final.Zunqiu.Wang

Zunqiu Wang

12/5/2021

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

```
## Loading required package: ggplot2
```

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

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(glmnet)
```

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

```
## Loaded glmnet 4.1-2
```

```
library(e1071)
```

Q1

```
# assume i rolled 10^5 times
t <- 10^5
# create sum of 5 sided dice funtion
count <- 0
dice <- function(num, side, times) {
  results <- replicate(times, sum(sample(1:side, num, replace = TRUE)))
  for (i in 1:t) {
  if (results[i] >= 15 & results[i] <= 20) {
    count = count + 1
  }
  }
  return(count)
}

dice(5,6,t)
```

```
## [1] 55551
```

```
dice(5,6,t)/10^5
```

```
## [1] 0.5595
```

Q2

```
x <- rnorm(100)
ep <- rnorm(100)
y <- 0.1 + 2 * x + ep
```

a

```
t.test(y, x)
```

```
##
##  Welch Two Sample t-test
##
## data:  y and x
## t = 0.28704, df = 135.71, p-value = 0.7745
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.4433402  0.5938877
## sample estimates:
##   mean of x   mean of y
##  0.03956547 -0.03570831
```

from t test, mean of y and mean of x dont differ significantly at 95% level

b
- calculate mean of y $\bar{y}$ and x $\bar{x}$

```
y.bar <- mean(y[1:5])
y.bar
```

```
## [1] -0.1447467
```

```
x.bar <- mean(x[1:5])
x.bar
```

```
## [1] -0.01137821
```

- calculate $se_{diff} = \sqrt{\frac{s_y^2}{n_y} + \frac{s_x^2}{n_x}}$
  s_y is standard deviation of y and s_x is standard deviation of x
  n_y is number of observations from y and n_x is number of observations from x

```
se.diff <- sqrt(sd(y[1:5])^2/5 + sd(x[1:5])^2/5)
se.diff
```

```
## [1] 1.281371
```

- calculate t value $= \frac{\bar{x}-\bar{y}}{se_{diff}}$

```
t.val <- (x.bar - y.bar)/se.diff
t.val
```

```
## [1] 0.1040826
```

- calculate

$$df = \frac{(s_y^2/n_y + s_x^2/n_x)^2}{\frac{(s_y^2/n_y)^2}{n_y-1} + \frac{(s_x^2/n_x)^2}{n_x-1}}$$

```
sy <- sd(y[1:5])
sx <- sd(x[1:5])
```

```r
df <- (sy^2/5 + sx^2/5)^2 / ((sy^2/5)^2/(5-1) + (sx^2/5)^2/(5-1))
df
```

```
## [1] 5.500271
```

```r
qt(0.975,df)
```

```
## [1] 2.501826
```

Since t value=0.1040826 and t crit=2.501826, t value < t crit. Therefore, we fail to reject the null hypothesis mean of y is equal to mean of x for the first 5 observations.

c

```r
y.bar
```

```
## [1] -0.1447467
```

```r
sy
```

```
## [1] 2.621451
```

```r
for (i in 6:100000) {
# Recalculate the standard error and CI
stand_err <- sy / sqrt(i)
ci <- y.bar + c(qt(0.01/2, i-1), qt(1-0.01/2, i-1))*stand_err
if (ci[2] < 0)
break # condition met, leave the for loop
}

i # minimum observations needed
```

```
## [1] 2181
```

```r
ci
```

```
## [1] -2.894610e-01 -3.242557e-05
```

```r
new_t <- (y.bar - 0) / stand_err
new_p_val <- pt(new_t, i-1)*2
new_p_val # confirm p=0.01 level
```

```
## [1] 0.009983347
```

```r
i-5 # additional observations needed
```

```
## [1] 2176
```

Q3

```r
y <- 0.1 + 0.2 * x + ep
```

a

```r
lm.3 <- lm(y ~ x)
summary(lm.3)
```

```
##
## Call:
## lm(formula = y ~ x)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.93732 -0.70038 -0.01064  0.61814  1.87553
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.11465    0.09356   1.225  0.22337
## x            0.30259    0.08924   3.391  0.00101 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.935 on 98 degrees of freedom
## Multiple R-squared:  0.105,  Adjusted R-squared:  0.09587
## F-statistic:  11.5 on 1 and 98 DF,  p-value: 0.001006
```

```
qt(0.975, 98)
```

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

```
# 95% CI lower bound
lm.3$coefficients[2] - qt(0.975, 98) * coef(summary(lm.3))[4]
```

```
##         x
## 0.1254995
```

```
# 95% CI upper bound
lm.3$coefficients[2] + qt(0.975, 98) * coef(summary(lm.3))[4]
```

```
##         x
## 0.4796795
```

the coefficient on x is the slope of the line of best fit and implies that we are 95% confindence that an increase in x by 1.0 will result in an increase in y by the estimated slope. since there is uncertainty, the standard error in slope will help determine the 95% CI

b

```
# retrieve t value of coefficient of x
coef(summary(lm.3))[[6]]
```

```
## [1] 3.390812
```

```
pt(1.847, lower.tail = FALSE, 98) * 2
```

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

It confirms that it matches regression output and it indicates that at p=0.05 threhold level, we fail to reject the null hypothesis that coefficient is 0, which means change in x wont have effect on y

c

```
pf(3.41, 1, 98, lower.tail = F)
```

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

F test is to measure overall significance of model as well by asking whether any coefficients are significantly different from 0. Since at p=0.05 level, we fail to reject the null hypothesis that none of coefficients are significantly different from 0

d

```r
x5 <- x[1:5]
y5 <- y[1:5]
x5.bar <- mean(x5)
y5.bar <- mean(y5)

# calculate slope
slope <- sum((x5-x5.bar)*(y5-y5.bar))/sum((x5-x5.bar)^2)
slope
```

```
## [1] 0.242416
```

```r
intercept <- y5.bar - slope * x5.bar
intercept
```

```
## [1] -0.1215077
```

```r
#calculate se
y5.pred <- intercept + slope * x5
error <- y5 - y5.pred
se.slope <- sqrt(sum(error^2)/(5-2)) * (1/sqrt(sum(x5-x5.bar)^2))
se.slope
```

```
## [1] 2.3648e+16
```

```r
# calculate adjusted R^2
tss <- sum((y5-y5.bar)^2)
sse <- sum((y5-y5.pred)^2)
#dft=n-1
dft <- 5-1
#df2=n-k-1
dfe <- 5-1-1
adj.r2 <- (tss/dft-sse/dfe) / (tss/dft)
adj.r2
```

```
## [1] -0.2568931
```

Q4

```r
y <- 0.1 + 0.2 * x - 0.5 * x^2 + ep
```

a

```r
fit.r4 <- lm(y ~ x + I(x^2))
summary(fit.r4)
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.97988 -0.61633 -0.01342  0.61830  1.92586
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.18306    0.11577   1.581  0.11709
## x            0.29551    0.08951   3.301  0.00135 **
## I(x^2)      -0.56247    0.06227  -9.032 1.65e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.935 on 97 degrees of freedom
## Multiple R-squared:  0.502,  Adjusted R-squared:  0.4918
## F-statistic:  48.9 on 2 and 97 DF,  p-value: 2.061e-15
```

They are both significant
b

```
(0.1 + 0.2 * 2 - 0.5 * 2^2) - (0.1 + 0.2 * 1 - 0.5 * 1^2)
```

```
## [1] -1.3
```

c

```
predict(fit.r4, data.frame(x=-0.7)) - predict(fit.r4, data.frame(x=-0.5))
```

```
##          1
## -0.1940939
```

Q5

```
x2 <- rnorm(100, mean=-1, sd=1)
y <- 0.1 + 0.2 * x - 0.5 * x * x2 + ep
```

a

```
x.mean <- mean(x)
(0.1 + 0.2 * x.mean - 0.5 * x.mean * 1) - (0.1 + 0.2 * x.mean - 0.5 * x.mean * 0)
```

```
## [1] 0.01785416
```

b

```
lm.complete <- lm(y ~ x + x2 + x:x2)
predict(lm.complete, data.frame(x=-0.7, x2=1)) - predict(lm.complete, data.frame(x=-0.5, x2=1))
```

```
##          1
## 0.07929373
```

c

```
lm.reduced <- lm(y~x)
r2.complete <- summary(lm.complete)$r.squared
r2.reduced <- summary(lm.reduced)$r.squared
#df1=2, which is additional variables in complete model
#n=100, k=3, so df2=100-3-1
fstat <- ((r2.complete - r2.reduced) / 2) / ((1 - r2.complete) / (100 - 3 - 1))
```

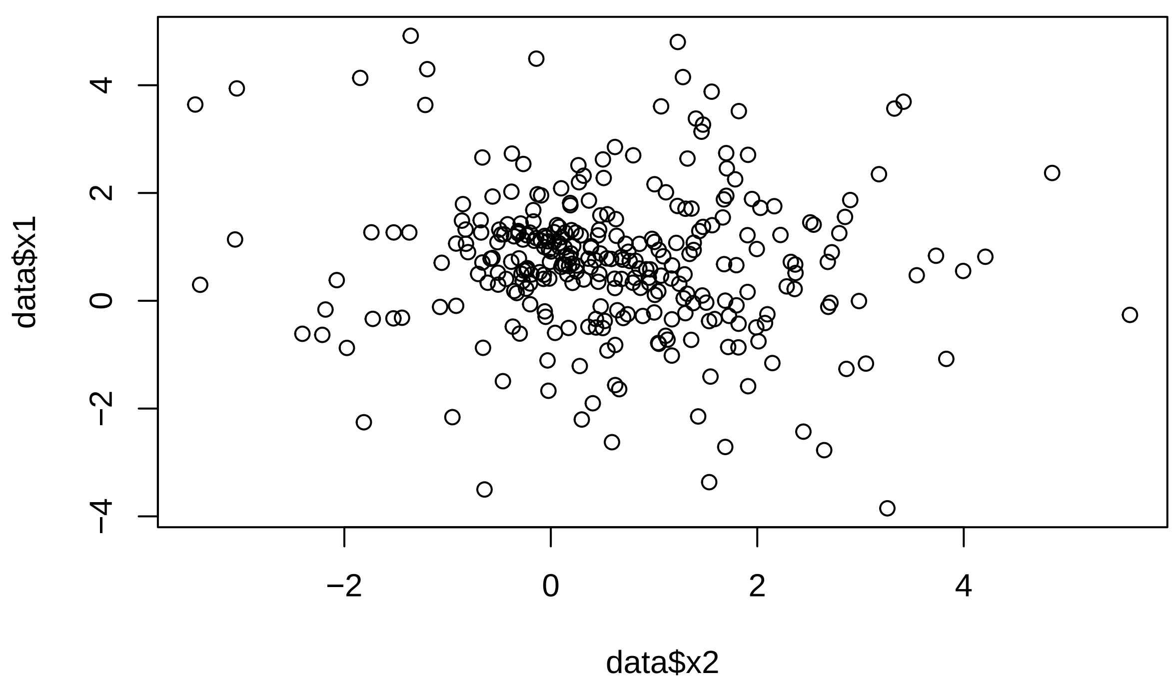
```r
# p value
pf(fstat,2,(100-3-1),lower.tail=F)
```

```
## [1] 1.073654e-08
```

the p value is way less than threshold p=0.05, so the F test reject the null hypothesis that complete regression model do no better than reduced regression model.

Q6
a

```r
n <- 100
f <- c(rep("a", n), rep("b", n), rep("c", n))
x1 <- c(rnorm(n, 1, 2), rnorm(n, 0, 1), rnorm(n, 1, 0.5))
x2 <- c(rnorm(n, 1, 2), rnorm(n, 1, 1), rnorm(n, 0, 0.5))
data <- as.data.frame(cbind(x1, x2, f))
data$f <- as.factor(data$f)
plot(data$x2, data$x1)
```




```r
kout <- kmeans(data[,1:2], 3, 1000)
kout$centers
```

```
##           x1        x2
## 1  1.8660783  1.971075
## 2 -0.6663173  1.128106
## 3  1.0869306 -0.299753
```

```
data$cluster <- kout$cluster
table(data$f, data$cluster)
```

```
##
##      1  2  3
##   a 41 21 38
##   b 17 64 19
##   c  5  6 89
```

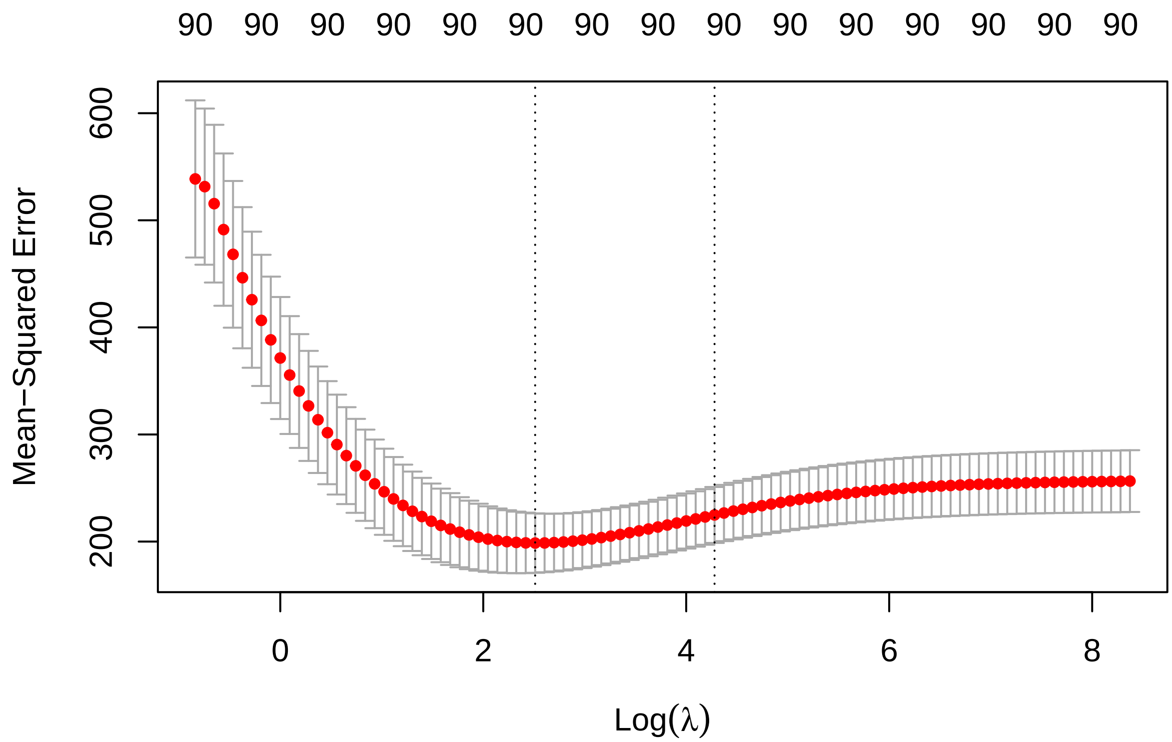
compare centroids from kmeans output with true means from normal distribution, I see they are not similar. And it is quite challenging to distinguish clusters of the plot from normal distribution.
However, centroids from kmeans is pretty obvious: cluster c/3 has a positive on x1 around 1 and negative or around 0 on x2, cluster b/2 has a negative on x1 around -1 and positive around 1 on x2, cluster a/1 has both positive and around 2.
Regarding similarity between original cluster and cluster after kmeans, c cluster has very high accuracy, while b cluster is mostly classified as correct b or incorrecly a or b with each around 50%, a cluster is poorly classified and distributed all over 3 cluster with 30%.

Q7
a

```
X <- matrix(rnorm(200*90, 0, 1), nrow=200)
beta <- c(rep(2, 30), rep(-1, 30), rep(0, 30))
episilon <- rnorm(200, 0, 10)

y <- X %*% beta + episilon

#ridge,lasso,elastic net model with cv
ridge.fit <- cv.glmnet(X[1:100,], y[1:100], alpha=0)
lasso.fit <- cv.glmnet(X[1:100,], y[1:100], alpha=1)
elnet.fit <- cv.glmnet(X[1:100,], y[1:100], alpha=0.5)

ridge.fit
```

```
##
## Call:  cv.glmnet(x = X[1:100, ], y = y[1:100], alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min  12.32    64   198.6 27.77      90
## 1se  72.18    45   224.8 26.11      90
```

```
plot(ridge.fit)
```




```r
min(ridge.fit$cvm)# lowest error for ridge
```

```
## [1] 198.5938
```

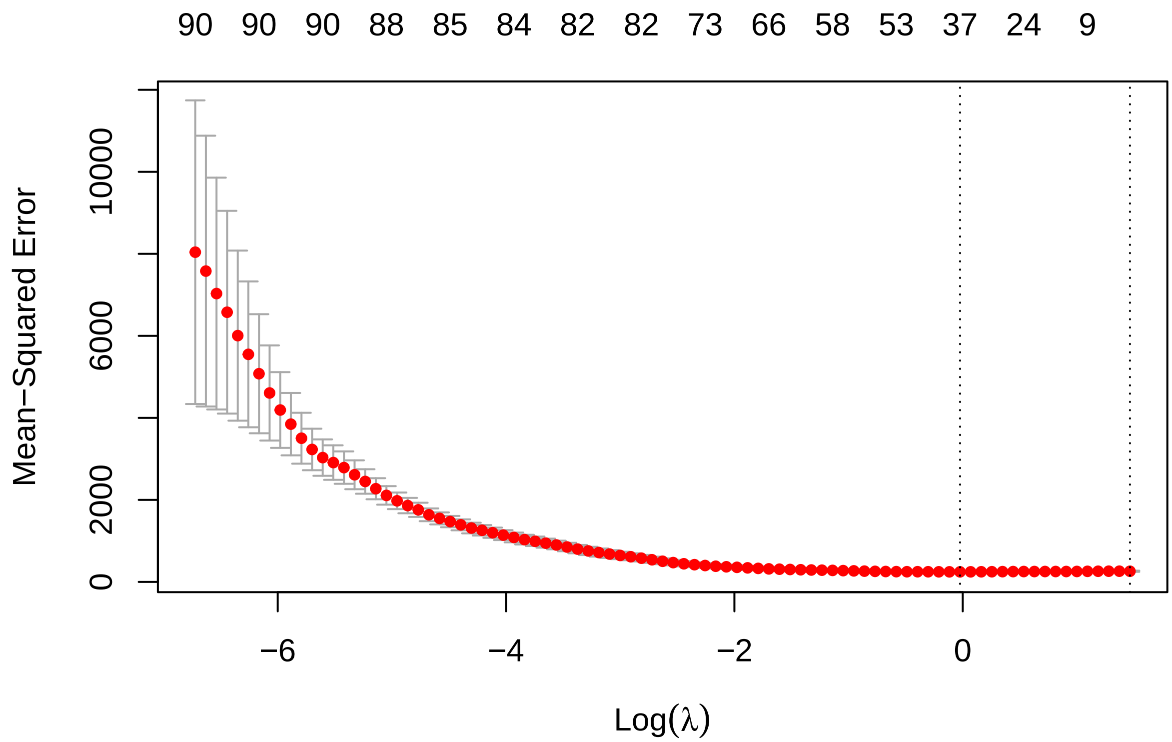
```r
ridge.fit$lambda.min
```

```
## [1] 12.32324
```

```r
lasso.fit # find min lmbda and its lowest MSE
```

```
##
## Call:  cv.glmnet(x = X[1:100, ], y = y[1:100], alpha = 1)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min  0.977    17   245.2 33.23      37
## 1se  4.327     1   260.1 14.39       0
```

```r
plot(lasso.fit)
```



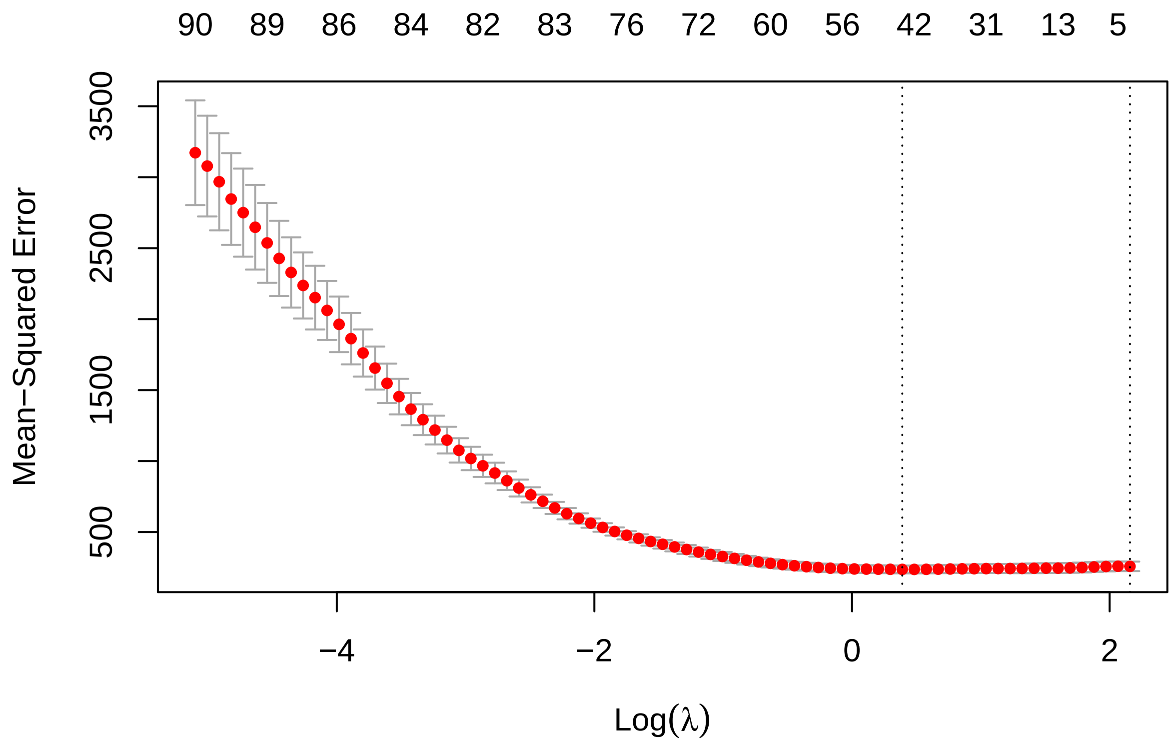

```
min(lasso.fit$cvm) # lowest error for lasso
```

```
## [1] 245.251
```

```
elnet.fit # find min lmbda and its lowest MSE
```

```
##
## Call:  cv.glmnet(x = X[1:100, ], y = y[1:100], alpha = 0.5)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min  1.478    20   237.0 27.04      47
## 1se  8.654     1   258.6 33.38       0
```

```
plot(elnet.fit)
```




```
min(elnet.fit$cvm) # lowest error for elastic net
```

```
## [1] 237.0356
```

Ridge regression with alpha=0, lambda=12.32 produces lowest MSE so i choose this model for testing.

b

```
fit.coef <- coef(ridge.fit)[, "s1"]
fit.30.acc <- sum(round(fit.coef[2:31]) == 2)
fit.60.acc <- sum(round(fit.coef[32:61]) == -1)
fit.90.acc <- sum(round(fit.coef[62:91]) == 0)
sum(fit.30.acc, fit.60.acc, fit.90.acc)
```

```
## [1] 31
```

Overall, after rounding to leave only the digit to compare with the simulated data, about 31/90 is approximately matched.

c

```
beta.fit <- coef(ridge.fit)[, "s1"]
X200 <- X[101:200, ]
y200 <- y[101:200]
#predicted y
yhat.200 <- cbind(1,X200) %*% beta.fit
```

```r
# mse
mean((y200 - yhat.200)^2)
```

```
## [1] 171.4463
```

d

```r
# multiple regression
lm.100 <- lm(y[1:100] ~ X[1:100,])
# beta
lm.beta.fit <- coef(lm.100)
# predicted
yhat.lm <- cbind(1,X200) %*% lm.beta.fit
# mse
mean((y200 - yhat.lm)^2)
```

```
## [1] 759.6831
```

The MSE from regular regression is about 5 times than ridge regression. One possible reason is that regular regression is overfitting in sample data and predicted poorly on out of sample data. The ridge regression will drop any variable due to having a high number of independent variables.

Q8

```r
y2 <- cut(y,breaks=c(-Inf,0,Inf),labels=c("0","1"))
```

a

```r
df <- data.frame(X_100=X[1:100,], y2_100=y2[1:100])
costvalues <- 10^seq(-3,2,1)
# linear kernel
svm.l <- tune(svm, y2_100 ~ ., data=df, ranges=list(cost=costvalues), kernel="linear")
# best cost = 0.01
svm.l$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = y2_100 ~ ., data = df, ranges = list(cost = costvalues),
##     kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##
## Number of Support Vectors:  67
```

```r
# linear kernel in smaple accuracy
yhat.in.l <- predict(svm.l$best.model, df)
sum(yhat.in.l == df$y2_100)/length(df$y2_100) # 0.99 accuracy
```

```
## [1] 0.99
```

```r
# radial kernel
svm.r <- tune(svm, y2_100 ~ ., data=df, ranges=list(cost=costvalues), kernel="radial")
# best cost = 10
svm.r$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = y2_100 ~ ., data = df, ranges = list(cost = costvalues),
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  100
```

```r
# linear kernel in smaple accuracy
yhat.in.r <- predict(svm.r$best.model, df)
sum(yhat.in.r == df$y2_100)/length(df$y2_100) # 1 accuracy
```

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

b

```r
y2_200 <- y2[101:200]
# choose radial kernel predict with test, accuracy since it has higher in sample accuracy
yhat.out.r <- predict(svm.r$best.model, X[101:200,])
table(yhat.out.r, y2_200)
```

```
##           y2_200
## yhat.out.r  0  1
##          0 31 11
##          1 18 40
```

```r
sum(yhat.out.r == y2_200)/length(y2_200) #0.68 accuracy
```

```
## [1] 0.71
```