Building Statistical Models in R Linear Regression

Ariel Felices

1/20/2022

# Building Statistical Models in R: Linear Regression

## Task One: Getting Started

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

### 1.1: Get the working directory

getwd()

## [1] "C:/Users/user/Documents/R/Building-Statistical-Model-in-R-Linear-Regression"

## Task Two: Import packages and dataset

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

### 2.1: Importing required packages

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5 v purrr 0.3.4  
## v tibble 3.1.6 v dplyr 1.0.8  
## v tidyr 1.2.0 v stringr 1.4.0  
## v readr 2.1.2 v forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.1.3

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(ggpubr)  
library(broom)  
library(ggfortify)

### 2.2: Import the mpg.csv dataset

data <- read.csv("mpg.csv", header = T, sep = ",")

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

View(data)

dim(data) #This will show the number of rows and columns

## [1] 234 12

## Task Three: Explore the dataset

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

### 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

### 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)) #To find how many missing values in the whole dataset.

## [1] 0

sapply(data, function(x) sum(is.na(x))) #To find how many missing values in each of the variable, use this function sapply.

## 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

colnames(data)

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

OR

names(data)

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

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

data <- data[, -1]

dim(data)

## [1] 234 11

colnames(data)

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

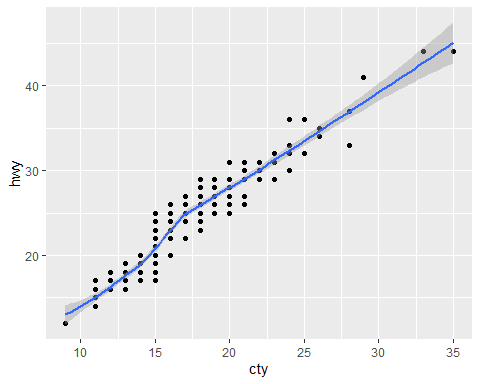
## Task Four: Data Visualizations

In this task, we 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 = data) +  
 geom\_point(aes(x = cty, y = hwy)) +  
 stat\_smooth(aes(x = cty, y = hwy))

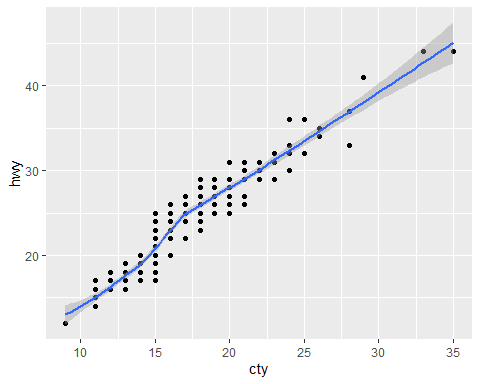
## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



But to simplify the above code, do the following:

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

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



### 4.2: Find the correlation between the variables

cor(data$cty, data$hwy) #the correlation coefficient measures the level of association between two variables X and Y.

## [1] 0.9559159

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, we will learn how to build a simple linear regression model.

### 5.1: Create a simple linear regression model using the variables use the lm function to determine the beta coefficients of this linear model.

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

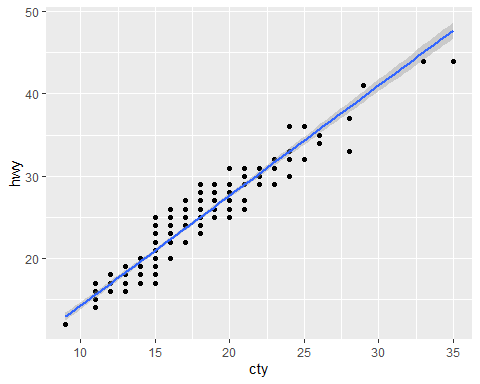
model #call the data

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



## Task Six: Model Assessment I

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

### 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

## Task Seven: Model Assessment II

In this task, we 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

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

Let’s calculate the percentage error to know how much error we have here.

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

## [1] 7.475581

## Task Eight: Model Prediction

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

### 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

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

model\_diag\_metrics <- augment(model)

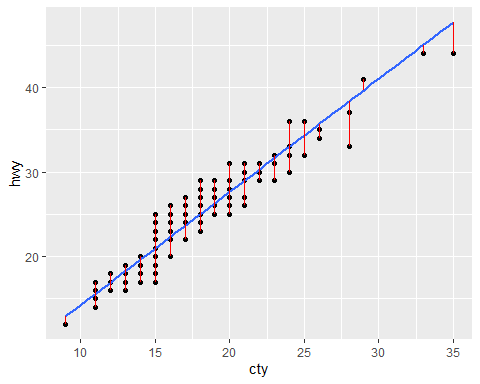
head(model\_diag\_metrics)

## # A tibble: 6 x 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

### 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)

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



### 8.4: Predict new values using the model

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

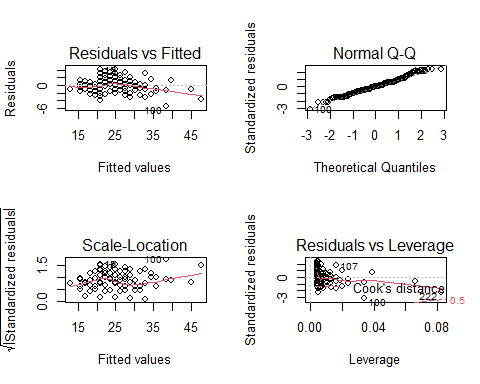
## 1 2 3   
## 28.97861 37.00334 19.61642

## Task Nine: Assumptions Check: Diagnostic Plots

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

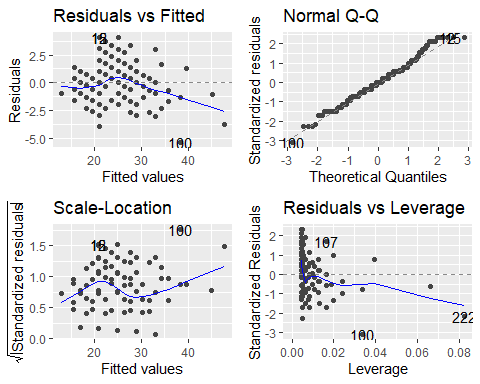
### 9.1: Plotting the fitted model

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



Better Version

autoplot(model)

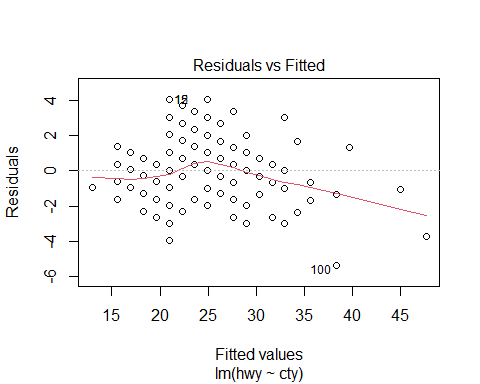


### 9.2: Return par back to default

par(mfrow = c(1, 1))

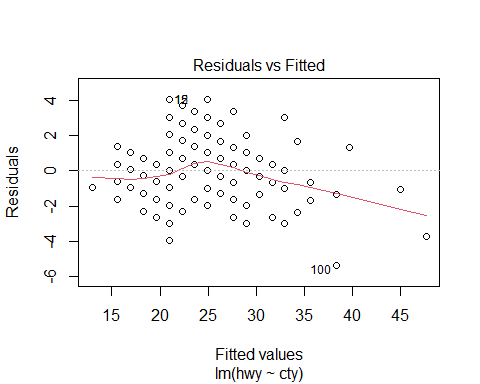
### 9.3: Return the first diagnostic plot for the model

plot(model, 1)



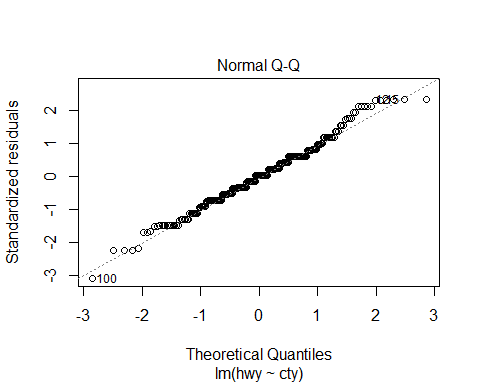
Build another regression model

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



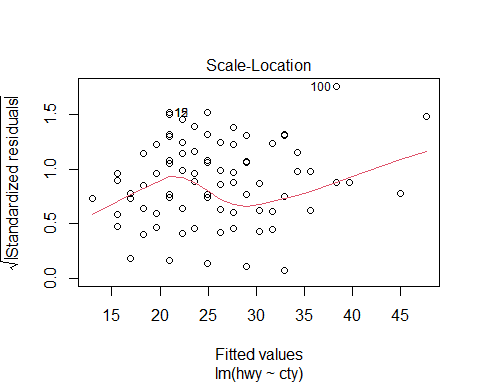
### 9.4: Return the second diagnostic plot for the model

plot(model, 2)



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

plot(model, 3)



## Task Ten: Multiple Regression

In this task, we 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

### 10.4: Plot the fitted multiple regression model

autoplot(mul\_reg\_model)

