Regressions

Monika Baloda (mbalo005@ucr.edu)

In this R-markdown, I do basic regressions. also show some visualizations and data cleaning steps before jumping to regressions. Main steps done in this exercise can be put in the following points:

1. Population regression line vs least squares regression line
2. Estimating the coefficients β_0 and β_1
3. Assess accuracy of the regression model
4. The R^2 statistics
5. Simulations of least squares regression

Load necessary packages

```
library(tidyverse) # for `ggplot2`, `dplyr`, and more
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.1     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.0     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
library(datasets) # for `state` data set
library(nycflights13) # for the 2013 NYC flights data set
```

Simple linear regression

1) Population regression line vs least squares regression line

In this question, we will reproduce the left panel of Figure 3.3 in the **ISLR** textbook.

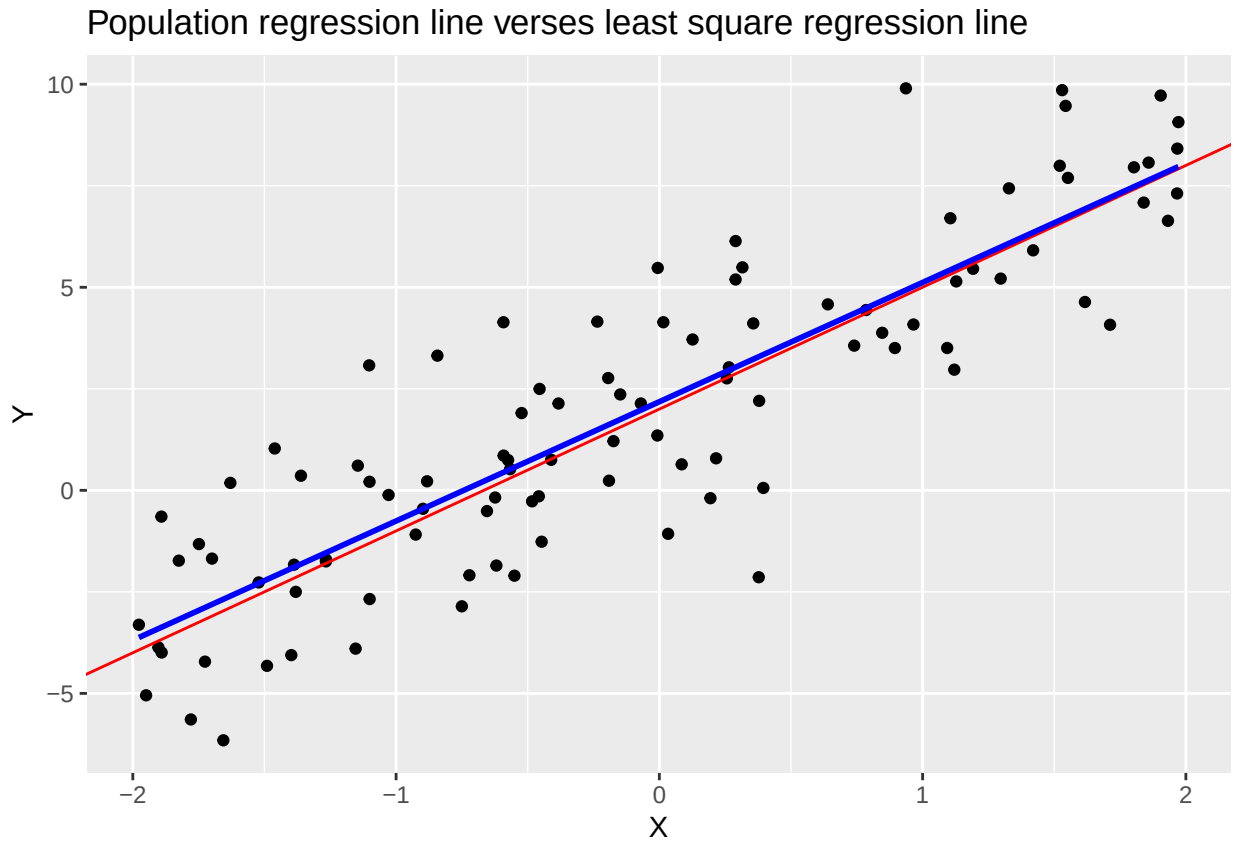
We write R code to simulate 100 data points from the following model.

$$Y = 2 + 3X + \epsilon$$
$$X \sim Uniform(A = -2, B = 2)$$
$$\epsilon \sim N(\mu = 0, \sigma = 2)$$

Then WE use `ggplot2` to make a scatter plot of the 100 data points $(x_1, y_1), \cdots (x_{100}, y_{100})$. Add the population regression line $Y = 2 + 3X$ in red color; add the least squares fit of your simulated data as a blue line.

ANSWER

```r
#setting the seed of the random number
set.seed(209)
n = 100
#creating n random numbers from a uniform distribution between -2 and 2
x =runif(n, min = -2, max = 2)
eps= rnorm(n, mean = 0, sd = 2)
y= 2 + 3*x + eps

data= data.frame(x, y)

ggplot(data, aes(x, y)) +
  geom_point() +
  geom_abline(intercept = 2, slope = 3, colour = "red") +
  geom_smooth(method = "lm", se = FALSE, colour = "blue") +
  labs(x = "X", y = "Y", title = "Population regression line verses least square regression line")
## `geom_smooth()` using formula = 'y ~ x'
```



2) Estimating the coefficients β_0 ***and*** β_1

The analytical solution of the least squares regression $Y = \beta_0 + \beta_1 X + \epsilon$ is

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$
$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Using R code we calculate $\hat{\beta}_0$ and $\hat{\beta}_1$ using the above equations.

ANSWER

```
# Calculate beta_0 and beta_1
beta_1= sum((x - mean(x)) * (y - mean(y)))/ sum((x -mean(x))^2)
beta_0= mean(y) - beta_1 * mean(x)

# Print the estimated coefficients
cat("beta_0:", beta_0, "\n")
## beta_0: 2.181763
cat("beta_1:", beta_1, "\n")
## beta_1: 2.936674

# to fit a linear model to the data
model= lm(y ~ x)

# extract the estimated coefficients of a fitted model
coef(model)
## (Intercept)           x
##    2.181763    2.936674
```

Question Compare your results with the coefficients calculated by the `lm` function. Are they the same?

Answer Yes, we get approximately the same coefficients from both the methods.

3) Assess accuracy of the regression model

Recall that the accuracy of a linear regression fit is typically assessed using the **residual standard error (RSE)** and the R^2 **statistic**.

i) Calculating the residual standard error (RSE)

Residual standard error (RSE) is an estimate of σ, the standard deviation of ϵ.

$$\begin{aligned}\text{RSE} = \hat{\sigma} &= \sqrt{\frac{1}{n-2}\text{RSS}} \\ &= \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}\end{aligned}$$

Now writing R code to calculate RSE using the above equation. **ANSWER**

```
#Simulate data
set.seed(209)
x =runif(100, -2, 2)
y = 2 + 3 * x + rnorm(100, mean = 0, sd = 2)

#Fit a linear regression model
lm.fit = lm(y ~ x)

# Calculating RSE using the analytical formula
n =length(y)
rss= sum(resid(lm.fit)^2)
rse=sqrt(rss / (n - 2))
rse
## [1] 1.942596
```

Question Compare your result with the residual standard error calculated by the `lm` function. Are they the same? How does the RSE value relate to our noise model $\epsilon \sim N(\mu = 0, \sigma = 2)$?

Answer The calculated RSE is approximately 1.942596. Comparing this to the residual standard error calculated by the lm function using the summary(lm.fit) command, we observed that they are the same approximatly.

ii) The R^2 statistics

The R^2 value can be obtianed using the following equations.

$$\text{TSS} = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

$$\text{RSS} = \sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2$$

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

Writing codes to calculate R^2 statistics.

```
#Compute R-squared manually
TSS =sum((y - mean(y))^2)
RSS =sum(residuals(lm.fit)^2)
R2 =1 - RSS/TSS
R2
## [1] 0.7569145
#summary(lm.fit) will return a summary of the model's performance
summary(lm.fit)$r.squared
## [1] 0.7569145
```

Question Compare your result with the R-squared value calculated by the `lm` function. Are they the same?

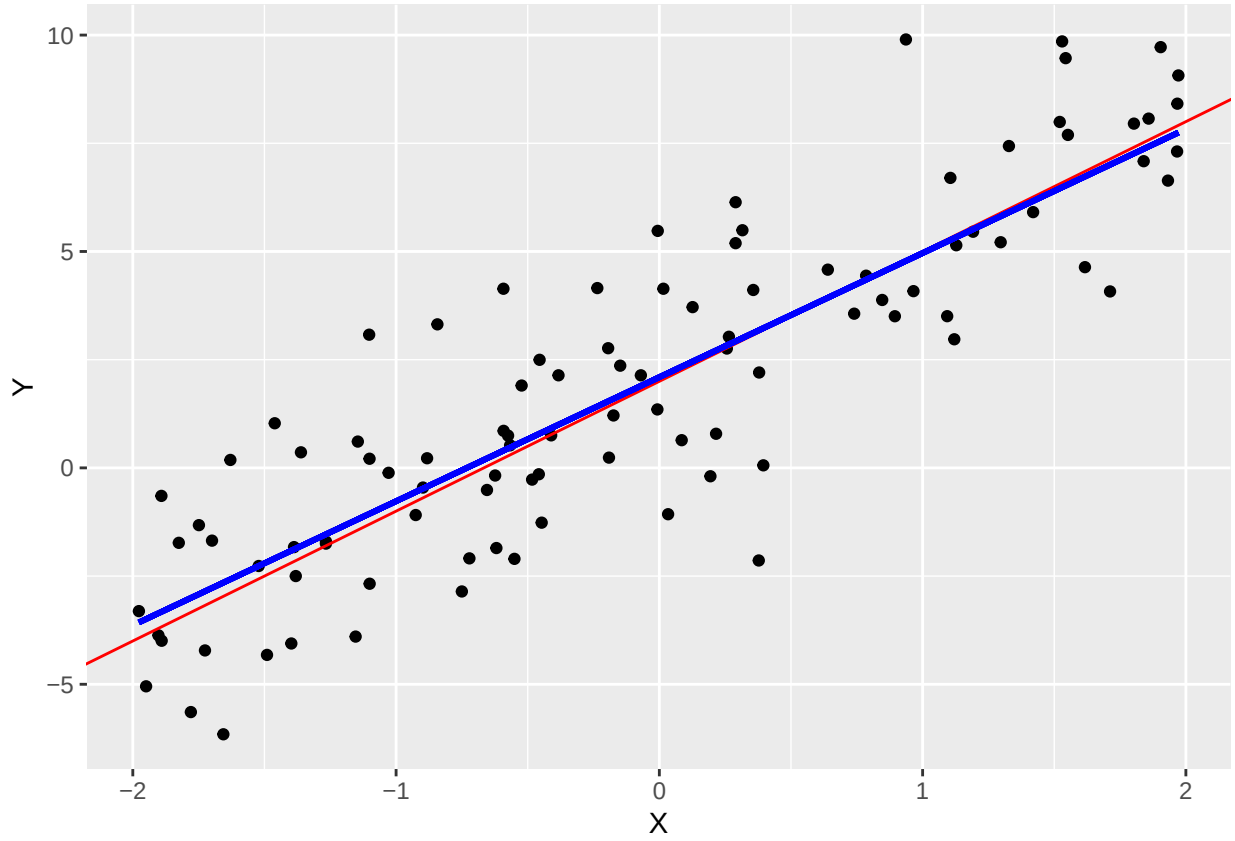
Answer The R^2 value calculated manually is 0.7725554, and the R-squared value that we calculated by the lm function by using summary output is 0.7602692. As we can see that these values are not exactly the same, but they are close to each other.

4. Simulations of least squares regression

Now, let's repeat the simulation 10 times, and reproduce the right panel of Figure 3.3 in the **ISLR** textbook.

```
library(ggplot2)

#set seed for reproducibility
set.seed(209)

# scatter plot of 100 points
n =100
A = -2
B =2
sigma = 2
x = runif(n, A, B)
#true population regression line has an intercept of 2 and a slope of 3
y = 2 + 3*x + rnorm(n, 0, sigma) #larger the value of sigma, more spread out the noise values will be

#creating a data frame with x and y
df=data.frame(x, y)
```

```r
gg= ggplot(df, aes(x, y)) +
  geom_point()+
  geom_abline(intercept = 2, slope = 3, colour= "red") +
  labs(x = "X", y = "Y")

for (i in 1:10) {
  y_sim = 2 + 3*x + rnorm(n, 0, sigma)
  # fit a linear regression model
  lm.fit = lm(y_sim ~ x)
  # add the least squares regression line to the plot
  #se=FALSE to suppress the display of confidence intervals around the lines
  gg= gg + geom_smooth(method = "lm", se = FALSE, color = "blue",
                       aes(y = y_sim, fill = NULL))}
gg
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



***