3. Use 10-fold CV to estimate the MSPE for ridge, LASSO-min, and LASSO-1SE. That is,

(a) Set the seed to 2928893 before running the sample.int() function.
(b) Create 10 folds
(c) Run the three analyses on each training set
i. Find the best versions of each for that training set
ii. Use those best versions to compute the prediction error on the validation set

```
library(dplyr)
library(MASS)   # For ridge regression
library(glmnet) # For LASSO
source("Helper Functions.R")
data = na.omit(airquality[, 1:4])
data$TWcp = data$Temp*data$Wind
data$TWrat = data$Temp/data$Wind

set.seed(2928893)



##################
get.folds = function(n, K) {
  ### Get the appropriate number of fold labels
  n.fold = ceiling(n / K) # Number of observations per fold (rounded up)
  fold.ids.raw = rep(1:K, times = n.fold) # Generate extra labels
  fold.ids = fold.ids.raw[1:n] # Keep only the correct number of labels

  ### Shuffle the fold labels
  folds.rand = fold.ids[sample.int(n)]

  return(folds.rand)
}


### Number of folds
K = 10

### Construct folds
n = nrow(data) # Sample size
folds = get.folds(n, K)
```

```r
### Create a container for MSPEs. Let's include ordinary least-squares
### regression for reference
all.models = c("LS", "Hybrid", "Ridge", "LASSO-Min", "LASSO-1se")
all.MSPEs = array(0, dim = c(K, length(all.models)))
colnames(all.MSPEs) = all.models
### Begin cross-validation
for(i in 1:K){
  ### Split data
  data.train = data[folds != i,]
  data.valid = data[folds == i,]
  n.train = nrow(data.train)

  ### Get response vectors
  Y.train = data.train$Ozone
  Y.valid = data.valid$Ozone

  # LS
  fit.ls = lm(Ozone ~ ., data = data.train)
  pred.ls = predict(fit.ls, newdata = data.valid)
  MSPE.ls = get.MSPE(Y.valid, pred.ls)
  all.MSPEs[i, "LS"] = MSPE.ls

  #Hybrid Stepwise

  fit.start = lm(Ozone ~ 1, data = data.train)
  fit.end = lm(Ozone ~ ., data = data.train)

  step.BIC = step(fit.start, list(upper = fit.end), k = log(n.train),
                  trace = 0)

  pred.step.BIC = predict(step.BIC, data.valid)

  err.step.BIC = get.MSPE(Y.valid, pred.step.BIC)
```

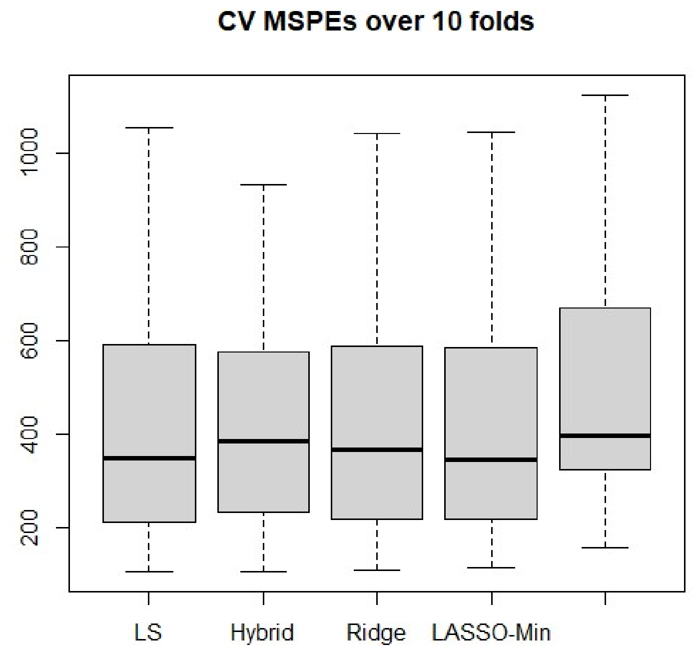
```
all.MSPEs[i, "Hybrid"] = err.step.BIC



#ridge regression
lambda.vals = seq(from = 0, to = 100, by = 0.05)

fit.ridge = lm.ridge(Ozone ~ ., lambda = lambda.vals,
                     data = data.train)

ind.min.GCV = which.min(fit.ridge$GCV)
lambda.min = lambda.vals[ind.min.GCV]

all.coefs.ridge = coef(fit.ridge)
coef.min = all.coefs.ridge[ind.min.GCV,]

matrix.valid.ridge = model.matrix(Ozone ~ ., data = data.valid)

### Now we can multiply the data by our coefficient vector. The
### syntax in R for matrix-vector multiplication is %*%. Note that,
### for this type of multiplication, order matters. That is,
### A %*% B != B %*% A. Make sure you do data %*% coefficients.
### For more information, see me in a Q&A session or, better still,
### take a course on linear algebra (it's really neat stuff)
pred.ridge = matrix.valid.ridge %*% coef.min

### Now we just need to calculate the MSPE and store it
MSPE.ridge = get.MSPE(Y.valid, pred.ridge)
all.MSPEs[i, "Ridge"] = MSPE.ridge

matrix.train.raw = model.matrix(Ozone ~ ., data = data.train)
matrix.train = matrix.train.raw[,-1]

### LASSO
all.LASSOs = cv.glmnet(x = matrix.train, y = Y.train)
```

```r
  ### Get both 'best' lambda values using $lambda.min and $lambda.1se
  lambda.min = all.LASSOs$lambda.min
  lambda.1se = all.LASSOs$lambda.1se

  ### Get the coefficients for our two 'best' LASSO models
  coef.LASSO.min = predict(all.LASSOs, s = lambda.min, type = "coef")
  coef.LASSO.1se = predict(all.LASSOs, s = lambda.1se, type = "coef")

  ### Get which predictors are included in our models (i.e. which
  ### predictors have non-zero coefficients)
  included.LASSO.min = predict(all.LASSOs, s = lambda.min,
                               type = "nonzero")
  included.LASSO.1se = predict(all.LASSOs, s = lambda.1se,
                               type = "nonzero")

  matrix.valid.LASSO.raw = model.matrix(Ozone ~ ., data = data.valid)
  matrix.valid.LASSO = matrix.valid.LASSO.raw[,-1]
  pred.LASSO.min = predict(all.LASSOs, newx = matrix.valid.LASSO,
                           s = lambda.min, type = "response")
  pred.LASSO.1se = predict(all.LASSOs, newx = matrix.valid.LASSO,
                           s = lambda.1se, type = "response")

  ### Calculate MSPEs and store them
  MSPE.LASSO.min = get.MSPE(Y.valid, pred.LASSO.min)
  all.MSPEs[i, "LASSO-Min"] = MSPE.LASSO.min

  MSPE.LASSO.1se = get.MSPE(Y.valid, pred.LASSO.1se)
  all.MSPEs[i, "LASSO-1se"] = MSPE.LASSO.1se
}

### Make a boxplot of MSPEs. I would like to include the number of folds
### in the title. This can be done by using the paste0() function,
### which concatenates strings (i.e. attaches them end-to-end), and
### can be provided numeric variables.
boxplot(all.MSPEs, main = paste0("CV MSPEs over ", K, " folds"))
```

(d) **Report the separate MSPEs from each fold,** $MSPE_v$, $v = 1, . . . , 10$ **and the MSPE for the full data.**

```
> all.MSPEs
             LS    Hybrid     Ridge LASSO-Min LASSO-1se
 [1,]  260.4455 328.0475  280.6840  314.2722  415.7364
 [2,]  322.9743 403.9859  363.7170  316.8250  236.9153
 [3,]  517.3466 577.2344  539.3697  586.1201  717.2987
 [4,]  106.3010 106.9023  108.3192  114.6189  157.0742
 [5,]  192.1326 178.0752  185.4001  190.1739  323.5662
 [6,]  374.3378 369.4887  371.7604  373.4553  668.8504
 [7,]  212.5903 233.2169  219.6295  217.3012  366.8750
 [8,]  592.5841 545.8137  587.6590  585.7460  643.3282
 [9,] 1055.8045 933.6909 1042.9887 1045.5646 1125.8526
[10,]  729.2416 786.7373  725.3818  650.6659  377.7573
```

(e) **Make a boxplot of the 10 CV error estimates showing the boxes for least squares, hybrid stepwise, ridge, and LASSO. Comment on any apparent differences in how the methods seem to perform.**



(f) **Repeat this using relative MSPE.**