Stat 401 (Spring 2020) - Project 3

Jason Park/116333818

- 0.1 expected values
- 0.2 standard deviations y_hat

#1 Constructing Functions for Linear Regression Models #
#Define R functions to compute test statistics related to linear regression models depending #of bivariate input data (x, y). #1. Sxx, Sxy, and Syy. #function for S_xx, s_xy, and s_yy

```
s_xx <- function(x){
  sum(x*x) - ((sum(x))^2) / length(x)
}

s_xy <-function(x,y){
  sum(x*y) - (sum(x)*sum(y))/length(x)
}
s_yy <- function(y){
  sum(y*y) - ((sum(y))^2) / length(y)
}
```

#2. etimates for ß0, ß1, s. #beta_0_hat: estimates beta_0 #sig_hat: estimates standard deviation #y_hat: beta_0_hat + beta_1_hat* x # sse = sum(y-y_hat)^2 # vatiance_hat : sse/ (n-2)

```
beta_0_hat <- function(x,y){
  mean(y) - beta_1_hat(x,y)*mean(x)
}
#beta_1_hat: estimate beta_1
beta_1_hat <- function(x,y){
  s_xy(x,y)/ s_xx(x)
}

y_hat <- function(x,y){
  beta_0_hat(x,y) + beta_1_hat(x,y)*x
}

residuals<- function(x,y){
  y - y_hat(x,y)
}

sse <- function(x,y){
  sum(residuals(x,y)^2)
}


sig_hat <- function(x,y){
  var_hat <- sse(x,y)/(length(x)-2)
  sqrt(var_hat)
}
```

#3. coefficient of determination r # R^2 = 1- (sse/sst)

```
sst <- function(y){
  s_yy(y)
}
r_sq <- function(x,y){
  1- (sse(x,y)/ sst(y))
}
```

#4. standard errors for ßˆ1 and Yˆ . #sd error _beta_1_hat

```
sd_beta_1_hat <- function(x,y){
  sig_hat(x,y)/ sqrt(s_xx(x))
}

  #sd error- y_hat

s_y_hat <-function(x,y, x_star){
  n <- length(x)
  inside <- (1/n)+(((x_star - mean(x))^2/(s_xx(x))))
  sig_hat(x,y)*sqrt(inside)
}
```

#5. correlation coefficient r.

```
corr<- function(x,y){
  s_xy(x,y)/ (sqrt(s_xx(x))*sqrt(s_xx(y)))
}
```

#Run simulations to compute the empirical expected values, standard deviations, and confidence intervals for the random variables (defined in class/text). Verify that the empirical #values match the theoretical values coming from formulas in the text. Use the following #global assignments for your simulations: # beta_0 <- -20 #beta_1 <- 3.2 #sig <- 26

#1. ˆ beta_1 - the estimator for the slope parameter.

```
beta_0 <- -20
beta_1 <- 3.2
sig <- 26
x_true <- seq(60,100, length.out = 10)
y_true <- beta_0 + (beta_1)* (x_true)
size <- length(x_true)
set.seed(10)
beta_1_hat_sim <- function(num_reps){
  replicate(num_reps,{
              error <-rnorm(size, 0, sig)
              y <- y_true + error
              beta_1_hat(x_true, y)
            }
            )
}
B <- 10000
beta_1s <- beta_1_hat_sim(B) #estimates for the beta_1
mean_beta_1_hat_sim <- mean(beta_1s)
mean_beta_1_hat_sim
```

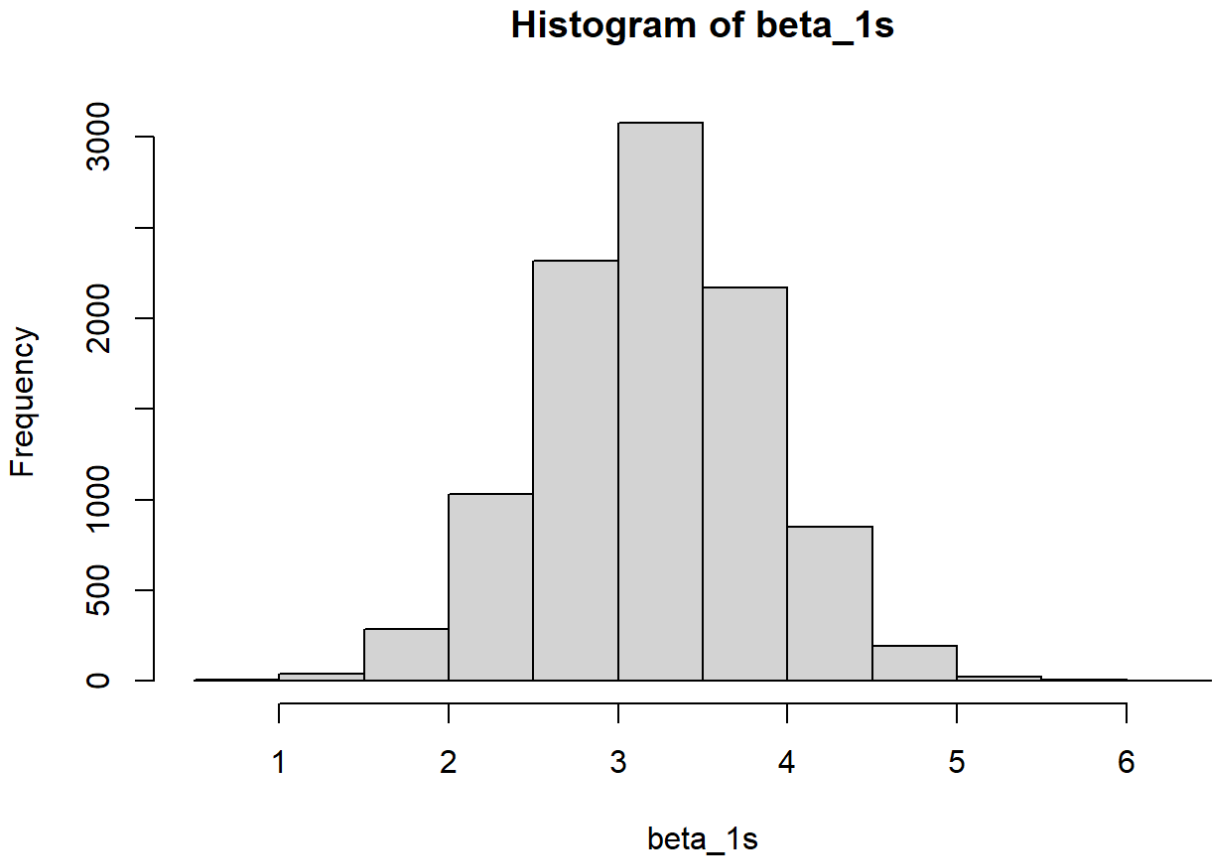
```
## [1] 3.204593
```

#variance & SD of beta_1_hat_sim

```
sd_beta_1_hat <- function(x,y){
  sig_hat(x,y)/ sqrt(s_xx(x))
}


######Standard deviations for beta_1_hat_sim#########################
sd(beta_1s)
```

```
## [1] 0.6471653
```

```
hist(beta_1s)
```




```
#Verify empirical values with theoretical values
sig/sqrt(s_xx(x_true))
```

```
## [1] 0.6440638
```

##Ci for B_1_hat_sim################################## #global variables

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.0     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
beta_0 <- -20
beta_1 <- 3.2
sig <- 26
x_true <- seq(60,100, length.out = 10)
y_true <- beta_0 + (beta_1)* (x_true)
alpha <- 0.05
B <-10000
prop_beta_1_hat_sim_insideCI <-function(alpha){
  between_check <- replicate(B, {
    sample_size <- length(x_true)
    error <-rnorm(sample_size, 0, sig)
  y <- y_true + error
  beta_hat <-beta_1_hat(x_true, y)
  se<- sd_beta_1_hat(x_true, y)
  df <- sample_size -2
  t_alpha <- qt(alpha/2, df, lower.tail = FALSE)
  between(beta_1, beta_hat - t_alpha*se, beta_hat + t_alpha*se)

  })
  mean(between_check)
}

prop_beta_1_hat_sim_insideCI(alpha)
```

```
## [1] 0.9521
```

#2. Yˆ - the estimator for the expected value of Y for a fixed value of x, *ie* $\mu_{Y \cdot x}$.

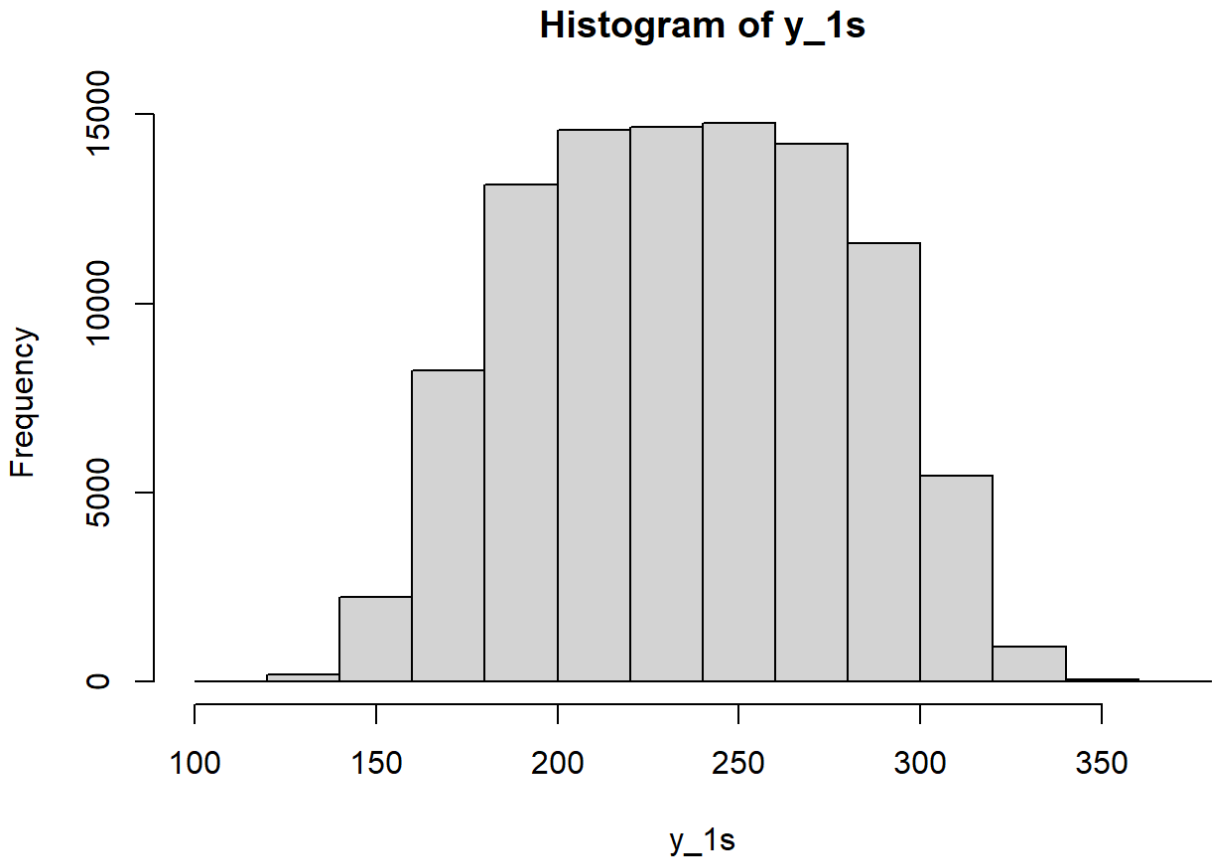
```
#2. Yˆ - the estimator for the expected value of Y for a fixed value of x*, ie μY ·x* .
beta_0 <- -20
beta_1 <- 3.2
sig <- 26
x_true <- seq(60,100, length.out = 10)
y_true <- beta_0 + (beta_1)* (x_true)
size <- length(x_true)

set.seed(10)
y_hat_sim <- function(num_reps){
  replicate(num_reps, {
            error <- rnorm(size, 0, 26)
            y<- y_true + error
            beta_0_hat(x_true, y)+ beta_1_hat(x_true, y)*(x_true)
            })}
## expected values
B <- 10000
y_1s <- y_hat_sim(B)
mean(y_1s)
```

```
## [1] 235.8451
```

```
sd(y_1s)
```

```
## [1] 42.54931
```

```
hist(y_1s)
```



##

standard deviations y_hat

##veryfy empirical values wiht theretical values

```
s_y_1 <- function(x,y){
  (sig_hat(x,y))*sqrt((1/length(x))+( (x - mean(x))^2/s_xx(x,y)))
}


sqrt((26)^2 *((1/size)+((x_true - mean(x_true))^2/s_xx(x_true))))
```

```
##  [1] 15.281599 12.960546 10.900097  9.275579  8.345567  8.345567  9.275579
##  [8] 10.900097 12.960546 15.281599
```

```
sd_beta_1_hat <- function(x,y){
  sig_hat(x,y)/ sqrt(s_xx(x))
}
```

######################CI for Y_hat##

library(tidyverse) x_star <- 60

s_y_hat <-function(x, y){ n <- length(x) inside <- (1/n)+(((x_star - mean(x))^2/(s_xx(x)))) sig_hat(x,y)*sqrt(inside) } beta_0 <- -20 beta_1 <- 3.2 sig <- 26 x_true <- seq(60,100, length.out = 10) y_true <- beta_0 + (beta_1) (x_true) size <- length(x_true) center <- beta_0 + beta_1*x_star alpha <- 0.05 B <-10000 prop_y_hat_sim_insideCI <-function(alpha){ between_check <- replicate(B, { sample_size <- length(x_true) error <-rnorm(sample_size, 0, sig) y <- y_true + error y_hat <- beta_0_hat(x_true, y)+ beta_1_hat(x_true, y) (x_true) se<- s_y_hat(x_true, y) df <- sample_size -2 t_alpha <- qt(alpha/2, df, lower.tail = FALSE) between(center, y_hat - t_alphase, y_hat + t_alphase)

}) mean(between_check) }

prop_y_hat_sim_insideCI(alpha)

#3.V =1/2*ln((1+R)/(1-R))

```
corr <- function(x,y){
  s_xy(x,y)/ (sqrt(s_xx(x))*sqrt(s_xx(y)))
}
v<- function(x,y){
  (1/2)*log((1+corr(x,y))/(1-corr(x,y)))
}

beta_0 <- -20
beta_1 <- 3.2
sig <- 26
x_true <- seq(60,100, length.out = 10)
y_true <- beta_0 + beta_1*x_true
size <- length(x_true)
set.seed(10)
v_rep <- function(num_reps){
  replicate(num_reps, {
    error <- rnorm(size, v(x_true, y_true), 26)
    error
  })}
```

0.1 expected values

#B <- 10000 #v_1s <- v_rep(B) #mean(v_1s)

0.2 standard deviations y_hat

#sd(v_1s) #hist(v_1s)

#V_s <- replicate(B, { y <- y_true + rnorm(length(x_true), 0, 35) v(x_true, y) }) # #mu_V <- mean(v_s)

#(exp(x*mu_v)-1)/ (exp(2*mu_v)+1)

#plot(x_ture, y_true) #y_hat <-(beta_0_hat(x_true, y)) + (beta_1_hat(x_true, y)*x)

0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1

0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.1

##3 Analyzing the Iris Dataset #1.load the data set using the command data.

```
library(datasets)
data(iris)
```

#2.2. use the head, names, summary commands to do some preliminary obersvations of #the data.

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
 names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
 summary(iris)
```

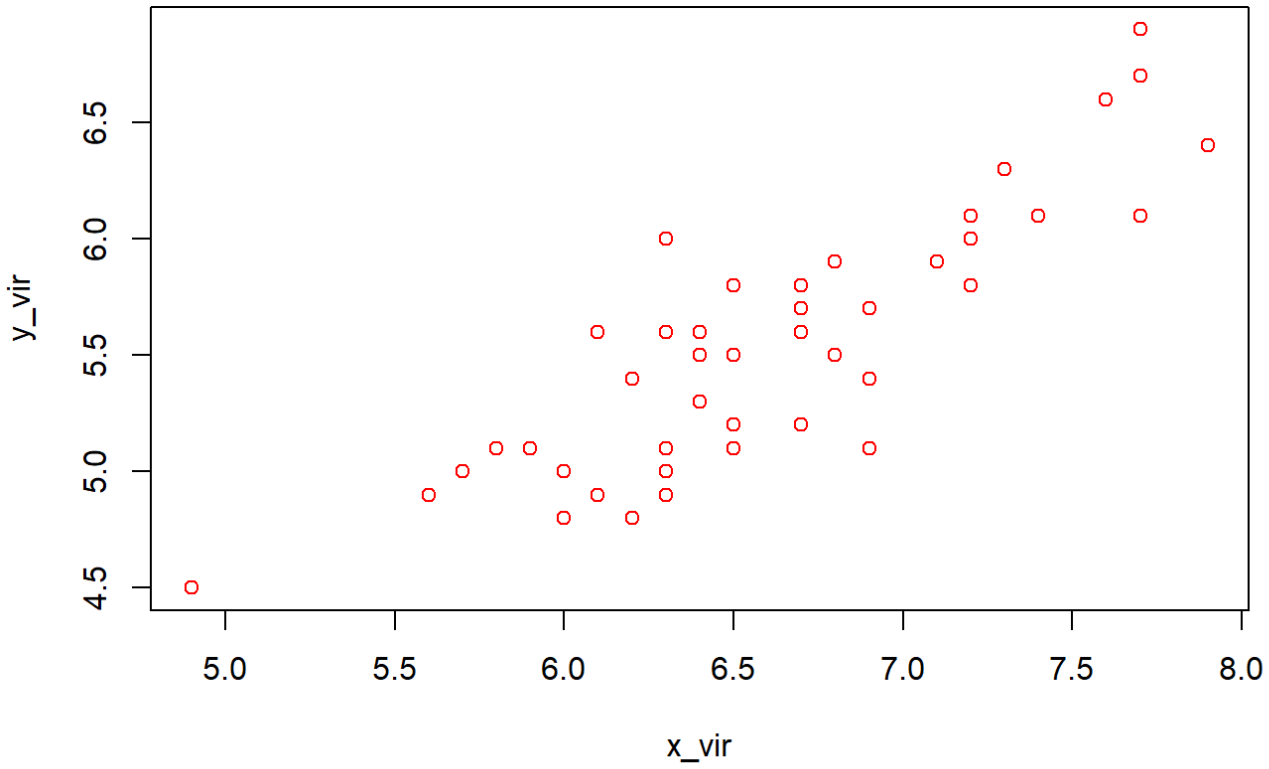
```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

#3.use the filter command to get separate datasets for each of the species, #you should have three separate datasets, #and you should name them by the species names, that # is virginica, setosa, and versicolor.

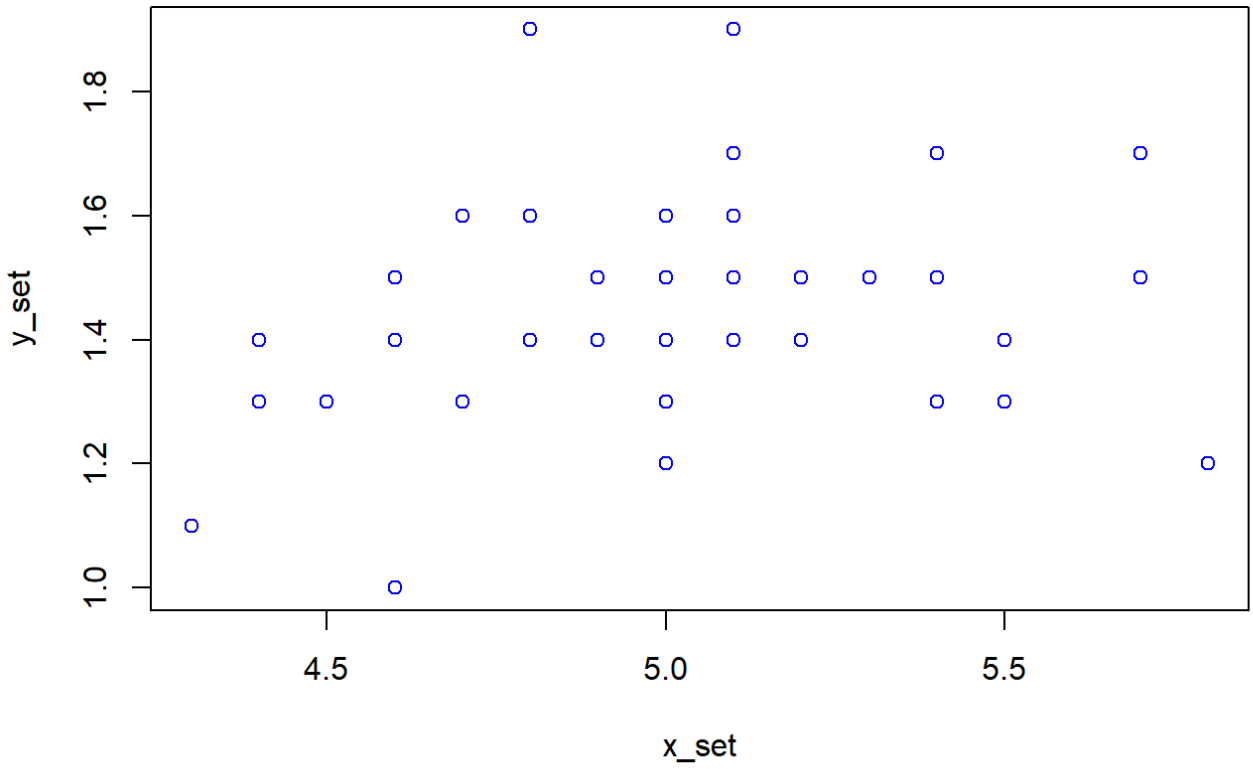
```
library(dplyr)

 virginica <- filter(iris, Species == 'virginica')
 versicolor<-  filter(iris, Species == 'versicolor')
 setosa <-  filter(iris, Species == 'setosa')
```

#4.(a)

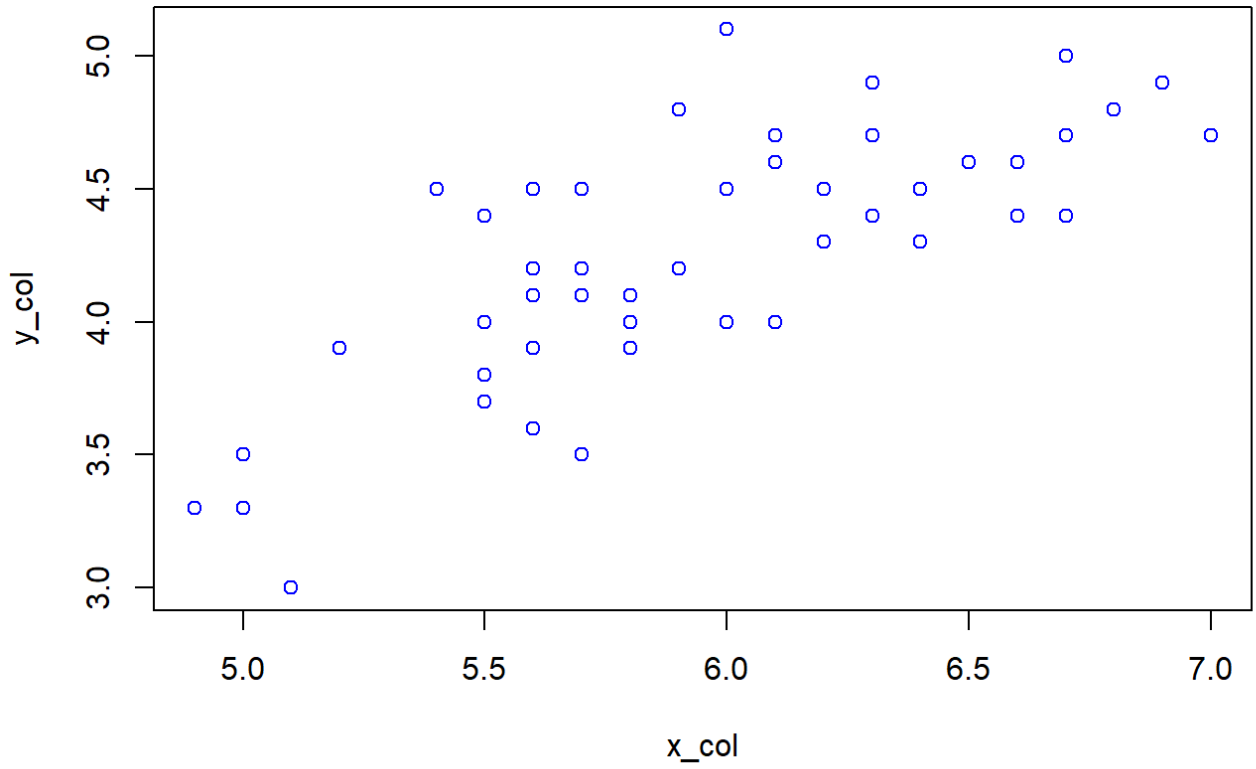
```
 #positive relationship in virginica
 x_vir <- virginica$Sepal.Length
 y_vir <- virginica$Petal.Length
 plot(x_vir,y_vir, col ='red')
```




```
# no relationship in setosa
x_set <- setosa$Sepal.Length
y_set <- setosa$Petal.Length
plot(x_set,y_set, col ='blue')
```




```
#positive relationship in versicolor
x_col <- versicolor$Sepal.Length
y_col <- versicolor$Petal.Length
plot(x_col,y_col, col ='blue')
```



#4.(b)

```
corr_vir <- corr(x_vir, y_vir)
corr_vir
```

```
## [1] 0.8642247
```

```
corr_set <- corr(x_set, y_set)
corr_set
```

```
## [1] 0.2671758
```

```
corr_col <- corr(x_col, y_col)
corr_col
```

```
## [1] 0.754049
```

#4.(c)

```
beta_1_hat(x_vir, y_vir)
```

```
## [1] 0.7500808
```

```
beta_0_hat(x_vir, y_vir)
```

```
## [1] 0.610468
```

```
 sd_beta_1_hat(x_vir, y_vir)
```

```
## [1] 0.06302606
```

```
 r_sq(x_vir, y_vir)
```

```
## [1] 0.7468844
```

```
 lm_vir <- lm(y_vir ~ x_vir)
 summary(lm_vir)
```

```
##
## Call:
## lm(formula = y_vir ~ x_vir)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.68603 -0.21104  0.06399  0.18901  0.66402
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.61047    0.41711   1.464     0.15
## x_vir        0.75008    0.06303  11.901  6.3e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2805 on 48 degrees of freedom
## Multiple R-squared:  0.7469, Adjusted R-squared:  0.7416
## F-statistic: 141.6 on 1 and 48 DF,  p-value: 6.298e-16
```

```
 beta_1_hat(x_set, y_set)
```

```
## [1] 0.1316317
```

```
 beta_0_hat(x_set, y_set)
```

```
## [1] 0.8030518
```

```
 sd_beta_1_hat(x_set, y_set)
```

```
## [1] 0.0685269
```

```
 r_sq(x_set, y_set)
```

```
## [1] 0.07138289
```

```
lm_set <- lm(y_set ~ x_set)
summary(lm_set)
```

```
##
## Call:
## lm(formula = y_set ~ x_set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.40856 -0.08027 -0.00856  0.11708  0.46512
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.80305    0.34388   2.335   0.0238 *
## x_set        0.13163    0.06853   1.921   0.0607 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1691 on 48 degrees of freedom
## Multiple R-squared:  0.07138,    Adjusted R-squared:  0.05204
## F-statistic:  3.69 on 1 and 48 DF,  p-value: 0.0607
```

```
beta_1_hat(x_col, y_col)
```

```
## [1] 0.6864698
```

```
beta_0_hat(x_col, y_col)
```

```
## [1] 0.1851155
```

```
sd_beta_1_hat(x_col, y_col)
```

```
## [1] 0.08630708
```

```
r_sq(x_col, y_col)
```

```
## [1] 0.5685898
```

```
lm_col <- lm(y_col ~ x_col)
summary(lm_col)
```

```
##
## Call:
## lm(formula = y_col ~ x_col)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.68611 -0.22827 -0.04123  0.19458  0.79607
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.18512    0.51421   0.360     0.72
## x_col        0.68647    0.08631   7.954 2.59e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3118 on 48 degrees of freedom
## Multiple R-squared:  0.5686, Adjusted R-squared:  0.5596
## F-statistic: 63.26 on 1 and 48 DF,  p-value: 2.586e-10
```