Data 621 - Blog 2

Leticia Salazar

November 8, 2022

Contents

Simple Linear Regression

In simple linear regression, we look to find a relationship between two quantitative variables. We want to see how strong this relationship is and if the value of the dependent variable is at a certain value with the independent variable.

Below I'll demonstrate how to create a simple linear regression model using a common package in R called `diamonds` from the library `ggplot2`

Load Libraries

Here we'll use `ggplot2` to be able to load the data set, but it is also a great visualization library. `Tidyverse` is also a great library for exploratory analysis of the data.

```r
library(ggplot2)
library(tidyverse)
library(gtsummary) # great to summarize statistical tables
library(jtools) # use of summ() to view statistical info
```

Loading Data

Using the function `head()` will display the first few rows, therefore we can see the data set without it loading fully.

```
head(diamonds)
```

```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

Data Exploration

Once loaded we can explore the data by using the `summary()`. This will summarize the variables including the following information for the numeric variables:

- minimum value
- first quartile (25th percentile)
- median value
- mean value
- third quartile (75th percentile)
- maximum value

For the rest of the variables we get the frequency count of the values.

```
summary(diamonds)
```

```
##      carat               cut        color        clarity          depth      
##  Min.   :0.2000   Fair     : 1610   D: 6775   SI1    :13065   Min.   :43.00  
##  1st Qu.:0.4000   Good     : 4906   E: 9797   VS2    :12258   1st Qu.:61.00  
##  Median :0.7000   Very Good:12082   F: 9542   SI2    : 9194   Median :61.80  
##  Mean   :0.7979   Premium  :13791   G:11292   VS1    : 8171   Mean   :61.75  
##  3rd Qu.:1.0400   Ideal    :21551   H: 8304   VVS2   : 5066   3rd Qu.:62.50  
##  Max.   :5.0100                     I: 5422   VVS1   : 3655   Max.   :79.00  
##                                     J: 2808   (Other): 2531                  
##      table           price             x                y         
##  Min.   :43.00   Min.   :  326   Min.   : 0.000   Min.   : 0.000  
##  1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710   1st Qu.: 4.720  
##  Median :57.00   Median : 2401   Median : 5.700   Median : 5.710  
##  Mean   :57.46   Mean   : 3933   Mean   : 5.731   Mean   : 5.735  
##  3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540   3rd Qu.: 6.540  
##  Max.   :95.00   Max.   :18823   Max.   :10.740   Max.   :58.900  
##
```

```
##        z
##  Min.   : 0.000
##  1st Qu.: 2.910
##  Median : 3.530
##  Mean   : 3.539
##  3rd Qu.: 4.040
##  Max.   :31.800
##
```

Another exploratory function is `glimpse()` which provides a rundown of the columns along with how many rows and columns the data set has.

```
glimpse(diamonds)
```

```
## Rows: 53,940
## Columns: 10
## $ carat   <dbl> 0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23, 0.~
## $ cut     <ord> Ideal, Premium, Good, Premium, Good, Very Good, Very Good, Ver~
## $ color   <ord> E, E, E, I, J, J, I, H, E, H, J, J, F, J, E, E, I, J, J, J, I,~
## $ clarity <ord> SI2, SI1, VS1, VS2, SI2, VVS2, VVS1, SI1, VS2, VS1, SI1, VS1, ~
## $ depth   <dbl> 61.5, 59.8, 56.9, 62.4, 63.3, 62.8, 62.3, 61.9, 65.1, 59.4, 64~
## $ table   <dbl> 55, 61, 65, 58, 58, 57, 57, 55, 61, 61, 55, 56, 61, 54, 62, 58~
## $ price   <int> 326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339, 340, 34~
## $ x       <dbl> 3.95, 3.89, 4.05, 4.20, 4.34, 3.94, 3.95, 4.07, 3.87, 4.00, 4.~
## $ y       <dbl> 3.98, 3.84, 4.07, 4.23, 4.35, 3.96, 3.98, 4.11, 3.78, 4.05, 4.~
## $ z       <dbl> 2.43, 2.31, 2.31, 2.63, 2.75, 2.48, 2.47, 2.53, 2.49, 2.39, 2.~
```

One of my recent favorites functions in R is `gtsummary` which was introduced to me by a peer. It offers a cleaner way to present these statistical information and there's various of customization options. For more information check references for the link on this function.

```
sum1 <- diamonds %>%
  select(c("carat", "depth", "table", "price"))
sum2 <- tbl_summary(sum1)
sum2
```

```
## Table printed with ‘knitr::kable()‘, not {gt}. Learn why at
## https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
## To suppress this message, include ‘message = FALSE‘ in code chunk header.
```

Characteristic	**N = 53,940**
carat	0.70 (0.40, 1.04)
depth	61.80 (61.00, 62.50)
table	57.00 (56.00, 59.00)
price	2,401 (950, 5,324)

Data Preparation

In most data sets, you have to prepare the data before you can start plotting or in this case building models. Since our data set doesn't need any cleaning or transformations we can go straight to model building. I have provided some tips below on how you can start prepping your data.

- Remove duplicate or irrelevant observations
- Fix structural errors
    - these include inconsistencies, typos, "n/a" and "not applicable" in a column, etc
- Filter unwanted outliers
    - careful, not all outliers are harmful when analyzing data
- Handle missing data
    - you can drop the observations with missing values but be mindful before removing it since you can lose information
    - you can input missing values but this may cause to lose integrity of the data
    - you can alter the data to use the null values effectively
- Validate
    - Ensure that the data makes sense in order to present accurate information

---

Model Building

To fit a linear model in R we use the `lm()` function. Some parameters to consider for the function are:

```
lm(fitting_formula, dataframe)
```

- **fitting_formula**: determines the formula for the linear model
- **dataframe**: determines the name of the data frame

Using the we get the statistical information on the selected model. It contains the following information:

- **P-value**: it's associated with the model coefficients, any value less than 0.05 we can say that there's a statistically significant association between the variables.
- **Residual Standard Error**: it's the average distance that the observed values fall from the regression line. The lower the value the closer it is for the regression line to match with the observed data.
- **Multiple** R^2: it's the percentage of the variation in the variables being examined. The larger the R^2 value the better the explanatory variables are able to predict the value of the response variable.
- **Adjusted** R^2: the same as multiple R^2 but it takes into account the number of samples and variables you're using
- **F-Statistic**: global test to check if your model has at least one significant variable (non-zero). When combined with the p-value, you are able to tell the overall significance of the regression model.
- **T-value**: the predictor variable that's calculated as (Estimate) / (Standard Error)
- **Estimate**: shows us the average increase in the response variable associated with a one unit increase in the predictor variable, assuming all other predictor variables are constant.

```r
# model 1 will consist of looking at the data overall
model <- lm(diamonds)
summ(model)
```

Observations	53940
Dependent variable	carat
Type	OLS linear regression

F(23,53916)	102541.84
R^2	0.98
Adj. R^2	0.98

	Est.	S.E.	t val.	p
(Intercept)	-1.66	0.02	-69.51	0.00
cut.L	-0.02	0.00	-14.53	0.00
cut.Q	0.01	0.00	8.19	0.00
cut.C	-0.01	0.00	-7.08	0.00
cut^4	0.00	0.00	2.24	0.02
color.L	0.11	0.00	97.30	0.00
color.Q	0.04	0.00	41.46	0.00
color.C	0.01	0.00	6.24	0.00
color^4	-0.01	0.00	-5.97	0.00
color^5	0.01	0.00	7.13	0.00
color^6	0.00	0.00	2.69	0.01
clarity.L	-0.18	0.00	-86.67	0.00
clarity.Q	0.11	0.00	60.54	0.00
clarity.C	-0.06	0.00	-37.39	0.00
clarity^4	0.02	0.00	14.26	0.00
clarity^5	-0.01	0.00	-12.46	0.00
clarity^6	-0.00	0.00	-2.08	0.04
clarity^7	0.00	0.00	0.23	0.82
depth	0.01	0.00	43.28	0.00
table	0.00	0.00	12.07	0.00
price	0.00	0.00	231.49	0.00
x	0.24	0.00	134.73	0.00
y	0.01	0.00	4.92	0.00
z	0.00	0.00	2.18	0.03

Standard errors: OLS

Using the `summ()` or `summary()` functions we get a clear view of the model fit values. Let's create a model to view the relationship between carat and price:

```r
# two variables used are carat over price
model_2 <- lm(carat ~ price, diamonds)
#summary(model_2)
summ(model_2)
```

The equation for a simple regression model is: $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_i x_i + \hat{\epsilon}_i$ where ϵ is the residual standard error ($\hat{\sigma}$).

Observations	53940
Dependent variable	carat
Type	OLS linear regression

F(1,53938)	304050.91
R^2	0.85
Adj. R^2	0.85

	Est.	S.E.	t val.	p
(Intercept)	0.37	0.00	330.17	0.00
price	0.00	0.00	551.41	0.00

Standard errors: OLS

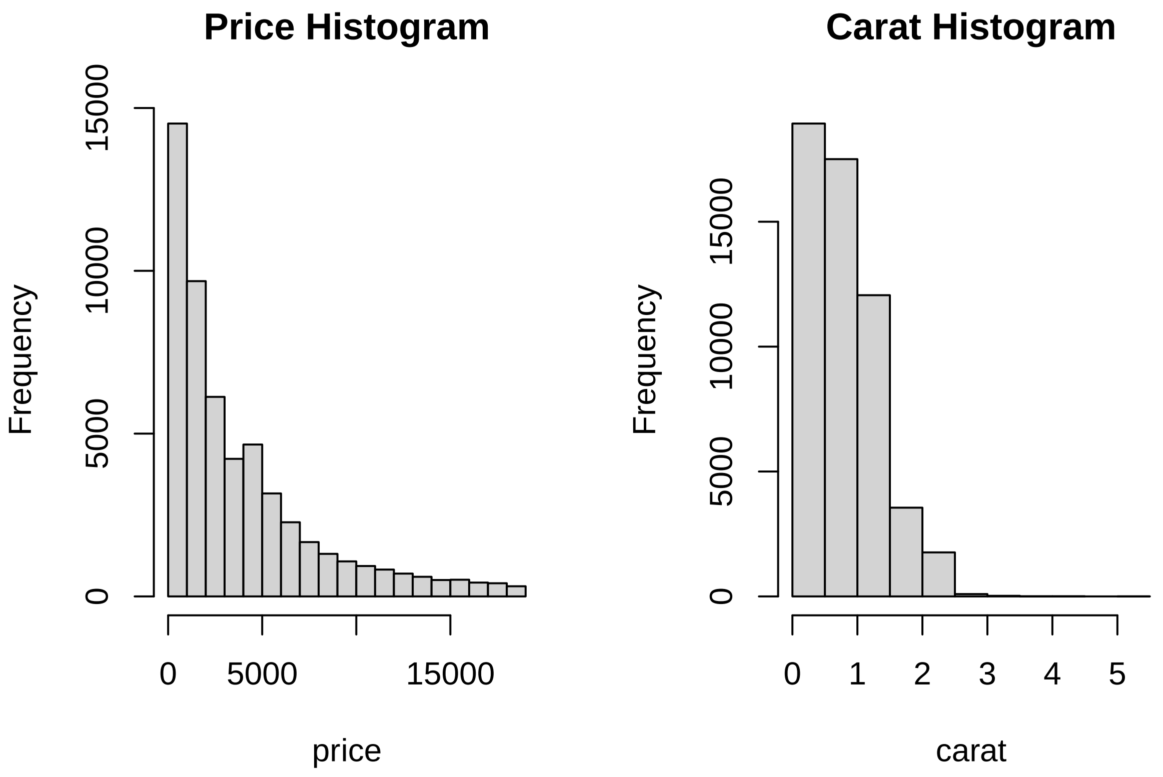
Plotting

Not only does displaying the model's `summary()` function helpful to see if the model we have created describes a relationship between dependent and independent variables. Plotting the model is helpful in viewing these findings.

For a simple linear regression model we create a histogram as well as a Normal Q-Q Plot. The histogram will allow us to see if there's a normal distribution.

Based on the plots below there's a right skeweness in the two variables we are using for `model_2`.

```
# histogram
par(mfrow=c(1,2))
hist(diamonds$price, main = "Price Histogram", xlab = "price")
hist(diamonds$carat, main = "Carat Histogram", xlab = "carat")
```

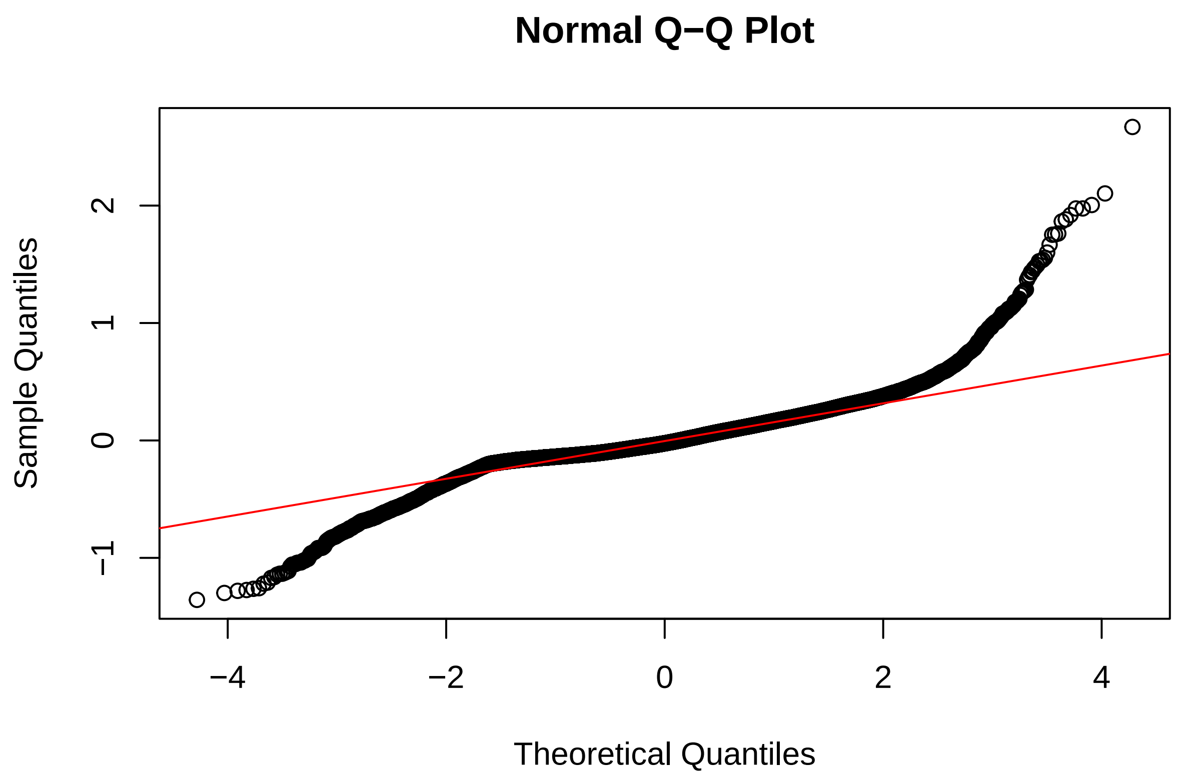


The Q-Q plot will show the distribution of the data against the expected normal distribution. Our model's normal Q-Q plot shows that the qqline (in red), which passes through the first and third quartiles, passes through most of the residuals but there a few that don't.

```r
# define residuals
res <- resid(model_2)

#create Q-Q plot for residuals
qqnorm(res)

#add a straight diagonal line to the plot
qqline(res, col = "red")
```



Based on our model and plots we can assume that the model follows a nearly normal distribution and there's a strong relationship between the carat and price.

References:

- Zach. (2020, October 26). How to perform simple linear regression in R (step-by-step). Statology. Retrieved October 24, 2022, from https://www.statology.org/simple-linear-regression-in-r/
- Guide to data cleaning: Definition, benefits, components, and how to clean your data. Tableau. (n.d.). Retrieved October 24, 2022, from https://www.tableau.com/learn/articles/what-is-data-cleaning

- How to use LM() function in R to fit linear models? GeeksforGeeks. (2021, December 19). Retrieved October 24, 2022, from https://www.geeksforgeeks.org/how-to-use-lm-function-in-r-to-fit-linear-models/#:~:text=The%20lm()%20function%20is,not%20in%20the%20data%20frame.
- Sjoberg DD, Whiting K, Curry M, Lavery JA, Larmarange J. Reproducible summary tables with the gtsummary package. The R Journal 2021;13:570–80. https://doi.org/10.32614/RJ-2021-053.
- Zach. (2020, October 26). How to perform simple linear regression in R (step-by-step). Statology. Retrieved November 1, 2022, from https://www.statology.org/simple-linear-regression-in-r/