Ajeya-kempegowda-assignment-2-ds5220

Ajeya Kempegowda

1/24/2019

Problem 1

Data processing and intialization block

```
sheet1<-read_excel("hw2_dataset.xlsx", sheet = 1)
colnames(sheet1) <- as.character(unlist(sheet1[1,]))
sheet1<-sheet1[-1,]
data_sheet1 <- sheet1[!map_lgl(sheet1, ~ all(is.na(.)))]
data_sheet1[, c(2:9)] <- sapply(data_sheet1[, c(2:9)], as.numeric)

#read data from sheet2
data_sheet_2<-read_excel("hw2_dataset.xlsx", sheet = 2)
#global var for data size
data_size = 50
#data simulator function
data_simulator <- function() {
  X <- runif(data_size, -2, 2)
  Y <- 2 + 3*X + rnorm(data_size, 0, 2)
  return(list(X, Y))
}

#get predictor and expected value
data<-data_simulator()
predictor<-data[[1]]
exp_value<-data[[2]]

#get scaled data for ridge and lass regression
get_scaled_data <- function(predictor, data_size) {
  scaled_predictor <- scale(predictor)
  scaled_input_matrix <- cbind(
    c(rep(1, data_size)),
    c(scaled_predictor),
    c(scaled_predictor ^ 2),
    c(scaled_predictor ^ 3),
    c(scaled_predictor ^ 4),
    c(scaled_predictor ^ 5)
    )
  return(scaled_input_matrix)
}
#get 2+3xi+e data
get_data <- function(predictor, data_size) {
  input_matrix <- cbind(
    c(rep(1, data_size)),
    c(predictor),
    c(predictor ^ 2),
```

```r
    c(predictor ^ 3),
    c(predictor ^ 4),
    c(predictor ^ 5)
    )
  return(input_matrix)
}
input_matrix<-get_data(predictor, data_size)
scaled_input_matrix<-get_scaled_data(predictor, data_size)
```

Common utilty functions

```r
get_variance_summary<-function(models){
  ssr = c()
  sse = c()
  for (i in models) {
  ssr <- c(ssr, sum(anova(i)[1, 2]))
  sse <- c(sse, anova(i)[2, 2])
}
#create a dataframe for convenience
  df <- data.frame("fit" = 1:length(models))
  df$ssr <- ssr
  df$sse <- sse
  #sst=sse+ssr
  df$sst <- df$ssr + df$sse
  df<-df %>% mutate(r_squared = 1-(sse/sst))
  return(df)
}
```

Summary statistics

```r
#Mean of columns
map_dbl(data_sheet1[, c(2:9)], mean)
```

```
##       x1       y1       x2       y2       x3       y3       x4       y4
## 9.000000 7.500909 9.000000 7.500909 9.000000 7.500000 9.000000 7.500909
```

```r
#Standard deviations
map_dbl(data_sheet1[, c(2:9)], sd)
```

```
##       x1       y1       x2       y2       x3       y3       x4       y4
## 3.316625 2.031568 3.316625 2.031657 3.316625 2.030424 3.316625 2.030579
```

```r
#correlation
cor(data_sheet1$x1, data_sheet1$y1)
```

```
## [1] 0.8164205
```

```r
cor(data_sheet1$x2, data_sheet1$y2)
```

```
## [1] 0.8162365
```

```r
cor(data_sheet1$x3, data_sheet1$y3)
```

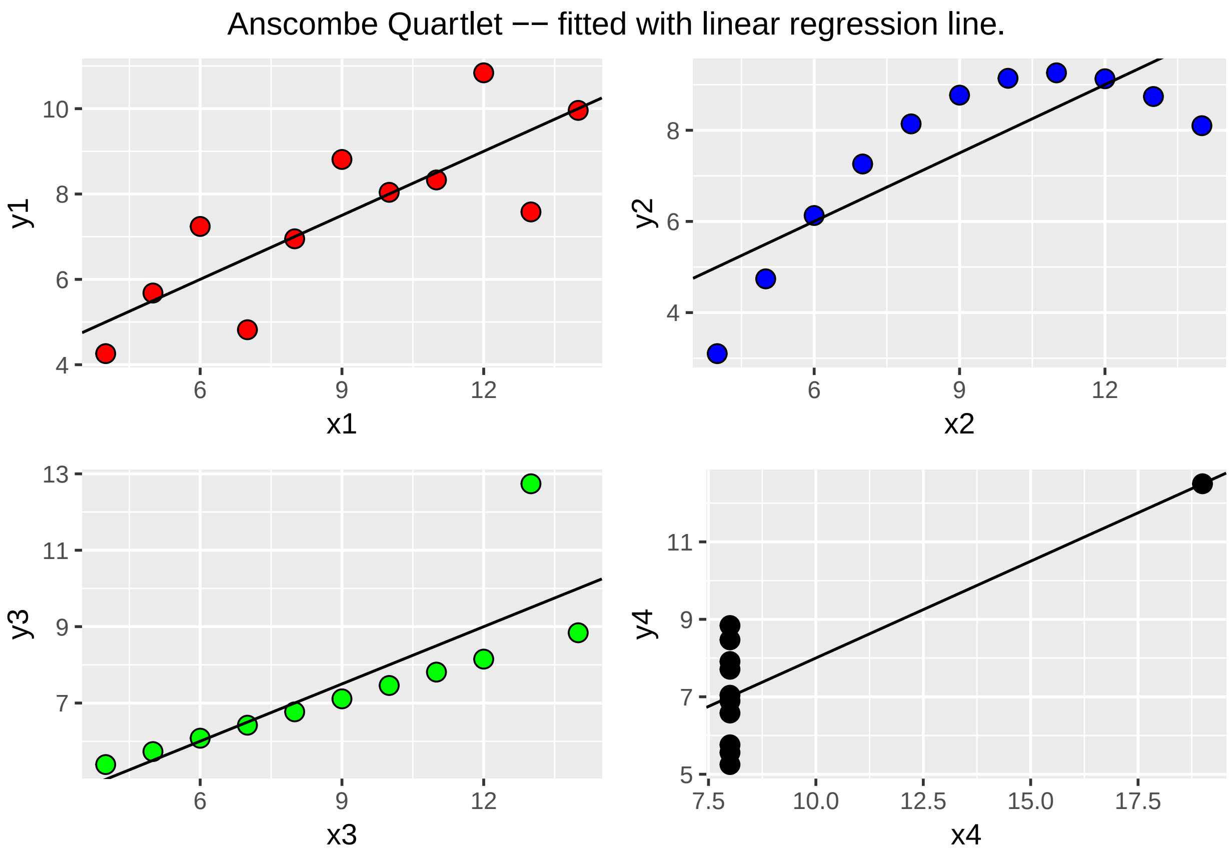
```
## [1] 0.8162867
```

```r
cor(data_sheet1$x4,  data_sheet1$y4)
```

```
## [1] 0.8165214
```

Fitting linear regression

```r
#define the LM regression
fit1 <- lm(y1 ~ x1, data=data_sheet1)
fit2 <- lm(y2 ~ x2, data=data_sheet1)
fit3 <- lm(y3 ~ x3, data=data_sheet1)
fit4 <- lm(y4 ~ x4, data=data_sheet1)

circle.size = 3
colors = list('red', 'blue', 'green', 'black')

#plot1 x1~y1
plot1 <-
  ggplot(data_sheet1, aes(x = x1, y = y1)) + geom_point(size = circle.size, pch =
  21, fill = colors[[1]]) +
  geom_abline(intercept = fit1$coefficients[1],
  slope = fit1$coefficients[2])

#plot2 x2~y2
  plot2 <-
  ggplot(data_sheet1, aes(x = x2, y = y2)) + geom_point(size = circle.size, pch =
  21, fill = colors[[2]]) +
  geom_abline(intercept = fit2$coefficients[1],
  slope = fit2$coefficients[2])

#plot3 x3~y3
  plot3 <-
  ggplot(data_sheet1, aes(x = x3, y = y3)) + geom_point(size = circle.size, pch =
  21, fill = colors[[3]]) +
  geom_abline(intercept = fit3$coefficients[1],
  slope = fit3$coefficients[2])

#plot x4~y4
  plot4 <-
  ggplot(data_sheet1, aes(x = x4, y = y4)) + geom_point(size = circle.size, pch =
  21, fill = colors[[4]]) +
  geom_abline(intercept = fit4$coefficients[1],
  slope = fit4$coefficients[2])

grid.arrange(plot1, plot2, plot3, plot4, top = 'Anscombe Quartlet -- fitted with linear regression line
```



Summaries of the linear regression fit

```r
get_variance_summary(list(fit1, fit2, fit3, fit4))
```

```
##   fit      ssr      sse      sst r_squared
## 1   1 27.51000 13.76269 41.27269 0.6665425
## 2   2 27.50000 13.77629 41.27629 0.6662420
## 3   3 27.47001 13.75619 41.22620 0.6663240
## 4   4 27.49000 13.74249 41.23249 0.6667073
```

Conclusion: From the above results we can see that the coeffcient of correlation, ssr, sse, sst and r_squared are almost similar across all data sets. However when we evaluate fit of the model, x1 and x3 seem to be fairly good predictors while x4 and x2 are not.

Hence, we can conclude that these statistics aren't sufficient to judge the quality of fit of linear regression.

Problem 2

```r
#EDA plot x1~Y
data_2_plot_1 <- ggplot(data_sheet_2, aes(x = X1, y = Y)) + geom_point()

#Fit a linear model
data_2_fit_1 <- lm(Y ~ X1, data_sheet_2)
```

```r
#Fit regression line
data_2_plot_2<-ggplot(data_sheet_2, aes(x = X1, y = Y)) + geom_point() +
  geom_abline(intercept = data_2_fit_1$coefficients[1],
  slope = data_2_fit_1$coefficients[2])

#Indicator variables
cat1.wgt<-ifelse(data_sheet_2$X2==1, 1, 0)
cat2.wgt<-ifelse(data_sheet_2$X2==2, 1, 0)
cat3.wgt<-ifelse(data_sheet_2$X2==3, 1, 0)
cat4.wgt<-ifelse(data_sheet_2$X2==4, 1, 0)

#FIT A LINEAR MODEL
fit_n <- lm(Y ~ cat1.wgt + cat2.wgt + cat3.wgt + cat4.wgt + X1, data_sheet_2)

#visualize
data_2_plot_3 <- ggplot(data_sheet_2) +
  geom_point(aes(x = X1, y = Y, color = as.factor(X2))) + labs(x = "X1") +
  theme(legend.position = "none") +
  geom_abline(intercept = fit_n$coefficients[1],
  slope = fit_n$coefficients[6])+
  geom_abline(intercept = fit_n$coefficients[1]+fit_n$coefficients[2],
  slope = fit_n$coefficients[6])+
  geom_abline(intercept = fit_n$coefficients[1]+fit_n$coefficients[3],
  slope = fit_n$coefficients[6])+
  geom_abline(intercept = fit_n$coefficients[1]+fit_n$coefficients[4],
  slope = fit_n$coefficients[6])+
  geom_abline(intercept = fit_n$coefficients[1]+fit_n$coefficients[5],
  slope = fit_n$coefficients[6])
get_variance_summary(list(data_2_fit_1, fit_n))
```

```
##   fit      ssr      sse      sst r_squared
## 1   1 2515.963 2151.423 4667.386 0.5390518
## 2   2 2019.313 1122.044 3141.357 0.6428154
```

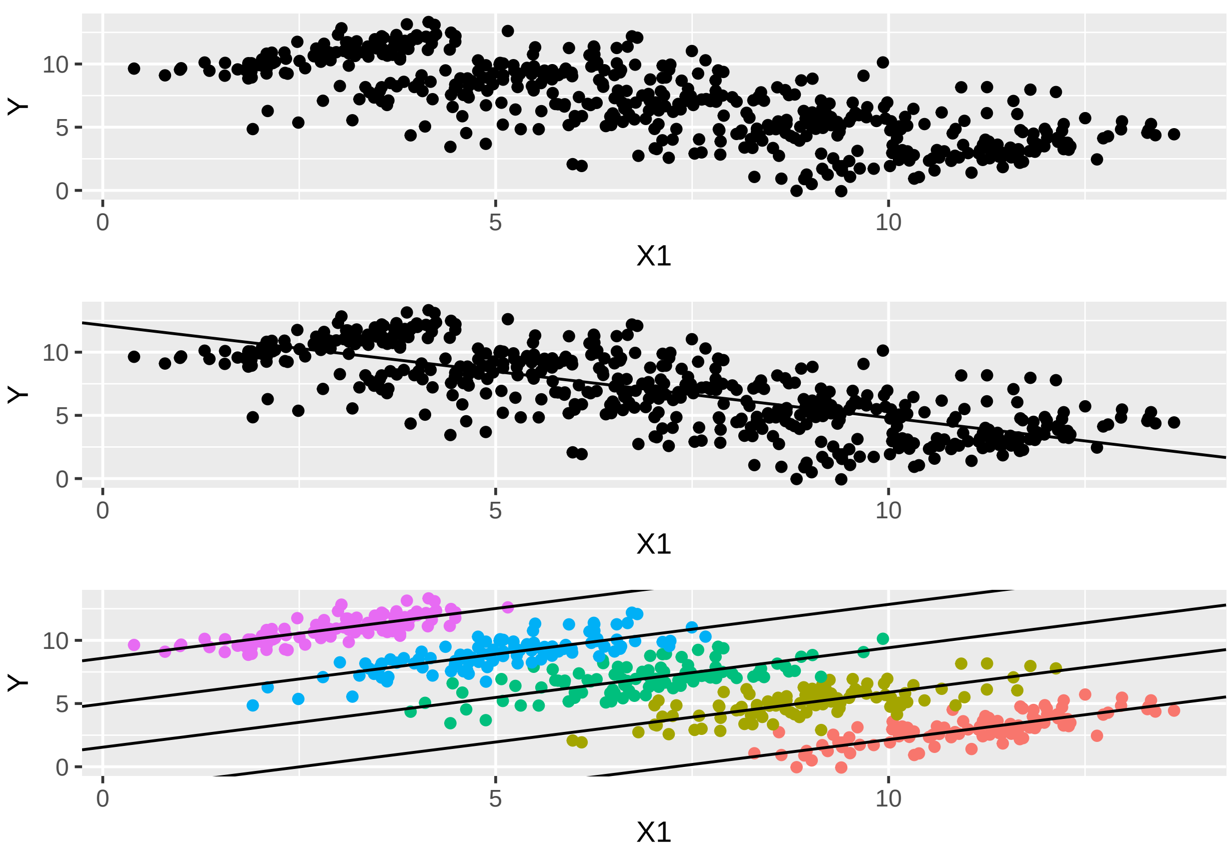
```r
summary(data_2_fit_1)
```

```
##
## Call:
## lm(formula = Y ~ X1, data = data_sheet_2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8797 -1.3597  0.1126  1.4091  5.2637
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.13016    0.23209   52.27   <2e-16 ***
## X1          -0.73209    0.03034  -24.13   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.078 on 498 degrees of freedom
## Multiple R-squared:  0.5391, Adjusted R-squared:  0.5381
## F-statistic: 582.4 on 1 and 498 DF,  p-value: < 2.2e-16
```

```r
summary(fit_n)
```

```
##
## Call:
## lm(formula = Y ~ cat1.wgt + cat2.wgt + cat3.wgt + cat4.wgt +
##     X1, data = data_sheet_2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3108 -0.5577 -0.0615  0.5130  2.0465
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.58684    0.12434   69.06   <2e-16 ***
## cat1.wgt    -14.30740    0.27829  -51.41   <2e-16 ***
## cat2.wgt    -10.56445    0.22111  -47.78   <2e-16 ***
## cat3.wgt     -7.03857    0.16889  -41.68   <2e-16 ***
## cat4.wgt     -3.61181    0.13289  -27.18   <2e-16 ***
## X1            0.78683    0.03139   25.07   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8158 on 494 degrees of freedom
## Multiple R-squared:  0.9296, Adjusted R-squared:  0.9288
## F-statistic:  1304 on 5 and 494 DF,  p-value: < 2.2e-16
```

```r
grid.arrange(data_2_plot_1, data_2_plot_2, data_2_plot_3)
```



Models: $h[\theta(x)] = \theta_0 + \theta_1 x_1$

$h[\theta(x)] = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + +\theta_3 x_3 + \theta_4 x_4 + \theta_5 x_5$

Conclusion: Simpson's paradox, also known as the amalgamation paradox, reversal paradox, or Yule-Simpson effect, is a paradox in which a statistical trend appears to be present when data are segmented into separate groups of data but disappears (or reverses) when the data is considered as a whole.

Using the above definition we can clearly see that when we plot X1 and Y (ref: plot1),we see a clear negative trend of the data and the same can be inferred when we fit a linear model with Y~X1 and obtain a regression line(plot2) with a negative slope. This seems to be pretty reasonable.

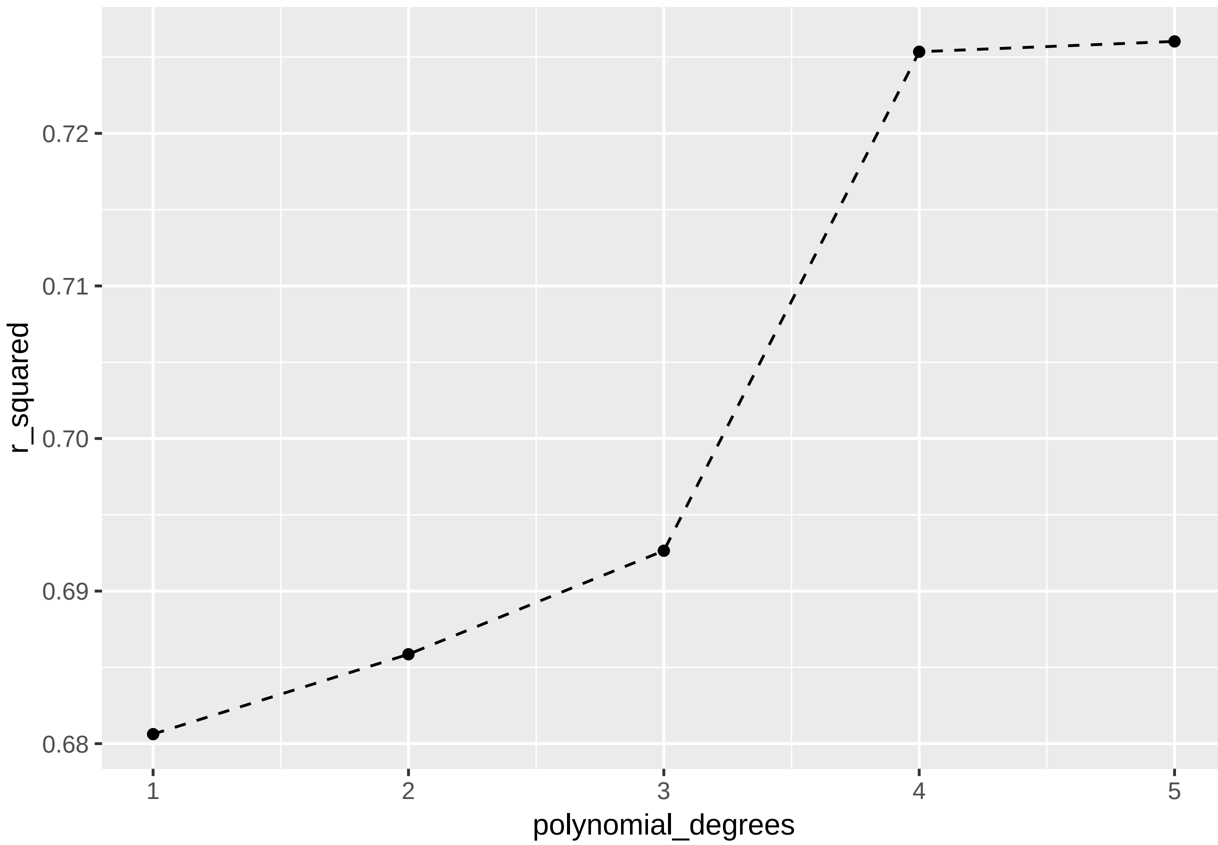
However, when we look closely at the data, we can see five subtle data clusters. To accomodate for this subtlety we change the existing model to a multiple linear regression (as expressed in the above mathematical statement) using indicator varibles. Now, we obtain five different regression lines which seem to express our data better.

We can clearly correlate this situation to Simpson's paradox as the whole statistical trend of the data changed when we re-modeled our initial model using a categorical variable.

Problem 3 - Properties of parameter estimates as function of the number of predictors.

3(a)

```r
#Global var for poly degrees
poly_deg = 5
r2 = c()
for (i in 1:5) {
  r2 <- c(r2, summary(lm(exp_value ~ poly(predictor, i)))$r.squared)
}
poly_summary_df <- data.frame("polynomial_degrees" = c(1:poly_deg))
poly_summary_df$r_squared <- r2
ggplot(
  data = poly_summary_df,
  aes(x = polynomial_degrees, y = r_squared, group =1)) +
geom_line(linetype = "dashed") + geom_point()
```



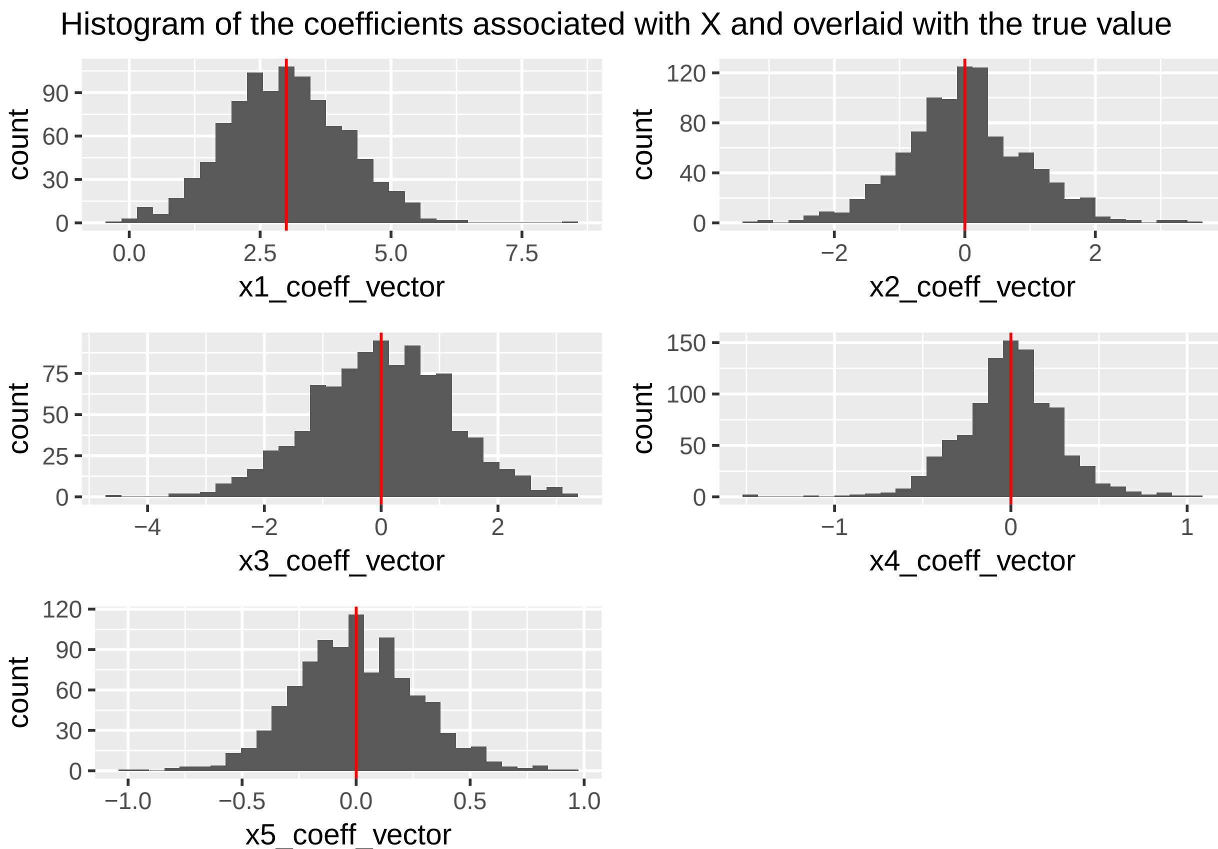
Conclusion: R-squared is a statistical measure of how close the data are to the fitted regression line. The higher the R-squared, the better the model fits the data.

We can see that as we fit more and more explanatory variables to our model, the value of r2 increase(towards 1). i.e r2 gives us a sense of variance (overfitting) in the model. Hence we need to careful increasing the model complexilty.

3(b) Histograms of the coefficients

```
x1_coeff_vector = c()
x2_coeff_vector = c()
x3_coeff_vector = c()
x4_coeff_vector = c()
x5_coeff_vector = c()
line_equation = c()
#Simulating the data as for 1,000 times and fitting a poly model =5
for (iter in 1:1000) {
  data <- data_simulator()
  x <- data[[1]]
  y <- data[[2]]
  coeffs<-summary(lm(y ~ x + I(x ^ 2) + I(x ^ 3) + I(x ^ 4) + I(x ^ 5)))$coefficients
  x_coeff1<-coeffs[2]
  x_coeff2<-coeffs[3]
  x_coeff3<-coeffs[4]
  x_coeff4<-coeffs[5]
  x_coeff5<-coeffs[6]
  intercept<-coeffs[1]
```

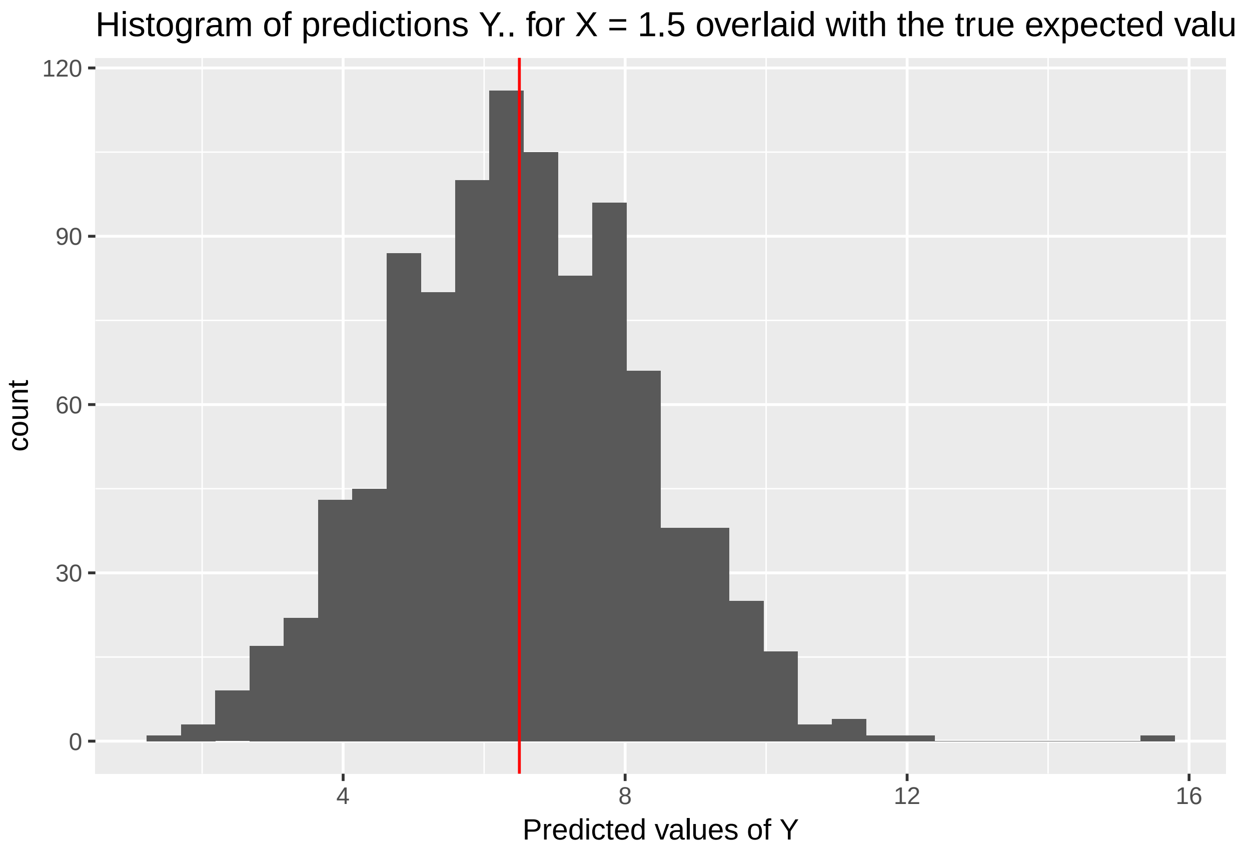
```
  equation=intercept+x_coeff1*1.5
  x1_coeff_vector <-c(x1_coeff_vector, x_coeff1)
  x2_coeff_vector <-c(x2_coeff_vector, x_coeff2)
  x3_coeff_vector <-c(x3_coeff_vector, x_coeff3)
  x4_coeff_vector <-c(x4_coeff_vector, x_coeff4)
  x5_coeff_vector <-c(x5_coeff_vector, x_coeff5)
  line_equation <-c(line_equation, equation)
}
#construct a data frame
x_coeffcients<-as.data.frame(x1_coeff_vector)
x_coeffcients$x2_coeff_vector<-x2_coeff_vector
x_coeffcients$x3_coeff_vector<-x3_coeff_vector
x_coeffcients$x4_coeff_vector<-x4_coeff_vector
x_coeffcients$x5_coeff_vector<-x5_coeff_vector

#Plot the histograms of the coefficient associated with X
histogram_1<-x_coeffcients %>% ggplot() +
  geom_histogram(mapping = aes(x = x1_coeff_vector)) +
  geom_vline(xintercept = 3, col = "red")
histogram_2<-x_coeffcients %>% ggplot() +
  geom_histogram(mapping = aes(x = x2_coeff_vector))+
  geom_vline(xintercept = 0, col = "red")
histogram_3<-x_coeffcients %>% ggplot() +
  geom_histogram(mapping = aes(x = x3_coeff_vector)) +
  geom_vline(xintercept = 0, col = "red")
histogram_4<-x_coeffcients %>% ggplot() +
  geom_histogram(mapping = aes(x = x4_coeff_vector)) +
  geom_vline(xintercept = 0, col = "red")
histogram_5<-x_coeffcients %>% ggplot() +
  geom_histogram(mapping = aes(x = x5_coeff_vector)) +
  geom_vline(xintercept = 0, col = "red")

grid.arrange(histogram_1, histogram_2, histogram_3,histogram_4,histogram_5,
             top="Histogram of the coefficients associated with X and overlaid with the true value")
```




```r
# Plot the histograms of predictions Y^ for X = 1.5 and overlay the true expected value
actual_val<-2+3*1.5
as.data.frame(line_equation) %>% ggplot(mapping = aes(x = line_equation)) +
  geom_histogram() + geom_vline(xintercept = actual_val, col = "red") +
  labs(x = "Predicted values of Y", title = "Histogram of predictions Y^ for X = 1.5 overlaid with the t
```

```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Histogram of predictions Y^ for X = 1.5 overlaid
## with the true expected value' in 'mbcsToSbcs': dot substituted for <cb>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Histogram of predictions Y^ for X = 1.5 overlaid
## with the true expected value' in 'mbcsToSbcs': dot substituted for <86>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Histogram of predictions Y^ for X = 1.5 overlaid
## with the true expected value' in 'mbcsToSbcs': dot substituted for <cb>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Histogram of predictions Y^ for X = 1.5 overlaid
## with the true expected value' in 'mbcsToSbcs': dot substituted for <86>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Histogram of predictions Y^ for X = 1.5 overlaid
## with the true expected value' in 'mbcsToSbcs': dot substituted for <cb>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Histogram of predictions Y^ for X = 1.5 overlaid
## with the true expected value' in 'mbcsToSbcs': dot substituted for <86>
```

```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Histogram of predictions Y^ for X = 1.5 overlaid
## with the true expected value' in 'mbcsToSbcs': dot substituted for <cb>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Histogram of predictions Y^ for X = 1.5 overlaid
## with the true expected value' in 'mbcsToSbcs': dot substituted for <86>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Histogram of predictions Y^ for X = 1.5 overlaid
## with the true expected value' in 'mbcsToSbcs': dot substituted for <cb>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Histogram of predictions Y^ for X = 1.5 overlaid
## with the true expected value' in 'mbcsToSbcs': dot substituted for <86>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Histogram of predictions Y^ for X = 1.5 overlaid
## with the true expected value' in 'mbcsToSbcs': dot substituted for <cb>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Histogram of predictions Y^ for X = 1.5 overlaid
## with the true expected value' in 'mbcsToSbcs': dot substituted for <86>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Histogram of predictions Y^ for X = 1.5 overlaid
## with the true expected value' in 'mbcsToSbcs': dot substituted for <cb>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on 'Histogram of predictions Y^ for X = 1.5 overlaid
## with the true expected value' in 'mbcsToSbcs': dot substituted for <86>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x,
## x$y, : conversion failure on 'Histogram of predictions Y^ for X = 1.5
## overlaid with the true expected value' in 'mbcsToSbcs': dot substituted for
## <cb>

## Warning in grid.Call.graphics(C_text, as.graphicsAnnot(x$label), x$x,
## x$y, : conversion failure on 'Histogram of predictions Y^ for X = 1.5
## overlaid with the true expected value' in 'mbcsToSbcs': dot substituted for
## <86>
```



Conclusion:

Fitting a model with higher order terms (increasing the polynomials in the hypothesis equation) usually results in low bias and high variance. The histograms clearly indicate high density around certain values(3, 0 and 6.5)

In 3(a) we saw that adding more variable increases r2 value or decreases the variance by giving a closer fit to the actual value, but it increases the model complexility after some threshold value.

In this case, except for x1 model co-effcient all other values are zero(insignificant). Therefore if we take only r2 into consideration fit higher order parameters, they might not contribute to anything to improve our model Also, it takes a toll on the computational efficiency. Hence, r2 can be misleading.

Problem 7 -Regularization

7(a)(i) Analytical solution for least squares regression

```
least_squares_analytical <- function(y, X){
  return(solve(t(X)%*%X)%*%t(X)%*%y)
}
```

7(a)(ii) Analytical solution for regularized ridge regression

```
ridge_analytical <- function(x, y, lambda) {
  return(solve(t(x) %*% x + lambda * diag(ncol(x))) %*% (t(x) %*% y))
```

```
}
```

7 (a)(iii) Batch gradient descent optimization for regularized lasso regression

```
lasso_batch_descent <- function(x, y, alpha = 0.001, convergence_fact = 0.0001, lambda){
  i <- 0
  is_converged=F
  N<-nrow(x)
  pred_val<-as.matrix(y)
  theta <- matrix(c(1,1),ncol(x),1)
  J_theta <-
    (1 / (2 * N)) * (t(x %*% theta - pred_val) %*% (x %*% theta - pred_val) +
    lambda * sum(abs(theta)))
  while (!is_converged) {
    theta <-
      theta - (alpha / N) * (2 * t(x) %*% (x %*% theta - pred_val) + lambda *
      sign(theta))
      new_cost <-
      (1 / (2 * N)) * (t(x %*% theta - pred_val) %*% (x %*% theta - pred_val) +
      lambda * sum(abs(theta)))
    if(abs(J_theta - new_cost) <= convergence_fact){
      is_converged = T
    }
    J_theta = new_cost
  }
  return(theta)
}
```

7(b)(i) Analytical implementation -Least squares

```
least_squares_analytical(as.matrix(exp_value),input_matrix)
```

```
##             [,1]
## [1,]  3.07968177
## [2,]  3.91165105
## [3,] -2.10873433
## [4,] -0.69465161
## [5,]  0.54789134
## [6,]  0.07538082
```

7(b)(ii) Analytical implementation -Ridge regression

```
ridge_analytical(input_matrix, exp_value, 0.5)
```

```
##              [,1]
## [1,]  2.891363298
## [2,]  3.373867221
## [3,] -1.798169310
## [4,] -0.245751027
## [5,]  0.468793829
```

```
## [6,] -0.008824966
```

7(b)(iii) Batch gradient descent implementation lasso regression

```
lasso_batch_descent(scaled_input_matrix, exp_value, lambda = 5)
```

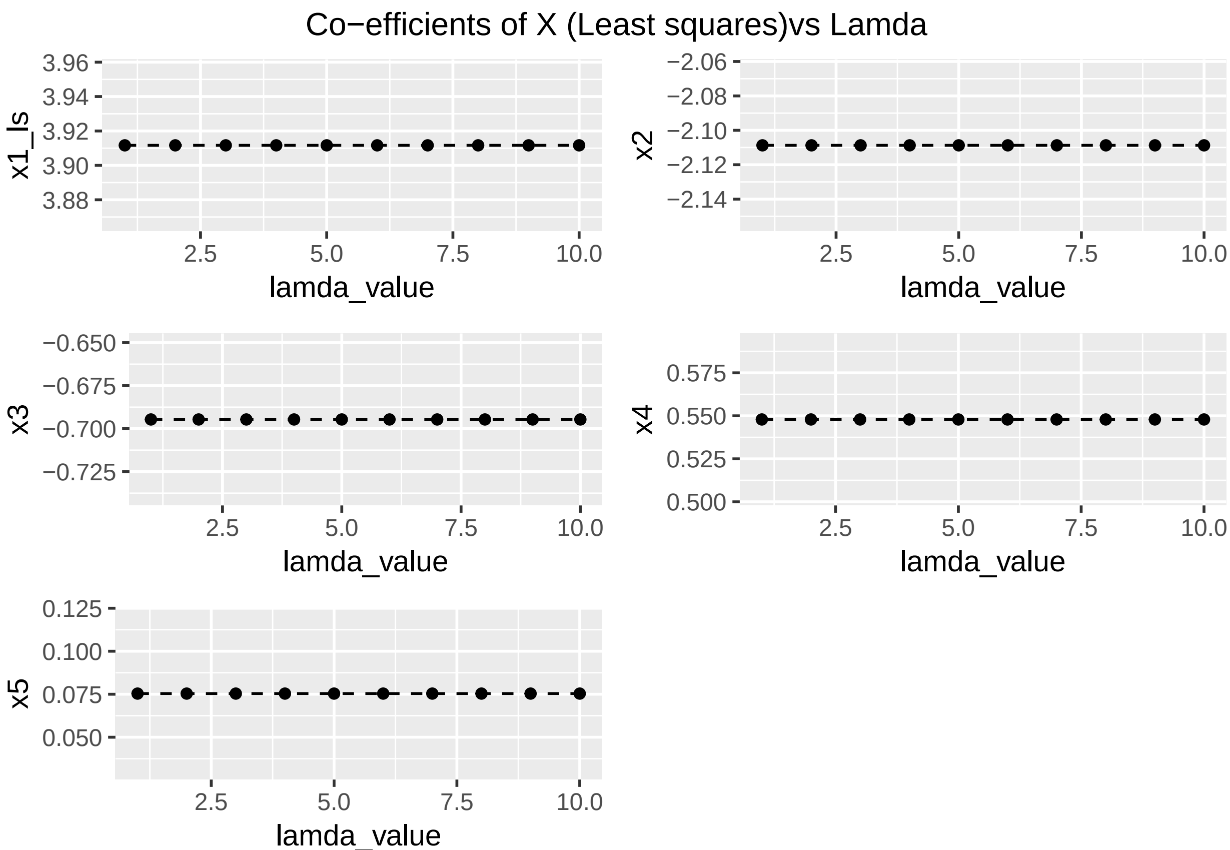
```
##               [,1]
## [1,]  2.5101723642
## [2,]  2.0664159108
## [3,] -0.0003131094
## [4,]  1.2904903347
## [5,] -0.0366007625
## [6,] -0.3496971733
```

7c Plot the coefficient associated with X, yhat predictions against different values of lamda

```
x1_ls = c()
x2_ls = c()
x3_ls = c()
x4_ls = c()
x5_ls = c()

x1_ridge = c()
x2_ridge = c()
x3_ridge = c()
x4_ridge = c()
x5_ridge = c()

x1_lasso = c()
x2_lasso = c()
x3_lasso = c()
x4_lasso = c()
x5_lasso = c()

yhat_ls = c()
yhat_ridge = c()
yhat_lasso = c()

lam_values=10
#Simulating the data as for 1,000 times and fitting a poly model =5
for (lam_val in 1:lam_values) {
  ls_fit<-least_squares_analytical(as.matrix(exp_value), input_matrix)
  ridge_fit<-ridge_analytical(input_matrix, exp_value, lam_val)
  lasso_fit<-lasso_batch_descent(scaled_input_matrix, exp_value, lambda=lam_val)
  #collect the coeffs from different algorithms
  x1_ls <-c(x1_ls, ls_fit[2])
  x2_ls <-c(x2_ls, ls_fit[3])
  x3_ls <-c(x3_ls, ls_fit[4])
  x4_ls <-c(x4_ls, ls_fit[5])
```

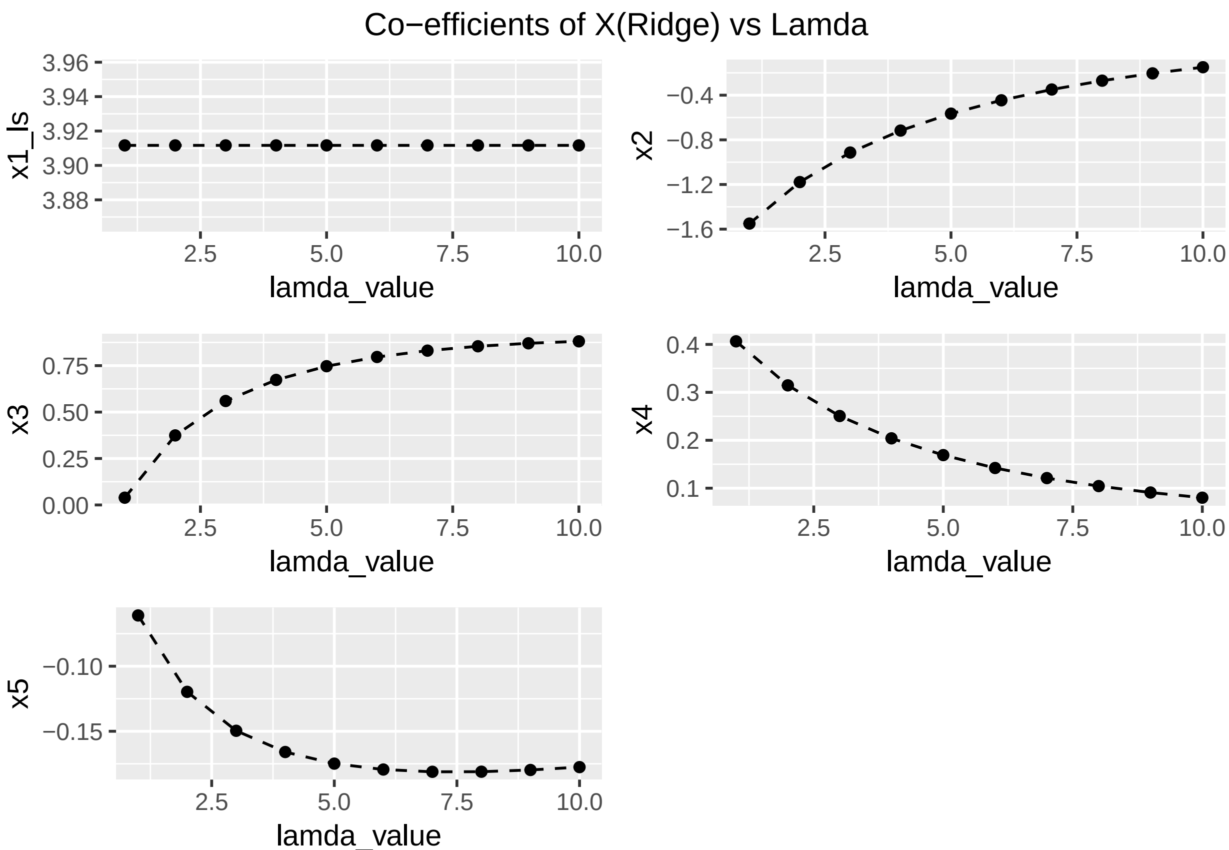
```
  x5_ls <-c(x5_ls, ls_fit[6])

  x1_ridge <-c(x1_ridge, ridge_fit[2])
  x2_ridge <-c(x2_ridge, ridge_fit[3])
  x3_ridge <-c(x3_ridge, ridge_fit[4])
  x4_ridge <-c(x4_ridge, ridge_fit[5])
  x5_ridge <-c(x5_ridge, ridge_fit[6])

  x1_lasso <-c(x1_lasso, lasso_fit[2])
  x2_lasso <-c(x2_lasso, lasso_fit[3])
  x3_lasso <-c(x3_lasso, lasso_fit[4])
  x4_lasso <-c(x4_lasso, lasso_fit[5])
  x5_lasso <-c(x5_lasso, lasso_fit[6])

  #calc yhat values from the coeffs obtained above
  yhat_ls <-c(yhat_ls, ls_fit[1]+ls_fit[2]*1.5)
  yhat_ridge <-c(yhat_ridge, ridge_fit[1]+ridge_fit[2]*1.5)
  yhat_lasso <-c(yhat_lasso, lasso_fit[1]+lasso_fit[2]*1.5)
}
x_coeffcients_ls <-data.frame("lamda_value" = 1:lam_values)
x_coeffcients_ls$x1<- x1_ls
x_coeffcients_ls$x2<- x2_ls
x_coeffcients_ls$x3<- x3_ls
x_coeffcients_ls$x4<- x4_ls
x_coeffcients_ls$x5<- x5_ls

x_coeffcients_ridge <-data.frame("lamda_value" = 1:lam_values)
x_coeffcients_ridge$x1<- x1_ridge
x_coeffcients_ridge$x2<- x2_ridge
x_coeffcients_ridge$x3<- x3_ridge
x_coeffcients_ridge$x4<- x4_ridge
x_coeffcients_ridge$x5<- x5_ridge

x_coeffcients_lasso <-data.frame("lamda_value" = 1:lam_values)
x_coeffcients_lasso$x1<- x1_lasso
x_coeffcients_lasso$x2<- x2_lasso
x_coeffcients_lasso$x3<- x3_lasso
x_coeffcients_lasso$x4<- x4_lasso
x_coeffcients_lasso$x5<- x5_lasso
#consolidate the observations to a dataframe
lam_df <- data.frame("lamda_value" = 1:lam_values)
lam_df$yhat_ls <- yhat_ls
lam_df$yhat_ridge <- yhat_ridge
lam_df$yhat_lasso <- yhat_lasso

#visualize
plot_ls_1<-ggplot(data=x_coeffcients_ls, aes(x=lamda_value, y=x1_ls, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()

plot_ls_2<-ggplot(data=x_coeffcients_ls, aes(x=lamda_value, y=x2, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()
```

```
plot_ls_3<-ggplot(data=x_coeffcients_ls, aes(x=lamda_value, y=x3, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()

plot_ls_4<-ggplot(data=x_coeffcients_ls, aes(x=lamda_value, y=x4, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()

plot_ls_5<-ggplot(data=x_coeffcients_ls, aes(x=lamda_value, y=x5, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()

grid.arrange(plot_ls_1, plot_ls_2, plot_ls_3,plot_ls_4,plot_ls_5, top="Co-efficients of X (Least square
```



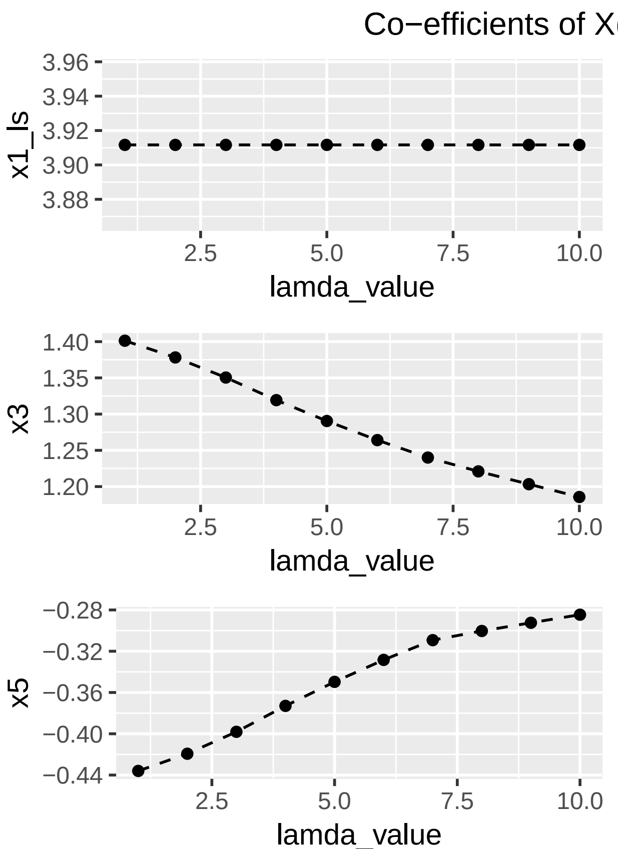

```
plot_ridge_1<-ggplot(data=x_coeffcients_ridge, aes(x=lamda_value, y=x1_ls, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()

plot_ridge_2<-ggplot(data=x_coeffcients_ridge, aes(x=lamda_value, y=x2, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()

plot_ridge_3<-ggplot(data=x_coeffcients_ridge, aes(x=lamda_value, y=x3, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()
```

```
plot_ridge_4<-ggplot(data=x_coeffcients_ridge, aes(x=lamda_value, y=x4, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()

plot_ridge_5<-ggplot(data=x_coeffcients_ridge, aes(x=lamda_value, y=x5, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()

grid.arrange(plot_ridge_1, plot_ridge_2, plot_ridge_3,plot_ridge_4,plot_ridge_5, top="Co-efficients of X
```

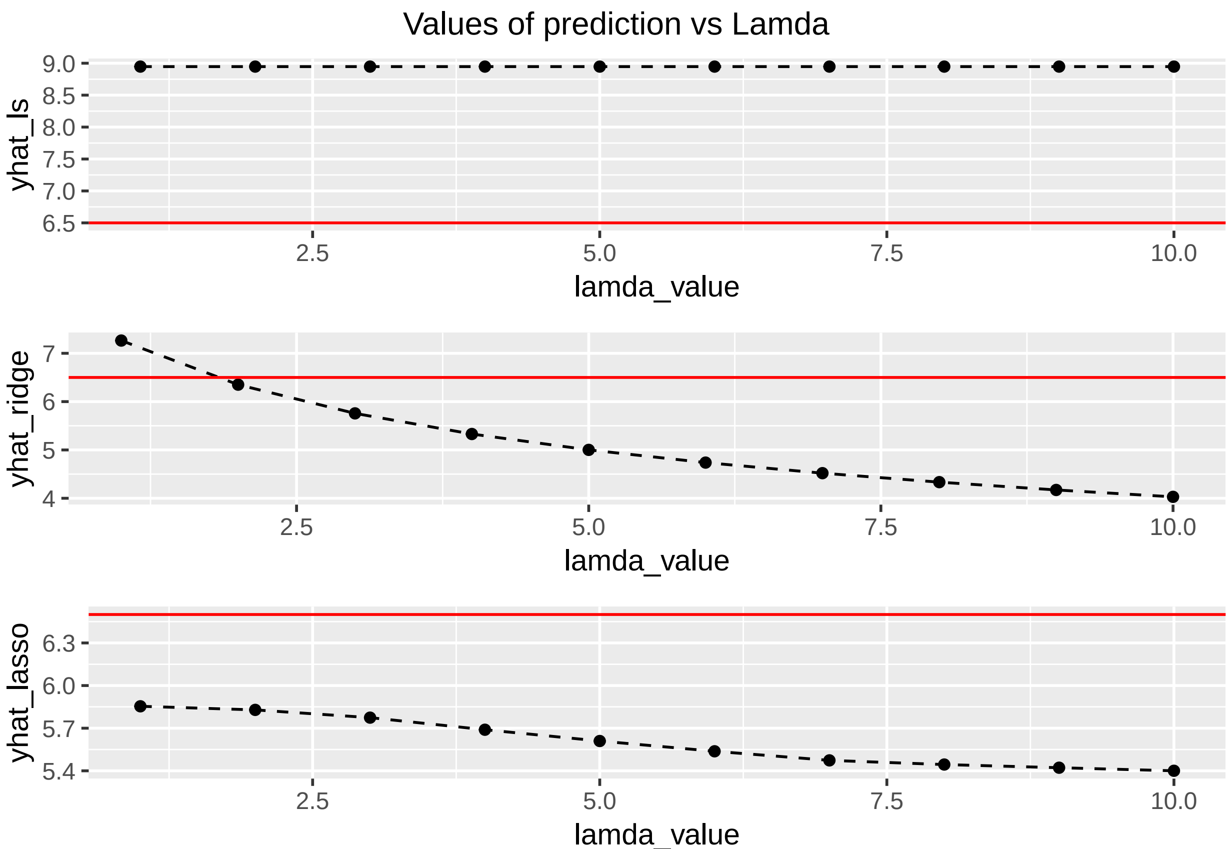



```
plot_lasso_1<-ggplot(data=x_coeffcients_lasso, aes(x=lamda_value, y=x1_ls, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()

plot_lasso_2<-ggplot(data=x_coeffcients_lasso, aes(x=lamda_value, y=x2, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()

plot_lasso_3<-ggplot(data=x_coeffcients_lasso, aes(x=lamda_value, y=x3, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()

plot_lasso_4<-ggplot(data=x_coeffcients_lasso, aes(x=lamda_value, y=x4, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()

plot_lasso_5<-ggplot(data=x_coeffcients_lasso, aes(x=lamda_value, y=x5, group=1)) +
```

```
  geom_line(linetype = "dashed")+
  geom_point()

grid.arrange(plot_lasso_1, plot_lasso_2, plot_lasso_3,plot_lasso_4,plot_lasso_5, top="Co-efficients of X
```




```
plot_yhat_ls<-ggplot(data=lam_df, aes(x=lamda_value, y=yhat_ls, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()+geom_hline(yintercept = 6.5, col = "red")

plot_yhat_ridge<-ggplot(data=lam_df, aes(x=lamda_value, y=yhat_ridge, group=1)) +
  geom_line(linetype = "dashed")+
  geom_point()+geom_hline(yintercept = 6.5, col = "red")

plot_yhat_lass<-ggplot(data=lam_df, aes(x=lamda_value, y=yhat_lasso, group=1)) +
  geom_line(linetype = "dashed")+geom_point()+geom_hline(yintercept = 6.5, col = "red")

grid.arrange(plot_yhat_ls, plot_yhat_ridge, plot_yhat_lass, top = "Values of prediction vs Lamda")
```



Conclusion: As we can see from the different x-coeffcients plots generated for the 3 algorithms, co-efficient of x1 seems to be more stable with significant(magnitude) value. The non x1 co-effcients in Lasso(mod fluctuation) and Ridge (exponential)have fluctions with lesser significant values in terms of maginitude.

The coeffs generated by least sqaures seem to stationary at a fixed value for all iterations since it does not have any hyper parameter in it.

The predictions generated by least squares seem to be more closer to the actual value compared to other algorithms. This can be explained by the bias introduced by Lasso and

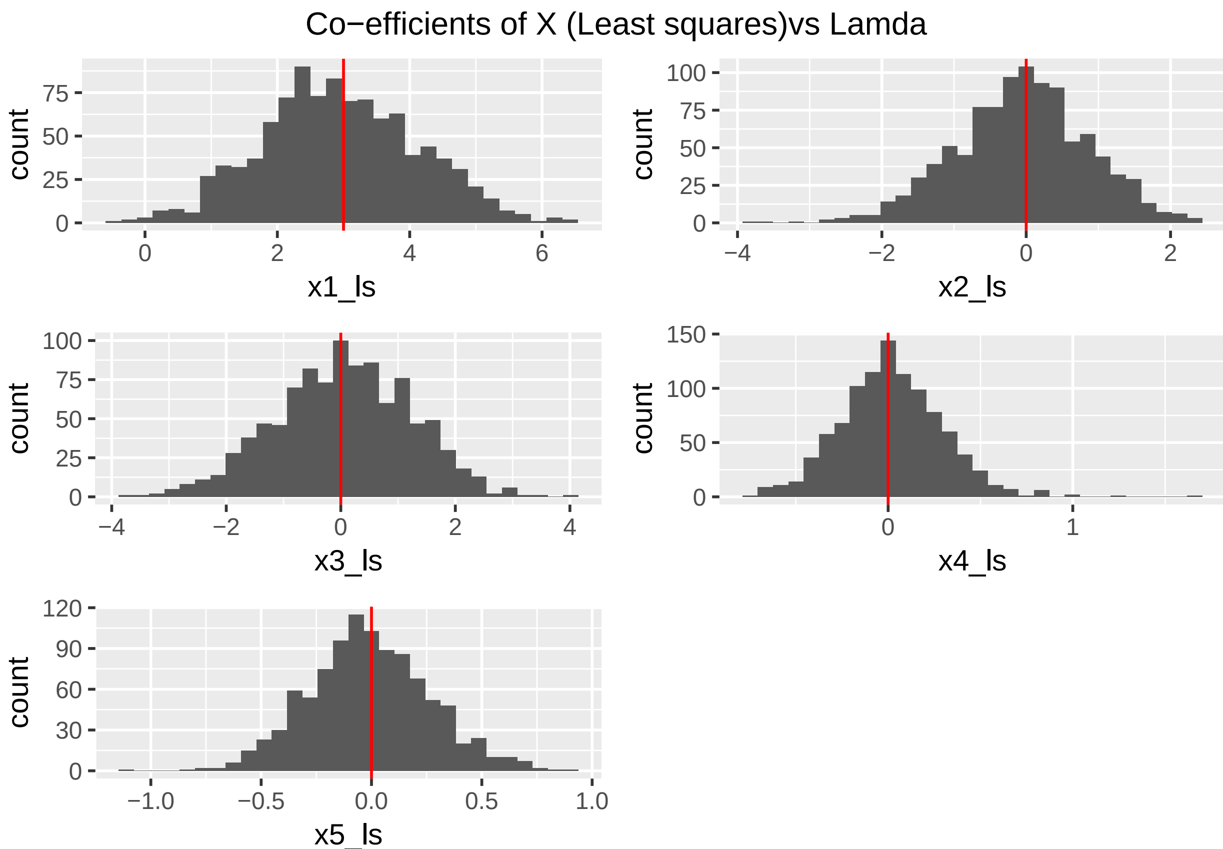
7d Plot the coefficient associated with X, yhat predictions against

lamda for 1000 iterations

```
x1_ls = c()
x2_ls = c()
x3_ls = c()
x4_ls = c()
x5_ls = c()

x1_ridge = c()
x2_ridge = c()
x3_ridge = c()
x4_ridge = c()
x5_ridge = c()

x1_lasso = c()
```

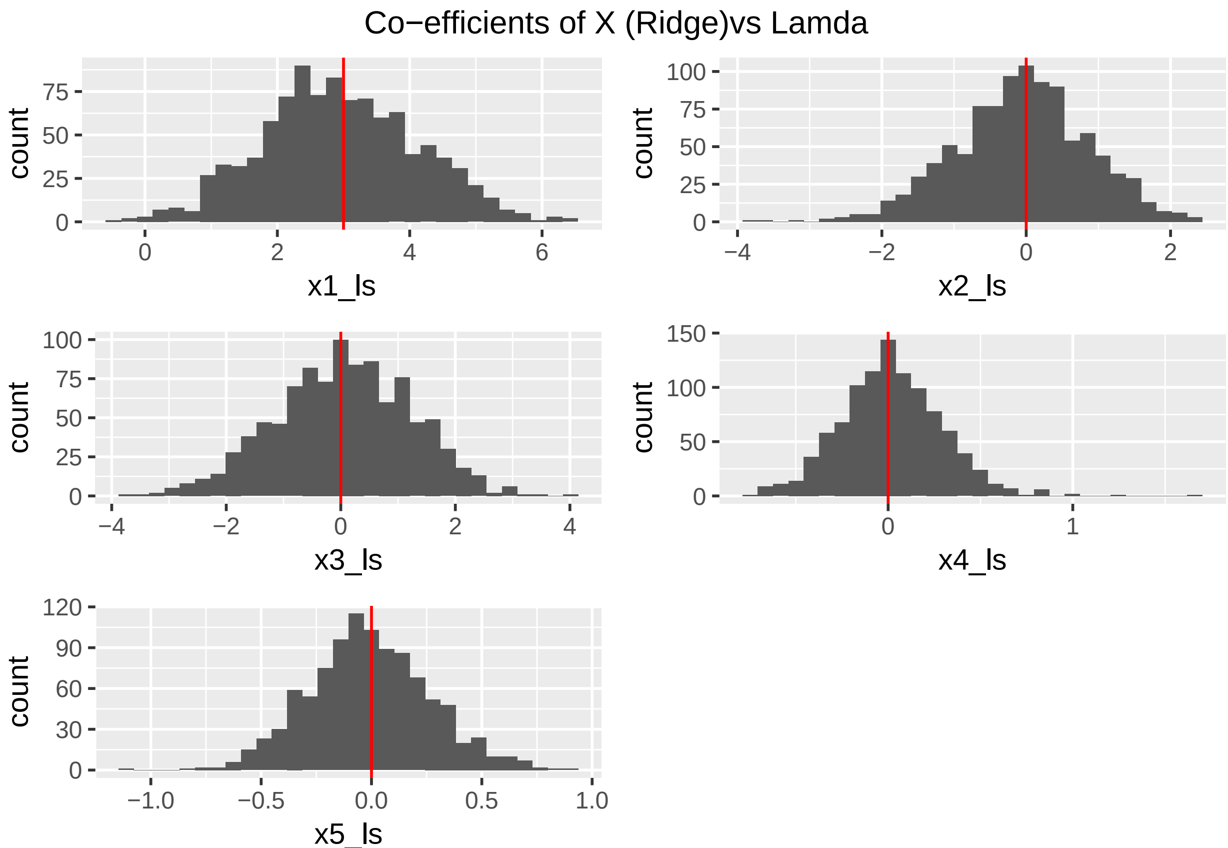
```
x2_lasso = c()
x3_lasso = c()
x4_lasso = c()
x5_lasso = c()

analytical_yhat=c()
lasso_yhat=c()
ridge_yhat=c()
for(index in 1:1000) {
  #generate y function and add outliers
  data<-data_simulator()
  predictor<-data[[1]]
  exp_value<-data[[2]]
  #get inputs
  input_matrix<-get_data(predictor, data_size)
  scaled_input_matrix<-get_scaled_data(predictor, data_size)
   # Calc slopes and add to a column vector
  ls_fit<-least_squares_analytical(as.matrix(exp_value), input_matrix)
  ridge_fit<-ridge_analytical(input_matrix, exp_value, 2)
  lasso_fit<-lasso_batch_descent(scaled_input_matrix, exp_value, lambda=2)

  x1_ls <-c(x1_ls, ls_fit[2])
  x2_ls <-c(x2_ls, ls_fit[3])
  x3_ls <-c(x3_ls, ls_fit[4])
  x4_ls <-c(x4_ls, ls_fit[5])
  x5_ls <-c(x5_ls, ls_fit[6])

  x1_ridge <-c(x1_ridge, ridge_fit[2])
  x2_ridge <-c(x2_ridge, ridge_fit[3])
  x3_ridge <-c(x3_ridge, ridge_fit[4])
  x4_ridge <-c(x4_ridge, ridge_fit[5])
  x5_ridge <-c(x5_ridge, ridge_fit[6])

  x1_lasso <-c(x1_lasso, lasso_fit[2])
  x2_lasso <-c(x2_lasso, lasso_fit[3])
  x3_lasso <-c(x3_lasso, lasso_fit[4])
  x4_lasso <-c(x4_lasso, lasso_fit[5])
  x5_lasso <-c(x5_lasso, lasso_fit[6])
  #Predict values
  analytical_yhat <-c(analytical_yhat, ls_fit[1]+ls_fit[2]*1.5)
  ridge_yhat <-c(ridge_yhat, ridge_fit[1]+ridge_fit[2]*1.5)
  lasso_yhat <-c(lasso_yhat, lasso_fit[1]+lasso_fit[2]*1.5)
}

x_coeffcients_ls <-data.frame(x1_ls)
x_coeffcients_ls$x2<- x2_ls
x_coeffcients_ls$x3<- x3_ls
x_coeffcients_ls$x4<- x4_ls
x_coeffcients_ls$x5<- x5_ls

x_coeffcients_ridge <-data.frame(x1_ridge)
x_coeffcients_ridge$x2<- x2_ridge
x_coeffcients_ridge$x3<- x3_ridge
```

```
x_coeffcients_ridge$x4<- x4_ridge
x_coeffcients_ridge$x5<- x5_ridge

x_coeffcients_lasso <-data.frame(x1_lasso)
x_coeffcients_lasso$x2<- x2_lasso
x_coeffcients_lasso$x3<- x3_lasso
x_coeffcients_lasso$x4<- x4_lasso
x_coeffcients_lasso$x5<- x5_lasso

yhat_df<-as.data.frame(analytical_yhat)
yhat_df$ridge_yhat<-ridge_yhat
yhat_df$lasso_yhat<-lasso_yhat


histogram_ls_1<-x_coeffcients_ls %>% ggplot() +
  geom_histogram(mapping = aes(x = x1_ls)) +
  geom_vline(xintercept = 3, col = "red")

histogram_ls_2<-x_coeffcients_ls %>% ggplot() +
  geom_histogram(mapping = aes(x = x2_ls)) +
  geom_vline(xintercept = 0, col = "red")

histogram_ls_3<-x_coeffcients_ls %>% ggplot() +
  geom_histogram(mapping = aes(x = x3_ls)) +
  geom_vline(xintercept = 0, col = "red")

histogram_ls_4<-x_coeffcients_ls %>% ggplot() +
  geom_histogram(mapping = aes(x = x4_ls)) +
  geom_vline(xintercept = 0, col = "red")

histogram_ls_5<-x_coeffcients_ls %>% ggplot() +
  geom_histogram(mapping = aes(x = x5_ls)) +
  geom_vline(xintercept = 0, col = "red")

grid.arrange(
  histogram_ls_1,
  histogram_ls_2,
  histogram_ls_3,
  histogram_ls_4,
  histogram_ls_5,
  top = "Co-efficients of X (Least squares)vs Lamda"
  )
```



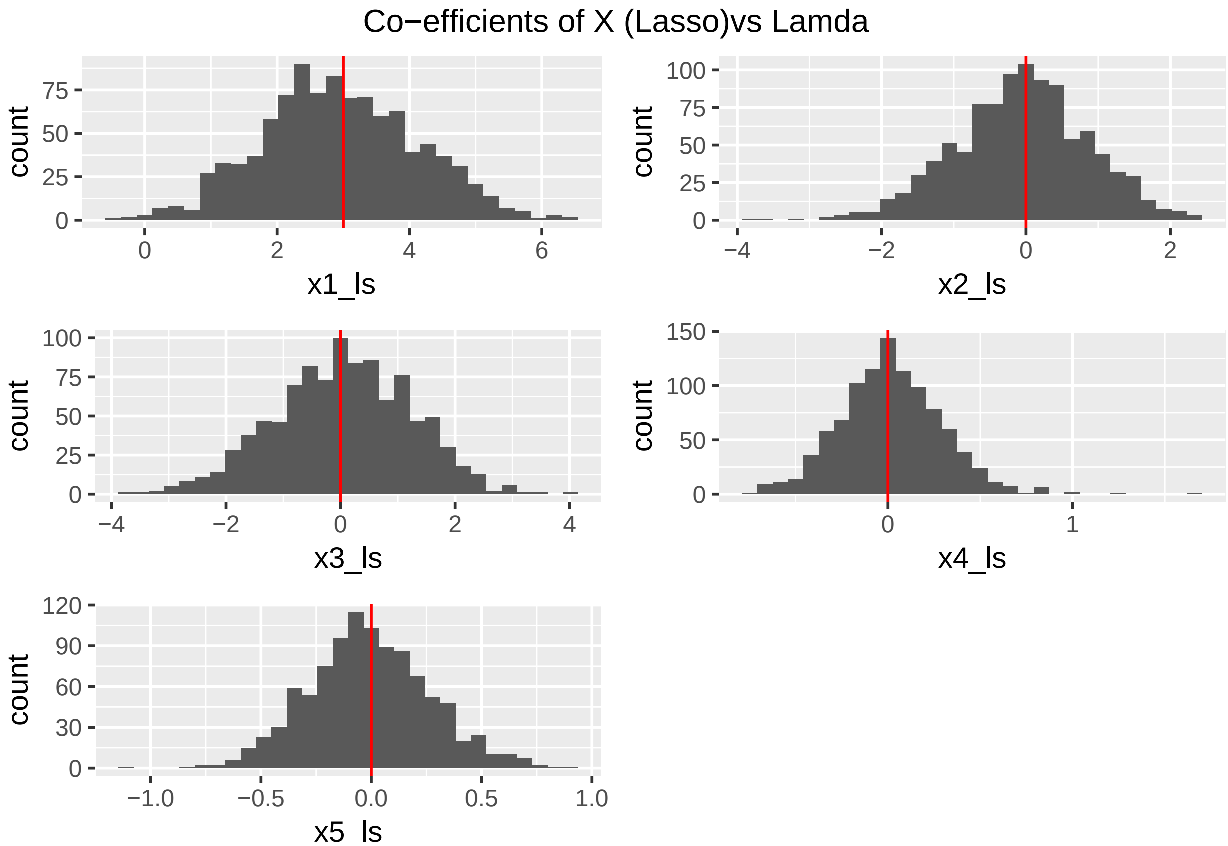

```
histogram_ridge_1<-x_coeffcients_ridge %>% ggplot() +
  geom_histogram(mapping = aes(x = x1_ls)) +
  geom_vline(xintercept = 3, col = "red")

histogram_ridge_2<-x_coeffcients_ridge %>% ggplot() +
  geom_histogram(mapping = aes(x = x2_ls)) +
  geom_vline(xintercept = 0, col = "red")

histogram_ridge_3<-x_coeffcients_ridge %>% ggplot() +
  geom_histogram(mapping = aes(x = x3_ls)) +
  geom_vline(xintercept = 0, col = "red")

histogram_ridge_4<-x_coeffcients_ridge %>% ggplot() +
  geom_histogram(mapping = aes(x = x4_ls)) +
  geom_vline(xintercept = 0, col = "red")

histogram_ridge_5<-x_coeffcients_ridge %>% ggplot() +
  geom_histogram(mapping = aes(x = x5_ls)) +
  geom_vline(xintercept = 0, col = "red")

grid.arrange(
  histogram_ridge_1,
  histogram_ridge_2,
  histogram_ridge_3,
  histogram_ridge_4,
  histogram_ridge_5,
  top = "Co-efficients of X (Ridge)vs Lamda"
```

```
)
```

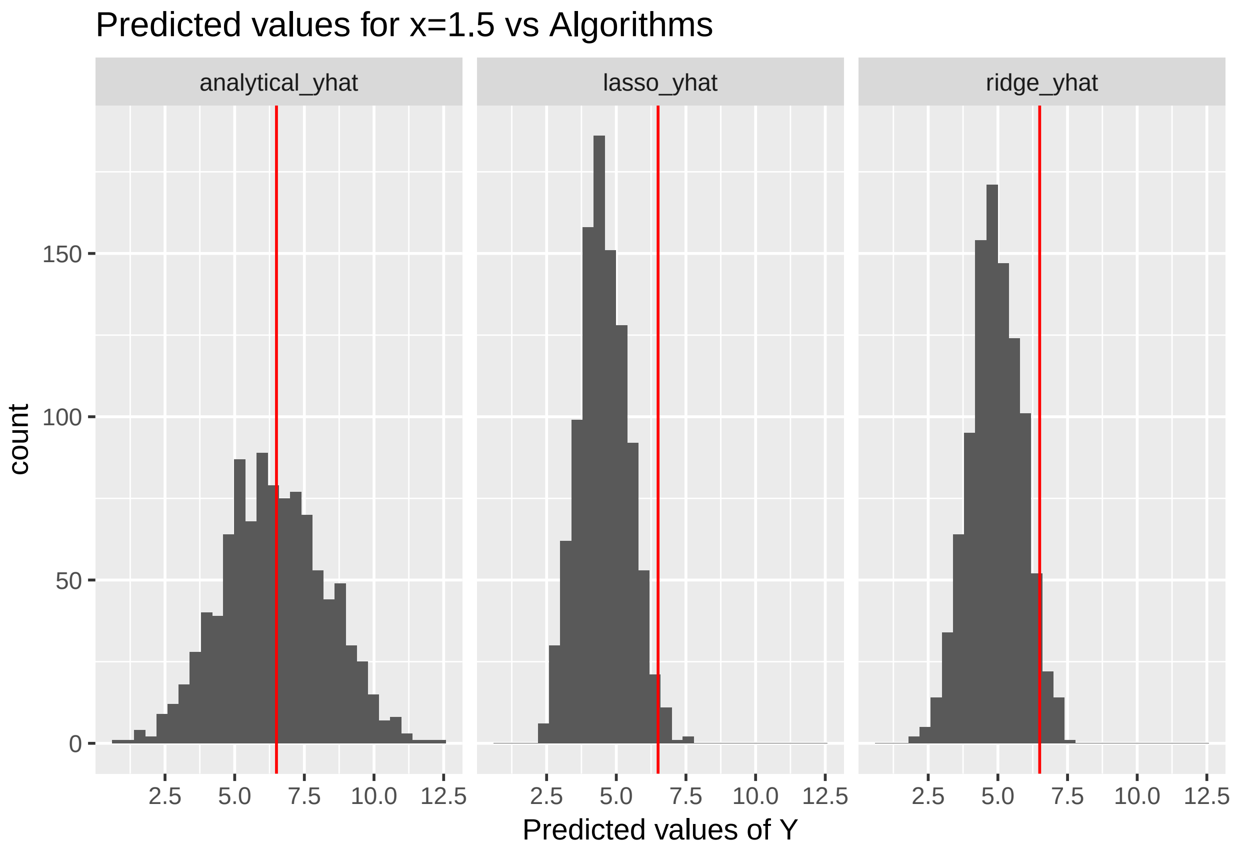



```
histogram_lasso_1<-x_coeffcients_lasso %>% ggplot() +
  geom_histogram(mapping = aes(x = x1_ls)) +
  geom_vline(xintercept = 3, col = "red")

histogram_lasso_2<-x_coeffcients_lasso %>% ggplot() +
  geom_histogram(mapping = aes(x = x2_ls)) +
  geom_vline(xintercept = 0, col = "red")

histogram_lasso_3<-x_coeffcients_lasso %>% ggplot() +
  geom_histogram(mapping = aes(x = x3_ls)) +
  geom_vline(xintercept = 0, col = "red")

histogram_lasso_4<-x_coeffcients_lasso %>% ggplot() +
  geom_histogram(mapping = aes(x = x4_ls)) +
  geom_vline(xintercept = 0, col = "red")

histogram_lasso_5<-x_coeffcients_lasso %>% ggplot() +
  geom_histogram(mapping = aes(x = x5_ls)) +
  geom_vline(xintercept = 0, col = "red")

grid.arrange(
  histogram_ls_1,
  histogram_ls_2,
  histogram_ls_3,
  histogram_ls_4,
```

```r
  histogram_ls_5,
  top = "Co-efficients of X (Lasso)vs Lamda"
  )
```




```r
yhat_df_new <- gather(yhat_df, key = "algorithm", value = "yhat-values")

yhat_df_new %>% ggplot(mapping = aes(x = `yhat-values`)) + geom_histogram() + facet_grid( ~
algorithm) + geom_vline(xintercept = 6.5 , col = "red")+
  labs(x="Predicted values of Y", title="Predicted values for x=1.5 vs Algorithms")
```



Conclusion: It's clear that Analytical solution doesn't introduce any bias into the model (the peak is centered around the true value). However, this is not the case with Ridge and Lasso regressions. They clearly introduce a bias into the system - the peak density of offset from the actual value.

The predicted values by the analytical solution fairly good. But, the span of the histogram is pretty broad, indicating the variance is high. However, the other two seem to have a narrow peaks. This clearly explains bias-variance tradeoff

References:

The following articles have been referenced : http://www.stat.columbia.edu/~fwood/Teaching/w4315/Fall2009/lecture_6.pdf https://www.spss-tutorials.com/anova-what-is-it/#formulas http://sphweb.bumc.bu.edu/otlt/MPH-Modules/QuantCore/PH717_MultipleVariableRegression/PH717_MultipleVariableRegression4.html https://www.safaribooksonline.com/library/view/hands-on-machine-learning/9781491962282/ch04.html?orpq

4 To find the estimates for θ using Ridge Regression, we start with the following equation.

$$\hat{\theta}_{ridge} = \underset{\theta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} [y_i - \theta_0 - \sum_{j=1}^{p} x_{ij}\theta_j]^2 + \lambda \sum_{j=1}^{p} \theta_j^2 \right\}.$$

for simplicity, we can rewrite the above equation as.

$$\hat{\theta}_{ridge} = \underset{\theta}{\operatorname{argmin}} \, || X\theta - Y||^2 + \lambda \theta^2$$

where, $\hat{\theta} \rightarrow$ theta estimates for Ridge Regression.
$\lambda \rightarrow$ hyper parameter that imposes penalty on co-efficients.

Taking the derivative & setting the value to 0.
{as the slope / gradient at minima is zero}.

$$= 2X^T(X\theta - Y) + 2\lambda\theta = 0.$$

$$= 2X^TX\theta - 2X^TY + 2\lambda\theta = 0$$

$$\theta(2X^TX + 2\lambda) = 2X^TY.$$

Simplifying: $\theta(X^TX + \lambda I) = X^TY$. ; $I \rightarrow$ Identity matrix of $p \times p$

$$\hat{\theta}_{Ridge} = (X^TX + \lambda I)^{-1} \cdot X^TY.$$

(5) Prove that $Var(\hat{\theta}_{LS}) \geq Var(\hat{\theta}_{Ridge})$.

→ Solution →

Ridge Estimation produces a biased Estimator of the theta parameter. Using the definition of $\hat{\theta}_{Ridge}$ & the modelling assumption on the mean function $E[y[X]] = X\theta$, we get.

$$E[\hat{\theta}_{ridge}[X]] = (X^TX + \lambda I)^{-1} X^TX\theta$$
$$= (X^TX + \lambda I)^{-1}(X^TX + \lambda I - \lambda I)\theta$$
$$= [I - \lambda(X^TX + \lambda I)^{-1}]\theta$$
$$= \theta - \lambda(X^TX + \lambda I)^{-1}\theta.$$

The above equation implies that λ is directly proportional to the ridge estimator $\hat{\theta}$.

Although the ridge estimators incur a greater bias, it possesses a smaller variance than the vector of Least square estimators. This can be proved by taking the trace of the variance matrices of the two methods.

i.e $$tr(Var[\hat{\theta}^{LS}[X]]) = tr(\sigma^2(X^TX)^{-1})$$
$$= \sigma^2 \sum_{j=1}^{p} \frac{1}{d_j^2}$$

Similarly for Ridge Estimators, we have

$$Var[\hat{\theta}^{ridge}[X]] = \sigma^2(X^TX + \lambda I)^{-1}(X^TX)(X^TX + \lambda I)^{-1}$$

Since X^TX & λI are Simultaenously diagonalizable, we have,

$$(VD^2V^T + \lambda I)^{-1} = V(D^2 + \lambda I)^{-1}V^T$$

Applying this formula twice in the above formula,

Scanned with CamScanner

$$(VD^2V^T+\lambda I)^{-1} = V(D^2+\lambda I)^{-1}V^T$$

$$\Rightarrow \; V(D^2+\lambda I)^{-1}V^T VD^2V^TV(D^2+\lambda I)^{-1}V^T$$

$$= \; V(D^2+\lambda I)^{-1}D^2(D^2+\lambda I)^{-1}V^T$$

This is a diagonalizable matrix, therefore we can simply take the trace (Σ Eigen values).

$$tr(Var[\hat{\theta}^{ridge}][X]) = \sigma^2 \sum_{j=1}^{p} \frac{d_j^2}{(d_j^2+\lambda)^2}$$

We can see that Ridge estimator has indeed systematically less total variance than least square estimator.

i.e $tr(Var[\hat{\theta}^{LS}][X]) \geq tr(Var[\hat{\theta}^{ridge}[X]])$.

(6) The objective function for Ridge Regression can be written as:

$$J(\theta) = (Y-X\theta)^T(Y-X\theta) + \lambda(W^T\theta - b).$$

Now, $\frac{\partial(J(\theta))}{\partial\theta} = 2X^T(X\hat{\theta}-Y)+\lambda W = 0$

$$= 2X^TX\hat{\theta} - 2X^TY + \lambda W = 0$$

given that $X^TX = I_p$.

$$\therefore \; 2I_p\hat{\theta} - 2X^TY + \lambda W = 0$$

$$2\hat{\theta} = 2X^TY - \lambda W$$

$$\hat{\theta} = X^TY - \frac{\lambda}{2}W. \longrightarrow (1)$$

further,

given that $w^T\hat{\theta} = b$.

$$\Rightarrow w^T\left(X^Ty - \frac{\lambda}{2}w\right) = b.$$

$$w^TX^Ty - \frac{1}{2}\lambda w^Tw = b.$$

$$\Rightarrow \frac{\lambda}{2} w^Tw = w^TX^Ty - b$$

$$= \frac{\lambda}{2} \|w\|_2^2 = w^TX^Ty - b.$$

$$\lambda = \frac{2(w^TX^Ty - b)}{\|w\|_2^2}.$$

Re-subtuting the value of λ in eq^n 1

$\hat{\theta}$ becomes $\Rightarrow$ $\hat{\theta} = X^Ty - \frac{(w^TX^Ty - b)}{\|w\|_2^2} \cdot w$

$$\therefore \quad \hat{\theta} = X^Ty - \frac{(w^TX^Ty - b)}{\|w\|_2^2} \cdot w.$$