ST552 Homework 2

Nick Sun

January 22, 2019

Part 1

Question 1

a.

Consider the simple linear regression model:

$$\begin{aligned} y_i = \beta_0 + \beta_1 x_i + \epsilon_i, i = 1, \ldots, n \\ \text{where } \epsilon_i \sim N(0, \sigma^2) \end{aligned}$$

First we will write out the form of y, X, and ϵ.

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

b.

Now we will calculate $X^T X, X^T y, (X^T X)^{-1}$.

$$X^T X = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \end{pmatrix} \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} = \begin{pmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{pmatrix} = \begin{pmatrix} n & n\bar{x} \\ n\bar{x} & \sum_{i=1}^n x_i^2 \end{pmatrix}$$

$$X^T y = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \end{pmatrix}$$

$$(X^T X)^{-1} = \begin{pmatrix} \frac{\sum x_i^2}{n(\sum x_i^2 - n\bar{x}^2)} & \frac{-\bar{x}}{\sum x_i^2 - n\bar{x}^2} \\ \frac{-\bar{x}}{\sum x_i^2 - n\bar{x}^2} & \frac{1}{\sum x_i^2 - n\bar{x}^2} \end{pmatrix} = \begin{pmatrix} \frac{\sum x_i^2}{n(\sum (x_i - \bar{x})^2)} & \frac{-\bar{x}}{\sum (x_i - \bar{x})^2} \\ \frac{-\bar{x}}{\sum (x_i - \bar{x})^2} & \frac{1}{\sum (x_i - \bar{x})^2} \end{pmatrix}$$

c.

Lastly, we will calculate the vector of coefficients $\hat{\beta} = (X^TX)^{-1}X^Ty$. Thankfully we've done most of the hard work already!

$$\hat{\beta} = (X^TX)^{-1}X^Ty = \begin{pmatrix} \frac{\sum x_i^2}{n(\sum(x_i-\bar{x})^2)} & \frac{-\bar{x}}{\sum(x_i-\bar{x})^2} \\ \frac{-\bar{x}}{\sum(x_i-\bar{x})^2} & \frac{1}{\sum(x_i-\bar{x})^2} \end{pmatrix} \begin{pmatrix} \sum y_i \\ \sum x_i y_i \end{pmatrix} = \begin{pmatrix} \frac{(\sum x_i^2)(\sum y_i) - (\sum x_i)(\sum x_i y_i)}{n(\sum(x_i-\bar{x})^2)} \\ \frac{\sum y_i(x_i-\bar{x})}{\sum(x_i-\bar{x})^2} \end{pmatrix}$$

Question 2

In this question, we are given a model with no intercept:

$$\begin{aligned} y_i = \beta_1 x_i + \epsilon_i, i = 1, \ldots, n \\ \text{where } \epsilon_i \sim N(0, \sigma^2) \end{aligned}$$

We can derive the form of the estimate for slope using the same process as above, except this time instead of a matrix we only really have to deal with vectors. First, expanding this model into matrix form.

For $y_i = \beta_1 x_i + \epsilon_i$, $i = 1, 2, \cdots, n$, where ϵ_i are independent $N(0, \sigma^2)$ random errors.

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \begin{pmatrix} \beta_1 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

Now we can calculate X^TX, X^Ty, and $(X^TX)^{-1}$

$$X^TX = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \sum x_i^2$$

$$(X^TX)^{-1} = \frac{1}{\sum x_i^2}$$

$$X^Ty = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \sum x_i y_i$$

Finally, our slope estimate formula is found by calculating the hat matrix:

$$\hat{\beta} = (X^TX)^{-1}X^Ty = \frac{\sum x_i y_i}{\sum x_i^2}$$

Part 2

Question 1

```
library(faraway)
head(teengamb)
```

```
##   sex status income verbal gamble
## 1   1     51   2.00      8    0.0
## 2   1     28   2.50      8    0.0
## 3   1     37   2.00      6    0.0
## 4   1     28   7.00      4    7.3
## 5   1     65   2.00      8   19.6
## 6   1     61   3.47      6    0.1
```

Consider the regression model:

$$gamble_i = \beta_0 + \beta_1 sex_i + \beta_2 status_i + \beta_3 income_i + \beta_4 verbal_i + \epsilon_i, i = 1, \ldots, 47$$

Part a.

Construct the design matrix X and response vector y in R

```
y <-  teengamb$gamble
design_matrix <- data.frame(intercept = rep(1,dim(teengamb)[1]),
                            sex = teengamb$sex, status = teengamb$status,
                            income = teengamb$income,
                            verbal = teengamb$verbal)
design_matrix <- cbind(rep(1,47),
                       teengamb$sex,
                       teengamb$status,
                       teengamb$income,
                       teengamb$verbal)

head(design_matrix)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    1   51 2.00    8
## [2,]    1    1   28 2.50    8
## [3,]    1    1   37 2.00    6
## [4,]    1    1   28 7.00    4
## [5,]    1    1   65 2.00    8
## [6,]    1    1   61 3.47    6
```

Part b.

Find the least squares estimates using matrix algebra in R and verify your answers by fitting a regression model using lm()

```r
xtx <- t(design_matrix) %*% design_matrix
xty <- t(design_matrix) %*% y
xtx_inv <- solve(xtx)

beta = xtx_inv %*% xty
beta
```

```
##              [,1]
## [1,]  22.55565063
## [2,] -22.11833009
## [3,]   0.05223384
## [4,]   4.96197922
## [5,]  -2.95949350
```

```r
model <- lm(gamble ~ sex + status + income + verbal, data = teengamb)
model$coefficients
```

```
##  (Intercept)          sex       status       income       verbal 
##  22.55565063 -22.11833009   0.05223384   4.96197922  -2.95949350
```

Our model coefficients are very similar to the coefficients we calculated using matrix algebra! Whoa.

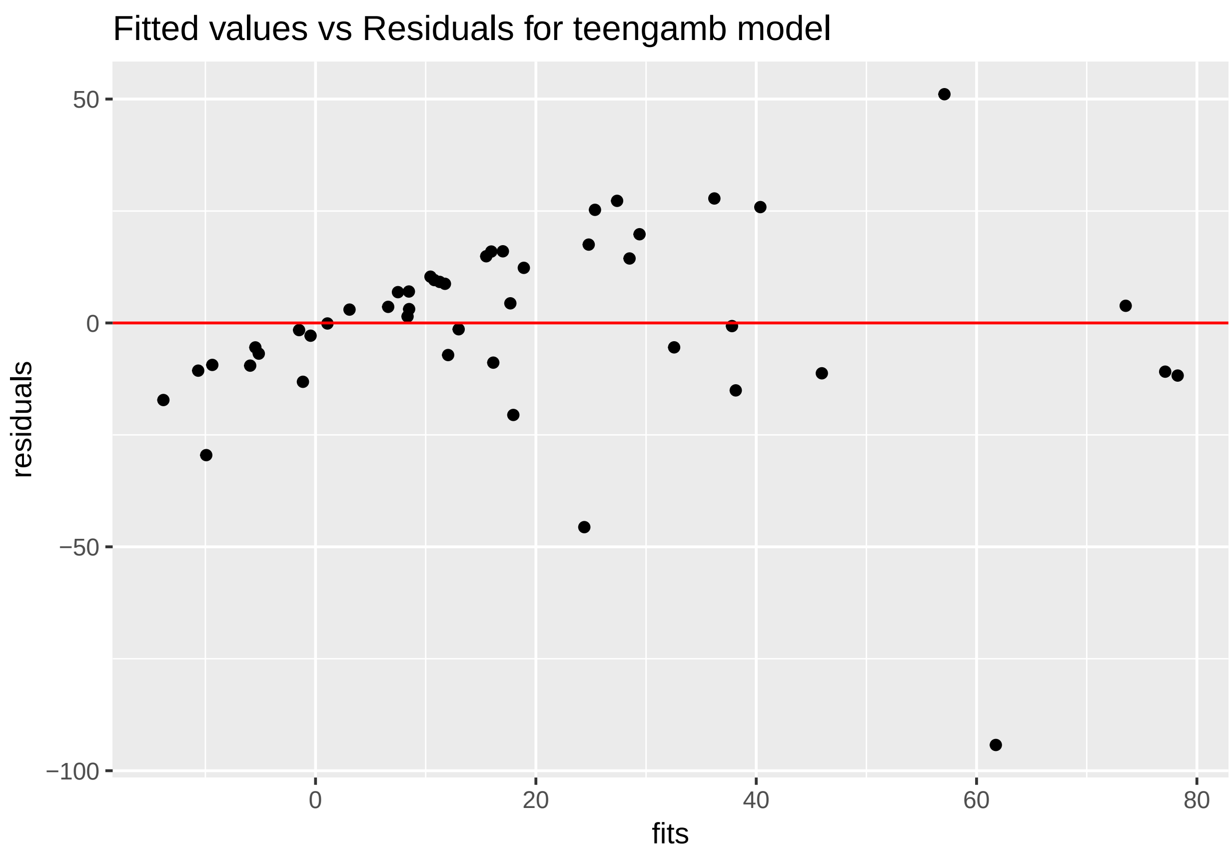
Part c.

Find the fitted values and residuals using matrix algebra in R and present a plot of residuals against the fitted values.

```r
fits <- design_matrix %*% beta
residuals <- fits - teengamb$gamble

fitvsres <- data.frame(fits = fits, residuals = residuals)

library(ggplot2)
ggplot(aes(x = fits, y = residuals), data = fitvsres) +
  geom_point() +
  ggtitle("Fitted values vs Residuals for teengamb model") +
  geom_hline(yintercept = 0, color = "red")
```



There seems to be some heteroskedasticity in the distribution of the residuals. This might be a good candidate for a transformation!

Question 2

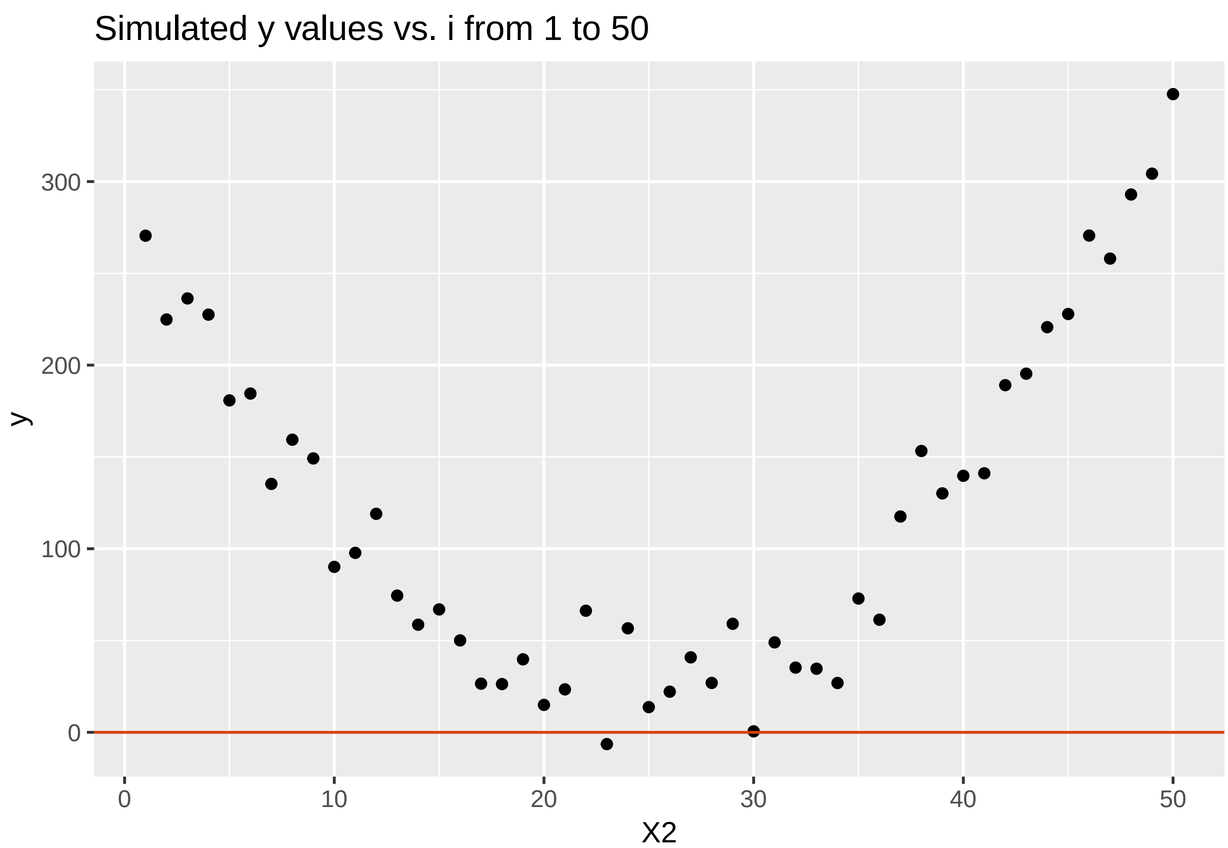
Consider the following regression model:

$$y_i = -5 + i + 0.5(i - 25)^2 + \epsilon_i, i = 1, \ldots, 50 \text{ and } \epsilon_i \sim Normal(0, 400)$$

Simulate a realization of this model. Set up the design matrix, vector of coefficients, and error vector.

```
X_matrix <- cbind(rep(1, 50), seq(from = 1, to = 50, by = 1), (seq(from = 1, to = 50, by = 1) - 25)^2)
errors <- rnorm(50, mean = 0, sd = sqrt(400))
betas <- c(-5, 1, 0.5)
y <- X_matrix %*% betas + errors

q2plotdata <- data.frame(X_matrix, y)
ggplot(aes(x = X2, y = y), data = q2plotdata) +
  geom_point() +
  geom_hline(yintercept=0, color = "#D73F09") +
  ggtitle("Simulated y values vs. i from 1 to 50")
```



Just for fun, let's see how well lm() can estimate the original coefficients of this model. Our coefficients were -5 for the intercept, 1 for β_1, and .5 for β_2.

```
test_model <- lm(formula = y ~ X2 + X3, data = q2plotdata); test_model$coefficients
```

```
## (Intercept)          X2          X3
##  -3.1209740   0.9426069   0.4756340
```

Honestly not bad, considering how much noise we added!

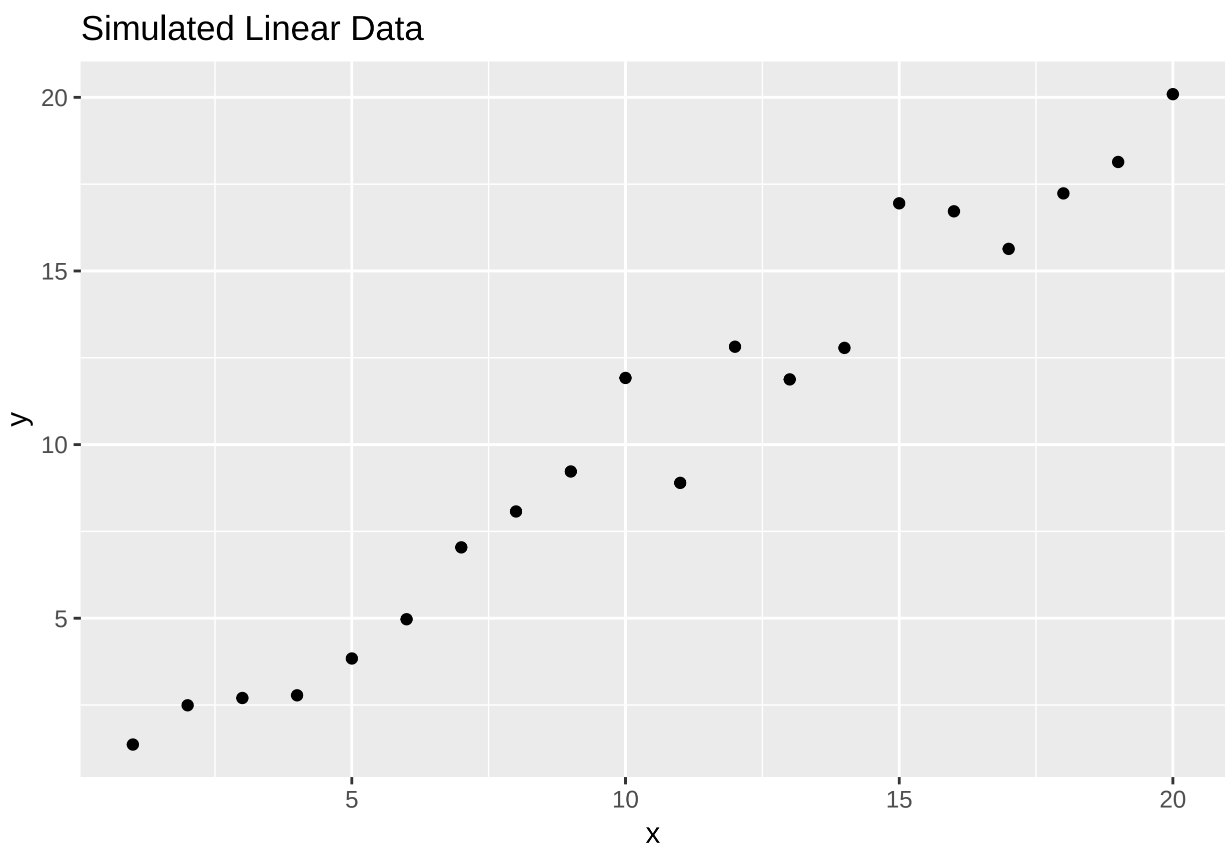
Question 3

Generate some artificial data:

```
set.seed(18492)
n <- 20
x <- 1:n
y <- x + rnorm(n)
```

Let's start by just plotting the data. We can see that it looks pretty linear.

```
q3plotdata <- data.frame(x, y)
ggplot(aes(x = x, y =y), data = q3plotdata) + geom_point() + ggtitle("Simulated Linear Data")
```



Fit a polynomial in x for predicting y, computing $\hat{\beta}$ both "by hand" using linear algebra and using the lm() function. Let's try the matrix algebra first. We begin by defining some functions that will create our design matrix.

```
matrix_degree_n <- function(x, degree) {

  output <- matrix(0, nrow = length(x), ncol = degree+1)
  output[,1] <- rep(1, length(x))
  output[,2] <- x
  if(degree >= 2){
    for(col in c(2:degree+1)) {
      output[,col] <- x^(col-1)
    }
  }
  return(output)
}

get_betas <- function(m, y) {
  xtx <- t(m) %*% m
  inv_xtx <- solve(xtx)
  xty <- t(m) %*% y
  betas <- inv_xtx %*% xty

  return(betas)
}
```

Now that we have a handy function to create the matrices in question, let's take a crack at comparing the output of out matrix algebra with the lm() function.

```r
m1 <- matrix_degree_n(x, 1)
get_betas(m1, y)
```

```
##            [,1]
## [1,] -0.1697881
## [2,]  0.9950225
```

```r
model1 <- lm(y ~ x); model1$coefficients
```

```
## (Intercept)           x
##  -0.1697881   0.9950225
```

```r
m2 <- matrix_degree_n(x, 2)
get_betas(m2, y)
```

```
##              [,1]
## [1,] -0.258282154
## [2,]  1.019157225
## [3,] -0.001149274
```

```r
model2 <- lm(y ~ x + I(x^2)); model2$coefficients
```

```
##  (Intercept)            x       I(x^2)
## -0.258282154  1.019157225 -0.001149274
```

```r
m3 <- matrix_degree_n(x, 3)
get_betas(m3, y)
```

```
##              [,1]
## [1,]  0.452669117
## [2,]  0.656388753
## [3,]  0.041001987
## [4,] -0.001338135
```

```r
model3 <- lm(y ~ x + I(x^2) + I(x^3)); model3$coefficients
```

```
##  (Intercept)            x       I(x^2)       I(x^3)
##  0.452669117  0.656388753  0.041001987 -0.001338135
```

```r
m4 <- matrix_degree_n(x, 4)
get_betas(m4, y)
```

```
##               [,1]
## [1,]  1.8364918394
## [2,] -0.4261471384
## [3,]  0.2600595005
## [4,] -0.0172912958
## [5,]  0.0003798372
```

```r
model4 <- lm(y ~ x + I(x^2) + I(x^3) + I(x^4)); model4$coefficients
```

```
##   (Intercept)             x        I(x^2)        I(x^3)        I(x^4)
##  1.8364918393 -0.4261471384  0.2600595005 -0.0172912958  0.0003798372
```

```r
m5 <- matrix_degree_n(x, 5)
tryCatch(get_betas(m5, y), error = function(c) message("Uh oh, computation error"))
```

```
##               [,1]
## [1,]  1.954691e+00
## [2,] -5.520943e-01
## [3,]  2.980770e-01
## [4,] -2.193203e-02
## [5,]  6.251111e-04
## [6,] -4.671885e-06
```

```r
model5 <- lm(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5)); model5$coefficients
```

```
##   (Intercept)             x        I(x^2)        I(x^3)        I(x^4)
##  1.954691e+00 -5.520943e-01  2.980770e-01 -2.193203e-02  6.251111e-04
##        I(x^5)
## -4.671885e-06
```

```r
m6 <- matrix_degree_n(x, 6)
tryCatch(get_betas(m6, y), error = function(c) message("Uh oh, computation error"))
```

```
## Uh oh, computation error
```

```r
model6 <- lm(y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6)); summary(model6)
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5) + I(x^6))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.17827 -0.51555  0.04396  0.46379  2.03905
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.733e+00  4.005e+00   0.682    0.507
## x           -1.597e+00  4.593e+00  -0.348    0.734
## I(x^2)       7.266e-01  1.736e+00   0.419    0.682
## I(x^3)      -9.819e-02  2.957e-01  -0.332    0.745
## I(x^4)       7.225e-03  2.500e-02   0.289    0.777
## I(x^5)      -2.781e-04  1.025e-03  -0.271    0.790
## I(x^6)       4.340e-06  1.622e-05   0.268    0.793
##
## Residual standard error: 1.239 on 13 degrees of freedom
## Multiple R-squared:  0.9707, Adjusted R-squared:  0.9572
## F-statistic: 71.78 on 6 and 13 DF,  p-value: 3.279e-09
```

Uh oh, we get an error here for k = 6. The actual logged error message *System is computationally singular* indicates that the matrix is now no longer invertible. This happens when the columns are linear combinations of one another – we've been playing with fire here since all the columns are functions of the first column! Checking the parameter estimates from the lm() function for k = 6 shows that all of the estimates are not statistically significant, even the intercept!

A natural question to ask is how the lm() function even calculates these estimates if the matrix is not invertible. The short answer is that the lm() function makes use of QR decomposition to turn the design matrix into an orthonormal matrix Q and a non-singular upper triangular matrix R. Now that $X = QR$, we can substitute it into $X^T X\hat{\beta} = X^T y$ to get a new equation $R\hat{\beta} = Q^T y$ which is easily solved for $\hat{\beta}$.