OZONE DATA

Refer Lecture 6, Problem 3, where you used 10-fold CV for comparing LASSO models
Using *the same 10 folds* estimate the MSPE for PLS

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

### Create a container for MSPEs. Let's include ordinary least-squares
### regression for reference
all.models = c("PLS")
all.MSPEs = array(0, dim = c(K, length(all.models)))
colnames(all.MSPEs) = all.models
```

```r
for(i in 1:K){
  ### Split data
  data.train = data[folds != i,]
  data.valid = data[folds == i,]
  n.train = nrow(data.train)

  ### Get response vector
  Y.valid = data.valid$per


  fit.pls = plsr(per ~ ., data = data.train, validation = "CV",
                 segments = 5)

  #Code from L7 Question
  mp.cv = fit.pls$validation
  Opt.Comps = which.min(sqrt(mp.cv$PRESS/nrow(data)))
  print(Opt.Comps)
  ### Investigate the fitted PLS model. Comment out the next two
  ### lines when running a CV loop

  ### The summary function gives us lots of information about how
  ### errors change as we increase the number of components
  # summary(fit.pls)

  ### The validationplot() function shows how MSPE from the internal
  ### CV of plsr() changes with the number of included components.
  # validationplot(fit.pls)

  ### Get the best model from PLS. To do this, we need to find the model
  ### that minimizes MSPE for the plsr() function's internal CV. It
  ### takes a few steps, but all the information we need is contained
  ### in the output of plsr().
  CV.pls = fit.pls$validation # All the CV information
  PRESS.pls = CV.pls$PRESS    # Sum of squared CV residuals
  CV.MSPE.pls = PRESS.pls / nrow(data.train)  # MSPE for internal CV
  ind.best.pls = which.min(CV.MSPE.pls) # Optimal number of components

  get.MSPE = function(Y, Y.hat){
    return(mean((Y - Y.hat)^2))
  }
```

```r

   ### Get predictions and calculate MSPE on the validation fold
   ### Set ncomps equal to the optimal number of components
   pred.pls = predict(fit.pls, data.valid, ncomp = ind.best.pls)
   MSPE.pls = get.MSPE(Y.valid, pred.pls)
   all.MSPEs[i, "PLS"] = MSPE.pls
}
```

(a) On each training set, run the PLS function, choosing the optimum number of components using 10-fold CV. You can figure out how many components is optimal according to CV using code like this, assuming that I have named my model, mod.pls

```
mp. cv = mod. p l s $ v a l i d a t i o n
Opt .Comps = which .min( s q r t (mp. cv$PRESS/nrow( i n s . dv ) ) )
```

Report the optimal number of components for each of the 10 folds.

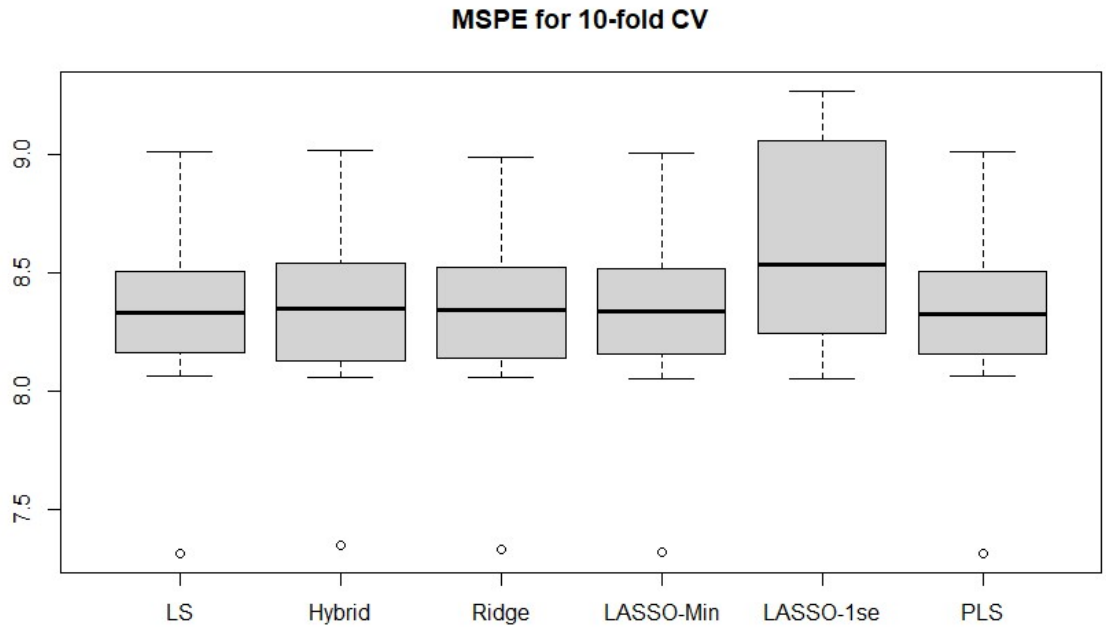
```
[1] 10
[1] 11
[1] 7
[1] 8
[1] 8
[1] 8
[1] 8
[1] 7
[1] 6
[1] 7
```

(b) Use the predict() function to predict responses on the test set. You need to specify ncomps for this function.

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

```
> all.MSPEs
           PLS
 [1,] 8.189934
 [2,] 7.309847
 [3,] 8.505827
 [4,] 8.515403
 [5,] 8.190235
 [6,] 8.162715
 [7,] 8.541857
 [8,] 8.063142
 [9,] 8.509003
[10,] 9.009496
```

(d) **ADD a boxplot for PLS to the boxplots you made for other models. Comment on how well PLS compares to the other models.**



-> PLS does pretty good because it seems to have the lowest MSPE value.

(e) **Remake the plot using relative MSPE and comment.**