HW: Week 9

36-350 – Statistical Computing

Week 9 – Spring 2022

Name: Ethan Vertal

Andrew ID: evertal

You must submit **your own** HW as a PDF file on Gradescope.

```
## Loading required package: emdbook
```

HW Length Cap Instructions

- If the question requires you to print a data frame in your solution e.g. `q1_out_df`, you must first apply **head(q1__out__df, 30)** and **dim(q1__out__df)** in the final knitted pdf output for such a data frame.
- Please note that this only applies if you are knitting the `Rmd` to a `pdf`, for Gradescope submission purposes.
- If you are using the data frame output for visualization purposes (for example), use the entire data frame in your exploration
- The **maximum allowable length** of knitted pdf HW submission is **30 pages**. Submissions exceeding this length *will not be graded* by the TAs. All pages must be tagged as usual for the required questions per the usual policy
- For any concerns about HW length for submission, please reach out on Piazza during office hours

Question 1

(20 points)

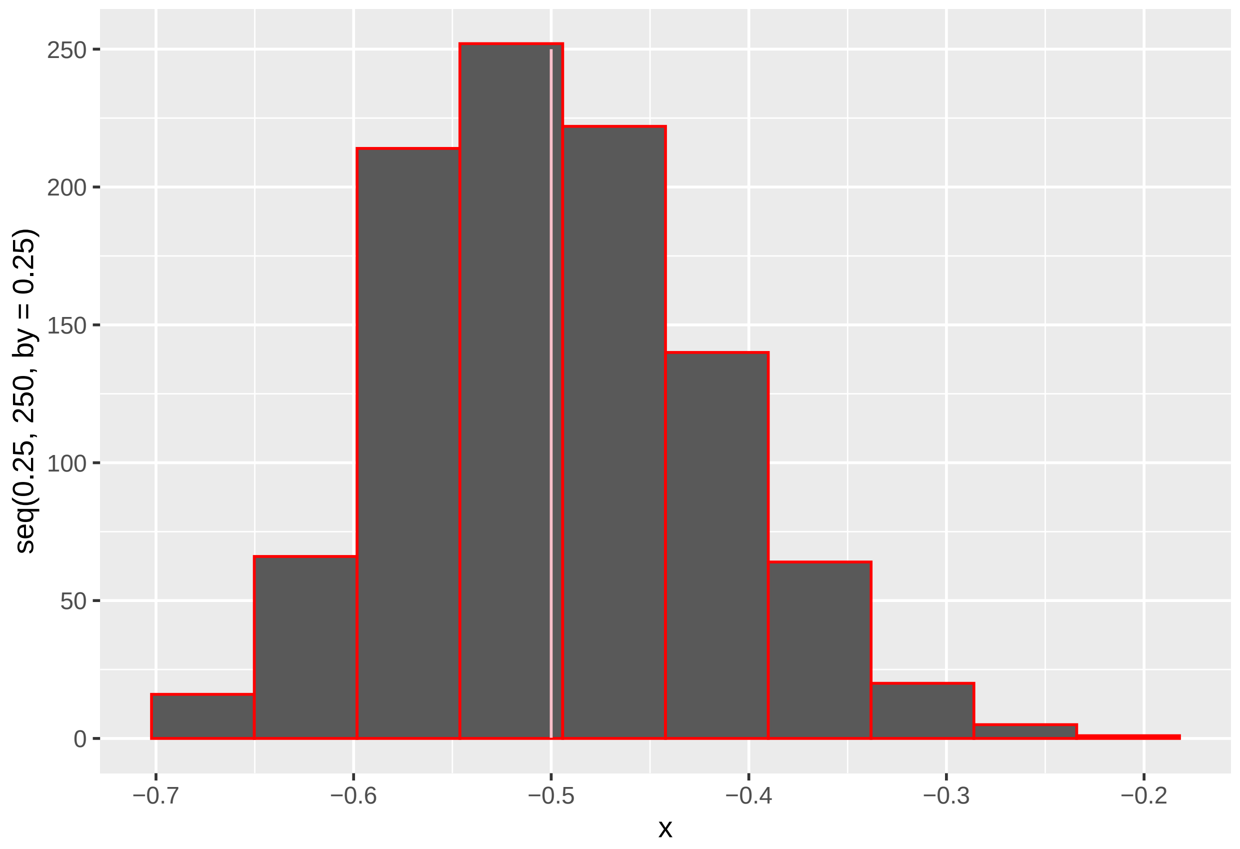
Display the sampling distribution for $R_{1,2}$, the off-diagonal element of a two-dimensional sample correlation matrix for a bivariate normal, given that $\mu_1 = \mu_2 = 2$, $\sigma_1 = 1$, $\sigma_2 = 2$, $\rho_{1,2} = -0.5$, and $n = 100$. Sample 1000 values and display them in a histogram; include a vertical line for the population value (-0.5). (Reminder: if I don't specify how exactly to visualize something, i.e., if I don't specify base `R` versus `ggplot`, then you can choose whichever.) Note that the final distribution that you see will not be normal.

```
set.seed(101)

n <- 100
cor.data <- rep(0, 1000)
mu <- c(1, 2)
sigma.1 <- 1
sigma.2 <- 2
rho.12 <- -0.5
```

```r
Sigma <- matrix(c(sigma.1^2, rho.12 * sigma.1 * sigma.2,
                  rho.12 * sigma.1 * sigma.2, sigma.2^2), nrow = 2)
for (ii in 1:1000) {
  data <- mvrnorm(n, mu, Sigma)
  data.df <- data.frame(x = data[,1], y = data[,2])
  cor.data[ii] <- cor(data.df)[1,2]
}

cor.data.df <- data.frame(x = cor.data)
ggplot(data = cor.data.df, mapping=aes(x=x)) + geom_histogram(bins=10, col="red") +
       geom_line(mapping = aes(x=c(-0.5), y=seq(0.25,250, by=0.25)), col = "pink")
```



Question 2

(20 points)

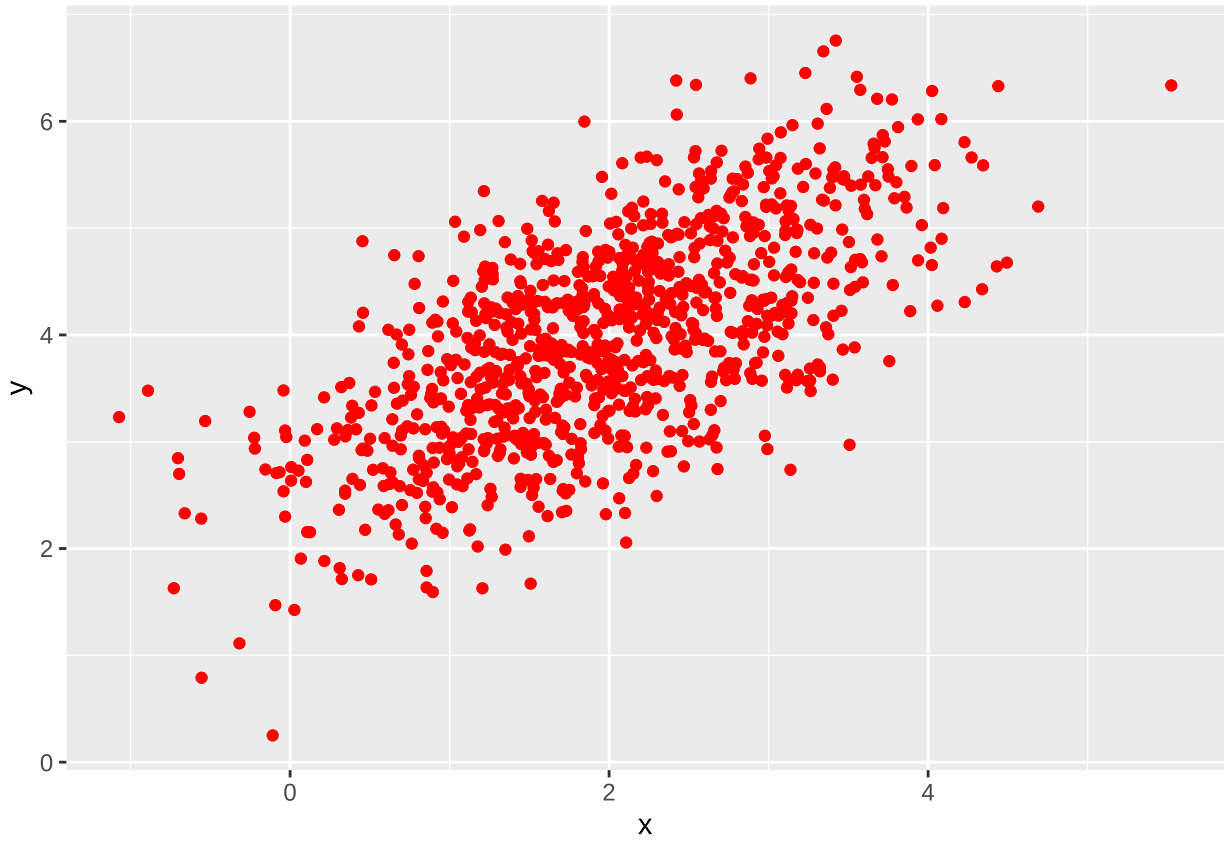
Assume you have a trivariate multivariate normal with $\mu = \{2, 3, 4\}$ and $\sigma = \{1, 2, 1\}$, and $\rho_{1,2} = 0.4$, $\rho_{1,3} = 0.7$, and $\rho_{2,3} = -0.2$. Sample 1000 data from the marginal distribution for (x_1, x_3), and display them via `ggplot`. Compute the sample means along each marginal axis: they should be approximately 2 and 4.

```r
set.seed(101)

mu <- c(2,3,4)
sigma.1 <- 1
sigma.2 <- 2
sigma.3 <- 1
rho.12 <- 0.4
rho.13 <- 0.7
rho.23 <- -0.2
Sigma <- matrix(c(sigma.1^2, rho.12*sigma.1*sigma.2, rho.13*sigma.1*sigma.3,
                 rho.12*sigma.1*sigma.2, sigma.2^2, rho.23*sigma.3*sigma.2,
                 rho.13*sigma.1*sigma.3, rho.23*sigma.2*sigma.3, sigma.3^2),
                 nrow=3)
p = c(1,3)
mu.marginal <- mu[p]
Sigma.marginal <- Sigma[p, p]

sample.data <- mvrnorm(1000, mu.marginal, Sigma.marginal)
sample.data.df <- data.frame(x = sample.data[,1], y = sample.data[,2])

ggplot(data = sample.data.df, mapping = aes(x=x, y=y)) + geom_point(col = "red")
```



```r
mean(sample.data.df$x)
```

```
## [1] 1.973867
```

```r
mean(sample.data.df$y)
```

```
## [1] 3.961851
```

Question 3

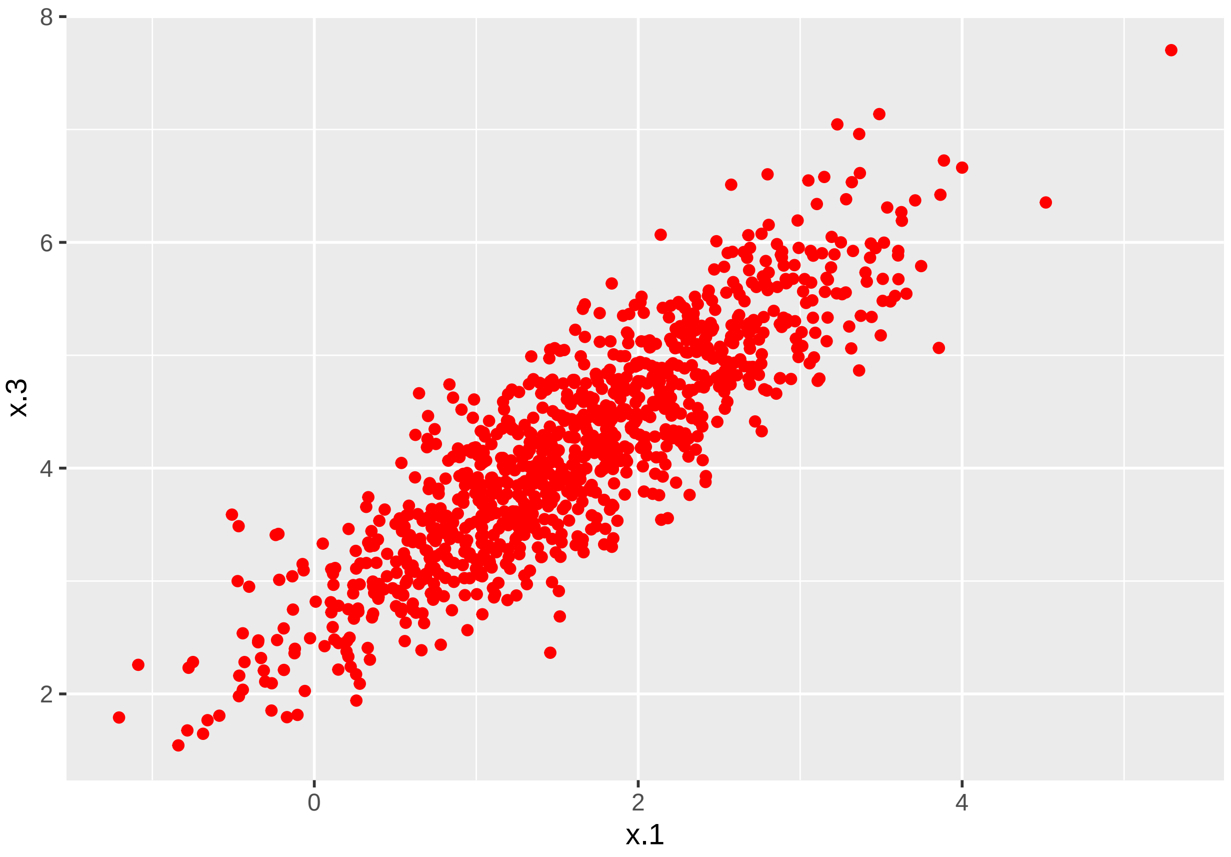
(20 points)

Repeat Q2, except here you should sample from the conditional distribution $f(x_1, x_3|x_2 = 1)$. Compute the conditional correlation coefficient ρ_{cond} between the data along the x_1 and x_3 axes, given $x_2 = 1$, and display the sample correlation matrix.

```r
set.seed(202)

k <- c(1,3)
d.minus.k <- 2
x.2 <- 1
Sigma.kk <- Sigma[k,k]
Sigma.kd <- Sigma[k,d.minus.k]
Sigma.dk <- Sigma[d.minus.k,k]
Sigma.dd <- Sigma[d.minus.k,d.minus.k]
Sigma.cond <- Sigma.kk - Sigma.kd %*% solve(Sigma.dd) %*% Sigma.dk

mu.cond <- mu[k] + Sigma.kd %*% solve(Sigma.dd) %*% matrix(1-mu[d.minus.k])
data <- mvrnorm(1000, mu.cond, Sigma.cond)
data.df <- data.frame(x.1=data[,1], x.3=data[,2])
ggplot(data <- data.df, mapping = aes(x=x.1, y=x.3)) + geom_point(col = "red")
```



```
pcond <- Sigma.cond[1,2] / (sqrt(Sigma.cond[1,1]) * sqrt(Sigma.cond[2,2]))
cat("pcond =", pcond)
```

```
## pcond = 0.868599
```

```
cor(data.df)
```

```
##           x.1       x.3
## x.1 1.0000000 0.8742421
## x.3 0.8742421 1.0000000
```

Question 4

(20 points)

Assume that you have a mixture model: you have 100 data sampled from a bivariate normal with $\mu = \{1, 1\}$ and $\sigma = \{1.2, 1.2\}$, with $\rho = 0.4$, and another 100 data sampled from a bivariate normal with $\mu = \{3, 3\}$, $\sigma = \{1, 1\}$, and $\rho = -0.6$. Plot your sampled data with separate colors for each component of the mixture. Then perform logistic regression to try to classify each sampled point as being from component 1 or component 2, and output the proportion of times you misclassify a point. (Don't worry about breaking your data up into training and testing sets, as this is a simple academic exercise; just use all 200 points to train your classifier, then output the training misclassification error.)

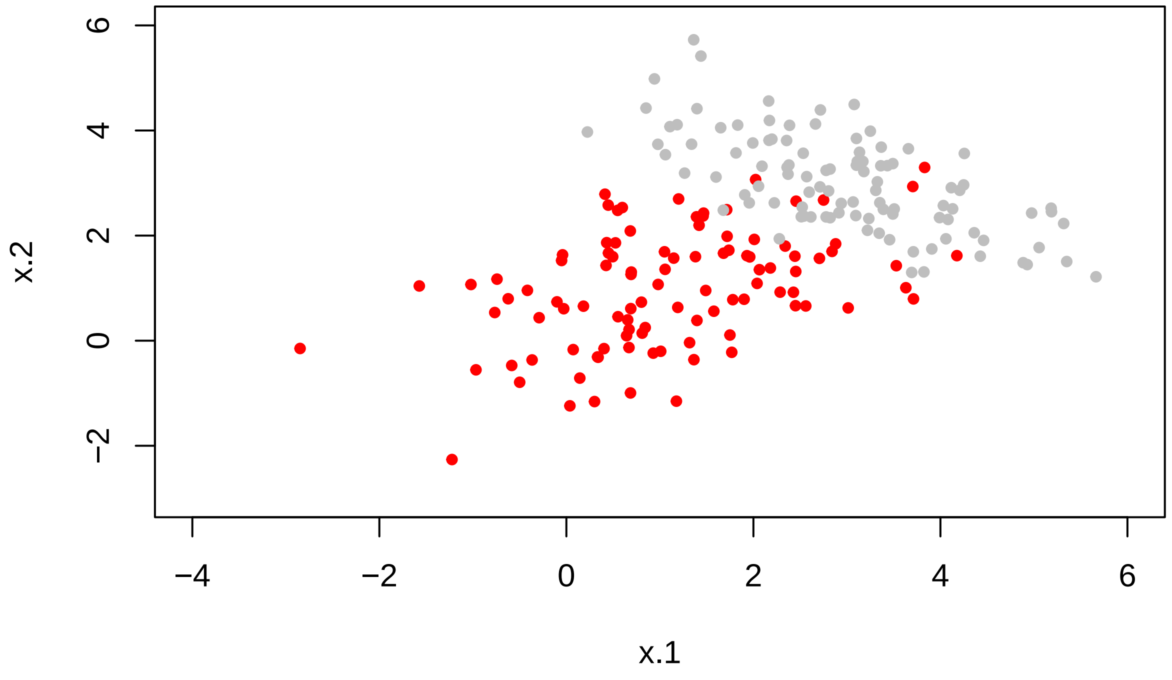
How to train your logistic classifier and get the misclassification rate?

- Assuming you already have a data frame with sampled x_1 and x_2 values in the first and second columns, add a third column with the labeled class. (Name this column `class`.) Use 0 for the first class, and 1 for the second.
- Use `glm()` with model formula `class~.`, your data frame, and the argument `family=binomial`.
- Use `predict()` with the output of `glm()` and with the argument `type="response"`. This will generate 200 predictions between 0 and 1.
- Round off all predictions to 0 or 1.
- Create a `table()` with the arguments being your rounded-off predictions and the labeled classes.
- Compute the proportion of table elements that are "off-diagonal" (upper-right and lower-left). Done.

```
set.seed(303)

mu1 <- c(1,1)
mu2 <- c(3,3)
sigma.1 <- c(1.2, 1.2)
sigma.2 <- c(1,1)
rho.1 <- matrix(c(1, .4, .4, 1), nrow=2)
rho.2 <- matrix(c(1, -0.6, -0.6, 1), nrow=2)
Sigma.1 <- rho.1 * (sigma.1 %o% sigma.1)
data.1 <- mvrnorm(100, mu1, Sigma.1)
data1.df <- data.frame(x.1 = data.1[,1], x.2 = data.1[,2], class = rep(0, 100))
Sigma.2 <- rho.2 * (sigma.2 %o% sigma.2)
data.2 <- mvrnorm(100, mu2, Sigma.2)
data2.df <- data.frame(x.1 = data.2[,1], x.2 = data.2[,2], class = rep(1, 100))

plot(data1.df$x.1, data1.df$x.2, col="red",
     xlim=c(-4, 6), ylim=c(-3,  6), pch=20,
     xlab = "x.1", ylab = "x.2")
points(data2.df$x.1, data2.df$x.2, col="gray", pch=20)
```



```
x.df <- rbind(data1.df, data2.df)
predictions<-round(predict(glm(class~., x.df, family = binomial), type="response"))

t <- table(predictions, x.df$class)
prop <- (t[1,2] + t[2,1]) / (t[1,1] + t[1,2] + t[2,1] + t[2,2])
cat("The proportion of \"off-diagonal\" elements in table:", prop)
```

```
## The proportion of "off-diagonal" elements in table: 0.05
```

In the following code chunk, we input seven measurements for each of 5000 asteroids. The data frame is `df`. There is another variable, `q`, that is also loaded and which we will utilize later.

```
load(url("http://www.stat.cmu.edu/~mfarag/350/HW_09_PCA.Rdata"))
names(df)
```

```
## [1] "diameter" "albedo"   "a"        "e"        "i"        "per_y"    "H"
```

Question 5

(20 points)

Perform PCA on the data frame `df`. Use the rule-of-thumb in the notes to determine the number of principal components (or PCs) to retain, and for those PCs, indicate the mapping from PC to original variables. (Also, display the proportion of variance explained by the PCs you retain.) Are any of the original variables "unimportant" within the context of the retained PCs?

```r
prc <- prcomp(df, scale=TRUE)
prc$rotation
```

```
##                  PC1         PC2          PC3         PC4         PC5
## diameter -0.10427784  0.64018993 -0.381807654 -0.06374223  0.08430670
## albedo    0.02926049  0.27356501  0.891476176 -0.02430516 -0.21790935
## a        -0.61875917 -0.08069235  0.091831781 -0.14969568  0.25344191
## e        -0.30222711  0.04700618 -0.006640299  0.94592379 -0.10576940
## i        -0.37669959 -0.08308185 -0.183319322 -0.21654799 -0.87665888
## per_y    -0.60414428 -0.08470000  0.131810599 -0.16680797  0.31781953
## H         0.08469313 -0.70179878 -0.005997209  0.05871658  0.02116336
##                   PC6          PC7
## diameter -0.648700557  0.039023087
## albedo   -0.285476999 -0.001974438
## a        -0.029727335 -0.717403265
## e        -0.006606603  0.020061597
## i        -0.003403910  0.046405205
## per_y     0.035335507  0.693093653
## H        -0.703919686  0.029514963
```

```r
#plot(1:7,cumsum(prc$sdev^2)/sum(prc$sdev^2),pch=19,xlab="PC",
#     ylab="Cumulative Explained Variance",ylim=c(0,1))
#abline(h=0.9,lty=2)
```

```
Based on the rule of thumb, I would retain the first 5 PCs.
PC1 maps about equally to a and per_y.
PC2 maps predominantly to diameter and H.
PC3 maps predominantly to albedo.
PC4 maps predominantly to e.
and PC 5 maps predominantly to i.
In terms of "unimportance" of original variables, the one that makes sense
would be i since PC5 is the only PC mapping predominantly to it and 5
is pretty low in terms of proportion of variance.
```

Question 6

(20 points)

Something that one can do with principal components is regression: after all, all you've done is transform your data to a new coordinate system. Below, linearly regress the variable `q` upon all the variables in `df`, and print the adjusted R^2 and the sum of squared errors for the model. Then repeat linear regression, except now regress the variable `q` upon only the retained PCs. Again, print out the adjusted R^2 and the sum of squared errors. Are the second value close to the first? (They often will be, but don't have to be.) (Hint: look at the names of the list elements that are output by the *summary* of your linear regression fits, as one of those list elements may help you with extracting the adjusted R^2. As far as the sum of squared errors, you need to simply compute the `sum()` of the difference between the observed values of `q` and the predicted values from `predict()`.)

```r
df.lm <- lm(q~., df)
retained <- data.frame(PC1 = prc$x[,1],
                       PC2 = prc$x[,2],
                       PC3 = prc$x[,3],
                       PC4 = prc$x[,4],
                       PC5 = prc$x[,5])
prc.lm <- lm(q~., retained)

cat("Adjusted R^2 for q~df =", summary(df.lm)$adj.r.squared, "\n")
```

```
## Adjusted R^2 for q~df = 0.7390329
```

```r
cat("Sum of squared error for q~df =", sum((q-predict(df.lm))^2), "\n\n")
```

```
## Sum of squared error for q~df = 2469.368
```

```r
cat("Adjusted R^2 for q~retained =", summary(prc.lm)$adj.r.squared, "\n")
```

```
## Adjusted R^2 for q~retained = 0.1842183
```

```r
cat("Sum of squared error for q~df =", sum((q-predict(prc.lm))^2), "\n")
```

```
## Sum of squared error for q~df = 7722.323
```

```
The second values are not close to the first values. The adjusted R^2 of the
second regression is much lower than the first and the SSE is much higher for
the second regression than the first.
```