Final

Christina Lee

12/3/2021

1. You roll five six-sided dice. Write a script in R to calculate the probability of getting between 15 and 20 (inclusive) as the total amount of your roll (ie, the sum when you add up what is showing on all five dice). Exact solutions are preferable but approximate solutions are ok as long as they are precise.

```
# Use expand.grid function to create a data frame with all values that can be formed with the
# combinations of all the vectors/factors passed into the function.

mean(15<=(a<-rowSums(expand.grid(rep(list(1:6), 5)))) & a<=20)
```

```
## [1] 0.5570988
```

2. Create a simulated dataset of 100 observations, where x is a random normal variable with mean 0 and standard deviation 1, and y = 0.1 + 2 * X + e, where epsilon is also a random normal error with mean 0 and sd 1.

```
set.seed(1)
# simulate a data set of 100 observations
x <- rnorm(100)
y.1 <- 0.1 + 2*x + rnorm(100)
```

a. Perform a t test for whether the mean of Y equals the mean of X using R.

$$H_a : \mu_y \neq \mu_x$$
$$H_0 : \mu_y = \mu_x$$

```
t.test(y.1,x) # conduct the t test
```

```
##
##  Welch Two Sample t-test
##
## data:  y.1 and x
## t = 0.76911, df = 136.16, p-value = 0.4432
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.2688031  0.6109616
## sample estimates:
## mean of x mean of y
## 0.2799667 0.1088874
```

```
qt(0.975,136.16) # upper threshold for two-tailed test
```

```
## [1] 1.97754
```

```
qt(0.025,136.16) # lower threshold for two-tailed test
```

```
## [1] -1.97754
```

Results: t statistic is 0.77, df is 136.16, threshold value is plus or minus 1.98 with a p-value of 0.44. Since the p-value $0.44 > 0.05$ we fail to reject the null. t statistic $0.77 < 1.98$ threshold value, we again fail to reject the null. Therefore, the difference in mean between y and x is not significantly different.

b. Now perform this test by hand using just the first 5 observations. Please write out all your steps carefully.

```
y1.FirstFive <- (y.1[1:5]) # extract first 5 observations from y
x.FirstFive <- (x[1:5]) # extract first 5 observations from x

y1.FirstFive # extracted 5 observations from y1
```

```
## [1] -1.7732743  0.5094025 -2.4821789  3.4485904  0.1044309
```

```
x.FirstFive # extracted 5 observations from x
```

```
## [1] -0.6264538  0.1836433 -0.8356286  1.5952808  0.3295078
```

```
# calculate basic stats for y1
y1.stat <- c(-1.7732743, 0.5094025, -2.4821789, 3.4485904, 0.1044309)

n_y1 <- length(y1.stat)
sample_mean_y1 <- mean(y1.stat)
Sd_y1 <- sd(y1.stat)
se_y1 <- Sd_y1 / sqrt(n_y1)


# print summary data for y1
n_y1
```

```
## [1] 5
```

```
sample_mean_y1
```

```
## [1] -0.03860588
```

```
Sd_y1
```

```
## [1] 2.316324
```

```
se_y1
```

```
## [1] 1.035892
```

```
# calculate basic stats for x
x.stat <- c(-0.6264538, 0.1836433, -0.8356286, 1.5952808, 0.3295078)

n_x <- length(x.stat)
sample_mean_x <-mean(x.stat)
Sd_x <- sd(x.stat)
se_x <- Sd_x / sqrt(n_x)


# print summary data for x
n_x
```

```
## [1] 5
```

```
sample_mean_x
```

```
## [1] 0.1292699
```

```
Sd_x
```

```
## [1] 0.9610394
```

```
se_x
```

```
## [1] 0.4297899
```

This is our equation for se_diff:

$$se_{diff} = \sqrt{se_1^2 + se_2^2}$$

Let's first calculate the se for y and x:

$$x = se_1 = \frac{sd}{\sqrt{n}} = \frac{0.96}{\sqrt{5}} = 0.43$$

$$y = se_2 = \frac{sd}{\sqrt{n}} = \frac{2.32}{\sqrt{5}} = 1.04$$

Therefore, our se_diff is equal to 1.13

$$se_{diff} = \sqrt{0.43^2 + 1.04^2} = 1.13$$

We can now calculate our t statistic.

$$Tstatistic = \frac{\bar{x}_x - \bar{x}_y}{se_{diff}}$$

$$Tstatistic = \frac{0.13 - (-0.04)}{1.13} = 0.15$$

Lastly, we need to calculate the degrees of freedom. Since our sd are not equal across the two groups, we calculate the df with this formula:

$$df = \frac{se_{diff}^4}{se_a^4/(n_a - 1) + se_b^4/(n_b - 1)}$$

$$df = \frac{1.13^4}{0.43^4/(5-1) + 1.04^4/(5-1)} = 5.4\overline{3}$$

```
qt(0.975,5.43) # upper threshold for two-tailed test.
```

```
## [1] 2.510545
```

```
qt(0.025,5.43) # lower threshold for two-tailed test.
```

```
## [1] -2.510545
```

```
2 * pt(0.15,5.43,lower.tail = FALSE) # p-value of two tailed test. alpha= 0.05
```

```
## [1] 0.886179
```

Results:
1. T statistics = 0.15. Threshold = 2.51. The t statistic is not above the critical threshold, so we fail to reject the null and conclude that the means are not significantly different. 2. The p-value is 0.89 > 0.05, so we again fail reject the null.

c. Assuming the mean and sd of the sample that you calculated from the first five observations would not change, what is the minimum total number of additional observations you would need to be able to conclude that the true mean of the population is different from 0 at the p = 0.01 confidence level?

```
alpha <- 0.01
mu <- 0

for (i in 5:30000) {
# Recalculate the standard error and CI
    stand_err <- Sd_y1 / sqrt(i)
    ci <- sample_mean_y1 + c(qt(alpha/2, i-1), qt(1-alpha/2, i-1))*stand_err
    if (ci[2] < mu)
        break     # condition met, exit loop
}
i
```

```
## [1] 23889
```

Here, we have calculated n = 23889. Let's now test to see if our t-statistics is now greater or less than the threshold value +/-2.51 if we increase our n to 23889:

$$Test\ statistic = \frac{\bar{x} - \mu_0}{\frac{sd}{\sqrt{n}}}$$

$$Test\ statistic = \frac{-0.04 - 0}{\frac{2.3}{\sqrt{23889}}}$$

$$Test\ statistic = -2.7$$

```
2*(1-pt(2.7,23888)) # p-value with new n and test statistic
```

```
## [1] 0.006938831
```

Our test statistic is now -2.7 and it is less and greater than the threshold value +/-2.51, which means we can reject the null hypothesis. In addition, the p-value is now $0.007 < 0.01$, which we can again reject the null. This indicates that we need minimum of 23889 total observations. Hence, we would need 23889 - 5 = 23884 additional observations to reject the null.

3. Generate a new 100-observation dataset as before, except now y = 0.1 + 0.2 * x + e

```
set.seed(1)
# simulate a data set of 100 new observations
x <- rnorm(100)
y.2 <- 0.1 + 0.2*x + rnorm(100)
```

a. Regress y on x using R, and report the results. Discuss the coefficient on x and its standard error, and present its 95% CI.

```
biv_model <- lm(y.2~x)
summary(biv_model)
```

```
##
## Call:
## lm(formula = y.2 ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8768 -0.6138 -0.1395  0.5394  2.3462
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.06231    0.09699   0.642   0.5221
## x            0.19894    0.10773   1.847   0.0678 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9628 on 98 degrees of freedom
## Multiple R-squared:  0.03363,    Adjusted R-squared:  0.02377
## F-statistic:  3.41 on 1 and 98 DF,  p-value: 0.06781
```

```
qt(0.975,98) # upper threshold for two-tailed test.
```

```
## [1] 1.984467
```

```
qt(0.025,98) # lower threshold for two-tailed test.
```

```
## [1] -1.984467
```

Result:
The coefficient for x is 0.19894, and its standard error is 0.10773, which the x coefficient is quite close to the truth, the error on x is relatively small too. The coefficient, however, is not significant $0.0678 > 0.05$. The 95% CI is 0.19894 +/- 1.98 x 0.10773 = [-0.014, 0.412].

b. Use R to calculate the p-value on the coefficient on x from the t statistic for that coefficient as shown in the regression in 3a, and confirm that your p-value matches what is shown in 3a. What does this p-value represent (be very precise in your language here)?

```
2*pt(1.847,98,lower.tail = FALSE) # p-value on the coefficient on x
```

```
## [1] 0.06776439
```

Result:
The calculated p-value matches with what is shown in 3a where p-value $= 0.0678 > 0.05$. This suggests that the coefficient is not significant –the effect of x on y is not significant.

c. Use R to calculate the p-value associated with the F statistic reported in your regression output. What does this test and its p-value indicate?

```
f <- 3.41
n <- length(x)
k <- 1

pf (f,k,(n-k-1), lower.tail = F) # p-value associated with F statistic.
```

```
## [1] 0.06782021
```

The p-value for the F statistic is $0.0678 > 0.05$ which means the overall significance of the model is insignificant, which is not surprising since the coefficient on x itself is also not significant.

d. Using just the first five observations from your simulated dataset, calculate by hand the coefficient on x,its standard error, and the adjusted R^2. Be sure to show your work, but you may use R for the simple math.

```
x.FirstFive <- (x[1:5]) # first five observation of x
x.FirstFive
```

```
## [1] -0.6264538  0.1836433 -0.8356286  1.5952808  0.3295078
```

```r
y2.FirstFive <- (y.2[1:5]) # first five observation of y2
y2.FirstFive
```

```
## [1] -0.6456574  0.1788445 -0.9780474  0.5770849 -0.4886831
```

```r
biv_model.2 <- lm(y2.FirstFive~x.FirstFive)
summary(biv_model.2) # regress x first five on y2 first five to confirm calculations
```

```
##
## Call:
## lm(formula = y2.FirstFive ~ x.FirstFive)
##
## Residuals:
##        1        2        3        4        5
##  0.07353  0.41791 -0.13489 -0.02048 -0.33607
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.3479     0.1458  -2.387   0.0970 .
## x.FirstFive   0.5927     0.1677   3.535   0.0385 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3223 on 3 degrees of freedom
## Multiple R-squared:  0.8064, Adjusted R-squared:  0.7418
## F-statistic: 12.49 on 1 and 3 DF,  p-value: 0.03851
```

```r
cov(x.FirstFive, y2.FirstFive) # covariance x and y first five
```

```
## [1] 0.5473849
```

```r
var(x.FirstFive) # variance of x first five
```

```
## [1] 0.9235968
```

First, calculate coefficient of B1 (x):

cov(x,y2) = 0.55
var(x) = 0.92

$$\beta_1 = \frac{cov(x, y)}{var(x)}$$

$$\beta_1 = \frac{0.55}{0.92}$$

$$\beta_1 = 0.60$$

Once we solved for B1, we can now solve for B0:

$$\beta_0 = \hat{y} - \beta_1 \hat{x}$$

$$\beta_0 = -0.27 - 0.60 * 0.13$$

$$\beta_0 = -0.35$$

Therefore, the best-fit line equation is then:

$$y = -0.35 + 0.60x$$

Calculate the predicted yi for each xi.

$$-0.728 = -0.35 + 0.60(-0.63)$$

$$-0.242 = -0.35 + 0.60(0.18)$$

$$-0.854 = -0.35 + 0.60(-0.84)$$

$$0.61 = -0.35 + 0.60(1.60)$$

$$-0.152 = -0.35 + 0.60(0.33)$$

Adjusted R and R2 by hand:

```
# Calculate adjusted R

tss <- 1.6
sse <- 0.32
n <-length(x.FirstFive)
k <- 1
dft <- n - 1
dfe <- n - k - 1
adj_r <- (tss/dft - sse/dfe) / (tss/dft)
adj_r  # about the same adjusted r in summary data above
```

```
## [1] 0.7333333
```

```
# Calculate r2
r2 <- (tss - sse)/ tss
r2 # about the same r2 in summary data above
```

```
## [1] 0.8
```

Standard error by hand:

Let's first calculate the se for y hat.

$$se_{\hat{y}} = \sqrt{\frac{\sum (y_i - \hat{y})^2}{n-2}}$$

$$se_{\hat{y}} = \sqrt{\frac{SSE}{n-2}}$$

$$se_{\hat{y}} = \sqrt{\frac{0.32}{5-2}}$$

$$se_{\hat{y}} = 0.33$$

Once we've calculated the se for y hat, solve for se of B1:

$$se_{\beta_1} = se_{\hat{y}} \sqrt{\frac{1}{\sum (x_i - \bar{x})^2}}$$

Solve for se of B1:

$$se_{\beta_1} = 0.33\frac{1}{\sqrt{3.72}}$$
$$se_{\beta_1} = 0.17$$

4. Now generate y = 0.1 + 0.2 * x - 0.5 * x^2 + e with 100 observations.

```
set.seed(1)
# simulate a data set of 100 new observations
x <- rnorm(100)
y.3 <- 0.1 + 0.2*x - 0.5*x^2 + rnorm(100)
```

a. Regress y on x and x2 and report the results. If x or x2 are not statistically significant, suggest why.

```
biv2_model <- lm(y.3 ~ x + I(x^2))
summary(biv2_model)
```

```
##
## Call:
## lm(formula = y.3 ~ x + I(x^2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9650 -0.6254 -0.1288  0.5803  2.2700
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.15672    0.11766   1.332   0.1860
## x            0.21716    0.10798   2.011   0.0471 *
## I(x^2)      -0.61892    0.08477  -7.302 7.93e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.958 on 97 degrees of freedom
## Multiple R-squared:  0.3602, Adjusted R-squared:  0.347
## F-statistic: 27.31 on 2 and 97 DF,  p-value: 3.912e-10
```

Since the coefficient on the x^2 term is negative, it means the curve is concave down (it rises and then falls), and there is a significant curved effect since the p-value is less than 0.05.

b. Based on the known coefficients that we used to create y, what is the exact effect on y of increasing x by 1 unit from 1 to 2?

$$y_2 = 0.1 + 0.2 * 2 + 0.5 * (2)^2 = -1.5$$
$$y_1 = 0.1 + 0.2 * 1 + 0.5 * (1)^2 = -0.2$$

$$y_2 - y_1 = -1.3$$

Increase x by 1 unit from 1 to 2 leads to a decrease of -1.3 in y.

c. Based on the coefficients estimated from 4(a), what is the effect on y of changing x from -0.5 to -0.7?

$$y_1 = 0.16 + 0.22(-0.5) - 0.62(-0.5)^2$$
$$y_2 = 0.16 + 0.22(-0.7) - 0.62(-0.7)^2$$

$$y_2 - y_1 = -0.30 - 0.105 = -0.20$$

Based on the coefficients estimated from 4(a), changing x from -0.5 to -0.7 will lead to a decrease of -0.20 in y.

5. Now generate x2 as a random normal variable with a mean of -1 and a sd of 1. Create a new dataset where y = 0.1 + 0.2* x - 0.5 * x * x2 + e.

```
set.seed(1)
x <- rnorm(100)
x2 <- rnorm(100,-1,1)

y.4 <- 0.1 + 0.2*x - 0.5*x*x2 + rnorm(100)
```

a. Based on the known coefficients, what is the exact effect of increasing x2 from 0 to 1 with x held at its mean?

```
mean(x)
```

```
## [1] 0.1088874
```

$$y_1 = 0.1 + 0.2(0.11) - 0.5(0.11) * 0$$
$$y_2 = 0.1 + 0.2(0.11) - 0.5(0.11) * 1$$
$$y_2 - y_1 = 0.067 - 0.12 = -0.055$$

Increasing x2 from 0 to 1 with x held at its mean will lead to a decrease of -0.055 in y.

b. Regress y on x, x2, and their interaction. Based on the regression-estimated coefficients, what is the effect on y of shifting x from -0.5 to -0.7 with x2 held at 1?

```
biv5_model <- lm(y.4 ~ x + x2 + x * x2)
summary(biv5_model)
```

```
##
## Call:
## lm(formula = y.4 ~ x + x2 + x * x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.92554 -0.43139  0.00249  0.65651  2.60188
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.10285    0.15470   0.665    0.508
## x           -0.07321    0.21598  -0.339    0.735
## x2          -0.02822    0.10970  -0.257    0.798
## x:x2        -0.73968    0.14847  -4.982 2.78e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.035 on 96 degrees of freedom
## Multiple R-squared:  0.4476, Adjusted R-squared:  0.4304
## F-statistic: 25.93 on 3 and 96 DF,  p-value: 2.262e-12
```

$$y_1 = 0.10 - 0.07(-0.5) - 0.74(-0.5) * 1$$
$$y_2 = 0.10 - 0.07(-0.7) - 0.74(-0.7) * 1$$
$$y_2 - y_1 = 0.667 - 0.505 = 0.162$$

By shifting x from -0.5 to -0.7 while holding x2 at 1 will lead to an increase of 0.162 in y.

c. Regress y on x alone. Using the R2 from this regression and the R2 from 5(b), perform by hand an F test of the complete model (5b) against the reduced, bivariate model. What does this test tell you?

```
biv6_model <- lm(y.4 ~ x)
summary(biv6_model)
```

```
##
## Call:
## lm(formula = y.4 ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9227 -0.7076  0.0501  0.6996  3.3161
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.1174     0.1162   1.011    0.315
## x             0.8352     0.1291   6.470 3.87e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.154 on 98 degrees of freedom
## Multiple R-squared:  0.2993, Adjusted R-squared:  0.2922
## F-statistic: 41.86 on 1 and 98 DF,  p-value: 3.873e-09
```

$$F = \frac{(R_c^2 - R_r^2)/df_1}{(1 - R_c^2)/df2}$$

$$F = \frac{(0.4476 - 0.2993)/2}{1 - 0.4476/100 - 3 - 1}$$

$$F = \frac{0.07414}{0.005754167} = 12.89$$

```
f.5 <- 12.89
n.5 <- 100
k.5 <- 3

pf (f.5,k.5-1,(n.5-k.5-1), lower.tail = F) # p-value associated with F statistic.
```

```
## [1] 1.099949e-05
```

The p-value is less than 0.05, which suggests that the complete model is significantly better than the reduced bivariate model in 5(b).

6. Generate a dataset with 300 observations and three variables: f, x1, and x2. f should be a factor with three levels, where level 1 corresponds to observations 1-100, level 2 to 101-200, and level 3 to 201-300. (Eg, f can be "a" for the first 100 observations, "b" for the second 100, and "c" for the third 100.) Create x1 such that the first 100 observations have a mean of 1 and sd of 2; the second 100 have a mean of 0 and sd of 1; and the third 100 have a mean of 1 and sd of 0.5. Create x2 such that the first 100 observations have a mean of 1 and sd of 2; the second 100 have a mean of 1 and sd of 1; and the third 100 have a mean of 0 and sd of 0.5. (Hint: It is probably easiest to create three 100-observation datasets first, and then stack them with rbind(). And make sure to convert f to a factor before proceeding.)

```
set.seed(1)

# f variable as factor with three levels.
f.var <- as.factor(c(rep("a", 100), rep("b", 100), rep("c", 100)))


X1<- c(rnorm(100,mean=1,sd=2),rnorm(100, mean=0, sd=1),rnorm(100, mean=1, sd=0.5))


X2 <- c(rnorm(100, mean=1, sd=2), rnorm(100, mean=1, sd=1), rnorm(100, mean=0, sd=0.5))
```

a. Using the k-means algorithm, peform a cluster analysis of these data using a k of 3 (use only x1 and x2 in your calculations; use f only to verify your results). Comparing your clusters with f, how many datapoints are correctly classified into the correct cluster? How similar are the centroids from your analysis to the true centers?

```
set.seed(1)

data <- data.frame(cbind(X1, X2, f.var))
kout <- kmeans(data[,1:2], centers=3, nstart=25)
kout$centers
```

```
##           X1          X2
## 1 -0.6477379  0.39957780
## 2  1.0283202  2.77781098
## 3  1.4261256 -0.07850055
```

```
data$cluster <- as.vector(kout$cluster)
data$f.var <- f.var
table(data[c("f.var","cluster")])
```

```
##      cluster
## f.var  1  2  3
##     a 18 41 41
##     b 60 24 16
##     c 10  0 90
```

```
true_centroids <- aggregate(data[,1:2], by=list(cat=data$f.var), FUN=mean)
true_centroids  # true centers from f
```

```
##   cat          X1          X2
## 1   a  1.21777473  1.10320372
## 2   b -0.03780808  0.96086576
## 3   c  1.01483677 -0.02225968
```

Total number of correctly classified data points are: 60 + 41 + 90 = 191
Total number of incorrectly classified data points are: 18 + 10 + 24 + 41 + 16 = 109
Accuracy = 191/(191+109)= 0.64. This means our model has achieved 64% accuracy.

The centroids of the first two clusters are very different from the true centers from f. The only similar cluster when compared to f is the third cluster where for f: x1= 1.015 and x2=-0.022) and for our analysis x1 = 1.426 and x2 = -0.079.

7. Generate a dataset of 200 observations, this time with 90 independent variables, each of mean 0 and sd 1. Create y such that:

$$y = 2x_1 + ... + 2x_{30} - x_{31} - ... - x_{60} + 0 * x_{61} + ... + 0 * x_{90} + \epsilon$$

where e is a random normal variable with mean 0 and sd 10. (Ie, the first 30 x's have a coefficient of 2; the next 30 have a coefficient of -1; and the last 30 have a coefficient of 0.)

```
set.seed(1)

# setting total num of observations and independent variables
X <- matrix(rnorm(200*90, 0, 1), nrow=200)

beta <- matrix(c(rep(2,30), rep(-1,30), rep(0,30)), ncol=1)   # coefficients

e <- matrix(rnorm(200,0,10), ncol=1)   # epsilon with mean=0 sd=10

y <- X%*%beta + e

df <- data.frame(cbind(X,y))   # create the data frame
names(df)[91] <- 'y'  # name the 91th variable in df to y
```

a. Perform an elastic net regression of y on all the x variables using just the first 100 observations. Use 10-fold cross-validation to find the best value of lambda and approximately the best value of alpha.

```
library(glmnet)
```

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

```
## Loaded glmnet 4.1-3
```

```
set.seed(1)
df_1st <- df[1:100,]  # extract first 100 observations.
x_train <- as.matrix(df_1st[,1:90])
y_train <- df_1st$y

Lambdalevels <- 10^seq(7,-6, length =100) # set lambda levels for glmnet to try.


# Estimate the elastic net model with alpha = 0, 0.5 and 1 using in-sample data.
set.seed(1)
Ridge <- cv.glmnet(x_train,y_train, alpha=0, lambda =Lambdalevels, nfolds=10)
Lasso <- cv.glmnet(x_train,y_train, alpha=1, lambda =Lambdalevels, nfolds = 10)
Elnet <- cv.glmnet(x_train,y_train, alpha=0.5, lambda =Lambdalevels, nfolds = 10)
```

```
Ridge # find best (minimum) lambda value for ridge.
```

```
##
## Call:  cv.glmnet(x = x_train, y = y_train, lambda = Lambdalevels, nfolds = 10,      alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min  3.678    50   174.7 30.63      90
## 1se 22.570    44   199.0 31.14      90
```

```
Lasso   # find best (minimum) lambda value for lasso.
```

```
##
## Call:  cv.glmnet(x = x_train, y = y_train, lambda = Lambdalevels, nfolds = 10,      alpha = 1)
```

```
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min 0.0722    63   196.4 25.44      74
## 1se 0.4431    57   216.5 18.23      58
```

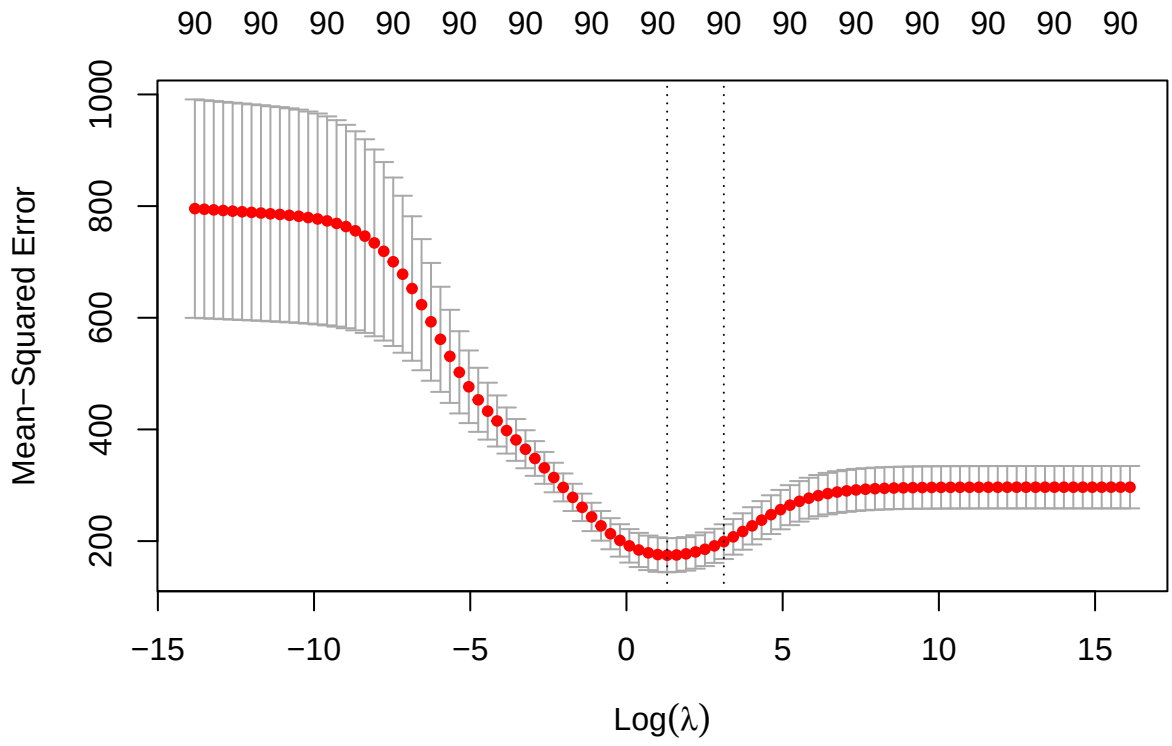
```
Elnet   # find best (minimum) lambda value for elnet.
```

```
##
## Call:  cv.glmnet(x = x_train, y = y_train, lambda = Lambdalevels, nfolds = 10,      alpha = 0.5)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min 0.5995    56   213.9 44.85      67
## 1se 2.7186    51   249.7 38.00      38
```

Results:
The best lambda value for full ridge (alpha=0) is 3.68 which gives an MSE of 174.7, full lasso (alpha=1) is 0.07 which gives an MSE of 196.4 and the elastic net (mix of both lasso/ridge) (alpha=0.5) is 0.60 which gives an MSE of 213.9. Hence, ridge with alpha of 0 has the best lambda value and associated MSE (lowest among all).

b. How accurate are your coefficients from (a)? Summarize your results any way you like, but please don't give us the raw coefficients from 90 variables.

```
plot(Ridge) # plot Ridge to see number of coefficients for lambda with best MSE
```




```
Bestlambda <- Ridge$lambda.min
#predict(Ridge, type="coefficients", s=Bestlambda)
```

Results:

Based on the plot of ridge, we can see that the best fit is with all 90 coefficients. To confirm the accuracy of coefficients from (a), I used the predict() to get the coefficients with best lambda and no coefficients have been shrunk to 0 which leaves us with 90 coefficients. Thus, I would say it is accurate.

c. Using the results from (b), predict y for the second 100 observations. How accurate is the MSE of your prediction?

```
set.seed(1)
df_2nd <- df[101:200,]  # extract second 100 observations.
x_test <- as.matrix(df_2nd[,1:90])
y_test <- df_2nd$y
```

```
yhat.Ridge <- predict(Ridge$glmnet.fit, s=Ridge$lambda.min, newx = x_test)

# mean square error for Lasso
mse.Ridge <- sum((y_test - yhat.Ridge)^2)/nrow(x_test)
mse.Ridge
```

```
## [1] 205.6364
```

The MSE for Ridge when using the training (in-sample) set is 174.7 and the MSE when using the testing (out-of sample) set is 205.6, which is higher. Therefore, I would say the MSE of my prediction is not too accurate since its MSE increased from 174.7 to 205.6

d. Compare the MSE of (c) to the out-of-sample MSE using ordinary multiple regression. Explain your results, including if the regular regression failed for any reason.

```
lmout <- lm(y_train ~ x_train)
yhat.reg <- cbind(1,x_test) %*% lmout$coefficients

# mean square error for multiple regression fit
mse.reg <- sum((y_test - yhat.reg)^2)/nrow(x_test)
mse.reg
```

```
## [1] 778.3815
```

By comparing the out-of-sample MSE, we can conclude that the MSE for Ridge (205.6) is better than multiple regression's MSE (778). My guess as to why the multiple regression has failed is likely due to overfitting the out-sample set with all 90 coefficients (all coefficients weighs the same amount of effect to the model), where although ridge retained all 90 variables, the degree of effects from each coefficients to the model depends on their significance.

8. Use the data from 7 to generate a new y2 that is 1 if y $>$ 0 and 0 otherwise.

```
library(e1071)
set.seed(1)


y2 <- 1*(y>0)
```

a. Using the same process as in 8, estimate an SVM model of y2 on all the x variables for the first 100 variables. Use 10-fold cross-validation to select the best kernel.

```
set.seed(1)

costvalues <- 10^seq(-3,2,1)

y2_train <- as.factor(y2[1:100])
```

```r
Dat_train <- data.frame(x=x_train, y=y2_train) # df with first 100 observ.
```

```r
# Estimate linear kernel
set.seed(1)
tuned_svm1 <- tune(svm,y~., data=Dat_train, ranges=list(cost=costvalues), kernel= "linear")

summary(tuned_svm1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.36
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03  0.42 0.09189366
## 2 1e-02  0.44 0.13498971
## 3 1e-01  0.38 0.14757296
## 4 1e+00  0.36 0.16465452
## 5 1e+01  0.36 0.16465452
## 6 1e+02  0.36 0.16465452
```

```r
# Estimate radial kernel
set.seed(1)
tuned_svm2 <- tune(svm,y~., data=Dat_train, ranges=list(cost=costvalues), kernel= "radial")

summary(tuned_svm2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.38
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-03  0.42 0.09189366
## 2 1e-02  0.42 0.09189366
## 3 1e-01  0.42 0.09189366
## 4 1e+00  0.41 0.07378648
## 5 1e+01  0.38 0.14757296
## 6 1e+02  0.38 0.14757296
```

According to the results, linear kernel outperformed radial kernel and has the highest in-sample accuracy

with best performance being about 36% wrong, or 64% correct (best cost setting at 1), while radial kernel's best performance being about 38% wrong, or 62% correct (best cost setting at 10).

b. Using the fitted model from (a), predict y2 for the second 100 observations, and report the accuracy of your predicted y2.

```
set.seed(1)
y2_test <- as.factor(y2[101:200])

Dat_test <- data.frame(x=x_test, y= y2_test) # df with second 100 observ.
```

Since linear kernel outperformed radial kernel in (8a), I will be using linear kernel here.

```
set.seed(1)
tuned_svm3 <- tune(svm,y~., data=Dat_train, ranges=list(cost=costvalues), kernel= "linear")


yhat.linear <- predict(tuned_svm3$best.model, newdata=Dat_test) # predict y with test data
sum(yhat.linear==Dat_test$y) / length(Dat_test$y) # percentage correct
```

```
## [1] 0.66
```

```
table(predicted=yhat.linear, truth=Dat_test$y) # confusion matrix
```

```
##          truth
## predicted  0  1
##         0 34 13
##         1 21 32
```

Based on the results, the out-of sample accuracy and in-sample accuracy for linear kernel are not much different, with the out-sample (testing set) being 66% correct and in-sample (training set) accuracy being 64% correct.