Building-Statistical-Models-in-R

Gian Carlo Sanfuego

2024-03-12

# Building Statistical Models in R: Linear Regression

What is a linear regression model? The linear regression– simple linear regression model is used to predict a quantitative outcome Y, on the basis of one single predictor variable X. The goal is to build a mathematical model or formula that defines Y as a function of the X variable. The model defines the linear relationship between an explanatory variable or independent variable or the X-variable, or the predictor variable depending on the context; or y which is sometimes refer to as outcome, response, dependent, or y variable. After building a statistical significant model, it is used to predict future outcomes on the basis of new X value.

In this project, we will use one of the openly available data set called mpg, which is a fuel economic data from 1999 to 2008 for 38 popular models of cars.

# Task One: Getting Started

In this task, you will learn change the panes and font size. Also, you will learn how to set and check your current working directory

### 1.1: Get the working directory

# setwd("C:/Users/gianc/OneDrive/Desktop/DATA ANALYSIS/PROJECTS/R/BUILDING STATISTICAL MODELS IN R - LINEAR REGRESSION")  
  
getwd()

## [1] "D:/Desktop/DATA ANALYSIS/PROJECTS/R/BUILDING STATISTICAL MODELS IN R - LINEAR REGRESSION"

# Task Two: Import packages and dataset

In this task, you will import the required packages and data for this project

Tidyverse is a collection of R packages designed for data science, including dplyr, ggplot2, tidyr, readr, purrr, and more, which provide tools for data manipulation, visualization, and analysis. While ggpubr, simplifies the creation of ‘ggplot2’ publication-ready plots by providing easy-to-use functions for common tasks like adding statistical summaries and customizing plot aesthetics. Another is the package broom, which converts statistical analysis objects from R into tidy data frames, making it easier to integrate with the rest of the tidyverse. Lastly, ggfortify, which extends ggplot2 to handle various statistical results, enabling the autoplot function to create plots for time series, principal components, and other models in a consistent and straightforward manner.

### 2.1: Importing required packages

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.3.3

## Warning: package 'ggplot2' was built under R version 4.3.3

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.1 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(ggpubr)

## Warning: package 'ggpubr' was built under R version 4.3.3

library(broom)

## Warning: package 'broom' was built under R version 4.3.3

library(ggfortify)

## Warning: package 'ggfortify' was built under R version 4.3.3

Let’s now import the data.

### 2.2: Import the mpg.csv dataset

data <- read.csv(file = "mpg.csv", header = T, sep = ',')

### 2.3: View and check the dimension of the dataset

# View(data)  
dim(data)

## [1] 234 12

# Task Three: Explore the dataset

In this task, you will learn how to explore and clean the data

What this do is to return the first 6 rows, and the last 6 rows of the data set

### 3.1: Take a peek using the head and tail functions

head(data)

## X manufacturer model displ year cyl trans drv cty hwy fl class  
## 1 1 audi a4 1.8 1999 4 auto(l5) f 18 29 p compact  
## 2 2 audi a4 1.8 1999 4 manual(m5) f 21 29 p compact  
## 3 3 audi a4 2.0 2008 4 manual(m6) f 20 31 p compact  
## 4 4 audi a4 2.0 2008 4 auto(av) f 21 30 p compact  
## 5 5 audi a4 2.8 1999 6 auto(l5) f 16 26 p compact  
## 6 6 audi a4 2.8 1999 6 manual(m5) f 18 26 p compact

tail(data)

## X manufacturer model displ year cyl trans drv cty hwy fl class  
## 229 229 volkswagen passat 1.8 1999 4 auto(l5) f 18 29 p midsize  
## 230 230 volkswagen passat 2.0 2008 4 auto(s6) f 19 28 p midsize  
## 231 231 volkswagen passat 2.0 2008 4 manual(m6) f 21 29 p midsize  
## 232 232 volkswagen passat 2.8 1999 6 auto(l5) f 16 26 p midsize  
## 233 233 volkswagen passat 2.8 1999 6 manual(m5) f 18 26 p midsize  
## 234 234 volkswagen passat 3.6 2008 6 auto(s6) f 17 26 p midsize

Now lets see the structure of the data to see if the data type is correct.

### 3.2: Check the internal structure of the data frame

str(data)

## 'data.frame': 234 obs. of 12 variables:  
## $ X : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ manufacturer: chr "audi" "audi" "audi" "audi" ...  
## $ model : chr "a4" "a4" "a4" "a4" ...  
## $ displ : num 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...  
## $ year : int 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...  
## $ cyl : int 4 4 4 4 6 6 6 4 4 4 ...  
## $ trans : chr "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...  
## $ drv : chr "f" "f" "f" "f" ...  
## $ cty : int 18 21 20 21 16 18 18 18 16 20 ...  
## $ hwy : int 29 29 31 30 26 26 27 26 25 28 ...  
## $ fl : chr "p" "p" "p" "p" ...  
## $ class : chr "compact" "compact" "compact" "compact" ...

### 3.3: Count missing values in the variables

sum(is.na(data))

## [1] 0

As the result shows, we have no NA values in our data set. But lets be more specific and check for null values per column.

To check the missing values per column:

sapply(data, function(x) sum(is.na(x)))

## X manufacturer model displ year cyl   
## 0 0 0 0 0 0   
## trans drv cty hwy fl class   
## 0 0 0 0 0 0

### 3.4: Check the column names for the data frame

names(data)

## [1] "X" "manufacturer" "model" "displ" "year"   
## [6] "cyl" "trans" "drv" "cty" "hwy"   
## [11] "fl" "class"

Since the first column is just a row number, we dont actually need it so we can just remove it by sub-setting the data set.

### 3.5: Drop the first column of the data frame

data <- data[,-1]

Let’s check if it works well.

dim(data)

## [1] 234 11

names(data)

## [1] "manufacturer" "model" "displ" "year" "cyl"   
## [6] "trans" "drv" "cty" "hwy" "fl"   
## [11] "class"

# Task Four: Data Visualizations

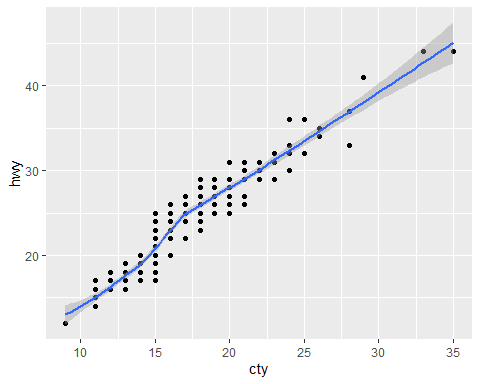
In this task, you will learn how to visualize the variables we will use to build the statistical model

### 4.1: Plot a scatter plot for the variables with cty on the x-axis

### hwy on the y-axis

ggplot(data, aes(x = cty, y = hwy)) +  
 geom\_point() +  
 stat\_smooth()

## `geom\_smooth()` using method = 'loess' and formula = 'y ~ x'

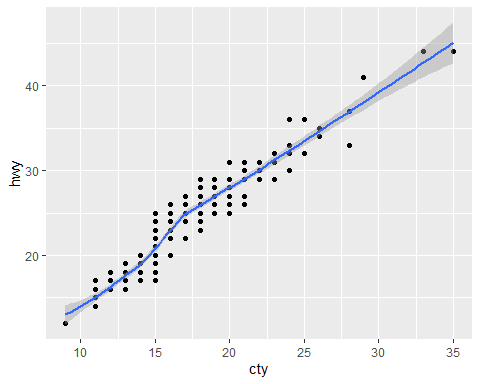


So in out graph, it suggests a linear, increasing relationship betwenn hwy and cty. This is actually a good thing because one important assumption of linear regression is that the relationship between the outcome variable and the predictor variable must be linear and additive.

Alternative

data %>%  
 ggplot(aes(x = cty, y = hwy)) +  
 geom\_point() +  
 stat\_smooth()

## `geom\_smooth()` using method = 'loess' and formula = 'y ~ x'



It is also possible to compute for the correlation coefficients between these two variables using the function cor( )

### 4.2: Find the correlation between the variables

cor(data$cty, data$hwy)

## [1] 0.9559159

0.9559159 correlation between hwy and cty suggest a very strong positive relationship between the predictor and response variable. Basically the correlation coefficients measures the level of association between two variables X and Y.

Its value ranges between minus one, that’s a perfect negative correlation (when X increases, Y decreases) or plus one, which is a perfect positive correlation (when X increases Y will also increase). A value close to zero suggests a weak relationship between the variables. A low correlation, say between -0.2 to 0.2 probably suggests that much of the variation of the outcome variable Y is not explained by the predictor variable X. In such a case, we will probably look for a better predictor variable. In our own example here, the correlation coefficient is large enough, so we can continue by building a linear model of y, as a function of x.

# Task Five: Model Building

In this task, you will learn how to build a simple linear regression model

### 5.1: Create a simple linear regression model using the variables

model <- lm(hwy ~ cty, data = data)  
model

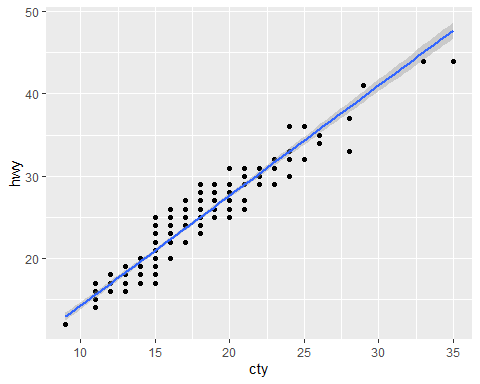
##   
## Call:  
## lm(formula = hwy ~ cty, data = data)  
##   
## Coefficients:  
## (Intercept) cty   
## 0.892 1.337

So what does this results mean, in the intercept part, what it means is that, when x is zero, when cty is zero, the value of hwy is 0.892. It is interpreted as the predicted hwy units for zero cty. More over, regression coefficient for cty which is the 1.337 is the slope. And in our model, the slope term is saying that for every 1 unit increase in x, there is an additional 1.337 increase in y. Meaning for every 1 mile per gallon increase in cty the required highway mile per gallon is increased by 1.337.

### 5.2: Plot the regression line for the model

ggplot(data, aes(x = cty, y = hwy)) +  
 geom\_point() +  
 stat\_smooth(method = lm)

## `geom\_smooth()` using formula = 'y ~ x'



As you can see, we also use the same plot above but this time, we added an argument method in stat\_smooth function to add our linear model.

Now, we already have our fitted model of hwy as a function of cty. But before using this formula or model to predict future, let’s make sure first that our model is statistically significant meaning there is statistical significant relationship between predictor and the response variables, also that the model we built fits very well for the data.

# Task Six: Model Assessment I

In this task, you will learn how to assess and interpret the result of a simple linear regression model

Let’s know use the summary function to assess our fitted model

### 6.1: Assess the summary of the fitted model

summary(model)

##   
## Call:  
## lm(formula = hwy ~ cty, data = data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -5.3408 -1.2790 0.0214 1.0338 4.0461   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.89204 0.46895 1.902 0.0584 .   
## cty 1.33746 0.02697 49.585 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.752 on 232 degrees of freedom  
## Multiple R-squared: 0.9138, Adjusted R-squared: 0.9134   
## F-statistic: 2459 on 1 and 232 DF, p-value: < 2.2e-16

The summary outputs shows 6 components including the call used to compute the model itself, we also have the residuals which provides a quick view of the distribution of the residuals, which by definition, have the mean of zero therefore, the median should not be far from zero, and the minimum and maximum should be roughly equal in absolute value. Then, we have the coefficients which shows the regression beta coefficients that we’ve seen before and their statistical significance which is marked by stars. We also have the Residual Standard Error (RSE), the R squared, and F statistics. These are metrics that are used to check how well the model fits our data.

As we can see, the first item shown in the output of our model is what the formula that is used to fit the model. The next is the residuals, it is actually the difference between the actual observed response value, hwy and the response value that the model predicted. We will take this further by considering plotting the residuals to see whether this is normally distributed. The next section in the output shows the coefficients of the model. Theoretically, in simple linear regression, coefficients are two unknown constants that represents the intercept and the slope terms in the linear regression. We also have the standard error (SE) which defines the accuracy of the beta coefficients It reflects how the coefficients varies under repeated sampling. It can be used to compute the confidence interval and the T statistics. T values and P values defines the statistical significance of the coefficients. The coefficient standard error measures the average amount that the coefficient estimates vary from the actual average value of our response variable. It measures the variability or the accuracy of the beta coefficient. We will ideally want a lower number relative to its coefficient. It can also be used to compute confidence interval

### 6.2: Calculate the confidence interval for the coefficients

confint(model)

## 2.5 % 97.5 %  
## (Intercept) -0.03189534 1.815978  
## cty 1.28431197 1.390599

It shows that there is approximately 95% chance that the interval 1.28431197 and 1.390599 will contain the true value of the slope, and similarly, for the intercept, this says that this is the 95% confidence interval.

So going back, looking at t values and p values, always remember that the higher the t statistics and the lower the p value, the more significant the predictor is. A small p value indicates that it is unlikely that we observe this relationship by chance. So in this model, the p value is very close to zero, it was supported by the significance code below coefficients, the more the star, the more significant. So we can reject the null hypothesis and accept the alternative hypothesis which means there is a significant association between variables. We also saw the relationship the moment we graph it but we are more sure this time since we test it.

# Task Seven: Model Assessment II

In this task, you will learn how to assess the accuracy of a simple linear regression model

### 7.1: Assess the summary of the fitted model

summary(model)

##   
## Call:  
## lm(formula = hwy ~ cty, data = data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -5.3408 -1.2790 0.0214 1.0338 4.0461   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.89204 0.46895 1.902 0.0584 .   
## cty 1.33746 0.02697 49.585 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.752 on 232 degrees of freedom  
## Multiple R-squared: 0.9138, Adjusted R-squared: 0.9134   
## F-statistic: 2459 on 1 and 232 DF, p-value: < 2.2e-16

Let’s now continue by checking how well the model fits the data using the last part in the summary output. This process is also called the goodness of fit. The overall quality of the linear regression fit can be accessed by using the following three quantities displayed in the model summary. You can use the RSE, R squared or the F statistics. RSE also known as the model sigma, is the residual variation representing the average variation of the observation points around the fitted regression line. This is the standard deviation of the residual errors. RSE provides an absolute measure of patterns in data that cant be explained by the model. When comparing model, the model with smaller RSE is better fitted model. Dividing it by the average value of outcome variable will give us the prediction error. Let’s see it below.

### 7.2: Calculate the prediction error of the fitted model

sigma(model)\*100/mean(data$hwy)

## [1] 7.475581

We have the prediction error of 7.475581 which is quite low and acceptable. Let’s focus now on R squared, a high value of R squared is a good indication, its value should lie between 0 and 1. In this example, we’ve obtain a value of 0.9138 that is roughly 91% of the variation found in the response variation which can be explained by the X variable. Adjusted R squared is used in multiple regression, the R squared is increased the moment you add more variable as a predictor, that is why adjusted r squared is more preferred to use. Lastly, F statistics, it gives the overall significance of the model.

# Task Eight: Model Prediction

In this task, you will learn how to check for metrics from the fitted model and make prediction for new values

The fitted or the predicted values are those values that you would expect to get given x values according to the built regression model, or maybe visually, the best fitting straight line

### 8.1: Find the fitted values of the simple regression model

fitted <- predict.lm(model)  
head(fitted, 3)

## 1 2 3   
## 24.96624 28.97861 27.64115

As you can see, theses are the first three fitted values, but we can get this more easily using augment function from the broom package, let’s try this code below

### 8.2: Find the fitted values of the simple regression model

model\_diag\_metrics <- augment(model)  
head(model\_diag\_metrics)

## # A tibble: 6 × 8  
## hwy cty .fitted .resid .hat .sigma .cooksd .std.resid  
## <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 29 18 25.0 4.03 0.00458 1.74 0.0123 2.31   
## 2 29 21 29.0 0.0214 0.00834 1.76 0.000000632 0.0123  
## 3 31 20 27.6 3.36 0.00661 1.74 0.0123 1.92   
## 4 30 21 29.0 1.02 0.00834 1.75 0.00144 0.585   
## 5 26 16 22.3 3.71 0.00445 1.74 0.0101 2.12   
## 6 26 18 25.0 1.03 0.00458 1.75 0.000805 0.591

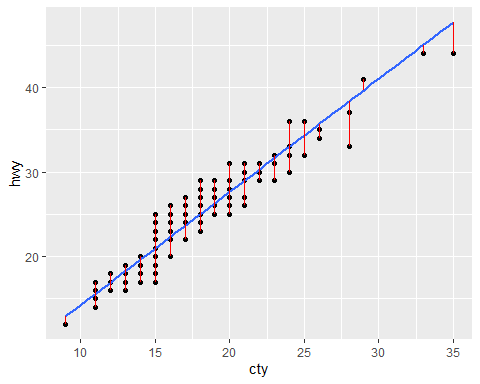
Now we can see from the result, .fitted, this is the fitted hwy, comparing the previous values, it is obviously equal to the new ones. So, basically, the augment function is more efficient by not just returning the fitted values but also other metrics like residuals.

### 8.3: Visualize the residuals of the fitted model

ggplot(model\_diag\_metrics, aes(cty, hwy)) +  
 geom\_point() +  
 stat\_smooth(method = lm, se = FALSE) +  
 geom\_segment(aes(xend = cty, yend = .fitted), color = "red", size = 0.3)

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

## `geom\_smooth()` using formula = 'y ~ x'



In this graph, the blue line is the fitted regression line, each of the points are the data points, while the red lines tells about the residual error.

## 8.4: Predict new values using the model

predict(  
 object = model,  
 newdata = data.frame(cty = c(21, 27, 14))  
)

## 1 2 3   
## 28.97861 37.00334 19.61642

As the result shows, when cty is 21, it gives as a 28.9 predicted hwy values. Same with the 2nd and 3rd values.

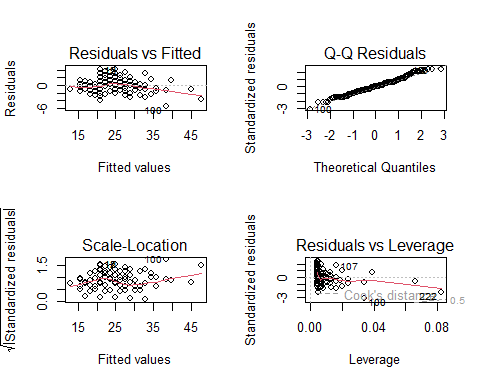
# Task Nine: Assumptions Check: Diagnostic Plots

In this task, you will learn how to perform diagnostics check on the fitted model

Linear regression makes several assumptions about the data, the linearity in which we are already done checking, and the next is the normality of the residuals, the residual errors are assumed to be normally distributed. The third is homogeneity of the residual variance, it is assumed to have a common or constant variance which is called the homoscedasticity. Fourth assumption is the independence or residual error terms. You would normally want to check whether or not these assumptions hold true. Possible problems includes non-linearity of the outcome-predictor relationship. Another is heteroscedasticity which means non constant variance in error terms, or maybe the presence of influential values in the data that can be outliers. All of this potential problems can be check by making diagnostic plots.

### 9.1: Plotting the fitted model

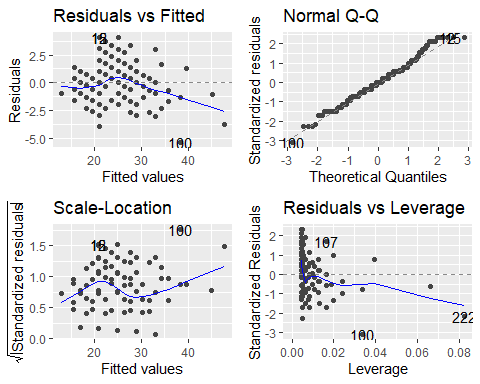
par(mfrow = c(2, 2)) ## This plots the figures in a 2 x 2  
plot(model)



Let’s use the autoplot for better version of this.

### Better Version

autoplot(model)



The residuals plots shows the residuals in four different ways, residuals vs. fitted which is used to check the linear relationship assumption. A horizontal line without a distinct pattern is an indication for a linear relationship. For the normal QQ plot, it is used to examine whether the residuals are normally distributed It is good if the residuals follow the dashed straight line. Then, for the scale location or the spread location, this is used to check for the homogeneity of the variance of the residuals. If there is a constant variance. Horizontal line with equally spread point is a good indication of homoscedasticity. And in this example, that is not the case. Lastly, the residuals vs. leverage, this is used to identify influential cases that is the extreme values that might influence the regression results when included or excluded from the analysis. Let’s start over and plot it all individually and assess what it means.

### 9.2: Return par back to default

dev.off()

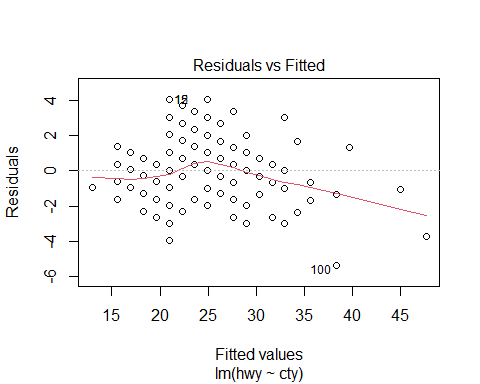
## null device   
## 1

### or

par(mfrow = c(1, 1))

### 9.3: Return the first diagnostic plot for the model

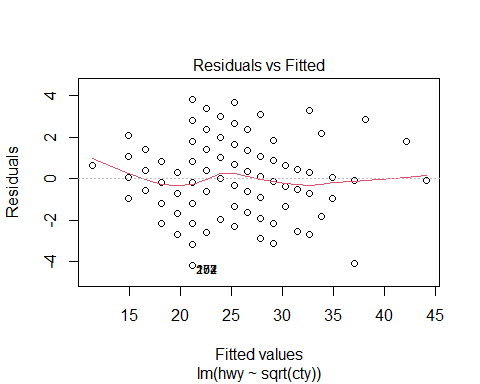
plot(model, 1)



The residual plot shows a clear curved pattern, which indicates that the linear model is not a good fit for the data. This suggests that there may be a non-linear relationship between the predictor and response variables. The presence of a pattern in the residual plot indicates a problem with the linear model, specifically that the assumption of linearity is not met. Given the non-linear pattern in the residuals, it’s a good idea to explore transformations of the predictor variable to try to linearize the relationship. In this case, taking the square root of the predictor variable (cty) seems to have helped, as shown in the second plot.

### Build another regression model

model1 <- lm(hwy ~ sqrt(cty), data = data)  
plot(model1, 1)

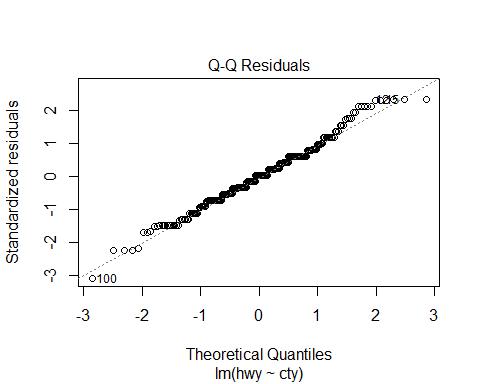


The plot shows a random scatter of points, with no apparent pattern or curvature. This suggests that the linear model, which includes the square root of the predictor variable cty, is a good fit for the data. The residuals appear to be randomly distributed around zero, indicating that the model is capturing the underlying relationship between the predictor and response variables. Additionally, the red line is somewhat on the horizontal line, and there is no specific pattern, which further supports the assumption of linearity. However, it’s worth noting that if the assumption of linearity is not met in other situations, a simple transformation of the model can be performed to address non-linear relationships.

Now for the second diagnostic plot, the QQ plot is used to verify or visually check that the normal assumption is satisfied. The normal probability plot of residuals should approximately follow a straight line.

### 9.4: Return the second diagnostic plot for the model

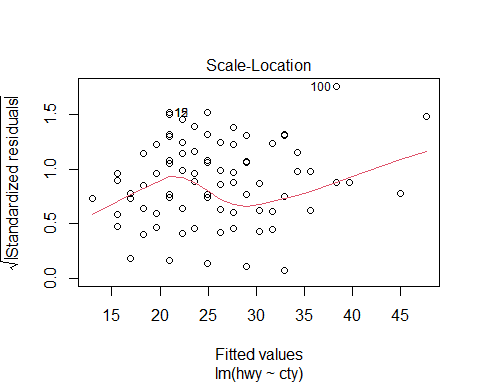
plot(model, 2)



You will notice that the residuals followed a straight line. The points approximately lie on the straight line, so we can assume that in our example, all the points fall approximately on the reference line. Therefore, we can assume that the residuals are normal. Let’s now check the third plot, homogeneity.

### 9.5: Return the third diagnostic plot for the model

plot(model, 3)



This plot shows if residuals are spread equally along the ranges of the predictor, it is good if you see the horizontal line with equally spread points. In this case, it can be seen that the variability or the variances of the residual points increases with the value of the fitted outcome variable. As the fitted value increases; you will see that the residual point also tend to increase, suggesting a non-constant variance in the residual errors and this is called heteroscedasticity. A possible solution to reduce the heteroscedasticity problem is to use a log or square root transformation on the outcome variable y.

# Task Ten: Multiple Regression

In this task, you will learn how to build and interpret the results of a multiple regression model

### 10.1: Build the multiple regression model with hwy on the y-axis ### and cty and cyl on the x-axis

mul\_reg\_model <- lm(hwy ~ cty + cyl, data = data)

### 10.2: This prints the result of the model

mul\_reg\_model

##   
## Call:  
## lm(formula = hwy ~ cty + cyl, data = data)  
##   
## Coefficients:  
## (Intercept) cty cyl   
## -0.07702 1.36425 0.08784

### 10.3: Check the summary of the multiple regression model

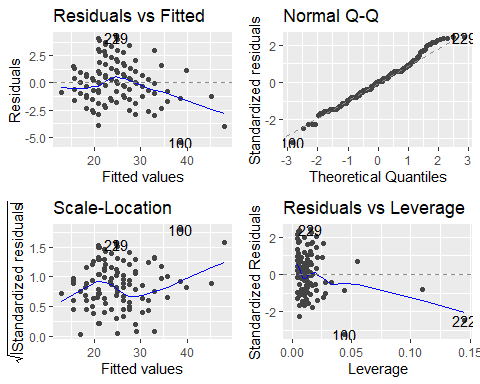
summary(mul\_reg\_model)

##   
## Call:  
## lm(formula = hwy ~ cty + cyl, data = data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -5.4735 -1.1952 0.0398 0.9934 4.1691   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.07702 1.40888 -0.055 0.956   
## cty 1.36425 0.04559 29.924 <2e-16 \*\*\*  
## cyl 0.08784 0.12040 0.730 0.466   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.754 on 231 degrees of freedom  
## Multiple R-squared: 0.914, Adjusted R-squared: 0.9132   
## F-statistic: 1227 on 2 and 231 DF, p-value: < 2.2e-16

Now we have built a multiple variable regression and as you can see in the result, the cyl variable that we added now is not significant, because if you look at the p-value is grater than 0.05. In fact, the standard error is 0.1204 and t-statistics or t-value is very low. It should be as large as possible from zero. So, its not significant variable. This means we should not include it in our model.

### 10.4: Plot the fitted multiple regression model

autoplot(mul\_reg\_model)



The four plots are diagnostic plots for a linear regression model.

Residuals vs Fitted: Shows the residual errors plotted against the fitted values. A random scatter of points suggests the model is adequate. In this case, there is some curvature, suggesting that a linear model might not be the best fit.

Normal Q-Q: Checks the normality of the residuals. The points should fall roughly along a straight line. This plot shows some deviations from normality, especially for larger residuals.

Scale-Location: Plots the absolute values of the standardized residuals against the fitted values. It helps identify non-constant variance in the data (heteroscedasticity). The plot suggests that the variance might be slightly increasing with the fitted values.

Residuals vs Leverage: Examines the influence of individual data points on the regression model. High leverage points can have a disproportionate impact on the model’s results. The plot shows a few points with high leverage (towards the right of the plot), but they are not necessarily outliers.

Overall, the plots suggest that the linear model might not be the best fit for the data, and there might be some issues with non-normality and non-constant variance. Further investigation is needed to determine the most appropriate model for the data.