# STA9690 - Homework 3

*Christopher Lang*

*October 30, 2017*

## Problem 2, Chapter 5, pg 197 - 198

### Part A

In bootstrap sampling, each observation $x_i$ has equal probability of being selected due to sampling with replacement. Therefore:

$$P(Y = x_i) = \frac{1}{n}$$

Where $Y$ is a random variable representing a boostrap observation for a bootstrap sample, and $x_i$ is a realization of $X$ from the original dataset

Since sampling is done with replacement, $P(Y = x_i) = \frac{1}{n}$ at every sampling procedure

Therefore, the probability of **not** sampling the $j^{th}$ observation from the original dataset for the **first** bootstrap observation is:

$$P(Y_1 \neq x_j) = 1 - \frac{1}{n}$$

### Part B

Similarly, the probability of **not** selecting the $j^{th}$ observation from the original dataset for the **second** boostrap observation remains the same:

$$P(Y_2 \neq x_j) = 1 - \frac{1}{n}$$

If we include part A, then:

$$P(Y_2 \neq x_j \cap Y_2 \neq x_j) = (1 - \frac{1}{n})^2$$

### Part C

When stating that the $j^{th}$ observation $x_j$ from the original dataset is not in the boostrap sample, we mean that $x_j$ is never sampled in any of the bootstrap observations

Since probabilities never change due to sampling with replacement, this bootstrap procedure follows a binomial distribution:

Let $Z$ the number of times the $x_j$ observation from the original dataset is sampled in a bootstrap sample of size $n$:

- $Z =$# of times $x_j$ observation is sampled
- $n =$ the size of the bootstrap sample (# of bootstrap observations)
- $p = \frac{1}{n}$, the probability of selecting the $j^{th}$ observation from the original dataset
- $q = 1 - \frac{1}{n}$, the probability of **not** selecting the $j^{th}$ observation from the original dataset

The probability distribution of $Z$ will follow the binomial distribution:

$$P(Z = k) = \binom{n}{k} p^k q^{n-k}$$

Then the probability of never selecting the $j^{th}$ observation in a boostrap sample is:

$$P(Z = 0) = \binom{n}{0} p^0 q^{n-0}$$

$$P(Z = 0) = (1)(1)q^n$$

$$P(Z = 0) = (1 - \frac{1}{n})^n$$

**Part D**

We borrow from part C the formula for never sampling the $j^{th}$ observation in a bootstrap sample:

$$P(Z = 0) = (1 - \frac{1}{n})^n$$

To find the probability of sampling the $j^{th}$ observation **at least once**, we formulate this as:

$$P(Z > 0|N = n) = 1 - P(Z = 0|N = n)$$

Therefore, if $N = 5$:

$$P(Z > 0|N = 5) = 1 - P(Z = 0|N = 5)$$

$$P(Z > 0|N = 5) = 1 - (1 - \frac{1}{5})^5$$

$$P(Z > 0|N = 5) \approx 0.6723$$

**Part E**

Similar to part D, we let $N = 100$:

$$P(Z > 0|N = 100) = 1 - P(Z = 0|N = 100)$$

$$P(Z > 0|N = 100) = 1 - (1 - \frac{1}{100})^{100}$$

$$P(Z > 0|N = 100) \approx 0.6340$$

**Part F**

Similar to part E, we let $N = 10000$:

$$P(Z > 0|N = 10000) = 1 - P(Z = 0|N = 10000)$$

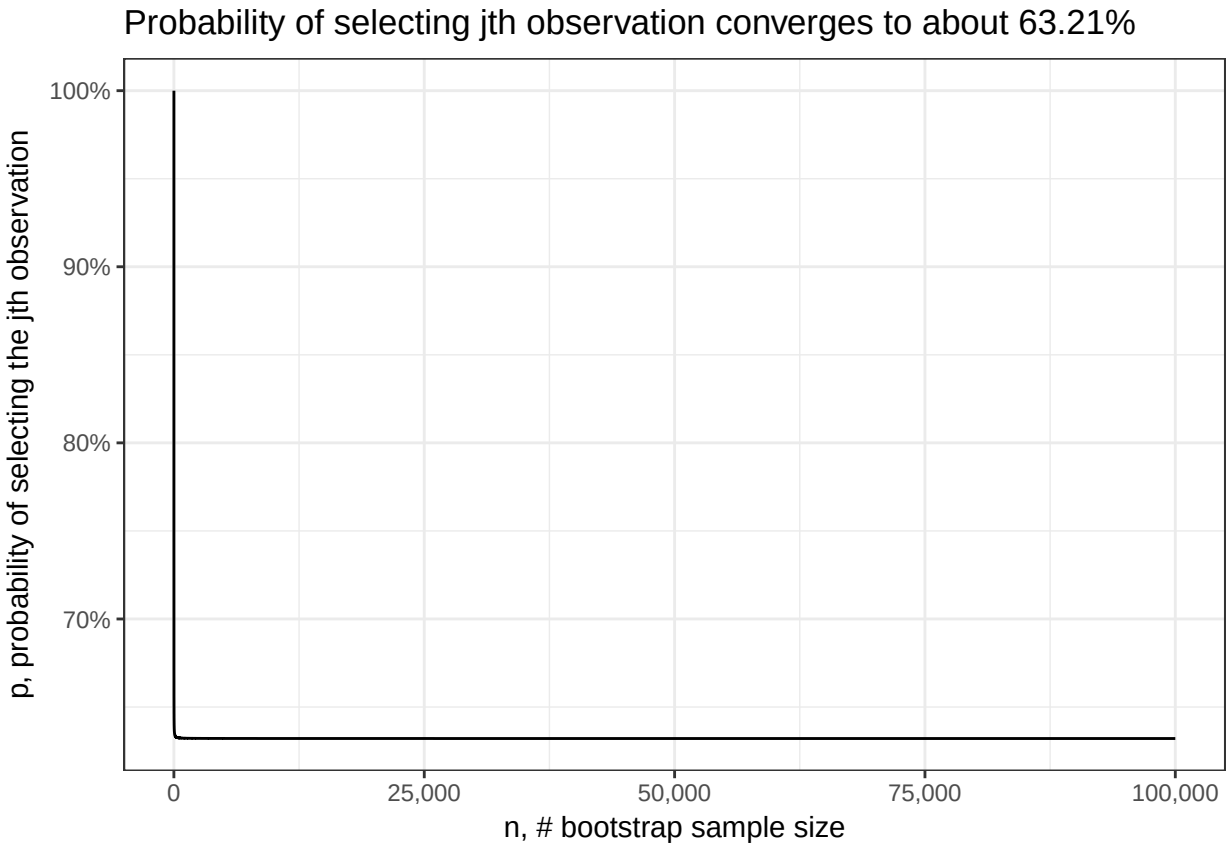
$$P(Z > 0|N = 10000) = 1 - (1 - \frac{1}{10000})^{10000}$$

$$P(Z > 0|N = 10000) \approx 0.6321$$

**Part G**

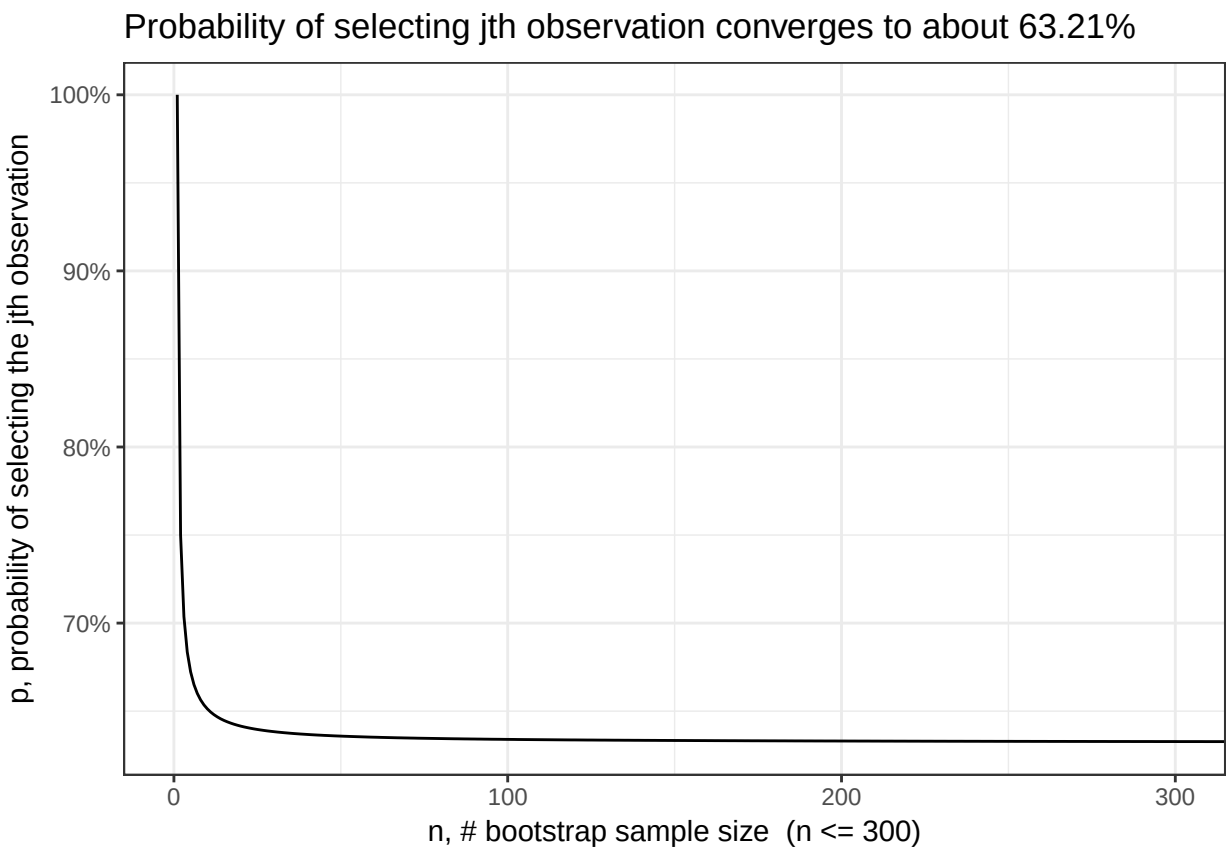
For computing the probability of selecting the $j^{th}$ observation for bootstrap sample sizes 1 through 100,000, we use the following R code:

```r
library(ggplot2)

x <- data.frame(n = seq(from = 1, to = 100000, by = 1))
x$p <- 1 - (1 - (1 / x$n))^x$n

ggplot(x) + aes(x = n, y = p) + geom_line() +
  scale_y_continuous(labels = scales::percent) +
  scale_x_continuous(labels=scales::comma) +
  theme_bw() +
  labs(x = 'n, # bootstrap sample size',
       y = 'p, probability of selecting the jth observation',
       title = "Probability of selecting jth observation converges to about 63.21%")
```




One of the most obvious behavior we observe is the exceptionally quick convergence towards some value, which is approximately 63.21%. This is however difficult to observe in the above chart due to the fact that it converges so quickly

We can observe the convergence by limiting the maximum $n$ value to 300:




**Part H**

For this part we copy in the code from the book but set the seed to 1 to get a constant mean

```
set.seed(1)

store <- rep(NA, 10000)

for (i in 1:10000) {
  store[i] <- sum(sample(1:100, rep = TRUE) == 4) > 0
}

mean(store)
```

```
## [1] 0.6408
```

In this case we get a mean value of 0.6408. This is to be expected, the mean represents the probability of sampling observation 4. The value of 0.6408 is close to the converged probability value of 63.21% from part G. We got our estimate much more quickly however due to the fact that those probabilities from part G was derived analytically, while here we estimated the probability numerically

## Problem 7, Chapter 5, pg 199 - 200

To begin this problem, we load the `Direction` dataset from the `ISLR` package:

```
library(ISLR)
data('Weekly')
```

### Part A

The follow code block fits a logistic regression model, predicting `Direction` using `Lag1` and `Lag2` from the `Weekly` dataset:

```
dataset <- Weekly
dataset <- model.frame(Weekly)
logit_model1 <- glm(Direction ~ Lag1 + Lag2, binomial(link='logit'), dataset)

summary(logit_model1)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial(link = "logit"),
##     data = dataset)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.623  -1.261   1.001   1.083   1.506
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.22122    0.06147   3.599 0.000319 ***
## Lag1        -0.03872    0.02622  -1.477 0.139672
## Lag2         0.06025    0.02655   2.270 0.023232 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1488.2  on 1086  degrees of freedom
## AIC: 1494.2
##
## Number of Fisher Scoring iterations: 4
```

### Part B

Similarly, we do the same as part A but **exclude the first observation**:

```
dataset <- Weekly[-1,]
dataset <- model.frame(Weekly)
logit_model2 <- glm(Direction ~ Lag1 + Lag2, binomial(link='logit'), dataset)

summary(logit_model2)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial(link = "logit"),
##     data = dataset)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.623  -1.261   1.001   1.083   1.506
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.22122    0.06147   3.599 0.000319 ***
## Lag1        -0.03872    0.02622  -1.477 0.139672
## Lag2         0.06025    0.02655   2.270 0.023232 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1488.2  on 1086  degrees of freedom
## AIC: 1494.2
##
## Number of Fisher Scoring iterations: 4
```

### Part C

We continue using `logit_model2`, which is the logistic regression model with the first observation removed when training. We then classify the first observation with $P(Direction = "UP"|Lag1, Lag2) > 0.50$

In our model, the logistic regression will return the probability of `Direction` being `DOWN`, since it is the first factor level. So we will substract by 1. Hence:

```
first_obs_predictors <- Weekly[1, c('Lag1', 'Lag2')]
first_obs_class <- Weekly[1, 'Direction']

prob <- predict(logit_model2, newdata = first_obs_predictors, type = 'response')
prob_up <- 1 - prob

if (prob_up > 0.50) {
  classification <- "Up"
} else {
  classification <- "Down"
}

if (classification == first_obs_class) {
  print("Correctly classified first observation")
} else {
  print("Incorrectly classified first observation")
}
```

```
## [1] "Correctly classified first observation"
```

As stated in the output, the logistic regression model we built correctly classified the first observation

### Part D

For this part, we develop a leave-one-out cross validation technique for fitting a logistic model. The idea is to leave out the $i^{th}$ observation out when training, then collecting the error on predicting the class for the $i^{th}$ observation

#### part i through part iv

```
n <- nrow(Weekly)
result <- rep(NA, n)

for (i in 1:n) {
  training_set <- Weekly[-i,]
  training_set <- model.frame(training_set)

  # The test set is the single observation that was left out
  # Using the logistic model, we will predict its class to estimate the LOOCV
  # test error
  test_set <- Weekly[i,]
  test_set <- model.frame(test_set)

  test_set_predictors <- test_set[, c('Lag1', 'Lag2')]
  test_set_class <- test_set[, 'Direction']

  model <- glm(Direction ~ Lag1 + Lag2, binomial(link='logit'), training_set)

  # Similar to part C, probability prediction is for Direction=Down, so we
  # inverse this by substracting from 1
  prob <- predict(model, newdata = test_set_predictors, type = 'response')
  prob_up <- 1 - prob

  # We keep the probability greater than 0.50 for classifying Up
  if (prob_up > 0.50) {
    classification <- "Up"
  } else {
    classification <- "Down"
  }

  if (classification == test_set_class) {
    result[i] <- 1
  } else {
    result[i] <- 0
  }
}

mean(result)
```

```
## [1] 0.4499541
```

The error rate of 45% is fairly high. It means that on a test set, the logistic model of using `Lag1` and `Lag2` as predictors of `Direction` correctly classifies the direction only 45% of the time. Changing the minimum required probability above/below 50%, or using a different set of predictors, could help improve the logistic model's performance

## Problem 8, Chapter 5, pg 200 - 201

### Part A

To begin, to use the code shown in the book:

```
set.seed(1)
y <- rnorm(100)  # This is meaningless, unless its to progress the random seed
x <- rnorm(100)
y <- x - 2 * x^2 + rnorm(100)
```

The parameter $n$ is 100, and $p$ is 2, of the equation form:

$$y = x + 2x^2 + \epsilon$$

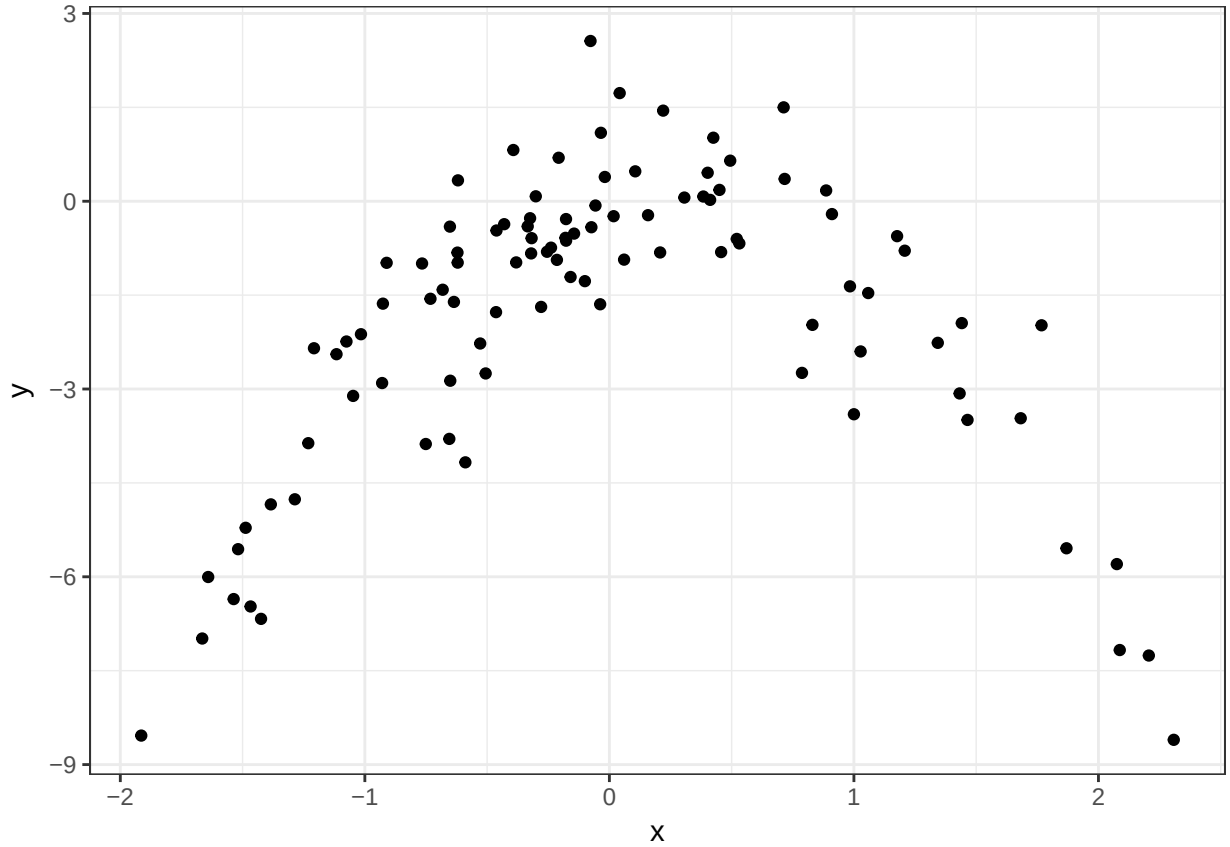
$$\epsilon \sim N(0, 1)$$

A polynomial regression model of order 2, with an error term normally distributed with mean 0, variance 1

### Part B

Below is the scatterplot of X and Y:

```
xdf <- data.frame(x = x, y = y)

ggplot(xdf) + aes(x, y) + geom_point() + theme_bw()
```




The scatterplot shows a parabolic-like pattern between the relationship of X and Y

**Part C**

For this part, we re-create the dataset found in Part B, fit four polynomial regression models, and estimate LOOCV test mean squared error

```
set.seed(2)

n <- 100  # Number of observations

poly_orders <- as.integer(1:4)
dset <- data.frame(x = rnorm(n), y = x - 2 * x^2 + rnorm(n))
result <- list()

for (k in poly_orders) {
  loocv_errors <- rep(NA, nrow(dset))

  for (i in seq_len(nrow(dset))) {
    training_set <- dset[-i,]
    training_set <- model.frame(training_set)

    test_set <- dset[i,]
    test_set <- model.frame(test_set)

    test_set_predictors <- test_set[, c('x')]
    test_set_predictors <- data.frame(x = test_set_predictors)
    test_set_y <- test_set[, 'y']

    model <- lm(y ~ poly(x, k), data = training_set)
    predicted_y <- predict(model, newdata = test_set_predictors)

    loocv_errors[i] <- test_set_y - predicted_y
  }

  r <- data.frame(k = k, mean_error = mean((loocv_errors)^2))

  result <- c(result, list(r))
}

result <- do.call(rbind, result)

print(result)
```

```
##   k mean_error
## 1 1   6.967153
## 2 2   6.838206
## 3 3   6.792999
## 4 4   6.851096
```

The four errors, printed one for each of the four polynomial models above

### Part D

We repeat the same code as Part C, but use `set.seed(126563)`:

```
##   k mean_error
## 1 1   6.624170
## 2 2   6.137644
## 3 3   6.329433
## 4 4   6.517017
```

We received different LOOCV MSE estimates. This is because when generating our dataset, we used `rnorm`, which generated psuedo-random numbers for $X$, which by definition also changed the values of $Y$. A different seed will generate a different stream of random numbers, affecting our LOOCV MSE estimates

### Part E

In part C, we found that a polynomial regression of order 2 had the smallest overall LOOCV MSE estimate. This aligns with our belief that an order 2 polynomial regression would have the lowest error due to the fact that the dataset used for training:

```r
dset <- data.frame(x = rnorm(n), y = x - 2 * x^2 + rnorm(n))
```

is of a order 2 polynomial linear regression equation form

### Part F

We first redo part C to get the actual models:

```r
set.seed(2)

n <- 100  # Number of observations

poly_orders <- as.integer(1:4)
dset <- data.frame(x = rnorm(n), y = x - 2 * x^2 + rnorm(n))
result <- list()

for (k in poly_orders) {
  model <- lm(y ~ poly(x, k), data = training_set)

  print(summary(model))
}
```

```
##
## Call:
## lm(formula = y ~ poly(x, k), data = training_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.9735 -1.0604  0.6171  1.5551  4.0005
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.0158     0.2524  -7.988 2.85e-12 ***
## poly(x, k)   -2.8232     2.5111  -1.124    0.264
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.511 on 97 degrees of freedom
## Multiple R-squared:  0.01286,    Adjusted R-squared:  0.002687
## F-statistic: 1.264 on 1 and 97 DF,  p-value: 0.2637
##
##
## Call:
## lm(formula = y ~ poly(x, k), data = training_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.4267 -1.3426  0.4905  1.5554  5.0422
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.0158     0.2426  -8.311 6.22e-13 ***
## poly(x, k)1  -2.8232     2.4134  -1.170  0.24498
## poly(x, k)2  -7.2443     2.4134  -3.002  0.00342 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.413 on 96 degrees of freedom
## Multiple R-squared:  0.09756,    Adjusted R-squared:  0.07876
## F-statistic: 5.189 on 2 and 96 DF,  p-value: 0.007245
##
##
## Call:
## lm(formula = y ~ poly(x, k), data = training_set)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.832 -1.323  0.523  1.620  4.291
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.0158     0.2398  -8.407 4.15e-13 ***
## poly(x, k)1  -2.8232     2.3857  -1.183  0.23962
## poly(x, k)2  -7.2443     2.3857  -3.036  0.00309 **
## poly(x, k)3  -4.2936     2.3857  -1.800  0.07508 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.386 on 95 degrees of freedom
## Multiple R-squared:  0.1273, Adjusted R-squared:  0.09976
## F-statistic:  4.62 on 3 and 95 DF,  p-value: 0.004627
##
##
## Call:
## lm(formula = y ~ poly(x, k), data = training_set)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.1762 -1.2798  0.5296  1.6598  4.4460
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.0158     0.2401  -8.396 4.68e-13 ***
## poly(x, k)1  -2.8232     2.3888  -1.182  0.24026
## poly(x, k)2  -7.2443     2.3888  -3.033  0.00313 **
## poly(x, k)3  -4.2936     2.3888  -1.797  0.07549 .
## poly(x, k)4   2.0742     2.3888   0.868  0.38744
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.389 on 94 degrees of freedom
## Multiple R-squared:  0.1343, Adjusted R-squared:  0.09742
## F-statistic: 3.644 on 4 and 94 DF,  p-value: 0.00834
```

- For the polynomial regression of order 1 (simple regression) the $\beta_1$ coefficient was not found to be significant
  - Not surprisingly, a simple regression will not fit the curved data we have
- For the polynomial regression of order 2, the $\beta_1$ remains not statistically significant, but for $\beta_2$ coefficient it is significant, but only at the 0.10 alpha level
  - This is inline with our cross validation result; order 2 is the best model due to the fact that the data itself has an order of 2
  - Perhaps with more data points we can further restrict the alpha level
- For the polynomial regression of order 3 and 4, $\beta_1$ remains not significant, $\beta_3$ and $\beta_4$ also not significant, but $\beta_2$ remains significant but at the 0.10 alpha level
  - Not surprisingly, order 3 and order 4 terms are not significant as our original training data does not have those terms

Overall this agrees with the cross validation assessment. It recommended the order 2 polynomial regression as it had the lowest MSE

## Problem 1, Chapter 6, pg 259

### Part A

Among the three selection methods, best subset, forward, back stepwise, the best subset selection method will select the model with $k$ predictors that has the lowest training *RSS*

This is because the best subset selection method will exhaustively train $2^p$ models, guaranteeing that the model with the lowest training RSS is present in the set, which the method will then select since it is the "best fitting" against training

In real life however, the use of selection based on a testing set, or cross validation, would greatly reduce the chance of picking a non-optimal model

In comparison, both forward and back stepwise selection may **or may not** select the lowest *RSS* model due to the way the algorithm works. Neither of them are exhaustive, and as they add/substract additional predictors, they only select the lowest *RSS* model based on the current set of predictors they have, and may very well take the wrong path and not select the lowest possible *RSS* model

### Part B

This is difficult to prove, and may not be possible to prove analytically. It would depend on the implementation of the best subset selection

If out of $2^p$ models in best subset, only $p + 1$ (as stated for this question) is selected for testing, there is no guarantee that the best subset model will select the lowest testing *RSS* model when compared to forward/backward stepwise selection. Depending on the data, the path taken by forward/background selection may end up selecting the best testing *RSS* vs. best subset

If all $2^p$ is used in best subset however, than best subset is guaranteed to select the best model, with the lowest testing *RSS*

### Part C

#### part i

***True***, the $k$-variable model identified by forward stepwise is a subset of *(k + 1)*-variable model selected by forward stepwise

#### part ii

***True***, the $k$-variable model identified by backward stepwise is a subset of *(k + 1)*-variable model selected by backward stepwise

#### part iii

***False***, there is no guarantee that $k$-variable backward stepwise model is a subset of *(k + 1)*-variable forward stepwise model. The foward and backward stepwise selection methods are unrelated in the way they select models

#### part iv

***False***, similar to part iii, there is no relation forward and backward stepwise selection and hence no guarantee that it'll be a subset

**part v**

***False***, with the assumption that best subset and best subset selection are different ways of selecting the best model. In best subset, we're assuming that the model with the lowest training *RSS* from the $2^p$ is selected, while in best subset selection, the model in the $p+1$ subset is selected. If so, then there is no guarantee that the best subset $k$-variable model has predictors that is a subset of the *(k+1)*-variable model

## Problem 2, Chapter 6, pg 259 - 260

### Part A

Answer ***number iii*** is correct

> *Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance*

The LASSO method contains a more restrictive budget constraint $s = \sum_{j=1}^{p} |\beta_j|$ than least square. This has the effect of forcibly reducing the coefficients down to zero, with the goal of reducing *RSS*. This also has the effect of increasing bias, but if this increase is small compared to the decrease in variance, than the LASSO method will produce a better model with a higher prediction accuracy

### Part B

Answer ***number iii*** is correct

> *Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance*

The reasoning for this is similar to part A. Ridge regression is similar to LASSO but with a less restrictive budget. However, ridge regression's budget constraint is still much higher than least squares, with a similar reasoning behind its improved prediction accuracy

### Part C

Answer ***number ii*** is correct

> *More flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias*

Non-linear models can be much more flexible than linear models with their capability to highly fit to the intricacies of the data. This is especially true if the data predictors does **not** have a linear relationship against the response vector

This has the affect of increasing variance, but if the increase remains smaller than the decrease in bias, a non-linear model can produce improved prediction accuracy vs. linear models

## Problem 3, Chapter 6, pg 260

### Part A

Answer ***number iv*** is correct

> *steadily decrease*

As the parameter $s$ steadily increases, we place less restriction on the budget constraint, allowing the $\beta$ coefficient's range of possible values to be greater

As seen in the previous question, if the increase in bias is less than its decrease in variance, a LASSO/ridge regression would have an improved prediction accuracy, and hence a lower training *RSS*

Therefore, if the above statement about bias and variance holds (part C and part D) holds, we should see a steady decrease in training *RSS*

### Part B

Answer ***number ii*** is correct

> *Decrease initially, and then eventually start increasing in a U shape*

On a test set, we have a greater uncertainty about the model's test bias and variance changes as $s$ increase. We have a reasonable expectation that a model with $s = 0$ will have a high test *RSS* at the very least, and increasing the budget $s$ should do no worse than a parameter-less model. Hence, we expect it to decrease initially

However, at some point, the value of $s$ begins producing an overly flexible model, which we may expect to start producing higher test *RSS* than previously, hence the U shape

### Part C

Answer ***number iii*** is correct

> *Steadily increase*

An ever increasing budget $s$ will produce a more flexible model, which will steadily increase the variance as the model begins to allow greater variability in the error of the estimates

### Part D

Answer ***number iv*** is correct

> *Steadily decrease*

An ever increasing budget $s$ will produce a more flexible model, which will steadily decrease the bias. Though a more flexible model has greater variability in the estimates, on average this variability will be closer to the actual values, decreasing the chance of a biased estimate

### Part E

Answer ***number v*** is correct

> *Remains constant*

Irreducible error is error that cannot be explained by any combination of the predictors. Hence, regardless of the constraint, this error can never be "explained out" by the predictors and will remain constant

## Problem 4, Chapter 6, pg 260 - 261

### Part A

Answer ***number iii*** is correct

*Steadily increase*

As $\lambda$ increases, the weight of the term $\sum_{j=1}^{p} \beta_j^2$ increases, reducing the influence of the $\beta_j$ coeffcient more and closer to zero. Hence, without a statistical or mathematical reason to increase $\lambda$, we will increase the training *RSS* as it deviates from the actual least square estimate

### Part B

Answer ***number ii*** is correct

*Decrease initially, and then eventually start increasing in a U shape*

When $\lambda$ is zero, the $\sum_{j=1}^{p} \beta_j^2$ term is removed from the model. As $\lambda$ increases away from zero and towards the least square estimate of $\lambda$, you will attain the lowest possible testing *RSS*. Once you continue to increase $\lambda$ away from the least square estimate however, the testing *RSS* should begin to increase again, giving the testing *RSS* curve a U shape

### Part C

Answer ***number iv*** is correct

*Steadily decrease*

As $\lambda$ increase, we expect to see a decrease in the variance. This is because with a higher value of $\lambda$, we are further restricting the models $\beta_j$ coefficient. With this greater penalty we expect to see reduced variance

### Part D

Answer ***number iii*** is correct

*Steadily increase*

As $\lambda$ increase, we expect to see an increase in the bias. Similar to part C, we are penalizing the model with a higher value of $\lambda$. We obtain reduced variance, but at a cost of higher bias

### Part E

Answer ***number v*** is correct

*Remains constant*

Irreducible error is error that cannot be explained by any combination of the predictors. Hence, regardless of the constraint or penalty we place on the model, this error can never be "explained out" by the predictors and will remain constant

## Problem 7, Chapter 6, pg 262

### Part A

Assumptions:

- The error term independently and identically distributed $\epsilon \sim N(0, \sigma^2)$
- Rewrite term $\epsilon_z$ as a vector $\epsilon_z = [\epsilon_{z1}, \epsilon_{z2}, ..., \epsilon_{zk}]$
- Rewrite term $x_z$ as a vector $x_z = [x_{z1}, x_{z2}, ..., x_{zk}]$
- Define $X$ as the matrix containing all $x_z$
- Rewrite term $\beta_z$ as a vector $\beta_z = [\beta_1, \beta_2, ..., \beta_k]$
- Rewrite regression equation as $y_z = x_z\beta + \epsilon_z$
- Define $y$ as a vector $y = [y_1, y_2, ..., y_k]$
- Hence, $y_z \sim N(x_t.\beta, \sigma^2)$

Let $Z$ be the number of observations so that $z = 1, 2, 3, ..., Z$

Then the liklihood function $L$ is:

$$L = \prod_{z=1}^{Z} N(x_t.\beta, \sigma^2) = (\frac{1}{\sqrt{2\pi\sigma^2}})^Z exp(\frac{-1}{2\sigma^2}(y_1 - x_1\beta)^2)exp((y_2 - x_2\beta)^2)...exp((y_z - x_z\beta)^2)$$

$$L = (\frac{1}{\sqrt{2\pi\sigma^2}})^Z exp(\frac{-1}{2\sigma^2}(y_1 - x_1\beta)^2)exp((y_2 - x_2\beta)^2)...exp((y_z - x_z\beta)^2)$$

$$L = (\frac{1}{\sqrt{2\pi\sigma^2}})^Z exp(\frac{-1}{2\sigma^2}(y_1 - x_1\beta)^2 \cdot (y_2 - x_2\beta)^2) \cdot ... \cdot (y_z - x_z\beta)^2)$$

$$L = (\frac{1}{\sqrt{2\pi\sigma^2}})^Z exp(\frac{-1}{2\sigma^2}(y - X\beta)'(y - X\beta))$$

### Part B

Assuming a double exponential distribution for the $\beta$ coefficients, we have the following posterior distribution:

$$P(\beta|y, X) \propto (\frac{1}{\sqrt{2\pi\sigma^2}})^Z exp(\frac{-1}{2\sigma^2}(y - X\beta)'(y - X\beta)) \cdot \frac{1}{2b}exp(-\frac{|\beta|}{b})$$

### Part C

**Part D**

Assuming independent and identically distributed normal distributions for the $\beta$ coefficients, we have the following prior:

$$P(\beta_1, \beta_2, ..., \beta_z) = P(\beta_1) \cdot P(\beta_2) \cdot ... \cdot P(\beta_z)$$
$$P(\beta_1, \beta_2, ..., \beta_z) = (\frac{1}{\sqrt{2\pi\sigma^2}} exp(\frac{-1}{2\sigma^2}(y - X\beta)'(y - X\beta)))^Z$$

Hence we replace the double exponential prior with the normal prior:

$$P(\beta|y, X) \propto (\frac{1}{\sqrt{2\pi\sigma^2}})^Z exp(\frac{-1}{2\sigma^2}(y - X\beta)'(y - X\beta)) \cdot ((\frac{1}{\sqrt{2\pi\sigma^2}} exp(\frac{-1}{2\sigma^2}(y - X\beta)'(y - X\beta)))^Z)$$

**Part E**