Linear Regression

Trey Davidson

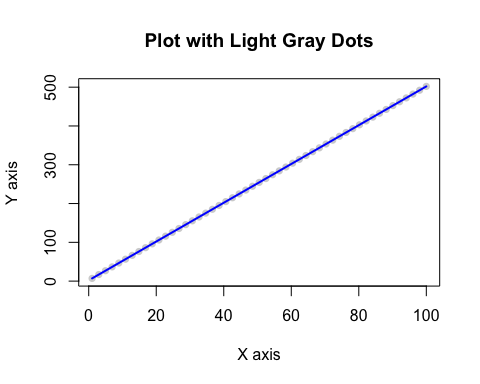
2024-11-16

# Simple Linear Regression Model

Assumes 1.) and are random variables 2.) e is normally distributed around 0

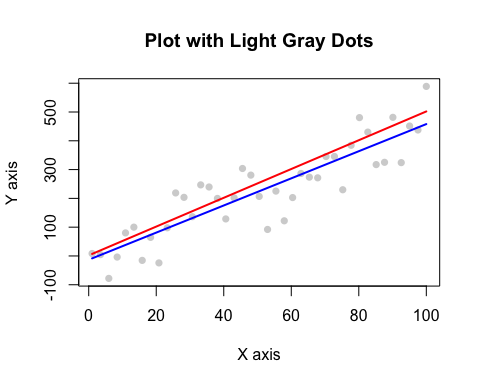
import numpy as np  
  
def lin\_regress\_one(x, y):  
 # Convert inputs to numpy arrays  
 x = np.array(x)  
 y = np.array(y)  
   
   
 x\_mean = np.mean(x)  
 y\_mean = np.mean(y)  
   
   
 numerator = np.sum((x - x\_mean) \* (y - y\_mean))  
 denominator = np.sum((x - x\_mean) \*\* 2)  
 beta\_1 = numerator / denominator  
   
   
 beta\_0 = y\_mean - beta\_1 \* x\_mean  
   
 return [beta\_0, beta\_1]

set.seed(10)  
n = 50  
l = 1  
h = 100  
beta\_1 = 2  
beta\_2 = 5  
  
t <- (h - l) / n  
  
x <- seq(l, h, by = t)  
y <- beta\_1 + x \* beta\_2 + rnorm(length(x), 0, 0)  
  
plot(x, y, col = "lightgray", pch = 16, main = "Plot with Light Gray Dots",  
 xlab = "X axis", ylab = "Y axis")  
  
# Use reticulate to call the Python function  
parameters <- py$lin\_regress\_one(x, y)  
  
parameters <- as.numeric(parameters)  
  
  
# Access the individual elements from the result  
intercept <- parameters[1]  
slope <- parameters[2]  
  
y\_hat <- intercept + slope \* x  
  
lines(x, y\_hat, col = "blue", lwd = 2)



You can see our model perfeclty matches the sample data. This happens because our real data has no variance. Meaning a perfect model can be found. Now if we raise the variance in our model the following will happen.

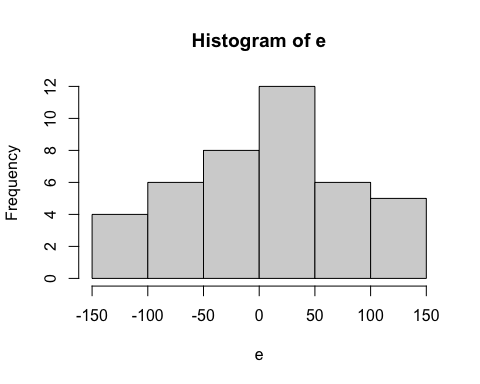
set.seed(10)  
n = 40  
l = 1  
h = 100  
beta\_1 = 2  
beta\_2 = 5  
  
t <- (h - l) / n  
  
x <- seq(l, h, by = t)  
y <- beta\_1 + beta\_2 \* x + rnorm(length(x), 0, 80)  
  
plot(x, y, col = "lightgray", pch = 16, main = "Plot with Light Gray Dots",  
 xlab = "X axis", ylab = "Y axis")  
  
# Use reticulate to call the Python function  
parameters <- py$lin\_regress\_one(x, y)  
  
parameters <- as.numeric(parameters)  
  
  
# Access the individual elements from the result  
intercept <- parameters[1]  
slope <- parameters[2]  
  
y\_hat <- intercept + slope \* x  
  
lines(x, y\_hat, col = "blue", lwd = 2)  
  
lines(x, beta\_1 + x \* beta\_2, col = "red", lwd = 2)



Since we added variance to the model, the model we predict deviates from the model that is used to generate this data.This is because of + e thats in the real model vs our predictive model. We can calculate e using the following.

We call e a residual. According to one of our assumption, e is normally distributed at 0. We can see if this is true by using a histogram.

e <- y - y\_hat  
  
hist(e)

 As you can see e follows close to a normal dist. If we take the variance of e we can calculate how good of a fit our model is. The closer to 0 it is the better of a fit it is.

sourceCpp("LinearRegressionSupport.cpp")  
  
 var\_residuals <- variance\_residuals(e, length(x), 2)  
   
 std\_dev <- sqrt(var\_residuals)  
   
   
 var\_residuals

## [1] 5343.921

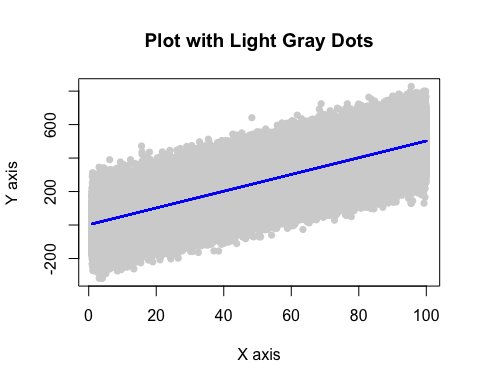
std\_dev

## [1] 73.10213

Since we know that e is close to norm dist and standard deviation = 73.10213, we know that aprox 68% of the data is withen 73.10213 of our predicted value.

Our standered deviation makes sense because our sample size is small at 40 and we used rnorm to generate our data and used a standard deviation of 80 for it. If we run the same code and increase sample size to 1 million, then we should expect our standard deviation of e to be 80.

set.seed(10)  
n = 1000000  
l = 1  
h = 100  
beta\_1 = 2  
beta\_2 = 5  
  
t <- (h - l) / n  
  
x <- seq(l, h, by = t)  
y <- beta\_1 + beta\_2 \* x + rnorm(length(x), 0, 80)  
  
plot(x, y, col = "lightgray", pch = 16, main = "Plot with Light Gray Dots",  
 xlab = "X axis", ylab = "Y axis")  
  
# Use reticulate to call the Python function  
parameters <- py$lin\_regress\_one(x, y)  
  
parameters <- as.numeric(parameters)  
  
  
# Access the individual elements from the result  
intercept <- parameters[1]  
slope <- parameters[2]  
  
y\_hat <- intercept + slope \* x  
  
  
lines(x, beta\_1 + x \* beta\_2, col = "red", lwd = 2)   
  
  
lines(x, y\_hat, col = "blue", lwd = 2)



e <- y - y\_hat  
  
var\_residuals <- variance\_residuals(e, length(x), 2)  
   
std\_dev <- sqrt(var\_residuals)  
   
   
var\_residuals

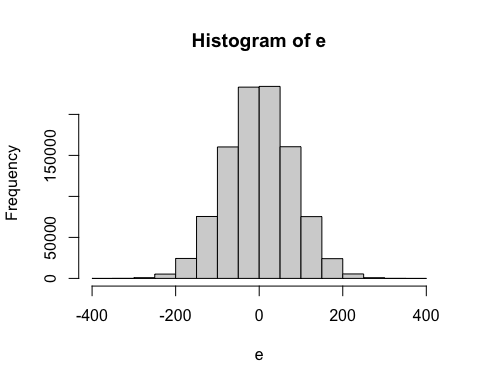
## [1] 6414.736

std\_dev

## [1] 80.09204

Like expected standard deviation of residuals is = to almost exactly 80. In addition, we would expect our residuals to follow exaclty the norm dist.

hist(e)



As you can see its symetrical around the mean of 0. This follows all of our assumptions.