Problem Set 7: Prediction Modeling

Linh Vu (Collab: Brice Laurent)

Due: November 10, 2023

This assignment is **due Friday, November 10 at 11:59pm**, handed in on Gradescope (remember, there are two separate submissions, one for your pdf, and another for you rmd file). Show your work and provide clear explanations when asked. **Incorporate the relevant R output in this R markdown file**. Only the key output should be displayed for each problem and the relevant parts should be **highlighted** in some way. Make sure that you write-up any interpretation of R-code in your own words (don't just provide the output). Note that, in order to lighten the load in the week before your take-home midterm, we will not be grading Question 4.

Collaboration policy (for this and all future problem sets): You are encouraged to discuss the problems with other students, but you must write up your solutions yourself and in your own words. Copying someone else's solution, or just making trivial changes is not acceptable.

Problem 1.

The closed-form solutions for the ridge estimates is calculated to be:

$$\hat{\vec{\beta}}_{ridge} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}^*)^{-1}\mathbf{X}^T\vec{y}$$

a) Derive the bias of $\hat{\vec{\beta}}_{ridge}$ for $\vec{\beta}$. You can leave it in matrix form. When will this bias be zero (in terms of λ)?

b) Derive the variance-covariance matrix of $\hat{\vec{\beta}}_{ridge}$. You can leave it in matrix form.

c) Letting $\lambda = 2$, complete the code below to calculate the estimated variance-covariance matrix of both $\hat{\vec{\beta}}_{ridge}$ and the ordinary least squares estimator, $\hat{\vec{\beta}}_{ols}$, for the following data set using matrix calculations in R. Check that the estimated OLS variance-covariance matrix is the same as the one computed by `lm`.

The estimated OLS variance-covariance matrix is the same as the one computed by ‘lm‘.

```
set.seed(139)
n = 100
rho = 0.95
z = rnorm(n)
x1 = sqrt(rho)*z + sqrt(1-rho)*rnorm(n)
x2 = sqrt(rho)*z + sqrt(1-rho)*rnorm(n)
x3 = rnorm(n)
y = x1+x3+rnorm(n)
X = cbind(1,x1,x2,x3)

lambda = 2
I.star = matrix(0,nrow=ncol(X),ncol=ncol(X))
```

```
diag(I.star)[2:nrow(I.star)]=1

beta.ridge = solve(t(X)%*%X+lambda*I.star)%*%t(X)%*%y
beta.ols = solve(t(X)%*%X)%*%t(X)%*%y

sigmasq.hat.ridge = sum((y-X%*%beta.ridge)^2)/(n-length(beta.ridge))
sigmasq.hat.ols = sum((y-X%*%beta.ols)^2)/(n-length(beta.ols))

# beta
beta.ridge
```

```
##            [,1]
##    -0.004315544
## x1  0.886349287
## x2 -0.025790360
## x3  0.927748404
```

```
beta.ols
```

```
##           [,1]
##    -0.01569774
## x1  1.08647333
## x2 -0.21294868
## x3  0.96019150
```

```
# SE
sigmasq.hat.ridge
```

```
## [1] 0.9125486
```

```
sigmasq.hat.ols
```

```
## [1] 0.9081255
```

```
# variance-covariance matrix
A = solve(t(X)%*%X + lambda*I.star) %*% t(X)
sigmasq.hat.ridge*A%*%t(A)              # ridge
```

```
##                           x1           x2            x3
##     0.0095082501 -0.002670199  0.003818594  0.0002773345
## x1 -0.0026701986  0.049320192 -0.044285503  0.0028253798
## x2  0.0038185944 -0.044285503  0.048577653 -0.0017290833
## x3  0.0002773345  0.002825380 -0.001729083  0.0101455494
```

```
A_ols = solve(t(X)%*%X) %*% t(X)
sigmasq.hat.ols*A_ols%*%t(A_ols)      # ols
```

```
##                           x1           x2            x3
##     0.0096815169 -0.005765768  0.006907392  0.0001297735
## x1 -0.0057657678  0.094383055 -0.088867490  0.0053038045
## x2  0.0069073925 -0.088867490  0.092837415 -0.0041171469
## x3  0.0001297735  0.005303805 -0.004117147  0.0106968601
```

```r
# lm
mod <- lm(y~x1+x2+x3)
vcov(mod)
```

```
##                (Intercept)           x1           x2            x3
## (Intercept)  0.0096815169 -0.005765768  0.006907392  0.0001297735
## x1          -0.0057657678  0.094383055 -0.088867490  0.0053038045
## x2           0.0069073925 -0.088867490  0.092837415 -0.0041171469
## x3           0.0001297735  0.005303805 -0.004117147  0.0106968601
```

d) How do the variance estimates for each of $\hat{\beta}_1$, $\hat{\beta}_2$, and $\hat{\beta}_3$ compare for the OLS and the ridge estimators in this problem? Which are/is affected the most? What are/is affect the least? Explain why this makes sense based on how $x1$, $x2$, and $x3$ were generated.

The variance estimates for $\hat{\beta}_1$ and $\hat{\beta}_2$ are almost double for OLS (when compared to Ridge), but the variance estimates for $\hat{\beta}_3$ are roughly the same for both methods. This makes sense because $x1$ and $x2$ are generated the same way (which includes 2 calls of rnorm() in the data generating process), with more variation than $x3$ (which only includes 1 call of rnorm() in the data generating process). Thus $x1$ and $x2$ have more variation and the variation estimates are larger for OLS than for Ridge (which shrinks everything towards 0 and makes variation estimates smaller). $x3$ has similar variation estimates for OLS and Ridge because it naturally has less variation due to how it is generated.

The training data set that will be used in the next 2 problems of this problem set is ‘bosflights18.csv’ which includes a subset of flights in and out of Boston’s Logan Airport for the year 2018. The variables in the data set include (for $n = 10,000$ flights):

flight_time: the total amount of time the flight takes from the time the plane takes off until the time it arrives at the destination gate.

year: year of flight (they are all from 2018)

month: month: 1 = January, 2 = February, etc.

dayofmonth: the calendar day of the month, from 1 to 31.

weekday: day of the week: 1 = Monday, 2 = Tuesday, etc.

carrier: the unique 2-digit carrier code of the flight. For details, see the list here: https://en.wikipedia.org/wiki/List_of_airlines_of_the_United_States

tailnum: the unique tail number of the aircraft

flightnum: the carrier’s specific flight number

origin: the originating airport. See http://www.leonardsguide.com/us-airport-codes.shtml.

dest: the destination airport.

bos_depart: an indicator if the flight departed out of Boston.

schedule_depart: the scheduled departure time in minutes across the day ranging from 0 to 1439. 7pm is 1140, for example.

depart: the actual departure time (in minutes)

wheels_off: the time of day the plane took off (in minutes)

distance: the distance of the flight, in miles.

weather_delay: an indicator if the delay is due to extreme weather.

nas_delay: an indicator if the delay is due to the national aviation system (air traffic control, for example).

security_delay: an indicator if the delay is due to a security issue at the terminal.

late_aircraft_delay: an indicator if the delay is due to a late arrival of a previous flight with the same aircraft.

carrier_delay: an indicator if the delay is due to a carrier (kind of a catch all if not the others).

More info on the delay indicators can be found at the Bureau of Transportation Statistics (BTS).

We are looking to predict `flight_time` based on all of the other predictors in the data set (all other variables could be measured at some point before the flight takes off). **Note**: a separate test set, ‘bosflights18_test.csv’ is also provided for later use (which is actually larger in size).

Problem 2. Exploratory Data Analysis and Linear Models

(a) Fit a simple linear model to predict flight time from distance (call it **lm1**). Report root mean squared error (RMSE) on both the train and test sets (do not adjust for degrees of freedom).

```r
# load data
flight      <- read.csv("data/bosflights18.csv")
flight_test <- read.csv("data/bosflights18_test.csv")

# model
lm1 <- lm(flight_time~distance, flight)

# rmse on train data
sqrt(mean(lm1$residuals^2))
```

```
## [1] 15.83715
```

```
# rmse on test data
pred.data <- as.data.frame(flight_test$distance)
colnames(pred.data) <- "distance"
sqrt(mean((flight_test$flight_time - predict(lm1, newdata = pred.data))^2))
```

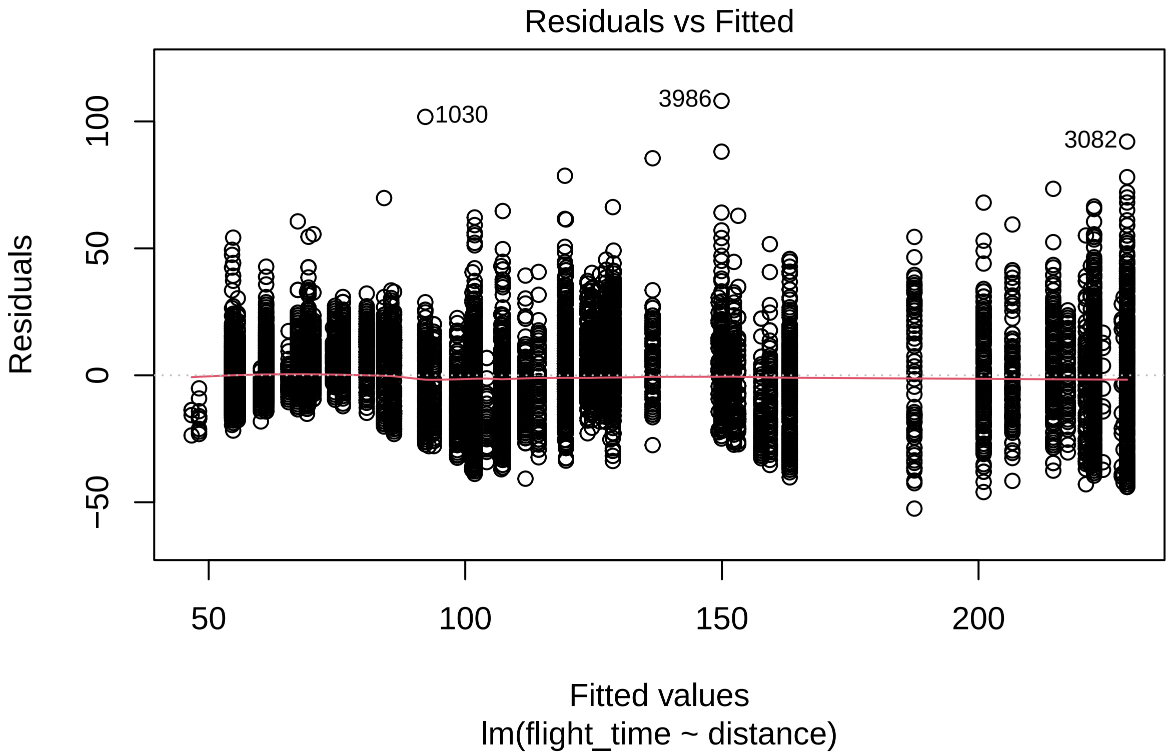
```
## [1] 15.89582
```

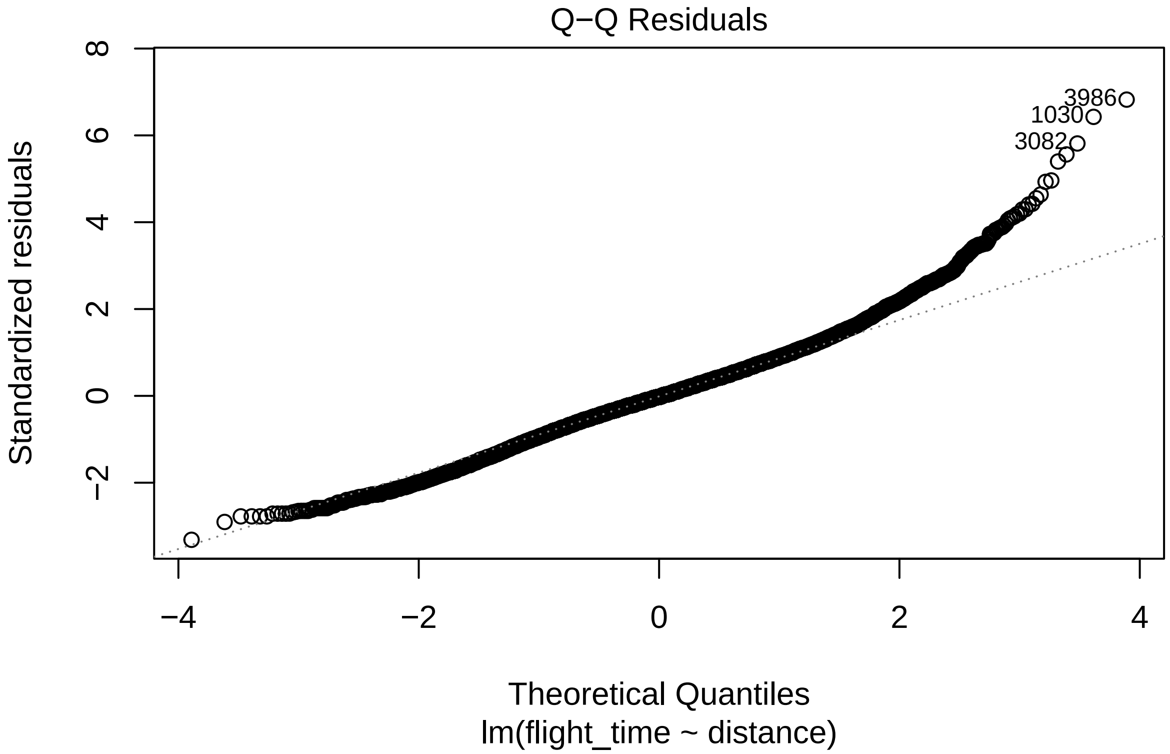
(b) Briefly interpret the β coefficients of **lm1** and comment on the quality of this model using a diagnostic plot.

```
# coef
coef(lm1)
```

```
## (Intercept)    distance
## 41.84099158  0.06919653
```

```
# diagnostic
plot(lm1, which=c(1,2))
```





The intercept of 41.841 means that if the distance is 0, the flight time is 41.841 units. The slope of 0.069 means that as the distance increases by 1 mile, the flight time increases by 41.841 unit.

Assumptions check: linearity is met because the red line in the 1st plot is straight; homoskedasticity is met because there is no weird funnel/fanning pattern in the residual plot. Normality is violated because the standardized residual quantiles don't closely follow the theoretical normal quantiles. The independence assumption might be violated (flight time of different flights can be dependent on each other due to weather or traffic).

(c) Fit three linear models:

- **lm2** that predicts flight time from the main effects of all of the included predictors (untransformed quantitative predictors, but be sure to handle categorical predictors appropriately).
- **lm3** that predicts flight time from the main effects of all of the included predictors but treats `distance` with a 15^{th} order polynomial function (do NOT use the `raw=T` argument in `poly`).
- **lm4** that predicts flight time from the main effects (treating `distance` with a 15^{th} order polynomial function) and the interactions of `bos_depart` with all the other predictors (including all polynomial terms of `distance`) [ignore other interactions].

  Report 3 things for each model: (1) R^2 on train (2) the number of non-`NA` β estimates and (3) the number of `NA` β estimates.

*Note: you should completely ignore 4 variables here: `year`, `day_of_month`, `flightnum`, and `tailnum`.

```
# handle factor var
flight$weekday      <- as.factor(flight$weekday)
flight_test$weekday <- as.factor(flight_test$weekday)

flight$month        <- as.factor(flight$month)
flight_test$month   <- as.factor(flight_test$month)

# subset
data <- subset(flight, select=-c(year, dayofmonth, flightnum, tailnum))

# fit models
lm2 <- lm(flight_time~., data)
lm3 <- lm(flight_time~. + poly(distance,15) - distance, data)
lm4 <- lm(flight_time~. + poly(distance,15)  - distance
          + bos_depart*(. - distance)
          + bos_depart*poly(distance,15), data)
# r2
summary(lm2)$r.sq
```

```
## [1] 0.9511597
```

```
summary(lm3)$r.sq
```

```
## [1] 0.9560594
```

```
summary(lm4)$r.sq
```

```
## [1] 0.9606136
```

```
# count NAs
table(is.na(coef(lm2)))
```

```
##
## FALSE  TRUE
##    79     2
```

```
table(is.na(coef(lm3)))
```

```
##
## FALSE  TRUE
##    93     2
```

```
table(is.na(coef(lm4)))
```

```
##
## FALSE  TRUE
##   146    42
```

For `lm2`, R^2 on train is 0.951. There are 69 non-‘NA‘ β estimates and 2 ‘NA‘ β estimates.

For `lm3`, R^2 on train is 0.956. There are 84 non-‘NA‘ β estimates and 2 ‘NA‘ β estimates.

For `lm2`, R^2 on train is 0.961. There are 128 non-‘NA‘ β estimates and 42 ‘NA‘ β estimates.

(d) Why are there `NA` estimates (be specific to this datset)?

The `NA` estimates ???

(e) Evaluate the three models from the previous part (**lm2**, **lm3**, and **lm4**) based on RMSE on both the train and test sets. Interpret the results as to which model is best for prediction and which models may be overfit.

```r
# rmse function
rmse <- function(model, data, pred.data){
  y    <- data$flight_time
  yhat <- predict(model, newdata = pred.data)
  sqrt(mean((y-yhat)^2))
}

# rmse on train data
train2 <- rmse(lm2, flight, flight)
train3 <- rmse(lm3, flight, flight)
train4 <- rmse(lm4, flight, flight)

# rmse on test data
test2 <- rmse(lm2, flight_test, flight_test)
test3 <- rmse(lm3, flight_test, flight_test)
test4 <- rmse(lm4, flight_test, flight_test)

# make table
rmse2e <- data.frame("train" = c(train2, train3, train4),
                     "test" = c(test2, test3, test4))
rownames(rmse2e) <- c("lm2", "lm3", "lm4")
rmse2e
```

```
##        train     test
## lm2 12.14254 12.07446
## lm3 11.51737 11.56716
## lm4 10.90419 10.98704
```

Model 4 has the best results because of lowest RMSE on both train and test datasets. Model 2 has highest train and test RMSEs (but not by much) and has a significantly simpler structure, so model 2 is appropriate. Models 3 and 4 probably overfit the data because RMSE for test data is higher than RMSE for train data (whereas the opposite is true for model 2).

Problem 3.

(a) Fit well-tuned Ridge and LASSO regression models using `cv.glmnet` based on the predictors used in the **lm4** model from the previous problem. Hint: the R command `model.matrix` may be helpful to get you started.

```r
library(glmnet)

# get predictors
X = model.matrix(lm4)[,-1]

# fit
```

```
ridge <- cv.glmnet(X, flight$flight_time, alpha=0)
lasso <- cv.glmnet(X, flight$flight_time, alpha=1)

ridge
```

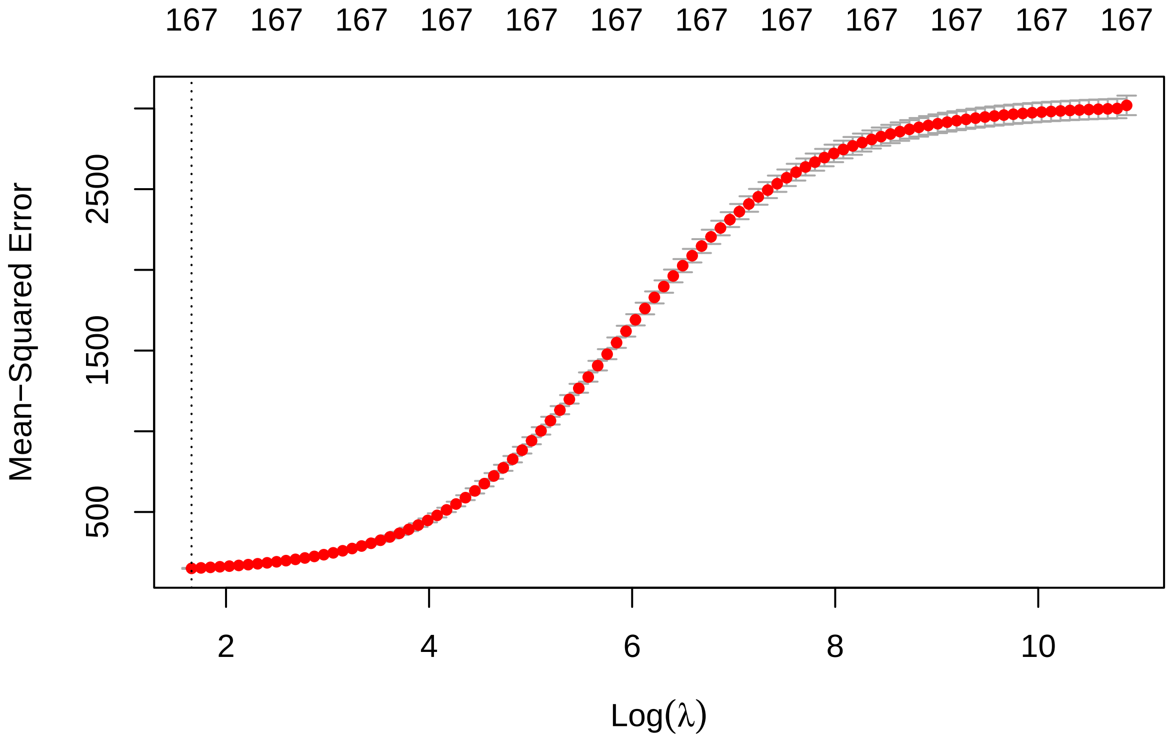
```
##
## Call:  cv.glmnet(x = X, y = flight$flight_time, alpha = 0)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min  5.261   100   150.7 2.742     167
## 1se  5.261   100   150.7 2.742     167
```

```
lasso
```

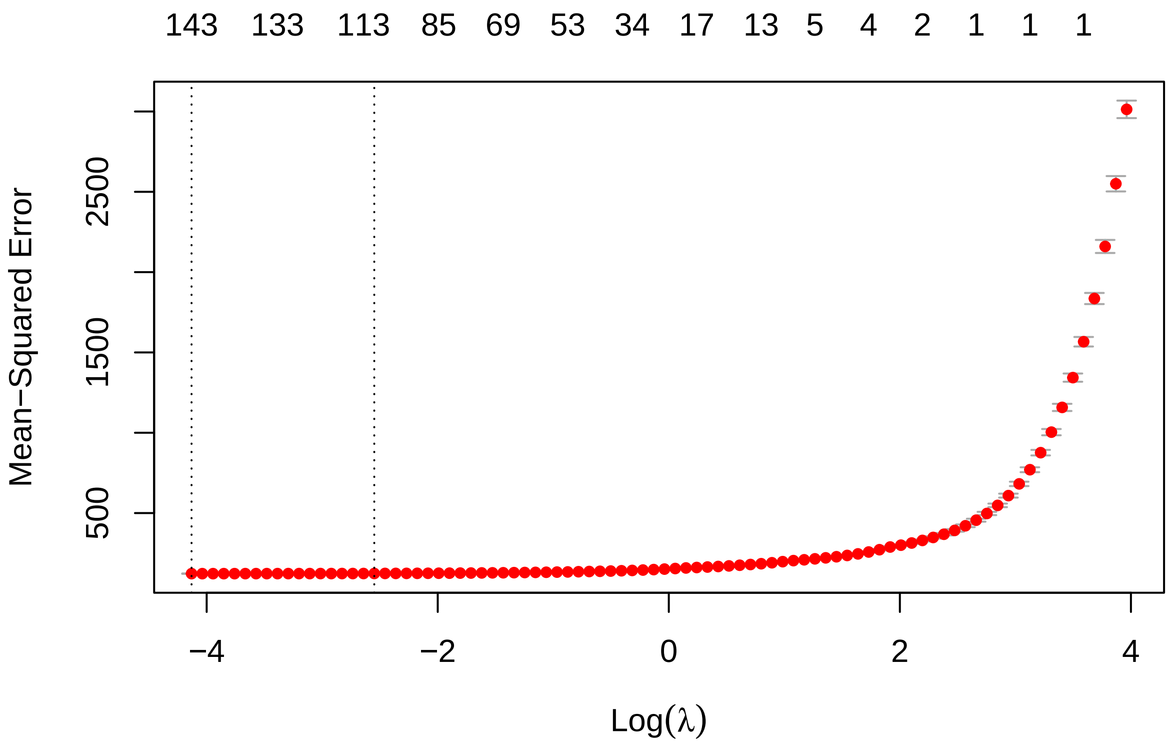
```
##
## Call:  cv.glmnet(x = X, y = flight$flight_time, alpha = 1)
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure    SE Nonzero
## min 0.01607    88   122.9 1.124     143
## 1se 0.07813    71   123.9 1.135     110
```

(b) For both the Ridge and LASSO models, plot the average MSE on the validation sets against the λ's you considered in the previous part. Report the best λ's. (This part should require almost no work if you did part (a)).

```
plot(ridge)
```


167
167
167
167
167
167
167
167
167
167
167
167
Mean−Squared Error
500
1500
2500
2
4
6
8
10
Log(λ)

```
plot(lasso)
```



```
ridge$lambda.min
```

```
## [1] 5.261206
```

```
lasso$lambda.min
```

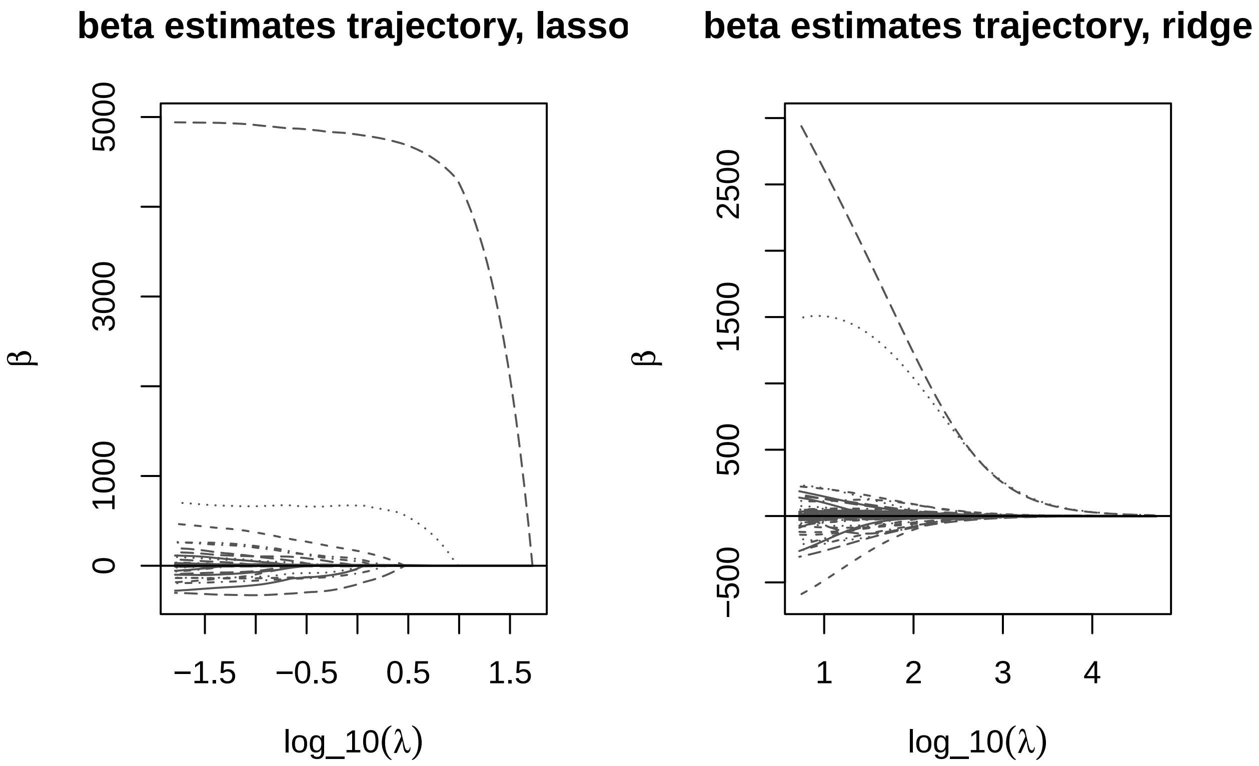
```
## [1] 0.01606696
```

The best lambda for Ridge is 5.2612, and the best lambda for LASSO is 0.0161.

(c) Provide the "$\hat{\beta}$ trajectory" plots of the main effects from these models (plot each β_j as a function of λ as a curve, and do this for all main/linear effects). Interpret what you see in 2-3 sentences.

```
par(mfrow=c(1,2))

# lasso
lassos = glmnet(X,flight$flight_time, alpha = 1,nlambda=100)

matplot(log(lassos$lambda,10),
        t(lassos$beta),
        type="l",col="gray33",lwd=1,
        xlab=expression(log_10(lambda)),
        ylab=expression(hat(beta)),
        main="beta estimates trajectory, lasso")
```

```
abline(h=0)

# ridge
lassos = glmnet(X,flight$flight_time, alpha = 0,nlambda=100)

matplot(log(lassos$lambda,10),
        t(lassos$beta),
        type="l",col="gray33",lwd=1,
        xlab=expression(log_10(lambda)),
        ylab=expression(hat(beta)),
        main="beta estimates trajectory, ridge")
abline(h=0)
```



LASSO shrinks the beta estimates to 0 at lower lambda values than Ridge (for LASSO, the beta estimates become 0 at around lambda of 1.7, while they approach 0 at around lambda of 4 for Ridge). Most beta estimates approach 0/become 0 quickly. The two largest beta estimates shrink in a very similar way to each other in both Ridge and LASSO methods.

(d) Choose a best regularized/penalized regression model and briefly justify your choice. Revisit the grid of λ's that were used (either explicitly by you, or automatically by R and comment on whether it's obvious that these penalized models will predict better than the original model.

```
test4^2
```

```
## [1] 120.715
```

```r
min(ridge$cvm)
```

```
## [1] 150.6799
```

```r
min(lasso$cvm)
```

```
## [1] 122.8579
```

LASSO is the better regularized regression model because it has lower MSE. ???

Problem 4. (This one will not be graded)

This problem is intended to investigate the effect of unimportant predictors on the sequential model selection and LASSO approaches for variable selection. Specifically, the simulation will have a response variable Y that is related to one predictor $Z_,Z_2$ but is unrelated to 10 other independent predictors, $X_1, ..., X_{10}$ (in fact, all predictors are independent). Set $n = 100$ and perform $nsims = 500$ iterations of the following simulation:

i) Sample $Z_1, Z_2, X_1, ..., X_{10}$ all independently from the standard normal distribution.

ii) Sample $Y|Z, \vec{X} \sim N(3 + 1 \cdot Z_1 + 2 \cdot Z_2, 5^2)$.

*Note, it is more efficient to sample all Z_1, Z_2 and $\vec{X}$ for each iteration from one `rnorm` function call, and then reorganize them into the separate variables for model fitting.

For each iteration, take three separate variable selection approaches for linear regression:

1) Fit the full linear regression model with all 12 predictors and keep track of which predictors' t-test for $H_0 : \beta_j = 0$ are significant in the presence of this model.

2) Perform backward sequential variable selection (via AIC) starting with the full model with all 12 predictors. Keep track of which variables are kept in the resulting model.

3) Perform a LASSO approach (be sure to choose an optimal λ carefully) on the full model with 12 predictors. Keep track of which variables' β coefficient estimates are not shrunken to zero.

```
library(glmnet)
n=100
nsims=200
data <- data.frame(Y = as.numeric(),
          Z1 = as.numeric(),
          Z2 = as.numeric(),
          X1 = as.numeric(),
          X2 = as.numeric(),
          X3 = as.numeric(),
          X4 = as.numeric(),
          X5 = as.numeric(),
          X6 = as.numeric(),
          X7 = as.numeric(),
          X8 = as.numeric(),
          X9 = as.numeric(),
          X10 = as.numeric())

mod1 <- list()

for(i in 1:nsims){

    # generate predictors
    for(j in 1:n){
      data[j,2:14] <- rnorm(13, 0, 1)
      data[j,"Y"] <- rnorm(1, 3 + 1*data[j,"Z1"] + 2*data[j,"Z2"], 5)
    }

  mod1 <- lm(Y~., data)
  mod2 <- lm(Y~., data)
```

```r
  a <- summary(mod1)$coefficients[-1,4]<0.05
  b <- summary(mod2)$coefficients[-1,4]<0.05
  rbind(a,b)
}
```

(a) Which method has the lowest rate of not *flagging* Z_1 or Z_2 as important (the Type II error rates)? Which method has the lowest Type I error rate?

*Note: it is important to keep track of Z_1 and Z_2 separately but the Xs can be treated interchangeable. Think to yourself: why? This is a rhetorical question and there is no need to answer it.

(b) Plot the distribution of the number of Type I errors for each method (500 'observations' that could range from 0 to 10, theoretically, in each of the 3 methods). Interpret what you see (note: what distribution should these follow if the results for each predictor are independent?).

(c) For each of the 3 methods of variable selection, provide a condition/situation in which that approach would be the preferred one. Which method would you take to build a best prediction model most commonly/reliably?