relocate(age, .before = kilometer) %>%
  #transform the following columns to categorical
  mutate(
    across(c(fuel_type, transmission, drivetrain), as_factor)
  ) %>%
  #drop missing values as they raise problems in regression
  drop_na()

# set default theme for all plots
theme_set(theme_minimal())
```

The previous steps have been chained together using the pipe operator, `%>%`. The tibble operator takes as input data(`tibble`), which is the output of some previous step and also passes the tibble to the next step.

We can then do a quick summary of the structure of the data after these transformations.

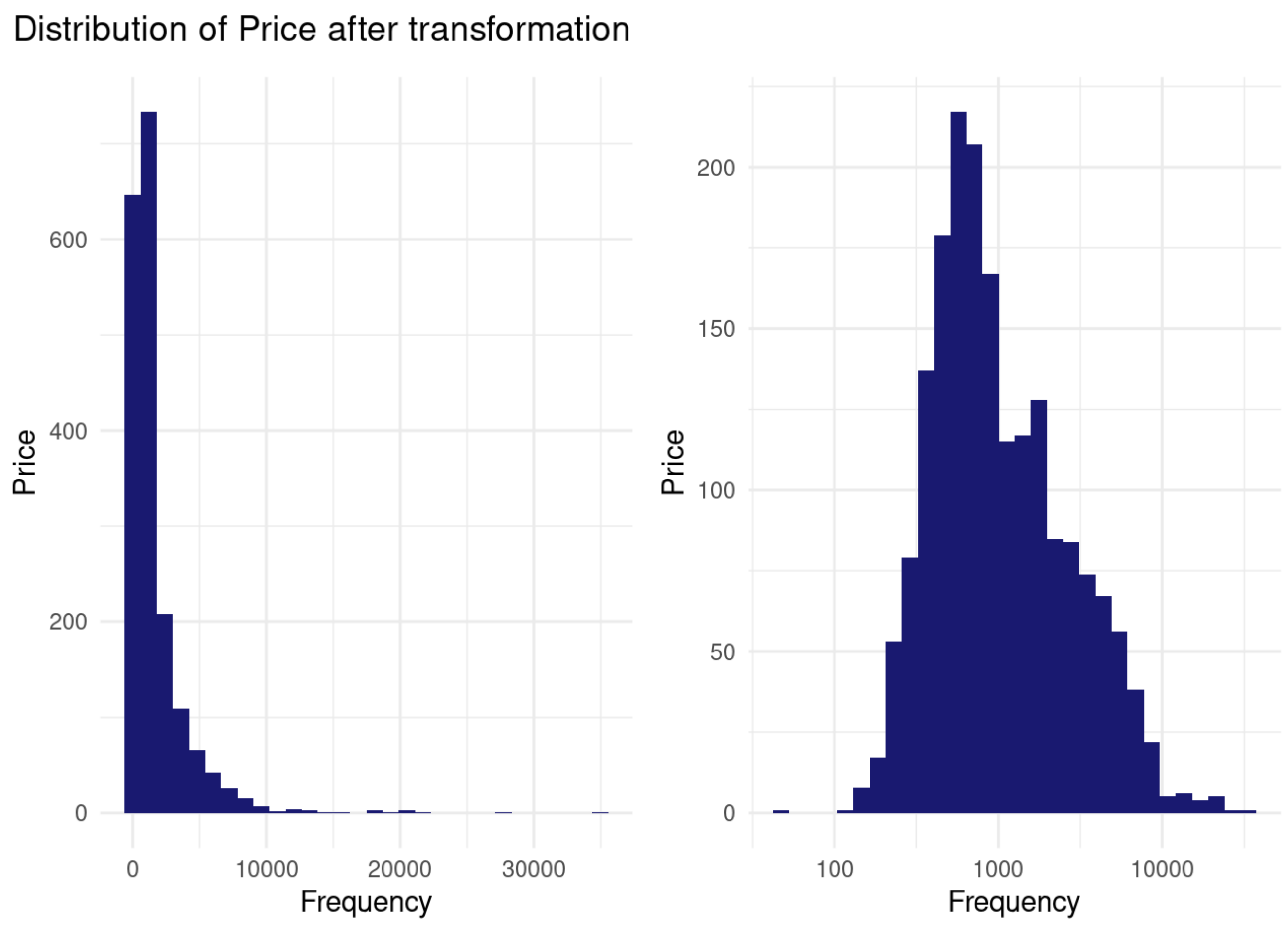
```r
glimpse(cars)
```

```
Rows: 1,874
Columns: 9
```

```
$ price             <dbl> 505.000, 450.000, 220.000, 799.000, 1950.000, 675.0…
$ age               <dbl> 6, 9, 12, 4, 5, 6, 6, 6, 8, 6, 5, 4, 4, 6, 7, 4, 5,…
$ kilometer         <dbl> 87150, 75000, 67000, 37500, 69000, 73315, 75000, 56…
$ fuel_type         <fct> Petrol, Diesel, Petrol, Petrol, Diesel, Petrol, Die…
$ transmission      <fct> Manual, Manual, Manual, Manual, Manual, Manual, Aut…
$ engine            <dbl> 1198, 1248, 1197, 1197, 2393, 1373, 1995, 1798, 146…
$ drivetrain        <fct> FWD, FWD, FWD, FWD, RWD, FWD, AWD, FWD, FWD, FWD, F…
$ sitting_capacity  <dbl> 5, 5, 5, 5, 7, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, …
$ fuel_tank_capacity <dbl> 35, 42, 35, 37, 55, 43, 51, 50, 50, 45, 28, 43, 50,…
```

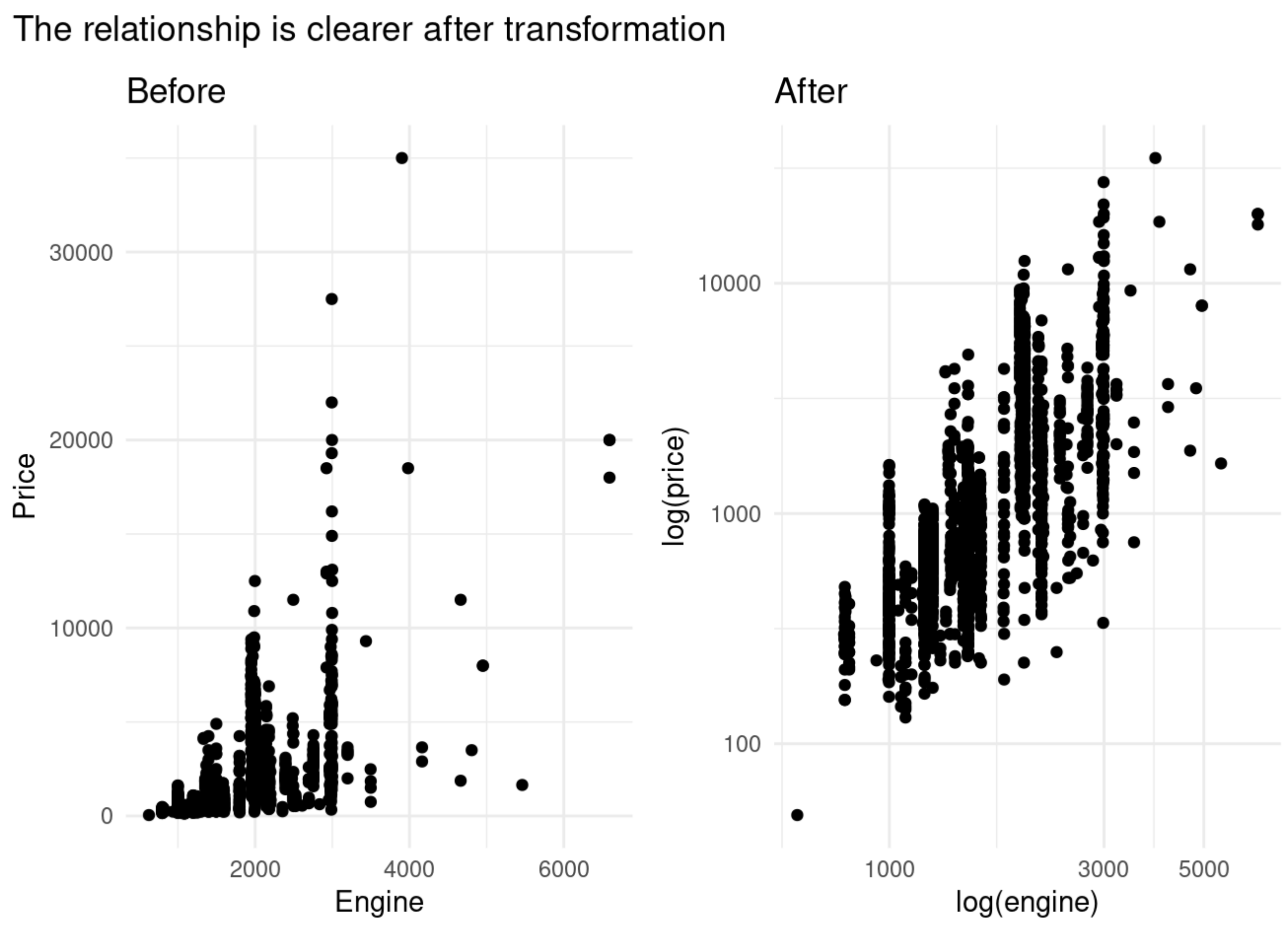
We observe that our variable of interest, `price`, is highly skewed but with a log transformation it becomes more evenly distribution.

```r
library(patchwork)

original_price <- cars %>%
  ggplot(aes(price))+
  geom_histogram(fill='midnightblue')+
  labs(x='Frequency',y='Price')


trans_price <- cars %>%
  ggplot(aes(price))+
  geom_histogram(fill='midnightblue')+
  scale_x_log10()+
  labs(x='Frequency',y='Price')


(original_price + trans_price)+
  plot_annotation('Distribution of Price after transformation')
```



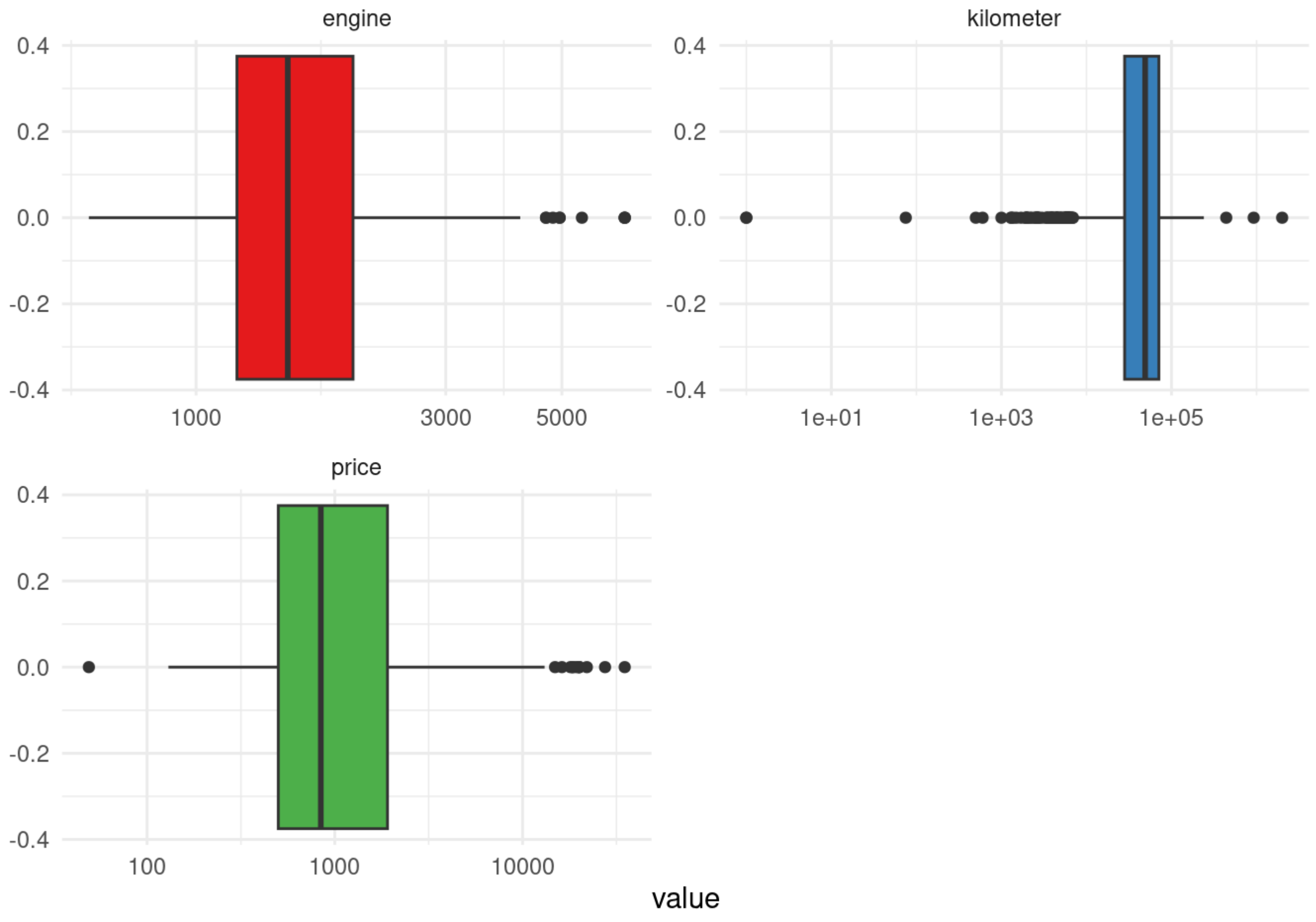
The log transformation pulls data that is further away closer and data that are clustered further apart.It is a very useful transformation especially in machine learning models where skewness and class imbalance limit their performance.Skewness can also make it hard or impossible to see the relationships between variables. A log transformation on either or both of these variables can applied. The other most commonly used transformation is the `square root` transformation. As shown below, now it's easier to see that there is a positive relationship between `engine` size and `price` after transforming both variables.

```
original <- cars %>%
  ggplot(aes(engine,price)) +
  geom_point()+
  labs(x='Engine', y = 'Price', title = 'Before')


transformed <- cars %>%
  ggplot(aes(engine,price)) +
  geom_point()+
  scale_x_log10()+
  scale_y_log10()+
  labs(x = 'log(engine)', y= 'log(price)', title = 'After')


(original + transformed)+
  plot_annotation(title = 'The relationship is clearer after transformation')
```



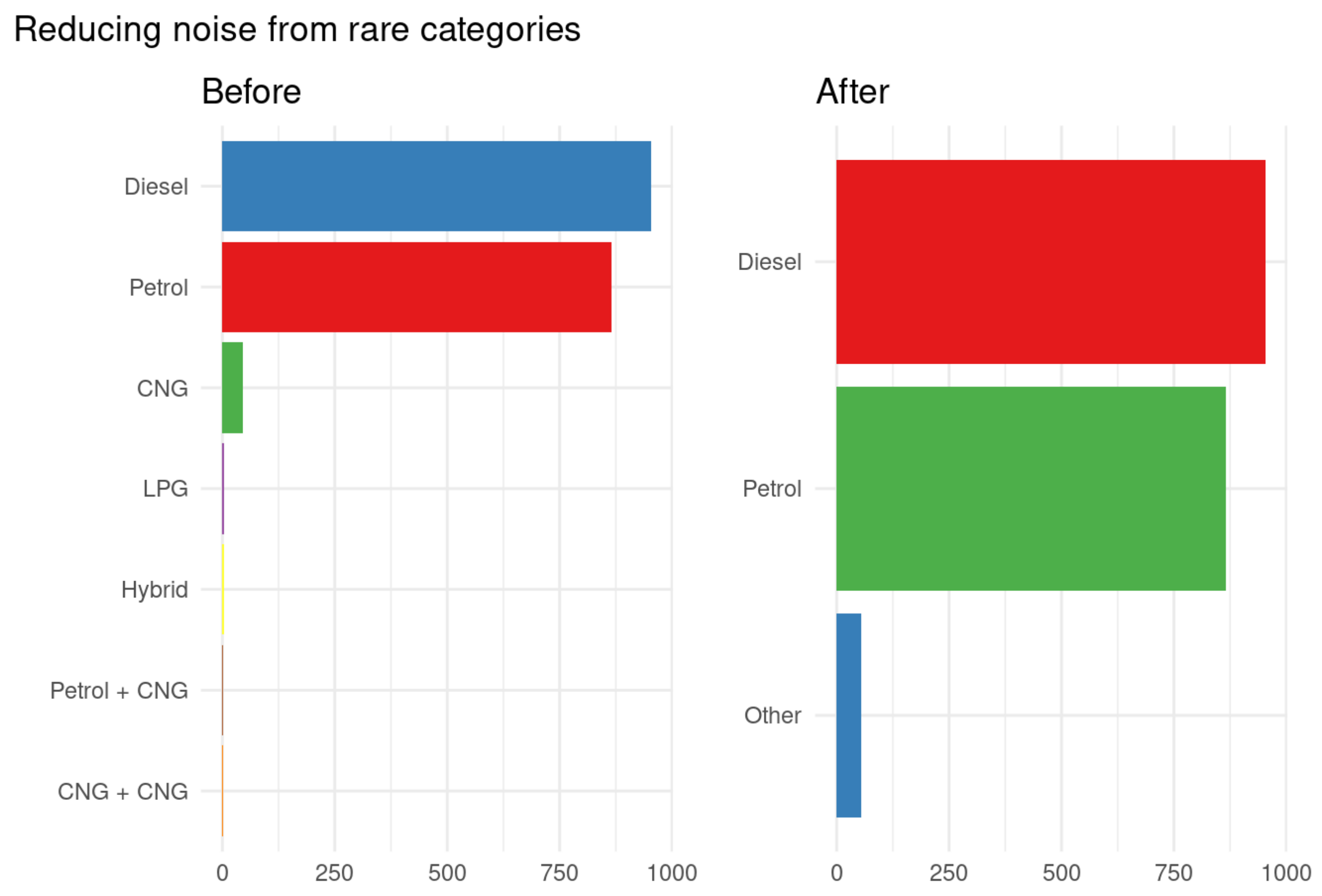
We can also use box plots to visualize the distribution of a few key variables. You may note that many of these variables are highly skewed(sparse). Therefore, a transformation is necessary.

```
cars %>%
  select(price,kilometer,engine) %>%
  pivot_longer(cols = everything()) %>%
  group_by(name) %>%
  ggplot(aes(value,fill=name))+
  geom_boxplot(show.legend = FALSE)+
  scale_x_log10()+
  scale_fill_brewer(palette = 'Set1')+
  facet_wrap(facets = 'name', scales = 'free', nrow = 2)
```



We see that most of the cars use petrol and diesel. Other categories have very low frequencies and will likely add more noise to the data. We can resolve this by collecting these into a `Other` group.

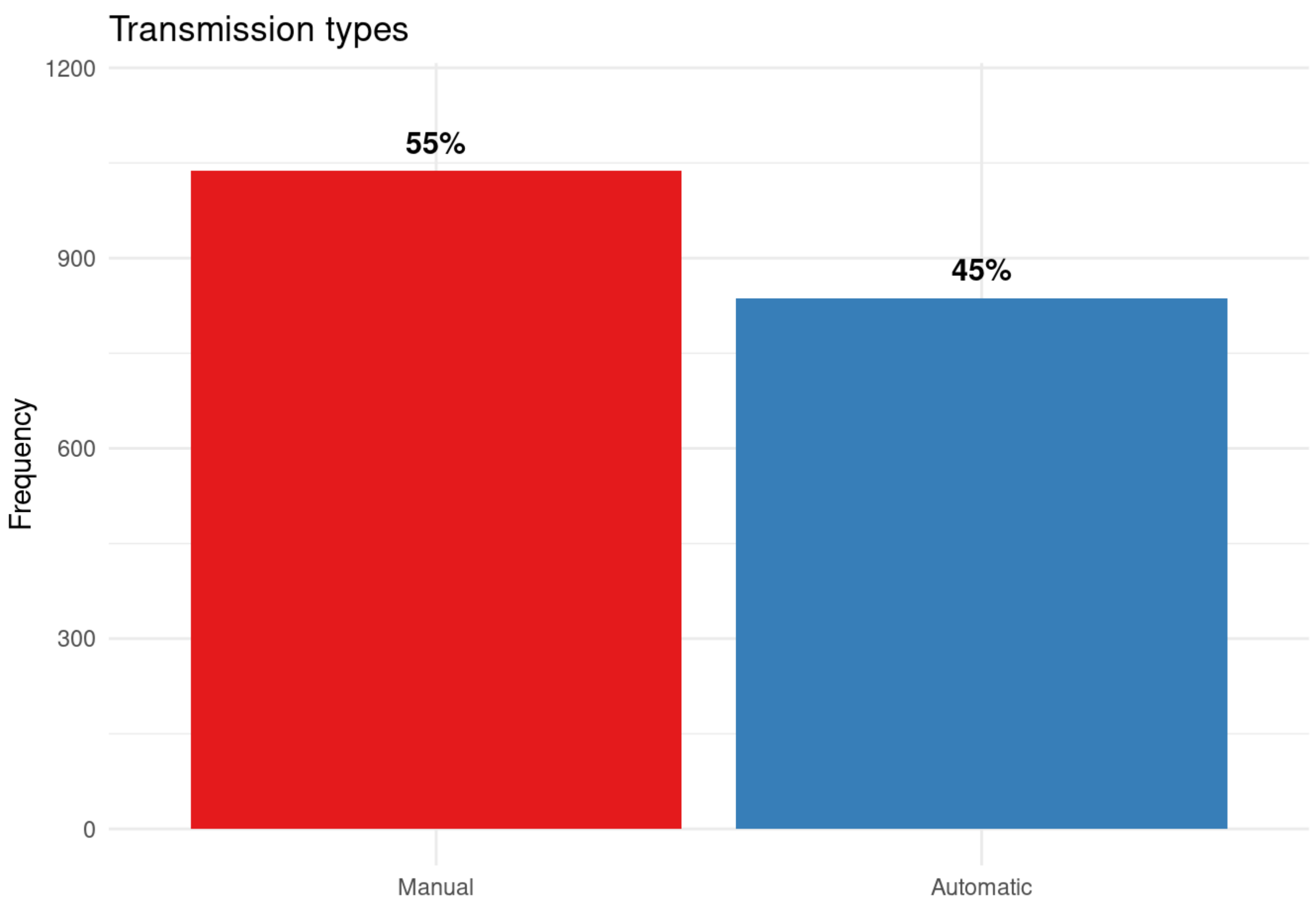
```r
all_types <- cars %>%
  count(fuel_type) %>%
  ggplot(aes(y=fct_reorder(fuel_type,n),x=n, fill = fuel_type))+
  geom_col(show.legend = F)+
  labs(x='',y='',title = 'Before')+
  scale_fill_brewer(palette = 'Set1')


reduced <- cars %>%
  mutate(
    fuel_type = case_when(
      fuel_type %in% c('Petrol', 'Diesel') ~ fuel_type,
      .default = 'Other')
  ) %>%
  count(fuel_type) %>%
  ggplot(aes(y=fct_reorder(fuel_type,n),x=n, fill = fuel_type))+
  geom_col(show.legend = F)+
  labs(x='',y='',title = 'After')+
  scale_fill_brewer(palette = 'Set1')


(all_types + reduced) +
  plot_annotation(title = "Reducing noise from rare categories")
```



```r
cars <- cars %>%
  mutate(
    fuel_type = case_when(
      fuel_type %in% c('Petrol', 'Diesel') ~ fuel_type,
      .default = 'Other')
  )
```

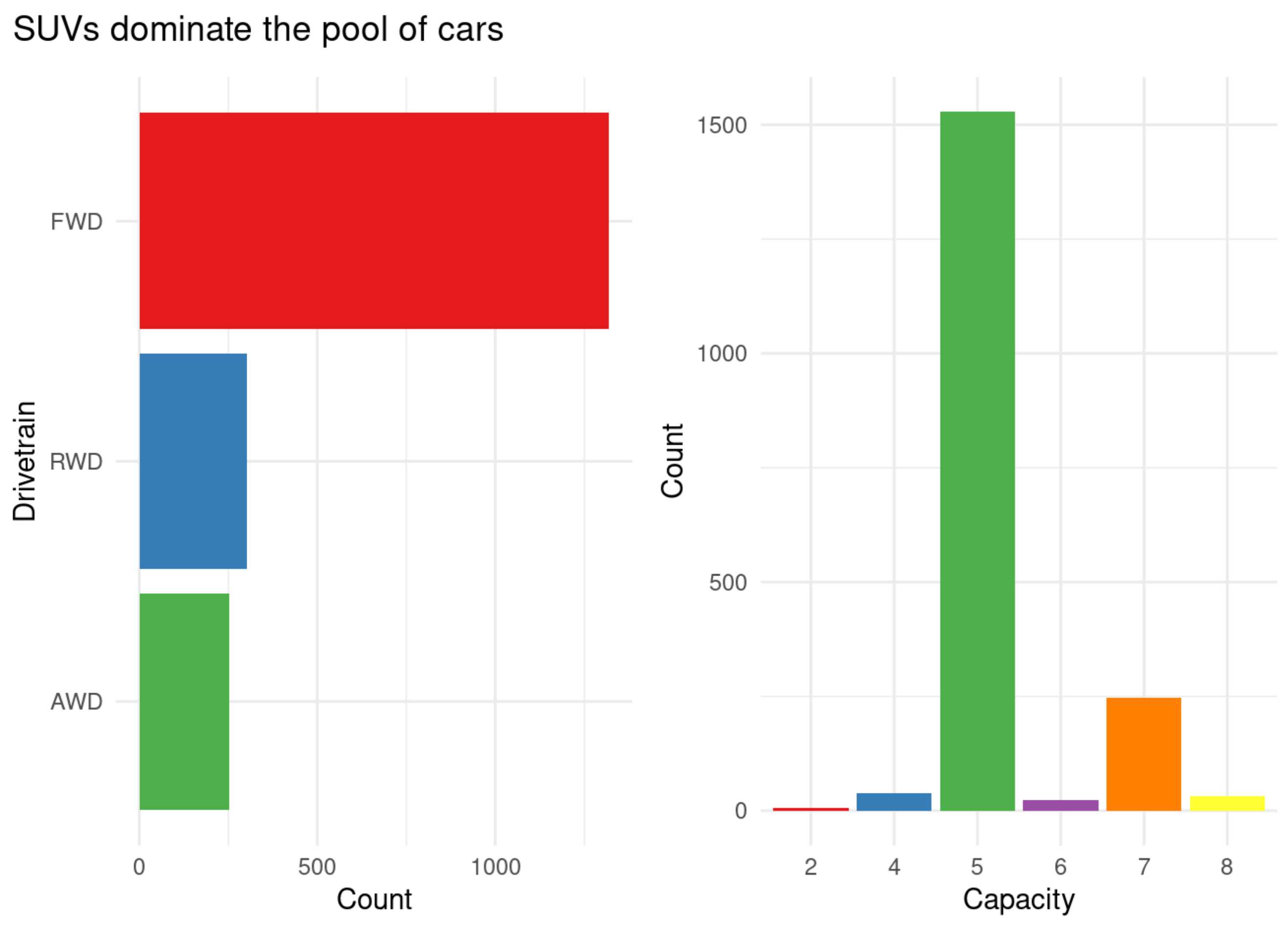
Most cars have manual transmission but by eye-balling, the difference in frequency between cars with automatic transmission is not much. We will verify this later by conducting a formal statistical test.

```r
cars %>%
  count(transmission) %>%
  mutate(
    perc = {scales::percent(n/sum(n))}
    ) %>%
  ggplot(aes(transmission,n,fill=transmission))+
  geom_col(show.legend = F)+
  geom_text(aes(x = transmission, y = n,label = perc, fontface = 'bold'), vjust = -.8,
  ylim(c(0,1150))+
  labs(y='Frequency',x='', title = 'Transmission types')+
  scale_fill_brewer(palette = 'Set1')
```



A high proportion of the cars are Front-Wheel Drive cars. This could mean that most of the cars are sedans or SUVs. This is also supported by the observation that there are more 5-seater cars.

```r
drivetrain_plot <- cars %>%
  count(drivetrain) %>%
  ggplot(aes(n,fct_reorder(drivetrain,n),fill=drivetrain))+
  geom_col(show.legend = F) +
  labs(y='Drivetrain',x='Count')+
  scale_fill_brewer(palette = 'Set1')


seater_plot <- cars %>%
  count(sitting_capacity) %>%
  mutate(sitting_capacity=as_factor(sitting_capacity)) %>%
  ggplot(aes(sitting_capacity,n,fill=sitting_capacity))+
  geom_col(show.legend = F)+
  labs(x='Capacity',y='Count')+
  scale_fill_brewer(palette = 'Set1')


drivetrain_plot + seater_plot +
  plot_annotation(
    title = 'SUVs dominate the pool of cars'
  )+
  theme(
    plot.title.position = 'plot'
  )
```

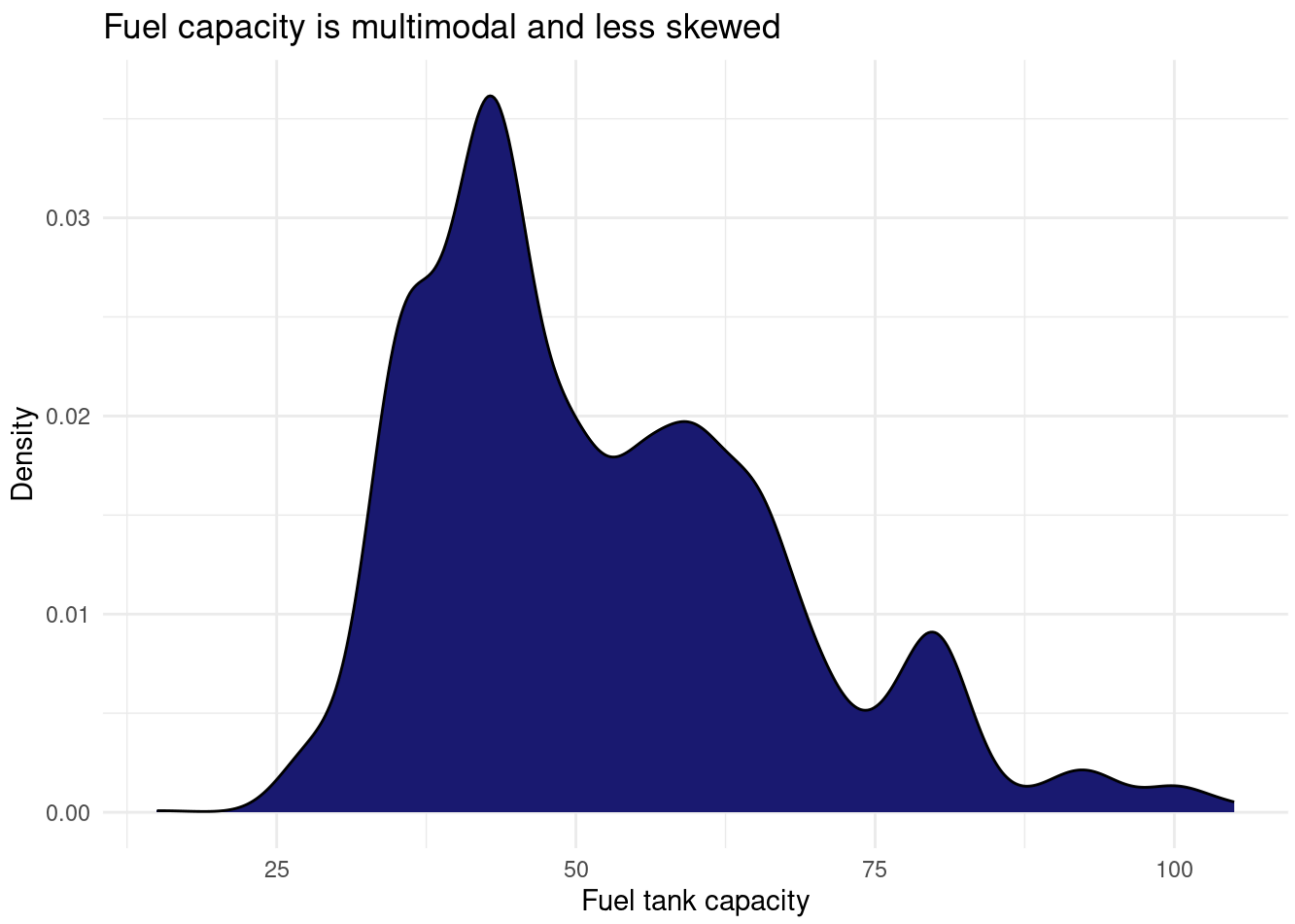


We can do the same transformation for the `sitting_capacity` variable as we did for the `fuel_type` variable.

```r
cars <- cars %>%
  mutate(
    sitting_capacity = case_when(
      sitting_capacity %in% c(5,7) ~ as.character(sitting_capacity),
      .default = 'Other'
    ),
    sitting_capacity = as_factor(sitting_capacity),
    fuel_type = as_factor(fuel_type)
  )
```

The fuel tank capacity variable appears to be multimodal. We could point the most common frequencies for different car types considered in the data.

```r
cars %>%
  ggplot(aes(fuel_tank_capacity))+
  geom_density(fill = 'midnightblue')+
  labs(x = 'Fuel tank capacity', y = 'Density',
       title = 'Fuel capacity is multimodal and less skewed')
```



Having done that bit of exploration, we can then head into regression modelling and analysis

Regression Modelling

We begin by setting a simple regression model with the price as the dependent variable and `engine`, `age` and `kilometer`s covered as the independent variables. We do this using the `lm()` function in R and the `y ~ x` interface for specifying the model. We begin with a simple model. The `summary()` function is used to summarize the results of the model including statistics like standard errors and p-values.

```r
spec_1 <- lm(price ~ age + kilometer + engine + fuel_tank_capacity,
                 data = cars)
summary(spec_1)
```

```

Call:
lm(formula = price ~ age + kilometer + engine + fuel_tank_capacity,
    data = cars)

Residuals:
    Min      1Q  Median      3Q     Max
-6806.0  -691.3  -124.0   450.9 28160.5

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)        -1.449e+03  1.664e+02  -8.706  < 2e-16 ***
```

```
age                -2.142e+02  1.327e+01 -16.141  < 2e-16 ***
kilometer          -4.248e-03  7.060e-04  -6.017 2.13e-09 ***
engine              1.671e+00  1.044e-01  16.001  < 2e-16 ***
fuel_tank_capacity  3.690e+01  4.367e+00   8.450  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1714 on 1869 degrees of freedom
Multiple R-squared:  0.5019,    Adjusted R-squared:  0.5008
F-statistic: 470.8 on 4 and 1869 DF,  p-value: < 2.2e-16
```

With this model, we see that a car's 'age' and the number of kilometers covered are negatively related with its price while the engine power and fuel capacity are positively related to its price.This p-values indicate that all variables specified in the model are highly significant in explaining the variation in the prices of the cars. Looking at the F-statistic from the table above, we can further conclude that the model is significant in explaining the relationship between `engine`, `age`, `kilometer` and `price`. The R^2 value reported at the bottom of the regression table measures the proportion of the variation in the dependent variable that can be explained by the independent variables included in the model. Despite the fact that the R^2 value is not a reliable measure of the performance of the model, it relatively low in this case implying that we could get better performance by adding more variables.

We can consider another model which includes interaction effects between the variables. Interaction effects occur when the effect of one variable on the dependent variable varies with the value of some other independent variable. We can include an interaction for `engine` and `fuel_tank_capacity` and interaction term for `age` and `kilometer` as they affect `price` in the same directions.

```r
spec_2 <- lm(price ~ age*kilometer + engine*fuel_tank_capacity,
             data = cars)
summary(spec_2)
```

```

Call:
lm(formula = price ~ age * kilometer + engine * fuel_tank_capacity,
    data = cars)

Residuals:
    Min      1Q  Median      3Q     Max
-9706.6  -656.6   -98.0   446.5 27390.5

Coefficients:
                            Estimate Std. Error t value Pr(>|t|)
(Intercept)                2.679e+03  4.349e+02   6.159 8.92e-10 ***
age                       -2.953e+02  2.256e+01 -13.088  < 2e-16 ***
kilometer                 -1.160e-02  2.442e-03  -4.751 2.18e-06 ***
engine                    -7.055e-01  2.625e-01  -2.687 0.007266 **
fuel_tank_capacity        -2.338e+01  7.427e+00  -3.149 0.001667 **
age:kilometer              1.174e-03  3.558e-04   3.301 0.000983 ***
engine:fuel_tank_capacity  3.627e-02  3.702e-03   9.798  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1669 on 1867 degrees of freedom
```

```
Multiple R-squared:  0.5283,    Adjusted R-squared:  0.5268 
F-statistic: 348.5 on 6 and 1867 DF,  p-value: < 2.2e-16
```

The interaction terms are significant and results in a higher R^2 value. It is plausible that they have an impact in the model and should be considered in our modelling.

The models above include continuous variables only. We can consider another model which includes the categorical variables in the data. Comparing the models can help us estimate the effects of these categorical variables on the dependent variable and the effect on the coefficients of the other independent variables.

```
spec_3 <- lm(price ~ age + kilometer + engine + fuel_tank_capacity+
               fuel_type + transmission + drivetrain +sitting_capacity,
             data = cars)
summary(spec_3)
```

```

Call:
lm(formula = price ~ age + kilometer + engine + fuel_tank_capacity + 
    fuel_type + transmission + drivetrain + sitting_capacity, 
    data = cars)

Residuals:
    Min      1Q  Median      3Q     Max 
-7859.9  -665.0  -137.2   424.6 27529.8 

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)    
(Intercept)           -1.357e+03  1.899e+02  -7.144 1.29e-12 ***
age                   -2.130e+02  1.281e+01 -16.629  < 2e-16 ***
kilometer             -2.497e-03  6.808e-04  -3.667 0.000252 ***
engine                 1.746e+00  1.107e-01  15.772  < 2e-16 ***
fuel_tank_capacity     3.254e+01  4.732e+00   6.877 8.32e-12 ***
fuel_typeDiesel       -5.135e+02  8.920e+01  -5.757 9.99e-09 ***
fuel_typeOther         3.639e+02  2.300e+02   1.582 0.113782    
transmissionAutomatic  3.385e+02  9.514e+01   3.557 0.000384 ***
drivetrainRWD          5.197e+02  1.410e+02   3.685 0.000235 ***
drivetrainAWD          9.439e+02  1.589e+02   5.940 3.39e-09 ***
sitting_capacity7     -1.410e+03  1.302e+02 -10.825  < 2e-16 ***
sitting_capacityOther -2.921e+02  1.845e+02  -1.583 0.113675    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1615 on 1862 degrees of freedom
Multiple R-squared:  0.5593,    Adjusted R-squared:  0.5567 
F-statistic: 214.8 on 11 and 1862 DF,  p-value: < 2.2e-16
```

From the regression results above , we see a warning on singularity for `sitting_capacity`. This is the result of multicollinearity. Multicollinearity means that one of the independent variables in the data can be written as a linear combination of of another independent variables. This makes the variable redundant as it does not provide additional information that is not yet in the model. The solution in this case is to remove the variable. This is done internally by `R` when it throws the warning. We can remove it from the formula for completeness.

```r
spec_3 <- lm(price ~ age + kilometer + engine + fuel_tank_capacity+
               fuel_type + transmission + drivetrain,
             data = cars)
```

When working with categorical independent variables, the interpretations of these coefficients is different. For example for the `drivetrain` variable, the coefficient for `drivetrainRWD` means that compared to the base class(chosen internally as `FWD`), the price of `RWD` cars is higher than the price of `FWD` cars by $519.7 on average.

Given that we observe significant categorical variables and interaction effects, we can consider another model which includes all of these features.

```r
spec_4 <- lm(price ~ age + kilometer + engine + fuel_tank_capacity+
               fuel_type + transmission + drivetrain+
               age*kilometer + engine*fuel_tank_capacity,
             data = cars)
summary(spec_4)
```

```

Call:
lm(formula = price ~ age + kilometer + engine + fuel_tank_capacity + 
    fuel_type + transmission + drivetrain + age * kilometer + 
    engine * fuel_tank_capacity, data = cars)

Residuals:
    Min      1Q  Median      3Q     Max 
-9248.1  -670.9   -50.6   495.7 27457.1 

Coefficients:
                            Estimate Std. Error t value Pr(>|t|)    
(Intercept)                2.765e+03  4.802e+02   5.759 9.88e-09 ***
age                       -2.635e+02  2.238e+01 -11.775  < 2e-16 ***
kilometer                 -8.736e-03  2.427e-03  -3.599 0.000327 ***
engine                    -7.468e-01  2.865e-01  -2.606 0.009222 ** 
fuel_tank_capacity        -2.986e+01  8.370e+00  -3.567 0.000370 ***
fuel_typeDiesel           -2.966e+02  9.579e+01  -3.097 0.001986 ** 
fuel_typeOther             1.765e+02  2.324e+02   0.759 0.447730    
transmissionAutomatic      6.853e+02  9.331e+01   7.345 3.07e-13 ***
drivetrainRWD              3.351e+02  1.379e+02   2.430 0.015192 *  
drivetrainAWD              5.636e+02  1.575e+02   3.578 0.000356 ***
age:kilometer              8.967e-04  3.511e-04   2.554 0.010732 *  
engine:fuel_tank_capacity  3.403e-02  3.939e-03   8.639  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1632 on 1862 degrees of freedom
Multiple R-squared:  0.5501,	Adjusted R-squared:  0.5474 
F-statistic: 206.9 on 11 and 1862 DF,  p-value: < 2.2e-16
```

We could have obtained similar results by taking the Bayesian approach in which we assume a prior distribution of the model parameters and estimate a posterior distribution that is obtained by updating the prior distribution based on the likelihood obtained from the data. We would proceed as follows:

```
library(rstanarm)
rstanarm::stan_glm(price ~ age + kilometer + engine+
                     fuel_tank_capacity+fuel_type +
                     transmission + drivetrain+ age*kilometer +
                     engine*fuel_tank_capacity,
                   data = cars,
                   refresh =0)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      price ~ age + kilometer + engine + fuel_tank_capacity + fuel_type + 
       transmission + drivetrain + age * kilometer + engine * fuel_tank_capacity
 observations: 1874
 predictors:   12
------
                          Median MAD_SD
(Intercept)               2764.1  487.9
age                       -262.4   23.4
kilometer                    0.0    0.0
engine                      -0.8    0.3
fuel_tank_capacity         -30.1    8.7
fuel_typeDiesel           -295.8   93.4
fuel_typeOther             179.3  232.3
transmissionAutomatic      687.3   90.8
drivetrainRWD              336.1  141.0
drivetrainAWD              565.0  157.7
age:kilometer                0.0    0.0
engine:fuel_tank_capacity    0.0    0.0

Auxiliary parameter(s):
      Median MAD_SD
sigma 1632.3   26.8

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

While this model specification is not so different from the other models, we can gain a better understanding of the variation in the coefficient estimates across the models by comparing all the models. This will enable us to see how the coefficients vary with the inclusion or exclusion of some of the factors.

To compare the models we can create a regression table as follows:

```
library(modelsummary)

models <- list(
  'Model 1' = spec_1,
  'Model 2' = spec_2,
  'Model 3' = spec_3,
```

```r
  'Model 4' = spec_4
  )

modelsummary(
  models,
  vcov = 'robust',
  fmt = 2,
  estimate = "{estimate}{stars}",
  coef_rename = c(
    'age' = 'Age', 'kilometer' = 'Kilometers',
    'engine' = 'Engine size',
    'fuel_tank_capacity' = 'Fuel tank capcity',
    'age:kilometer' = 'Age x Kilometers',
    'engine:fuel_tank_capacity' = 'Engine x Fuel tank capacity',
    'fuel_typeDiesel' = 'Fuel type: Diesel',
    'fuel_typeOther' = 'Fuel type: Other',
    'transmissionAutomatic' = 'Transmission: Automatic',
    'drivetrainRWD' = 'Drivetrain: RWD',
    'drivetrainAWD' = 'Drivetrain: AWD'
    ),
  gof_omit = "Num|Adj|AIC|BIC|Log|Std",
  title = 'Comparision of different specifications'
  )
```

Comparision of different specifications

	Model 1	Model 2	Model 3	Model 4
(Intercept)	−1448.53***	2679.00*	−1133.08**	2765.27**
	(228.67)	(1062.85)	(349.72)	(967.56)
Age	−214.20***	−295.26***	−200.73***	−263.47***
	(40.86)	(73.77)	(31.73)	(59.71)
Kilometers	0.00	−0.01*	0.00	−0.01*
	(0.01)	(0.00)	(0.01)	(0.00)
Engine size	1.67***	−0.71	1.54***	−0.75
	(0.33)	(0.64)	(0.37)	(0.64)
Fuel tank capcity	36.90***	−23.38	30.04**	−29.86
	(10.77)	(22.02)	(9.22)	(20.00)
Age x Kilometers		0.00		0.00
		(0.00)		(0.00)
Engine x Fuel tank capacity		0.04**		0.03**
		(0.01)		(0.01)
Fuel type: Diesel			−594.61***	−296.64*
			(174.54)	(140.87)
Fuel type: Other			267.16	176.46
			(188.00)	(184.12)

	Model 1	Model 2	Model 3	Model 4
Transmission: Automatic			571.32***	685.32***
			(91.70)	(75.64)
Drivetrain: RWD			259.60	335.13*
			(177.24)	(147.86)
Drivetrain: AWD			701.51**	563.56*
			(228.55)	(237.27)
R2	0.502	0.528	0.531	0.550
F	104.013	143.081	128.302	124.072
RMSE	1711.81	1665.80	1660.77	1626.92

One particular of the above modelling is multicollinearity and overfitting as can be seen in changes in the magnitude and sign of coefficients fir variables like fuel tank capacity and engine size. We can also see that as we add variables to the model the R^2 values increases while the root mean squared error decreases.This happens even when the variables are insignificant: `drivetrainRWD`.This is a limitations that comes with using R^2. To justify the addition of variables in the model we can conduct ANOVA analysis between the second and the fourth model.

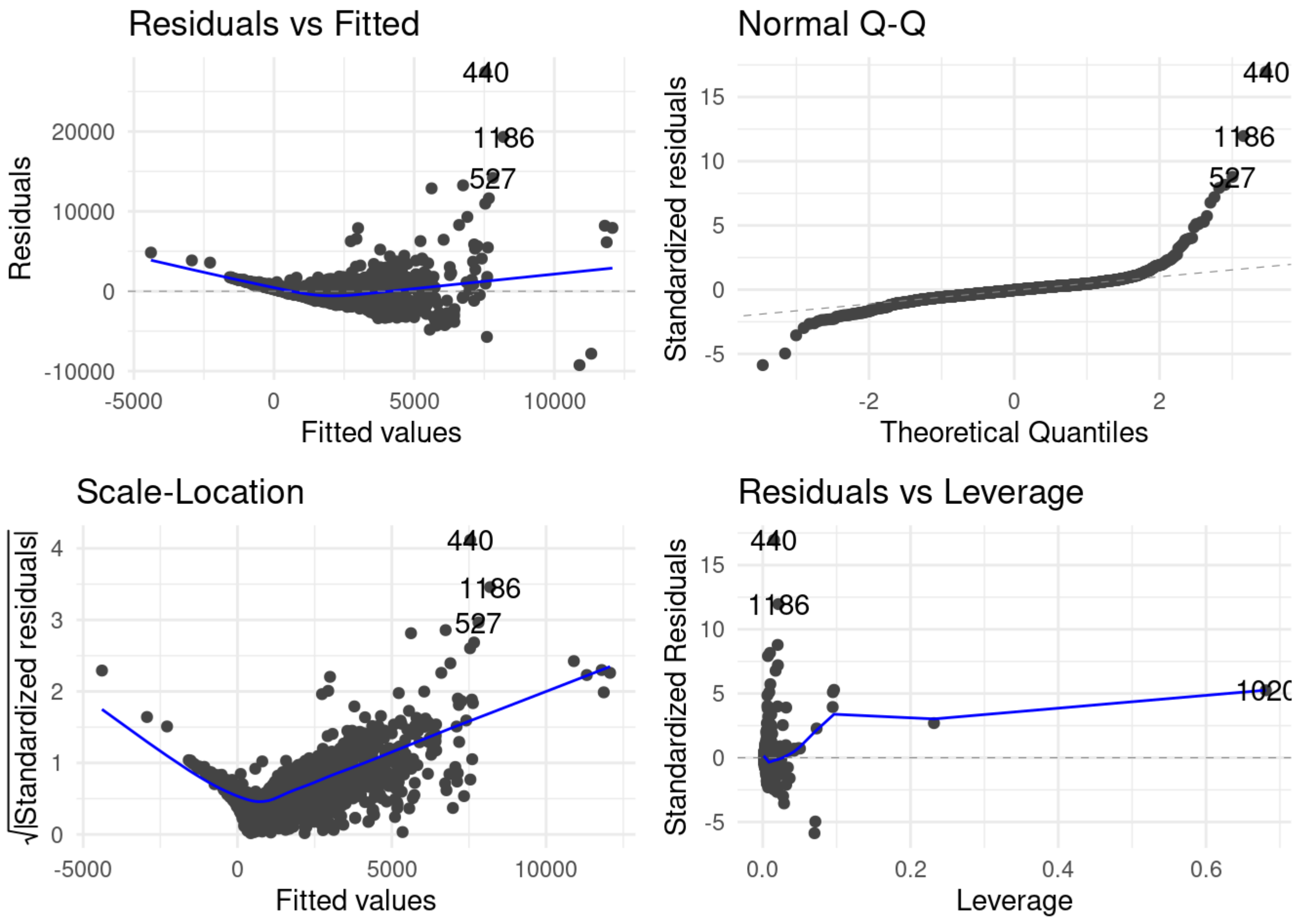
```
anova(spec_2, spec_4)
```

```
Analysis of Variance Table

Model 1: price ~ age * kilometer + engine * fuel_tank_capacity
Model 2: price ~ age + kilometer + engine + fuel_tank_capacity + fuel_type +
    transmission + drivetrain + age * kilometer + engine * fuel_tank_capacity
  Res.Df        RSS Df Sum of Sq     F    Pr(>F)
1   1867 5200126356
2   1862 4960239080  5 239887275 18.01 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We observe significant differences. We can therefore conclude that addition of extra variables in the models results in better performance. We can assume `spec_4` as the model to work with henceforth.

We can do some further investigation into this model to see whether it satisfies the assumptions of normally and homoskedastic errors. Homoskedastic errors are constant for all observations. We can do this by examining the diagnostic plots of the model.

```
library(ggfortify)
autoplot(spec_4)
```



The assumption of normality of residuals is violated as we can see many values on the edges of the `Normal Q-Q Plot`. The variance of the residuals also seems to vary with the fitted values in `Residual vs Fitted` plot. This is another violation. Violations of the OLS assumptions limit the performance and relevance of our models. We can conduct a formal test to support this using the Breusch-Pagan Test provided in the `lmtest` package

```r
library(lmtest)
bptest(spec_4)
```

```
	studentized Breusch-Pagan test

data:  spec_4
BP = 221.94, df = 11, p-value < 2.2e-16
```

Presence of high leverage points in the `Leverage vs Fitted Plot` also contribute to violation of homeskedasticity. To get around the violation of the homoskedasticity assumption, we can use heteroskedasticity-robust standard errors instead. These are calculated by doing an adjustment on the normal standard errors reported by the `lm` function. There are different variations of these estimators ranging from ‘HC1’ to ‘HC5’. The recommended estimator is the ‘HC3’ estimator.

```r
modelsummary(
  list('Model 4' = spec_4),
  vcov = 'robust',
  fmt = 2,
  estimate = "{estimate}{stars}",
  coef_rename = c(
    'age' = 'Age', 'kilometer' = 'Kilometers',
    'engine' = 'Engine size',
    'fuel_tank_capacity' = 'Fuel tank capcity',
    'age:kilometer' = 'Age x Kilometers',
    'engine:fuel_tank_capacity' = 'Engine x Fuel tank capacity',
    'fuel_typeDiesel' = 'Fuel type: Diesel',
    'fuel_typeOther' = 'Fuel type: Other',
    'transmissionAutomatic' = 'Transmission: Automatic',
    'drivetrainRWD' = 'Drivetrain: RWD',
    'drivetrainAWD' = 'Drivetrain: AWD'
    ),
  gof_omit = "Num|Adj|AIC|BIC|Log|Std")
```

	Model 4
(Intercept)	2765.27**
	(967.56)
Age	−263.47***
	(59.71)
Kilometers	−0.01*
	(0.00)
Engine size	−0.75
	(0.64)
Fuel tank capcity	−29.86
	(20.00)
Fuel type: Diesel	−296.64*
	(140.87)
Fuel type: Other	176.46
	(184.12)
Transmission: Automatic	685.32***
	(75.64)
Drivetrain: RWD	335.13*
	(147.86)
Drivetrain: AWD	563.56*
	(237.27)
Age x Kilometers	0.00
	(0.00)
Engine x Fuel tank capacity	0.03**
	(0.01)
R2	0.550
F	124.072
RMSE	1626.92

Using robust errors results in wider standard errors and reduced statistical significance as we can see with `engine`, `fuel_tank_capacity` and the joint effect of `age` and `kilometer`.

Inference

Having done the regression analysis, we can now proceed to performing statistical inference on these variables. Statistical inference involves making some prior claims about our data and using the data to test whether our claims are valid. For example, we can assume that the average selling price of the cars is $1,500 and then examine the data to test the validity of this claim. In statistical inference the claims we make are often of the form 'there is no difference, relationship or effect between variables'. This is referred to as the *Null hypothesis*. We test this hypothesis against the *Alternative hypothesis* which implies the existence of a difference, an effect or relationship between variables. The hypothesis are mutually exclusive.

The general workflow of statistical inference involves the following steps:

1. Specifying the variable(s) of interest and their relation,
2. Defining the null and alternative hypothesis
3. Generating data assuming the null hypothesis
4. Calculating the test statistics

In order to arrive at a conclusion, the null hypothesis is assumed to be true. We then find evidence in the data in favor of or against the null hypothesis. This evidence is represented by the test statistic calculated from the data. We then calculate the probability of obtaining a test statistic that is at least as extreme as the one we observe in the data assuming the null hypothesis is true. If such a probability is high enough, then it means that the data is consistent with the null hypothesis. If such a probability is low, then the null hypothesis is less likely to be true and we reject it. The threshold at which we determine whether the evidence is high enough or low enough is determined by the significance level. The significance level is the probability that our results are random. If we assume a 5% significance level, then it means that if the probability of observing such a test statistic is less than 5%, then we reject the null hypothesis and assume the alternative hypothesis to be true. To be clear, if we repeat the experiment many times assuming the null hypothesis is true, say a thousand times and in at least 5% of these experiments we observe a test statistic greater than or equal to the the test statistic that we calculate from the data, then we can conclude there is an effect or relationship and therefore reject the null hypothesis and assume the alternative hypothesis to be true.

Given that this is the general workflow in hypothesis testing, the `infer` package, which is part of the `tidymodels` framework, has unified this process by providing an interface to implement these for most of the tests in statistics. This is implemented through a set of four functions which generalize the steps above.

1. The `specify()` function is used to define the variable we are hypothesize or the relationship between the variables if there are two or more.
2. The `hypothesize()` function defines the kind of hypothesis we are testing: a point estimate or a test of independence.

3. The `generate()` function creates samples of the data according to the null hypothesis.

4. Finally, the `calculate()` function calculates the test statistic in each generated sample.

Following the above steps will obtain the distribution of the test statistic under the null hypothesis. After generating the distribution of the test statistic we calculate the actual value from the data and compare it to the distribution.

Examples

We can demonstrate this process by a few examples.

Mean Price

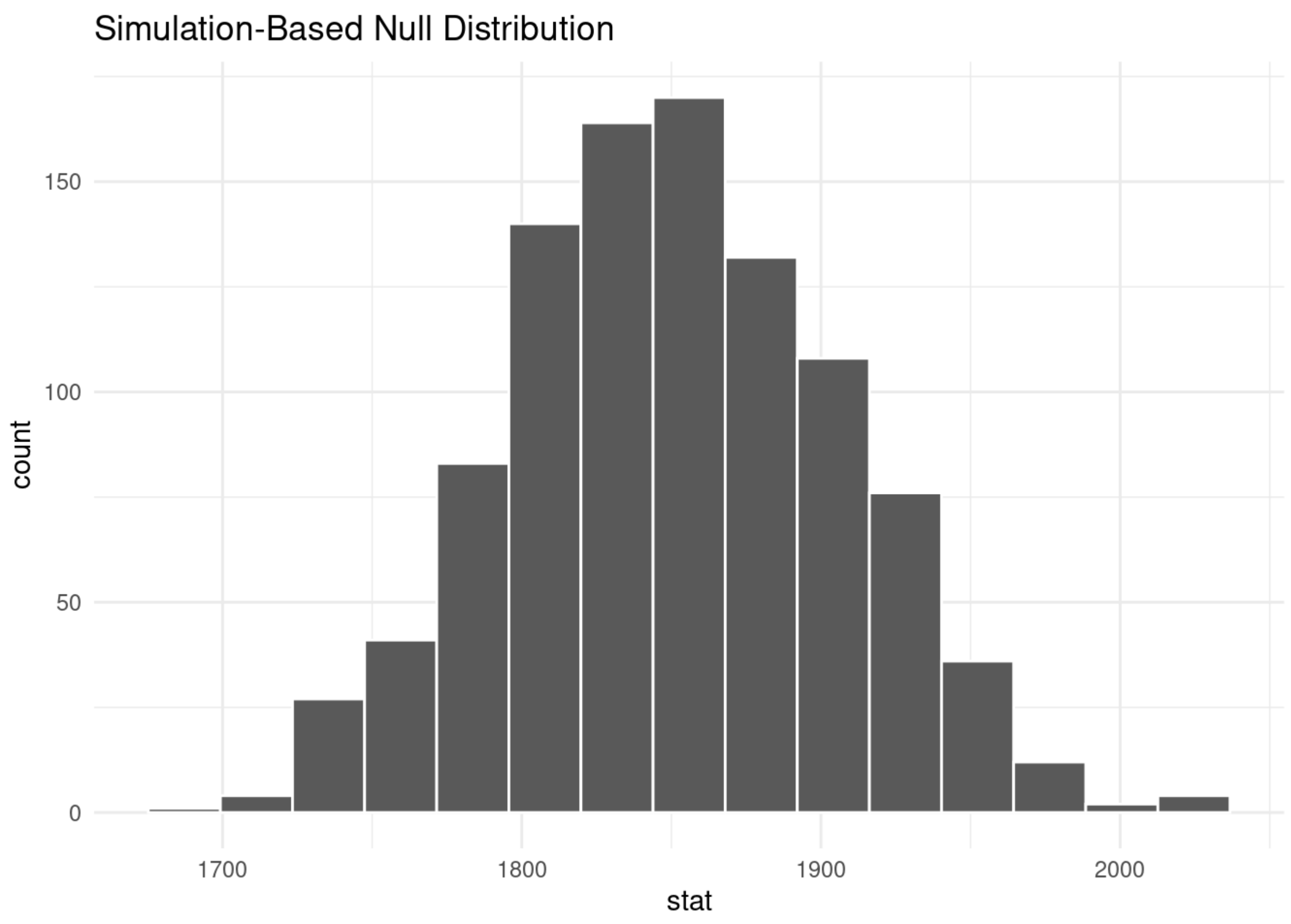
We can hypothesize that the mean selling price is $1850 and test against this using the data as follows.

We specify the variable of interest and calculate test statistic(standardized sample average) across the 1000 samples generated. Sampling is done with replacement('bootstrap').

```r
library(infer)

null_distribution <- cars %>%
  specify(response = price) %>%
  hypothesise(null = 'point', mu = 1850) %>%
  generate(reps = 1000, type = 'bootstrap') %>%
  calculate(stat = 'mean')
```

We can view this distribution of the test statistics under the null hypothesis.

```r
null_distribution %>%
  visualise()
```



To obtain the actual value from the data we can follow the previous code while removing the `hypothesize()` and `generate()` steps.

```
price_mean <- cars %>%
  specify(response = price) %>%
  calculate(stat = 'mean')
price_mean
```

```
Response: price (numeric)
# A tibble: 1 × 1
   stat
  <dbl>
1 1718.
```

For us to make conclusions about whether to reject the null or alternative hypothesis, we need to get the p-value. Since we are testing for whether the mean is equal to $1,850, our test is two-sided. The mean could be smaller than or bigger than $1,850. If we were testing that the mean was less or more than, then we would set this value to left or right respectively.

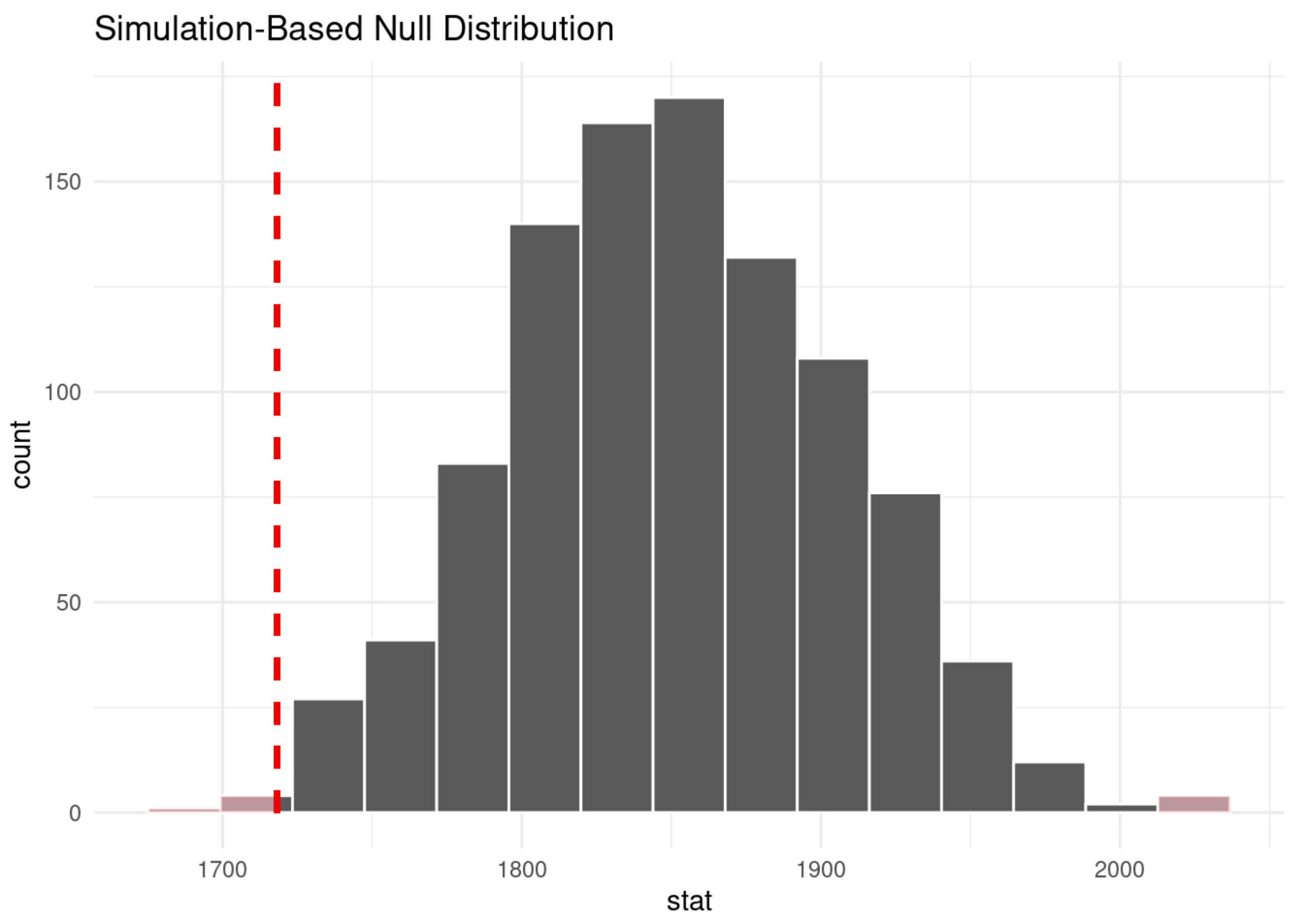
```
null_distribution %>%
  get_p_value(obs_stat = price_mean, direction = 'two-sided')
```

```
# A tibble: 1 × 1
  p_value
    <dbl>
1   0.006
```

The p-value is less than 0.05(5% significance level). This suggests strong evidence against the null hypothesis in favor of the alternative hypothesis that mean price is not equal to $1,850.

We can gain a better understanding by visualizing this result. The p-value is represented by the shaded area.

```
null_distribution %>%
  visualise() +
  shade_p_value(obs_stat =price_mean, direction = 'two-sided',
                linewidth =1, linetype = 2)
```

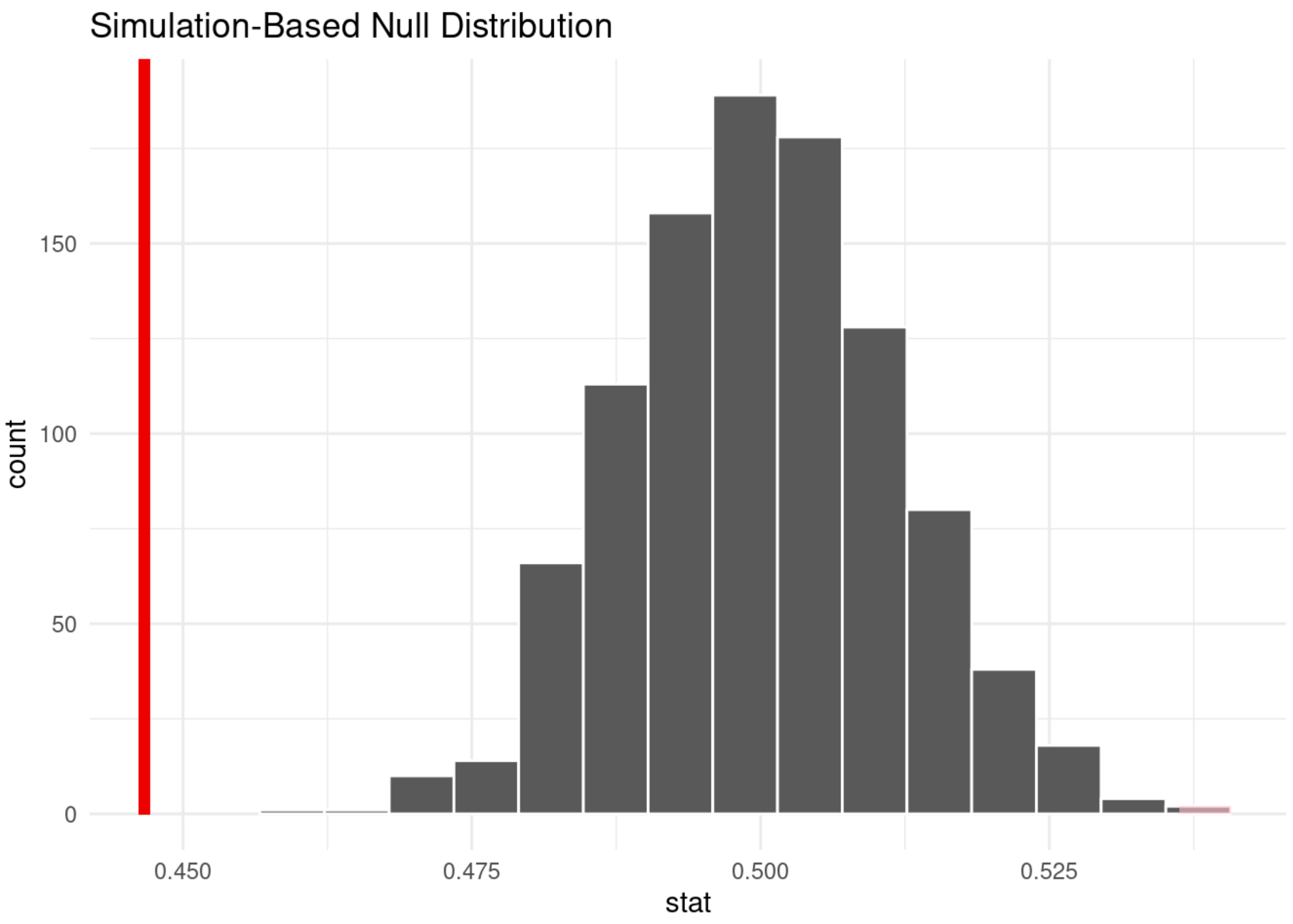


Proportion test

To test whether the proportion of cars with `Automatic` transmission is equal to 50% we can do a point estimate as follows.

```
# calculate the observed statistic
p_hat <- cars %>%
  specify(response = transmission, success = 'Automatic') %>%
  calculate(stat = 'prop')

null_dist <- cars %>%
  specify(response = transmission, success = 'Automatic') %>%
  hypothesise(null = 'point', p = .5) %>%
  generate(reps = 1000) %>%
  calculate(stat = 'prop')
```

```r
null_dist %>%
  visualise() +
  shade_p_value(
    obs_stat = p_hat, direction = 'two-sided'
  )
```



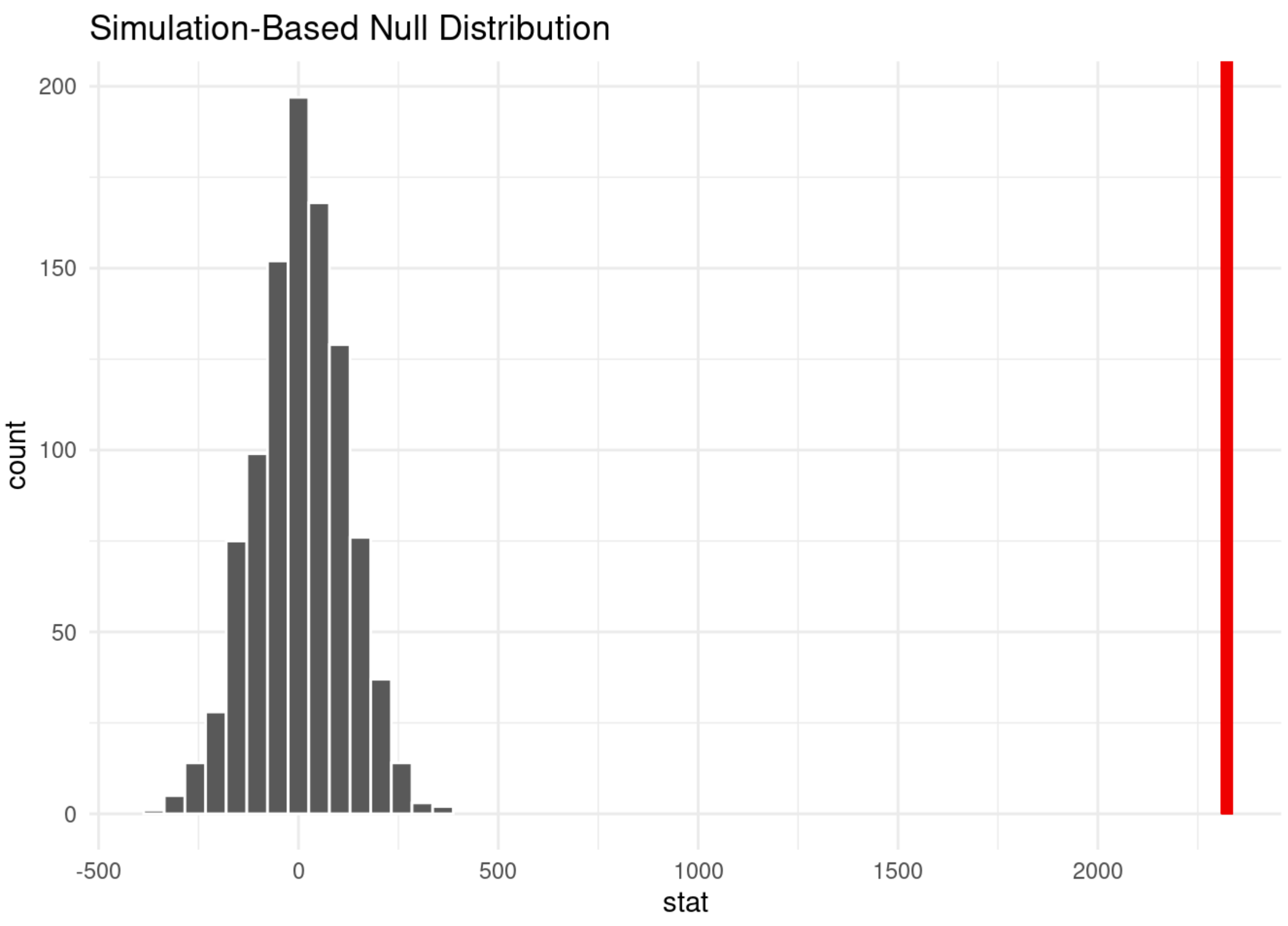
We see that the proportion of cars with Automatic transmission is not equal to 50%.

Difference in price by Transmission

We can extend the previous example to test that cars with `Automatic` transmission have a higher selling price than cars with `Manual` transmission.

Here we use permutation based sampling where the values of the variables are shuffled without replacement.

```r
null_distribution <- cars %>%
  specify(price ~ transmission) %>%
  hypothesise(null = 'independence') %>%
  generate(reps = 1000, type = 'permute') %>%
  calculate(stat = 'diff in means')

obs_diff <- cars %>%
  specify(price ~ transmission) %>%
  calculate(stat = 'diff in means', order = c('Automatic','Manual'))

null_distribution %>%
  visualise() +
  shade_p_value(obs_stat = obs_diff, direction = 'right')
```

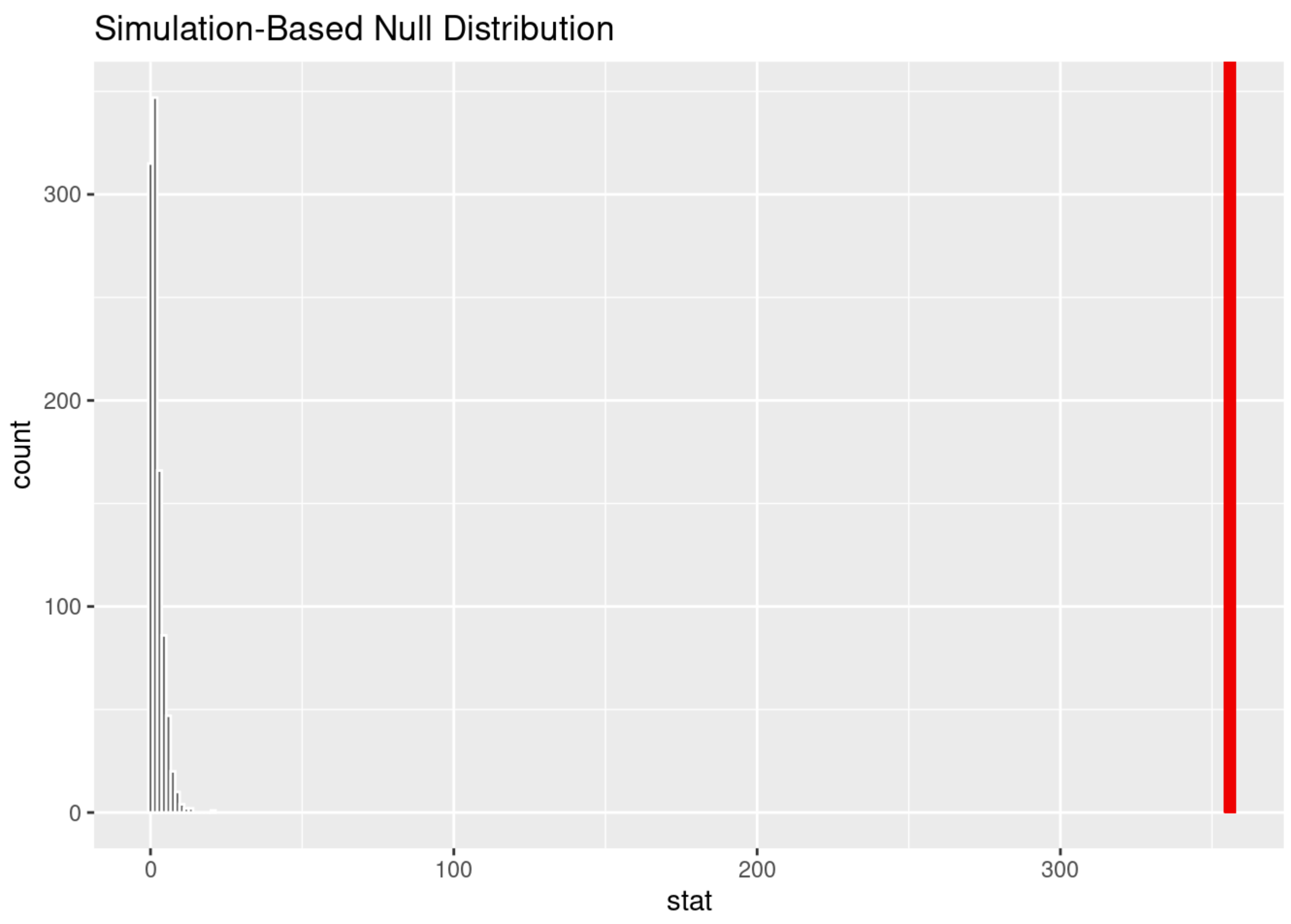


We see that on average selling price is higher for `Automatic` cars than for `Manual` cars.

Test for Association

Finally, we can test whether there is an association between `fuel_type` and `drivetrain`.

```r
null_distribution <- cars %>%
  specify(response = transmission, explanatory = drivetrain) %>%
  hypothesise(null = 'independence') %>%
  generate(reps = 1000, type = 'permute') %>%
  calculate(stat = 'Chisq')

obs_stat <-  cars %>%
  specify(response = drivetrain, explanatory = transmission) %>%
  calculate(stat = 'Chisq')

null_distribution %>%
  visualise()+
  shade_p_value(obs_stat = obs_stat, direction = 'right')+
  theme_grey()
```



From the results above, we reject the null hypothesis that there is no association between `drivetrain` and `transmission`.

Conclusion

The above exercise has demonstrated how regression analysis and statistical inference can be applied in a simple and intuitive way. We have built and explored various regression setups, optimized parameter values and built a a workflow to perform statistical inference.

While the results in this setup and given the structure are not extensible or generalizable, we have shown how the process can be conducted in R efficiently.