Assignment 1 - STATS 4850G/9850G

Instructor: Dr. Guowen Huang

Winter 2025

- **Student name: Andy Yuan**
- **Student number: 251159206**

Question 1

a)

Write your answer here.

Assumptions:

- X and Y are continuous RVs
- $Y_0 = f(x_0) + \epsilon$, there is error in our response's relationship with our covariates
- $\hat{Y_0} = f(\hat{x_0})$
- The error term is independent and identically distributed and has a mean of 0 with variance σ^2
- The variance is constant
- The error term and the covariates are independent

$$E\Big[\Big(Y_0 - \hat{f}(x_0)\Big)^2\Big] = E[f(x_0) + \epsilon - \hat{f}(x_0)]^2$$

$$= E[(f(x_0) - \hat{f}(x_0)) + \epsilon]^2$$

$$= E[(f(x_0) - \hat{f}(x_0))^2 + 2\epsilon(f(x_0) - \hat{f}(x_0)) + \epsilon^2]$$

$$= E(f(x_0) - \hat{f}(x_0))^2 + E(2\epsilon(f(x_0) - \hat{f}(x_0))) + E(\epsilon^2))$$

$$= E(f(x_0) - \hat{f}(x_0))^2 + 2E(\epsilon)E(f(x_0) - \hat{f}(x_0))) + E(\epsilon^2))$$

$$= E(f(x_0) - \hat{f}(x_0))^2 + 0 + E(\epsilon^2))$$

$$= E[f(x_0) - \hat{f}(x_0)]^2 + Var(\epsilon)$$

$$= E\{[f(x_0) - E(\hat{f}(x_0))] - [\hat{f}(x_0) - E(\hat{f}(x_0))]\}^2 + Var(\epsilon)$$

$$= E\{[f(x_0) - E(\hat{f}(x_0))]^2 + [\hat{f}(x_0) - E(\hat{f}(x_0))]^2 - 2[f(x_0) - E(\hat{f}(x_0))][\hat{f}(x_0) - E(\hat{f}(x_0))]\} + Var(\epsilon)$$

$$= E[f(x_0) - E(\hat{f}(x_0))]^2 + Var(\hat{f}(x_0)) - 2E[f(x_0) - E(\hat{f}(x_0))] \times [\hat{f}(x_0) - E(\hat{f}(x_0))] + Var(\epsilon)$$

$$= [f(x_0) - E(\hat{f}(x_0))]^2 + Var(\hat{f}(x_0)) - 2E[f(x_0) - E(\hat{f}(x_0))] \times [\hat{f}(x_0) - E(\hat{f}(x_0))] + Var(\epsilon)$$

$$= \{Bias[\hat{f}(x_0)]\}^2 + Var(\hat{f}(x_0)) - 2E[f(x_0) - E(\hat{f}(x_0))] \times [\hat{f}(x_0) - E(\hat{f}(x_0))] + Var(\epsilon)$$

$$= \{Bias[\hat{f}(x_0)]\}^2 + Var(\hat{f}(x_0)) + Var(\epsilon) - 2E[f(x_0)\hat{f}(x_0) - \hat{f}(x_0)E(\hat{f}(x_0)) - f(x_0)E(\hat{f}(x_0)) + (E(\hat{f}(x_0)))^2]$$

$$= \{Bias[\hat{f}(x_0)]\}^2 + Var(\hat{f}(x_0)) + Var(\epsilon) - 2[f(x_0)E(\hat{f}(x_0)) - (E(\hat{f}(x_0)))^2 - f(x_0)E(\hat{f}(x_0)) + (E(\hat{f}(x_0)))^2]$$

$$= \{Bias[\hat{f}(x_0)]\}^2 + Var(\hat{f}(x_0)) + Var(\epsilon) - 0$$

$$\therefore E\Big[\Big(Y_0 - \hat{f}(x_0)\Big)^2\Big] = \{Bias[\hat{f}(x_0)]\}^2 + Var(\hat{f}(x_0)) + Var(\epsilon)$$

b)

Write your answer here. (3 to 5 lines)

Bias-variance trade off decides which model is best to use according to your needs. The more flexible a model is, the less bias it has since it can precisely capture all of the training data. However, this makes it have a higher variance because it would not capture different data sets accurately. On the other hand, a simple model may have greater bias but a better test variance. In the end, we aim to use our training data to fit more data better.

c)

Write your answer here. (3 to 5 lines)

When the goal is prediction, it may not always be best to use the most flexible model. The most flexible model typically overfits the training data, introducing greater variance when the data we use later is not exactly like the training set. If the data sets we use vary in distribution, it is generally better to use a simpler model. Only if the data sets we use are almost completely homogeneous then it may be correct to use the most flexible model.

Question 2

Write your answer here.

Start by centering the predictors by translating them:

$$x_{ij} = (\bar{x}_j + (x_{ij} - \bar{x}_j))$$

Plug into the ridge regression problem:

$$\hat{\beta}^{ridge} = arg\underbrace{min}_{\beta}\left\{\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p}(\bar{x}_j + (x_{ij} - \bar{x}_j))\beta_j)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\right\}$$

$$\hat{\beta}^{ridge} = arg\underbrace{min}_{\beta}\left\{\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p}\bar{x}_j\beta_j - \sum_{j=1}^{p}(x_{ij} - \bar{x}_j)\beta_j)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\right\}$$

We can define a new centered intercept $\beta_0^c = \beta_0 + \sum_{j=1}^{p}\bar{x}_j\beta_j$

$$\hat{\beta}^{ridge} = arg\underbrace{min}_{\beta}\left\{\sum_{i=1}^{N}(y_i - \beta_0^c - \sum_{j=1}^{p}(x_{ij} - \bar{x}_j)\beta_j)^2 + \lambda\sum_{j=1}^{p}\beta_j^2\right\}$$

Only our centered intercept was defined to be the original intercept plus the weighted sum of the predictors. This is originally a number not attached to any covariate, so it needs to

be translated. Otherwise, for our other predictor coefficients, their covariates were already translated, so they will be the same value. Intuitively, this is because they are the 'slope' of a line, which doesn't change when a line is translated.

$$\hat{\beta}^c = arg \underbrace{min}_{\beta^c} \left\{ \sum_{i=1}^{N} (y_i - \beta_0^c - \sum_{j=1}^{p} (x_{ij} - \bar{x}_j)\beta_j^c)^2 + \lambda \sum_{j=1}^{p} (\beta_j^c)^2 \right\}$$

Thus, the ridge regression problem is equal to the centered ridge regression.

We defined our centered intercept as $\beta_0^c = \beta_0 + \sum_{j=1}^{p} \bar{x}_j \beta_j$

- In least-squares, $\hat{\beta}_0 = \bar{y} - \sum_{j=1}^{p} \bar{x}_j \beta_j$
- Substituting it into our centered intercept equation, $\hat{\beta}_0^c = \bar{y} - \sum_{j=1}^{p} \bar{x}_j \beta_j + \sum_{j=1}^{p} \bar{x}_j \beta_j = \bar{y}$
- Therefore, the centered intercept is equal to the mean of the responses $\bar{y}$

Question 3

a)

```
suppressWarnings(library(MASS))
data(Boston)
Boston$chas <- factor(Boston$chas,levels=c("0","1"),labels = c("no","yes"))
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

```r
# Write your code here.

boston_model <- lm(medv ~ crim + zn + indus + chas + nox +
                     rm + age + dis + rad + tax + ptratio +
                     black + lstat + chas*dis, data = Boston)
anova(boston_model)
```

```
## Analysis of Variance Table
##
## Response: medv
##             Df  Sum Sq Mean Sq  F value    Pr(>F)
## crim         1  6440.8  6440.8 287.4888 < 2.2e-16 ***
## zn           1  3554.3  3554.3 158.6502 < 2.2e-16 ***
## indus        1  2551.2  2551.2 113.8762 < 2.2e-16 ***
## chas         1  1529.8  1529.8  68.2858 1.328e-15 ***
## nox          1    76.2    76.2   3.4034 0.0656656 .
## rm           1 10938.1 10938.1 488.2304 < 2.2e-16 ***
## age          1    90.3    90.3   4.0292 0.0452675 *
## dis          1  1779.5  1779.5  79.4293 < 2.2e-16 ***
## rad          1    34.1    34.1   1.5236 0.2176646
## tax          1   329.6   329.6  14.7099 0.0001417 ***
## ptratio      1  1309.3  1309.3  58.4419 1.110e-13 ***
## black        1   593.3   593.3  26.4840 3.850e-07 ***
## lstat        1  2410.8  2410.8 107.6094 < 2.2e-16 ***
## chas:dis     1    78.6    78.6   3.5092 0.0616240 .
## Residuals  491 11000.2    22.4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

b)

Write your answer here.

An F-test can be used. According to the F-test using anova, (nox, rad, and the interaction between chas and dis) are not significant at the 5% level

c)

```r
# Write your code here.
suppressWarnings(library(faraway))
best_model <- step(boston_model,
```

```
                    trace = 0, steps = 100000) # best model using step function
sumary(best_model)
```

```
##                 Estimate  Std. Error  t value  Pr(>|t|)
## (Intercept)  36.9136582   5.0637163   7.2898 1.243e-12
## crim         -0.1075762   0.0326987  -3.2899 0.0010739
## zn            0.0475770   0.0135196   3.5191 0.0004731
## chasyes       6.6854987   2.2758230   2.9376 0.0034623
## nox         -18.8089327   3.6076690  -5.2136 2.730e-07
## rm            3.8468808   0.4059940   9.4752 < 2.2e-16
## dis          -1.5205996   0.1858495  -8.1819 2.396e-15
## rad           0.3084954   0.0634166   4.8646 1.546e-06
## tax          -0.0125030   0.0033858  -3.6928 0.0002466
## ptratio      -0.9425261   0.1287535  -7.3204 1.012e-12
## black         0.0091729   0.0026678   3.4383 0.0006349
## lstat        -0.5070014   0.0480213 -10.5578 < 2.2e-16
## chasyes:dis  -1.2932713   0.6880113  -1.8797 0.0607352
##
## n = 506, p = 13, Residual SE = 4.72414, R-Squared = 0.74
```

Notably, (crim, zn, chas, nox, rm, dis, rad, tax, ptratio, black, lstat, and the interaction between chas and dis) remained in the model, this means that (indus and age) are not present in the model, despite them having better p-values than (nox, rad, and the interaction between chas and dis) in the model with all covariates.

d)

Since this model is a simple linear model, the coefficients do not have to go under any transformation to be interpreted.

- **rad**: for every increase in index of accessibility to radial highways, the median value of owner-occupied homes increases by $308.4954
- **tax**: for every 10,000 dollar increase in full-value property-tax rate, the median value of owner-occupied homes decreases by $12.5030
- **chas**: if the tract bounds the river, the median value of owner-occupied homes increases by $6,685.4987
- **dis**: for every increase in the mean of distances to five Boston employment centres, the median value of owner-occupied homes decreases by $1,520.5996

e)

```r
suppressWarnings(library(glmnet))
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```r
x <- model.matrix(medv ~ crim + zn + chas + nox +
                    rm + dis + rad + tax + ptratio +
                    black + lstat + chas*dis, data = Boston)[,-1]
y <- Boston$medv
intercept <- mean(y)
ridge.mod <- glmnet(x, y, alpha = 0, standardize = TRUE)
# Considers a grid with 100 lambda values

cvfitridge <- cv.glmnet(x, y, alpha = 0,
                   type.measure = "mse") # default 10-fold CV

lambda_min <- cvfitridge$lambda.min
lambda_min # best ridge regression lambda value
```

```
## [1] 0.6777654
```

```r
ridge.mod.best <- glmnet(x, y, alpha = 0,
                         lambda = lambda_min) # best ridge model
coef(ridge.mod.best, trace = 0)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                          s0
## (Intercept)  28.178521304
## crim         -0.086505769
## zn            0.034253211
## chasyes       3.761892305
## nox         -13.092939784
## rm            4.036386741
## dis          -1.080619109
## rad           0.160211368
## tax          -0.006533597
## ptratio      -0.866718857
## black         0.009083408
## lstat        -0.476197363
## chasyes:dis  -0.322445883
```

```
cvfitlasso <- cv.glmnet(x, y, alpha = 1,
                    type.measure = "mse")
lambda_minlasso <- cvfitlasso$lambda.min
lambda_minlasso # best lambda value for lasso
```

```
## [1] 0.005759334
```

```
lasso.mod.best <- glmnet(x, y, alpha = 1,
                         lambda = lambda_minlasso) # best lasso model
coef(lasso.mod.best)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                          s0
## (Intercept)  36.509269258
## crim         -0.105852891
## zn            0.046714690
## chasyes       6.272013203
## nox         -18.471933340
## rm            3.852952338
## dis          -1.501045986
## rad           0.298798182
## tax          -0.012064740
## ptratio      -0.939211383
## black         0.009124309
## lstat        -0.508593405
## chasyes:dis  -1.163269511
```

f)

```
suppressWarnings(library(data.table))

ridge_coef <- as.matrix(coef(ridge.mod.best))
lasso_coef <- as.matrix(coef(lasso.mod.best))

comparison <- data.table(variable = rownames(ridge_coef),
                         AICBest = as.numeric(coef(best_model)),
                         RidgeBest = as.numeric(ridge_coef),
                         LassoBest = as.numeric(lasso_coef))
comparison
```

```
##        variable       AICBest     RidgeBest     LassoBest
```

```
##          <char>         <num>         <num>         <num>
##  1: (Intercept)  36.913658176  28.178521304  36.509269258
##  2:        crim  -0.107576198  -0.086505769  -0.105852891
##  3:          zn   0.047576997   0.034253211   0.046714690
##  4:     chasyes   6.685498727   3.761892305   6.272013203
##  5:         nox -18.808932675 -13.092939784 -18.471933340
##  6:          rm   3.846880762   4.036386741   3.852952338
##  7:         dis  -1.520599627  -1.080619109  -1.501045986
##  8:         rad   0.308495355   0.160211368   0.298798182
##  9:         tax  -0.012502986  -0.006533597  -0.012064740
## 10:     ptratio  -0.942526092  -0.866718857  -0.939211383
## 11:       black   0.009172857   0.009083408   0.009124309
## 12:       lstat  -0.507001397  -0.476197363  -0.508593405
## 13: chasyes:dis  -1.293271313  -0.322445883  -1.163269511
```

The coefficients between the lasso and AIC methods are similar. The ridge regression coefficients are noticeably more centered than both lasso and AIC, positive values are less than lasso and AIC and negative values are greater than lasso and AIC.

Question 4

```
library(faraway)
medpar <- read.csv("medpar.csv",header=TRUE)
medpar$died <- factor(medpar$died,levels=c("0","1"),labels=c("no","yes"))
medpar$age80 <- factor(medpar$age80,levels=c("0","1"),labels=c("no","yes"))
str(medpar)
```

```
## 'data.frame':    1495 obs. of  3 variables:
##  $ died : Factor w/ 2 levels "no","yes": 1 1 2 1 2 2 2 2 1 1 ...
##  $ los  : int  4 9 3 9 1 4 10 3 5 6 ...
##  $ age80: Factor w/ 2 levels "no","yes": 1 1 2 1 2 1 2 2 1 1 ...
```

a)

```
logis_model <- glm(died ~ los + age80, data = medpar, family = 'binomial')
sumary(logis_model)
```

```
##                Estimate Std. Error z value  Pr(>|z|)
## (Intercept) -0.5156598  0.0949301 -5.4320 5.573e-08
```

```
## los         -0.0299761  0.0077898 -3.8481  0.000119
## age80yes     0.6323079  0.1279893  4.9403 7.800e-07
##
## n = 1495 p = 3
## Deviance = 1880.42994 Null Deviance = 1922.86530 (Difference = 42.43536)
```

```
range(medpar$los)
```

```
## [1]   1 116
```

This model has a logit link, where $log(\frac{p}{1-p}) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}$

In this case, $log(\frac{p}{1-p}) = -0.5156598 - 0.0299761 * x_{i1} + 0.6323079 * x_{i2}$. Since x_{i2} is a factor variable, it effects the intercept term.

b)

- **los**: $exp(-0.0299761) = 0.9704687$, $1 - 0.9704687 = 0.0295313$. This can be interpreted as the odds of being dead in the hospital decrease by 2.95313% for every day you spend in the hospital
- **age80**: $exp(0.6323079) = 1.881949$. This can be interpreted as the odds of being dead in the hospital increase by 88.1949% when you the patient is 80 years or older vs not.

For the **age80** variable, there is a reasonable interpretation that if you are less than 80, your odds of dying are lower in the hospital. It is not perfect but generally acceptable for dying of old age.

For the **los** variable, it suggests that if you spend a lot of time in the hospital, you will not die. This is not true when applied to a healthy person who has never been to the hospital. Their odds of dying are less than that of the people in the hospital.

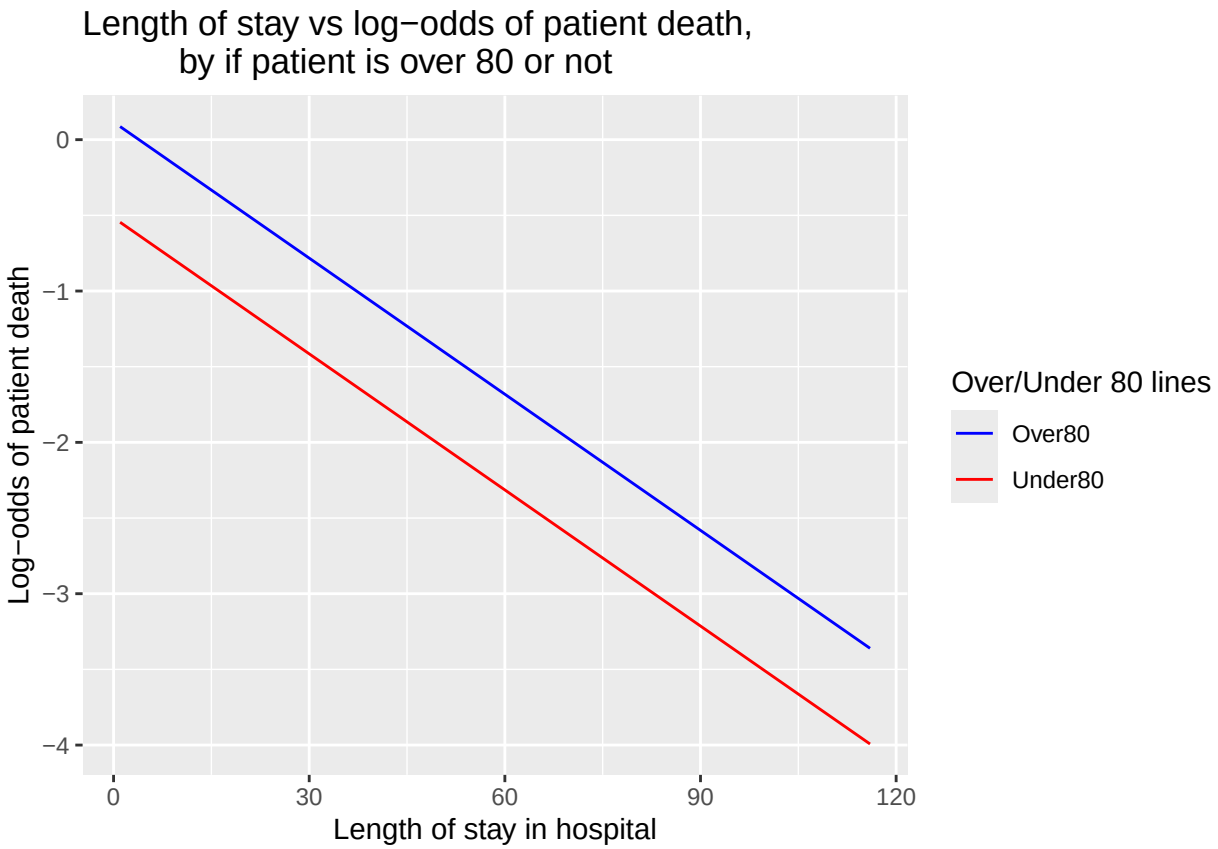
c)

Line 1: age80 = Yes $log(\frac{p}{1-p}) = 0.1166481 - 0.0299761 * x_{i1}$

Line 2: age80 = No $log(\frac{p}{1-p}) = -0.5156598 - 0.0299761 * x_{i1}$

```
suppressWarnings(library(ggplot2))
x <- medpar$los
line_1 <- 0.1166481 - 0.0299761*x
line_2 <- -0.5156598 - 0.0299761*x
medpardf <- data.frame(x, line_1, line_2)
colnames(medpardf) <- c("Los", "Line_1", "Line_2")
```

```
ggplot() +
  geom_line(data=medpardf, aes(x = Los, y = Line_1, colour = "Over80")) +
  geom_line(data=medpardf, aes(x = Los, y = Line_2, colour = "Under80")) +
  xlab("Length of stay in hospital") +
  ylab("Log-odds of patient death") +
  ggtitle("Length of stay vs log-odds of patient death,
          by if patient is over 80 or not") +
  scale_color_manual(name = "Over/Under 80 lines",
                     values = c("Over80" = "blue", "Under80" = "red"))
```



There are less log-odds that a patient would die if they were under 80. Their intercepts differ due to the factor variable "age" but have the same slope since that is controlled by the length of stay (los) variable.

d)

```
logis_model2 <- glm(died ~ los + age80 + los*age80, data = medpar,
                    family = 'binomial')
sumary(logis_model2)
```

```
##                Estimate Std. Error z value  Pr(>|z|)
## (Intercept)  -0.4864043  0.1024962 -4.7456 2.079e-06
## los          -0.0332303  0.0089714 -3.7040 0.0002122
## age80yes      0.5024633  0.2087283  2.4073 0.0160727
## los:age80yes  0.0143673  0.0181915  0.7898 0.4296568
##
## n = 1495 p = 4
## Deviance = 1879.82631 Null Deviance = 1922.86530 (Difference = 43.03899)
```

The link function is: $log(\frac{p}{1-p}) = -0.4864043 - 0.0332303 * x_{i1} + 0.5024633 * x_{i2} + 0.0143673 * x_{i1} * x_{i2}$. Since x_{i2} is a factor variable, it can effect both the intercept and the slope of the line

e)

- **los**: $exp(-0.0332303) = 0.9673158, 1 - 0.9673158 = 0.0326842$. This can be interpreted as the odds of being dead in the hospital decrease by 3.26842% for every day you spend in the hospital
- **age80**: $exp(0.5024633) = 1.652788$. This can be interpreted as the odds of being dead in the hospital increase by 65.2788% when you the patient is 80 years or older vs not.
- **los and age80 interaction term**: $exp(0.0143673) = 1.014471$. Since this term directly effects the **los** variable coefficient, the new **los** coefficient is: $exp(0.0143673 - 0.0332303) = 0.9813138, 1 - 0.9813138 = 0.0186862$. This can be interpreted as if the patient is over 80, the decrease in odds of being dead in the hospital actually increases by 1.4471%, making the length of stay in the hospital less effective at predicting/preventing death.

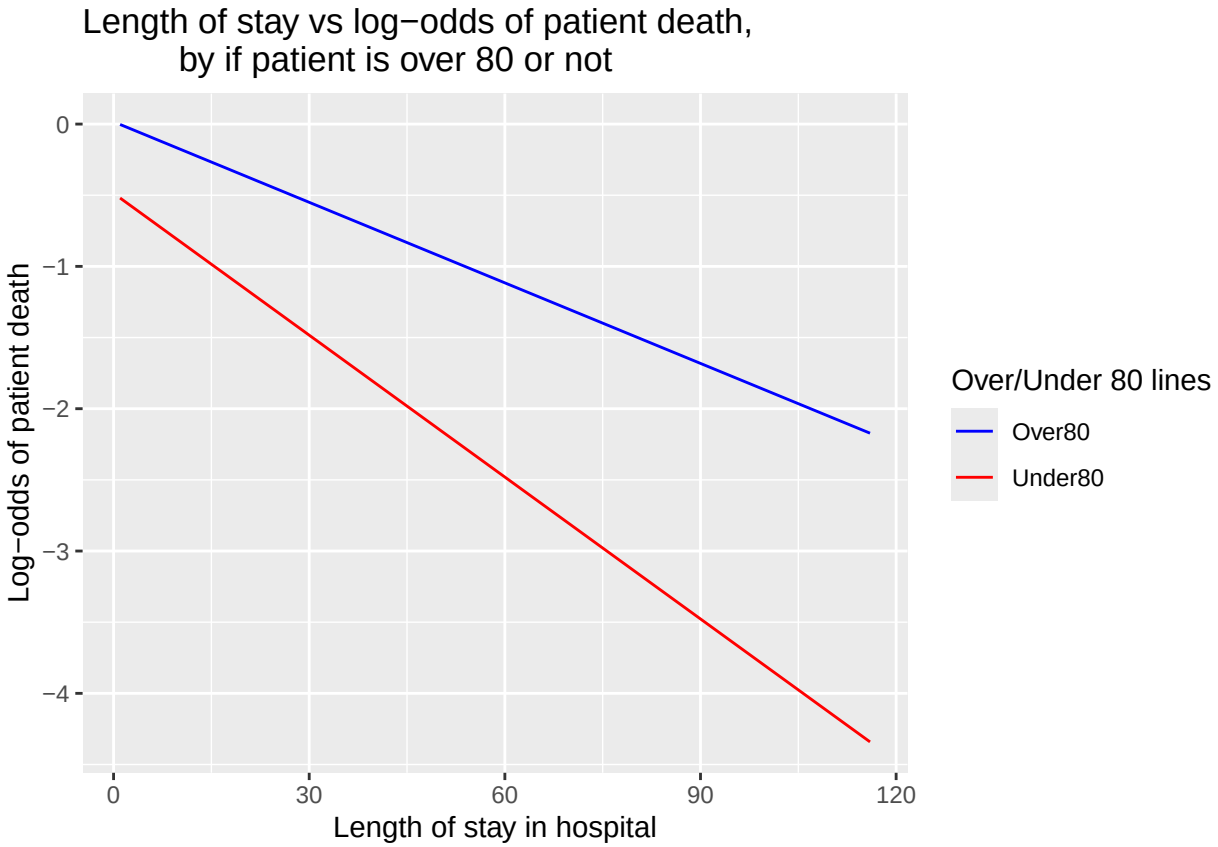
f)

Line 1: age80 = Yes $log(\frac{p}{1-p}) = 0.016059 - 0.018863 * x_{i1}$

Line 2: age80 = No $log(\frac{p}{1-p}) = -0.4864043 - 0.0332303 * x_{i1}$

```r
suppressMessages(library(ggplot2))
x <- medpar$los
line_1_int <- 0.016059 - 0.018863*x
line_2_int <- -0.4864043 - 0.0332303*x
medpardf2 <- data.frame(x, line_1_int, line_2_int)
colnames(medpardf2) <- c("Los", "Line_1_int", "Line_2_int")
ggplot() +
  geom_line(data=medpardf2, aes(x = Los, y = Line_1_int, colour = "Over80")) +
  geom_line(data=medpardf2, aes(x = Los, y = Line_2_int, colour = "Under80")) +
  xlab("Length of stay in hospital") +
```

```
  ylab("Log-odds of patient death") +
  ggtitle("Length of stay vs log-odds of patient death,
          by if patient is over 80 or not") +
  scale_color_manual(name = "Over/Under 80 lines",
                     values = c("Over80" = "blue", "Under80" = "red"))
```



In this plot, the patient under 80 has significantly better odds of not dying with a longer hospital stay than a patient 80 and up. The **age** variable now effects the **los** slope, which gives us better log-odds of the patient not dying if they are under 80.

Question 5

```
library(faraway)
pima_subset <- pima[,-c(4,5)]
pima_subset <- pima_subset[pima_subset$diastolic != 0,]
pima_subset <- pima_subset[pima_subset$glucose != 0,]
pima_subset <- pima_subset[pima_subset$bmi != 0,]
pima_subset$test <- factor(pima_subset$test,
levels=c("0","1"),labels = c("negative","positive"))
```

a)

```
data <- pima_subset
set.seed(123)
indices = sample(1:dim(data)[1], dim(data)[1])
training_size = round(dim(data)[1]*0.8)
training_set = data[indices[1:training_size],]
test_set = data[indices[(training_size+1):dim(data)[1]],]
```

```
pima_model <- glm(test ~ pregnant + glucose +
                    diastolic + bmi + diabetes + age,
                  family = binomial,data = training_set)
summary(pima_model) # model using training set
```

```
##
## Call:
## glm(formula = test ~ pregnant + glucose + diastolic + bmi + diabetes +
##     age, family = binomial, data = training_set)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.880928   0.904086  -9.823  < 2e-16 ***
## pregnant     0.146125   0.037813   3.864 0.000111 ***
## glucose      0.035471   0.004133   8.582  < 2e-16 ***
## diastolic   -0.008526   0.009693  -0.880 0.379089
## bmi          0.093821   0.017818   5.265  1.4e-07 ***
## diabetes     0.890574   0.347817   2.560 0.010453 *
## age          0.005942   0.011055   0.538 0.590896
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 738.46  on 578  degrees of freedom
## Residual deviance: 528.14  on 572  degrees of freedom
## AIC: 542.14
##
## Number of Fisher Scoring iterations: 5
```

b)

```r
best_pima_model <- step(pima_model, trace = 0)
summary(best_pima_model) # best model according to step function
```

```
##
## Call:
## glm(formula = test ~ pregnant + glucose + bmi + diabetes, family = binomial,
##     data = training_set)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.148901   0.798778 -11.454  < 2e-16 ***
## pregnant     0.150829   0.031733   4.753 2.00e-06 ***
## glucose      0.035350   0.003947   8.957  < 2e-16 ***
## bmi          0.089121   0.017144   5.199 2.01e-07 ***
## diabetes     0.905006   0.345947   2.616   0.0089 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 738.46  on 578  degrees of freedom
## Residual deviance: 529.03  on 574  degrees of freedom
## AIC: 539.03
##
## Number of Fisher Scoring iterations: 5
```

Only (pregnant, glucose, bmi, and diabetes) remain in the model after selecting for the best model based on AIC

c)

```r
suppressWarnings(library(ResourceSelection))
```

```
## ResourceSelection 0.3-6   2023-06-27
```

```r
set.seed(123)
hoslem.test(best_pima_model$y, fitted(best_pima_model))
```

```
##
##  Hosmer and Lemeshow goodness of fit (GOF) test
```

```
##
## data:  best_pima_model$y, fitted(best_pima_model)
## X-squared = 5.4118, df = 8, p-value = 0.7128
```

```
# performing hoslem-lemeshow test
```

In this case, H_0 is no lack of fit and H_1 is lack of fit. The test we conduct is the Hosmer-Lemeshow test that follows a chi-squared distribution. The test statistic conclusion is that this model is fit well to the training data since the p-value exceeds most conventional α's like 0.05. We fail to reject the null hypothesis and conclude the model is fit well to the data.

d)

```
summary(training_set$bmi)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.20   27.45   32.40   32.44   36.50   67.10
```

- $logodds_1 = 0.089121 * (27.45) + C$
- $logodds_2 = 0.089121 * (36.50) + C$
- $log\frac{odds_2}{odds_1} = 0.089121 * (36.50) - 0.089121 * (27.45) + C - C = 0.8065451$
- Odds ratio $= exp(0.8065451) = 2.240155$
- Standard error of odds ratio $= exp(0.089121 * (36.50) - 0.089121 * (27.45)) * 0.017144 = 0.03840522$
- The confidence interval of the odds ratio at a 95% confidence level is: $2.240155 \pm 1.96 * 0.03840522 = [2.164881, 2.315429]$

e)

```
suppressWarnings(library(dplyr))
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

```
## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
predict_pima <- predict(best_pima_model, newdata = test_set,
                        type = "response")
testm <- mutate(test_set, predprob = predict_pima)
testm <- mutate(testm, predout = ifelse(predict_pima > 0.5,
                                        "positive", "negative"))
xtabs(~ predout + test, testm)
```

```
##           test
## predout    negative positive
##   negative       82       29
##   positive        8       26
```

Sensitivity: $26/(8+26) = 0.7647059$

Specificity: $82/(82+29) = 0.7387387$

f)

ROC analysis

```
suppressMessages(library(pROC))
```

```
## Warning: package 'pROC' was built under R version 4.3.3
```

```
roc_obj <- roc(response=test_set$test, predictor=predict_pima)
```

```
## Setting levels: control = negative, case = positive
```

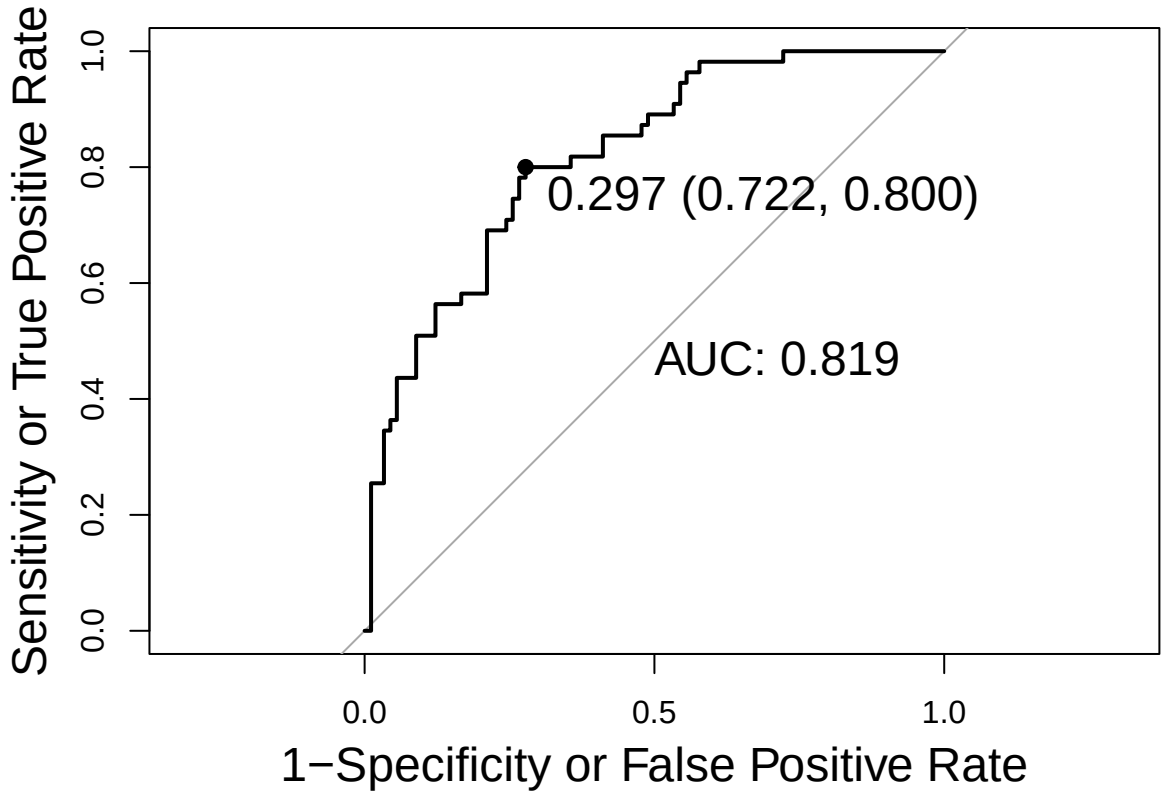
```
## Setting direction: controls < cases
```

```
AUC <- auc(roc_obj)
roc_logis <- c(coords(roc_obj, "b",
                      ret=c("threshold","se","sp","accuracy"),
                      best.method="youden"),AUC)
names(roc_logis) <- c("Threshold","Sensitivity","Specificity",
"Accuracy","AUC")
t(roc_logis)
```

```
##      Threshold Sensitivity Specificity Accuracy  AUC
## [1,] 0.2973867 0.8         0.7222222   0.7517241 0.8193939
```

ROC Curve

```
plot(roc_obj,legacy.axes=TRUE,print.auc=TRUE,print.thres=TRUE,
     print.thres.cex=1.5,print.auc.cex=1.5,cex.lab=1.5,
     xlab="1-Specificity or False Positive Rate",
     ylab="Sensitivity or True Positive Rate")
```



The best threshold for p is 0.297 from the roc() function. The specificity is 0.722 and the sensitivity is 0.8. Sensitivity increased while specificity decreased compared to the 0.5 threshold we calculated in part *e)*. The AUC is 0.819

g)

Constructing the LDA model

```
suppressMessages(library(MASS))
(lda_model <- lda(test ~ pregnant + glucose +
                    diastolic + bmi + diabetes + age,
                  data = training_set))
```

```
## Call:
## lda(test ~ pregnant + glucose + diastolic + bmi + diabetes +
##     age, data = training_set)
##
## Prior probabilities of groups:
##  negative  positive
## 0.6649396 0.3350604
##
## Group means:
##          pregnant  glucose diastolic      bmi  diabetes      age
## negative 3.236364 111.4935  70.34545 30.82052 0.4214805 31.15065
## positive 4.943299 143.1392  75.06701 35.65979 0.5449124 37.03608
##
## Coefficients of linear discriminants:
##                    LD1
## pregnant   0.108422518
## glucose    0.027891651
## diastolic -0.004762913
## bmi        0.064529315
## diabetes   0.629850702
## age        0.004088385
```

```
suppressMessages(library(pROC))
prob_pred = predict(lda_model, newdata = test_set)$posterior[,2]
roc_objLDA <- roc(response=test_set$test, predictor=as.numeric(prob_pred))
```

```
## Setting levels: control = negative, case = positive
```

```
## Setting direction: controls < cases
```

```
AUCLDA <- auc(roc_objLDA)
roc_LDA <- c(coords(roc_objLDA, "b",
                    ret=c("threshold","se","sp","accuracy"),
                    best.method="youden"),AUCLDA)
t(roc_LDA)
```

```
##      threshold sensitivity specificity accuracy
## [1,] 0.2978997 0.7636364   0.7666667   0.7655172 0.8280808
```

The threshold from the LDA is strikingly similar to the threshold found in part *f)*. Sensitivity and specificity are similar levels, but 'balance' each other out more in the LDA. The AUC is greater in the LDA compared to the logistic regression model, which suggests the LDA has a better predictive ability than the logistic regression model.

h)

```
training_set$test <- as.character(training_set$test)
training_set$test[training_set$test=="positive"] <- 1
training_set$test[training_set$test=="negative"] <- 0

test_set$test <- as.character(test_set$test)
test_set$test[test_set$test=="positive"] <- 1
test_set$test[test_set$test=="negative"] <- 0

suppressWarnings(library(class))
suppressWarnings(library(caret))
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:faraway':
##
##     melanoma
```

```
set.seed(2)
knn_pred = knn(train=training_set, test=test_set, cl=training_set$test, k=8)
table(test_set$test, knn_pred)
```

```
##    knn_pred
##      0  1
##   0 79 11
##   1 26 29
```

- Sensitivity: $29/(29+26) = 0.5272727$
- Specificity: $79/(79+11) = 0.8777778$

Sensitivity suffers in the kNN model while specificity rises. This trade off does not appear to be good since one outweighs the other while the other methods tend to balance out specificity and sensitivity.

Question 6

a)

```
set.seed(123)
n <- 1000
p <- 20
zero_prop <- 0.95

X <- matrix(rnorm(n * p), nrow = n, ncol = p) # design matrix
beta <- rnorm(p)
zero_betas <- sample(1:p, size = round(zero_prop * p))
beta[zero_betas] <- 0 # betas
epsilon <- rnorm(n, mean = 0, sd = 1) # error
beta # showing that only one beta is non-zero
```

```
##  [1]  0.000000  0.000000  0.000000  0.000000 -1.097963  0.000000  0.000000
##  [8]  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
## [15]  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
```

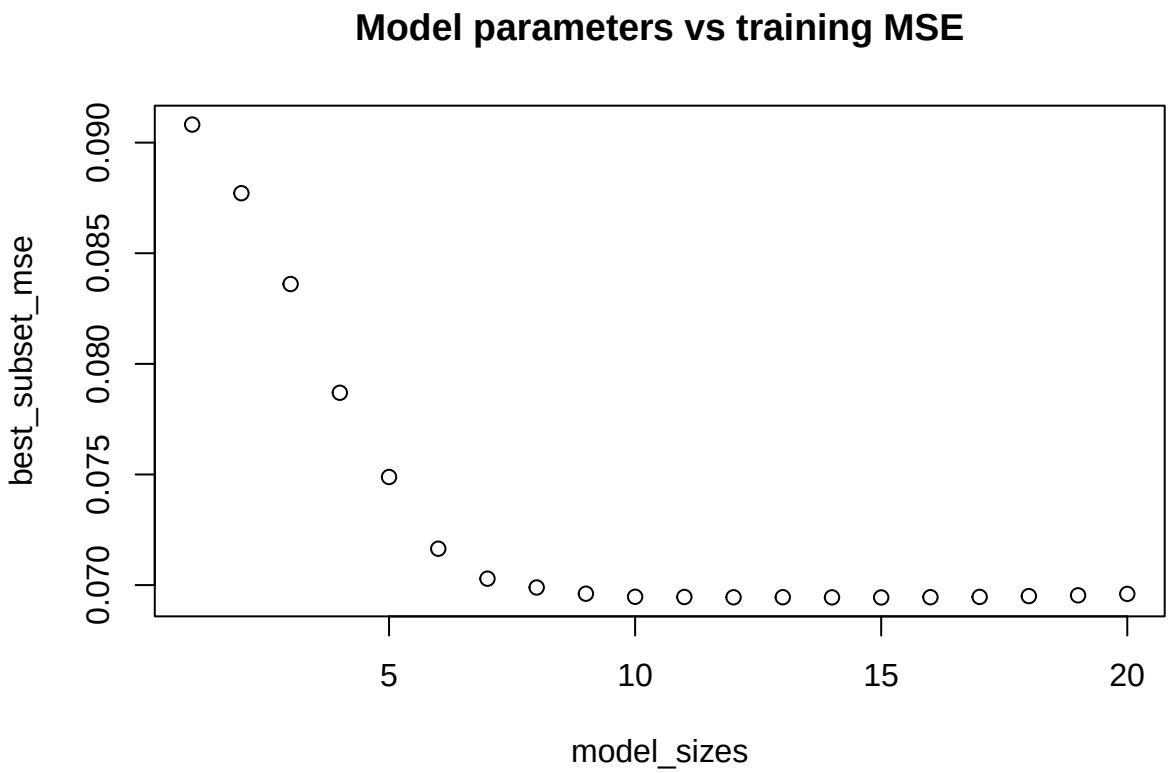
```
y <- X %*% beta + epsilon # response

dataset <- data.frame(y = y, X)
```

b)

```
dataq6 <- dataset
set.seed(123)
indicesq6 = sample(1:dim(dataq6)[1], dim(dataq6)[1])
#below divides into 100 training, 900 test
training_sizeq6 = round(dim(dataq6)[1]*0.1)
training_setq6 = dataq6[indicesq6[1:training_sizeq6],]
test_setq6 = dataq6[indicesq6[(training_sizeq6+1):dim(dataq6)[1]],]
```

c)

```r
suppressWarnings(library(leaps))
best_subsetq6 <- regsubsets(y~., data = training_setq6, nvmax = 20)
# regsubsets is best when used with summary
best_subset_summary <- summary(best_subsetq6)
model_sizes <- seq(1,20, by = 1)
best_subset_mse <- best_subset_summary$rss / (1000 - model_sizes - 1)
plot(model_sizes, best_subset_mse, main = "Model parameters vs training MSE")
```



We see that the more variables, the lower the MSE is for the training data

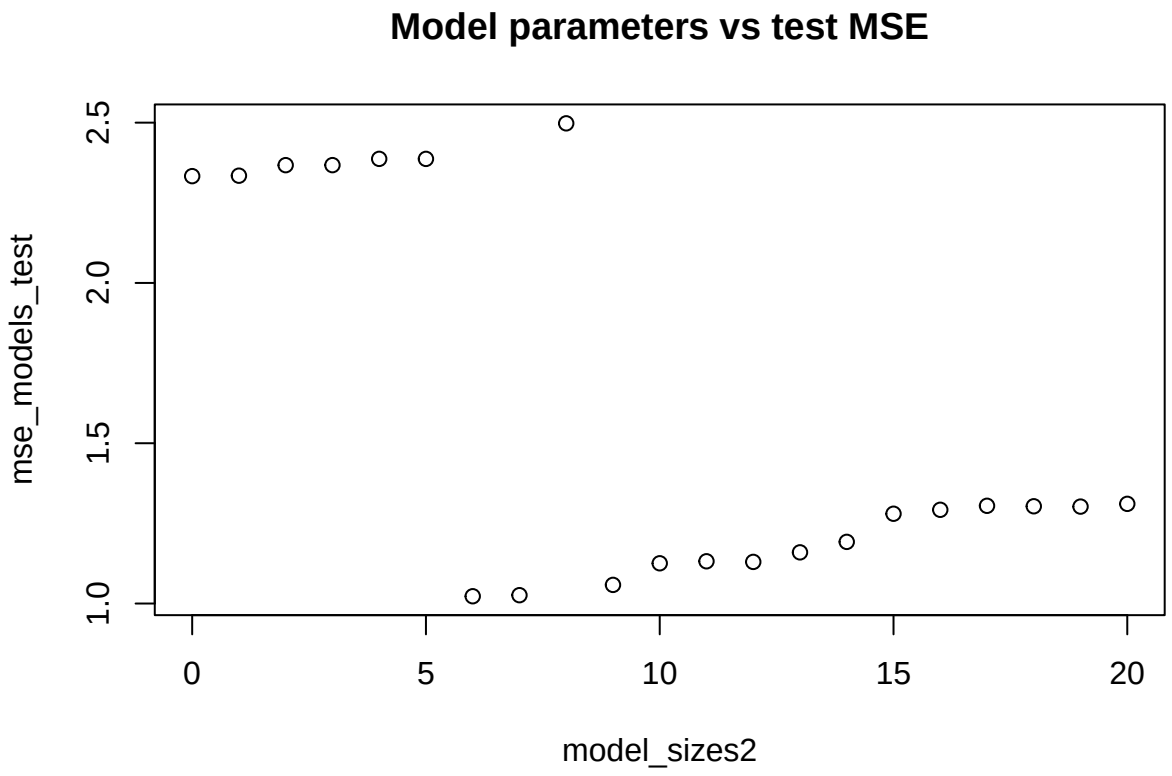
d)

I don't know of a way to extract the best models from regsubsets() and Google didn't work well for me, so I created each linear model manually. I also had to input the test data using predict, so I had to also manually do that.

```r
# creating each model from the function regsubsets()
model_sizes2 <- seq(0,20, by = 1)
size0 <- lm(y~1, data = training_setq6)
size1 <- lm(y~X3, data = training_setq6)
size2 <- lm(y~X3+X4, data = training_setq6)
size3 <- lm(y~X3+X4+X16, data = training_setq6)
size4 <- lm(y~X3+X4+X16+X15, data = training_setq6)
```

```r
size5 <- lm(y~X3+X4+X16+X15+X6, data = training_setq6)
size6 <- lm(y~X3+X4+X16+X15+X6+X5, data = training_setq6)
size7 <- lm(y~X3+X4+X16+X15+X6+X5+X14, data = training_setq6)
size8 <- lm(y~X3+X4+X16+X15+X6+X14+X1+X13, data = training_setq6)
size9 <- lm(y~X1+X3+X4+X5+X6+X13+X14+X15+X16, data = training_setq6)
size10 <- lm(y~X1+X3+X4+X5+X6+X12+X13+X14+X15+X16, data = training_setq6)
size11 <- lm(y~X1+X3+X4+X5+X6+X12+X13+X14+X15+X16+X19, data = training_setq6)
size12 <- lm(y~X1+X3+X4+X5+X6+X12+X13+X14+X15+X16+X18+X19,
             data = training_setq6)
size13 <- lm(y~X1+X3+X4+X5+X6+X9+X12+X13+X14+X15+X16+X18+X19,
             data = training_setq6)
size14 <- lm(y~X1+X3+X4+X5+X6+X9+X12+X13+X14+X15+X16+X18+X19+X20,
             data = training_setq6)
size15 <- lm(y~X1+X2+X3+X4+X5+X6+X9+X12+X13+X14+X15+X16+X18+X19+X20,
             data = training_setq6)
size16 <- lm(y~X1+X2+X3+X4+X5+X6+X9+X10+X12+X13+X14+X15+X16+X18+X19+X20,
             data = training_setq6)
size17 <- lm(y~X1+X2+X3+X4+X5+X6+X9+X10+X11+X12+X13+X14+X15+X16+X18+X19+X20,
             data = training_setq6)
size18 <- lm(y~X1+X2+X3+X4+X5+X6+X7+X9+X10+X11+X12+X13+X14+X15+
               X16+X18+X19+X20,
             data = training_setq6)
size19 <- lm(y~X1+X2+X3+X4+X5+X6+X7+X8+X9+X10+X11+X12+X13+X14+X15+
               X16+X18+X19+X20, data = training_setq6)
size20 <- lm(y~X1+X2+X3+X4+X5+X6+X7+X8+X9+X10+X11+X12+X13+X14+X15+
               X16+X17+X18+X19+X20, data = training_setq6)

# calculating MSE of each model with test data
mse_models_test <- c(
  sum((test_setq6$y - predict(size0, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size1, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size2, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size3, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size4, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size5, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size6, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size7, newdata = test_setq6))**2)
```

```
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size8, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size9, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size10, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size11, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size12, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size13, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size14, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size15, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size16, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size17, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size18, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size19, newdata = test_setq6))**2)
  / length(test_setq6$y),
  sum((test_setq6$y - predict(size20, newdata = test_setq6))**2)
  / length(test_setq6$y))

plot(model_sizes2, mse_models_test, main = "Model parameters vs test MSE")
```



e)

```
min(mse_models_test)
```

```
## [1] 1.022737
```

```
sum((test_setq6$y - predict(size6, newdata = test_setq6))**2) / length(test_setq6$y)
```

```
## [1] 1.022737
```

```
sum((test_setq6$y - predict(size7, newdata = test_setq6))**2) / length(test_setq6$y)
```

```
## [1] 1.025996
```

By the graph and a check using the MSE we calculated for each model, the model with 6 parameters has the lowest MSE. This result shows how a model that is less complex and doesn't necessarily capture all the nuances of the training data is actually more generalizable and applicable to other data sets.

f)

The true model to generate the model only had one non-zero beta, so the model is actually too complex for the true data.

```
summary(size6)
```

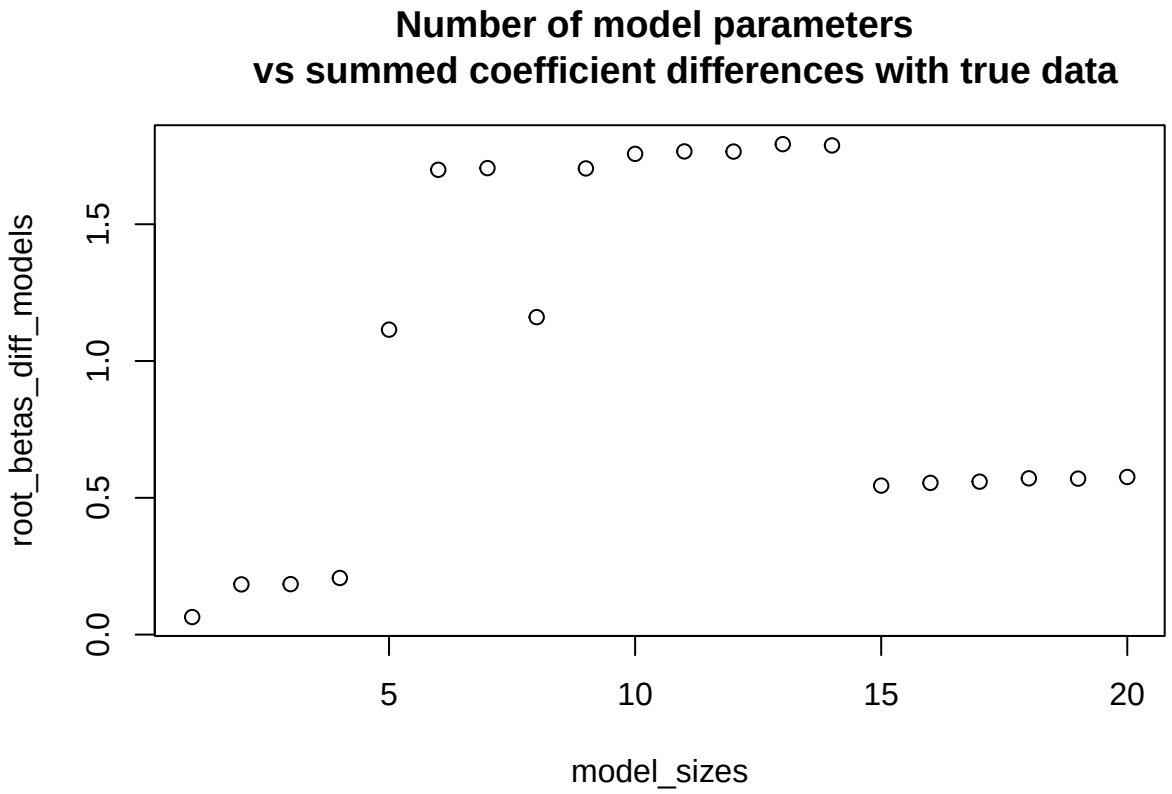
```
##
## Call:
## lm(formula = y ~ X3 + X4 + X16 + X15 + X6 + X5, data = training_setq6)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.14116 -0.65151  0.06285  0.61835  2.07031
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.03718    0.10027   0.371    0.712
## X3           0.01361    0.09799   0.139    0.890
## X4          -0.04608    0.09951  -0.463    0.644
## X16          0.01753    0.09480   0.185    0.854
## X15          0.01708    0.10157   0.168    0.867
## X6           0.01658    0.11378   0.146    0.884
## X5          -1.28127    0.10212 -12.546   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9856 on 93 degrees of freedom
## Multiple R-squared:  0.6351, Adjusted R-squared:  0.6116
## F-statistic: 26.98 on 6 and 93 DF,  p-value: < 2.2e-16
```

As we can see in the summary, most of the coefficients are small except for X5, which has great significance according to the t-test. This makes sense because X5 is the only non-zero coefficient in our true data.

g)

```
# calculating the given equation
root_betas_diff_models <- c(
  sqrt(sum((head(beta,1)-size1$coefficients[-1])**2)),
  sqrt(sum((head(beta,2)-size2$coefficients[-1])**2)),
  sqrt(sum((head(beta,3)-size3$coefficients[-1])**2)),
```

```
  sqrt(sum((head(beta,4)-size4$coefficients[-1])**2)),
  sqrt(sum((head(beta,5)-size5$coefficients[-1])**2)),
  sqrt(sum((head(beta,6)-size6$coefficients[-1])**2)),
  sqrt(sum((head(beta,7)-size7$coefficients[-1])**2)),
  sqrt(sum((head(beta,8)-size8$coefficients[-1])**2)),
  sqrt(sum((head(beta,9)-size9$coefficients[-1])**2)),
  sqrt(sum((head(beta,10)-size10$coefficients[-1])**2)),
  sqrt(sum((head(beta,11)-size11$coefficients[-1])**2)),
  sqrt(sum((head(beta,12)-size12$coefficients[-1])**2)),
  sqrt(sum((head(beta,13)-size13$coefficients[-1])**2)),
  sqrt(sum((head(beta,14)-size14$coefficients[-1])**2)),
  sqrt(sum((head(beta,15)-size15$coefficients[-1])**2)),
  sqrt(sum((head(beta,16)-size16$coefficients[-1])**2)),
  sqrt(sum((head(beta,17)-size17$coefficients[-1])**2)),
  sqrt(sum((head(beta,18)-size18$coefficients[-1])**2)),
  sqrt(sum((head(beta,19)-size19$coefficients[-1])**2)),
  sqrt(sum((beta-size20$coefficients[-1])**2)))

plot(model_sizes, root_betas_diff_models, main = "Number of model parameters
     vs summed coefficient differences with true data")
```



I don't see a connection between the plot from (d) and the plot I made here. I see that for the intermediate models, the difference between the true betas from generating the data and the estimated betas from the **regsubsets()** function are quite large. This may show that

the model with 6 parameters that we concluded was best using the test MSE may actually NOT be the best model. Obviously, the best model would be the one that only has **X5** as the predictor. This fact is partially reflected in the plot above because the model with only one predictor has the lowest difference between its coefficient value and the true beta value of 0. I doubt it is a coincidence that the best model according to our test MSE is the one where **X5** actually first appears as a predictor. The plot above does not show that the model with 6 predictors is best because of the bloat that the preceding 5 predictors created.