Application

Refer to the Air Quality data described previously, and the analyses we have done with Ozone as the response variable, and the five explanatory variables (including the two engineered features).

2. Add GAM on all variables to the 10-fold CV comparison that has been used for LASSO, Ridge, and other methods. Use the same folds for GAM that were used for the other methods.

```
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)

### Create a container for MSPEs. Let's include ordinary least-squares
### regression for reference
all.models = c("LS", "Hybrid", "Ridge", "LASSO-Min", "LASSO-1se", "GAM")
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
```

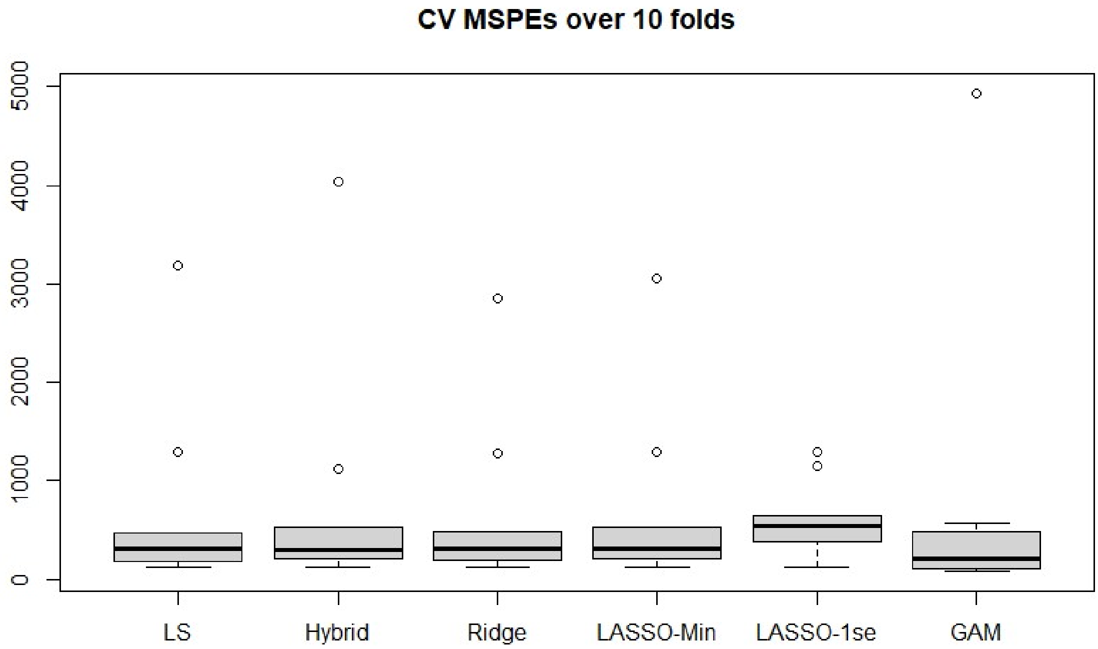
```r

# 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)
```

```r
  ### 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)

  ### 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

  ## GAM
  fit.gam = gam(Ozone ~ s(Solar.R) + s(Wind) + s(Temp) + s(TWcp) + s(TWrat),
                data = data.train)

  pred.gam = predict(fit.gam, data.valid)
  MSPE.gam = get.MSPE(Y.valid, pred.gam) # Our helper function
  all.MSPEs[i, "GAM"] = MSPE.gam
}

all.MSPEs

### 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"))
```

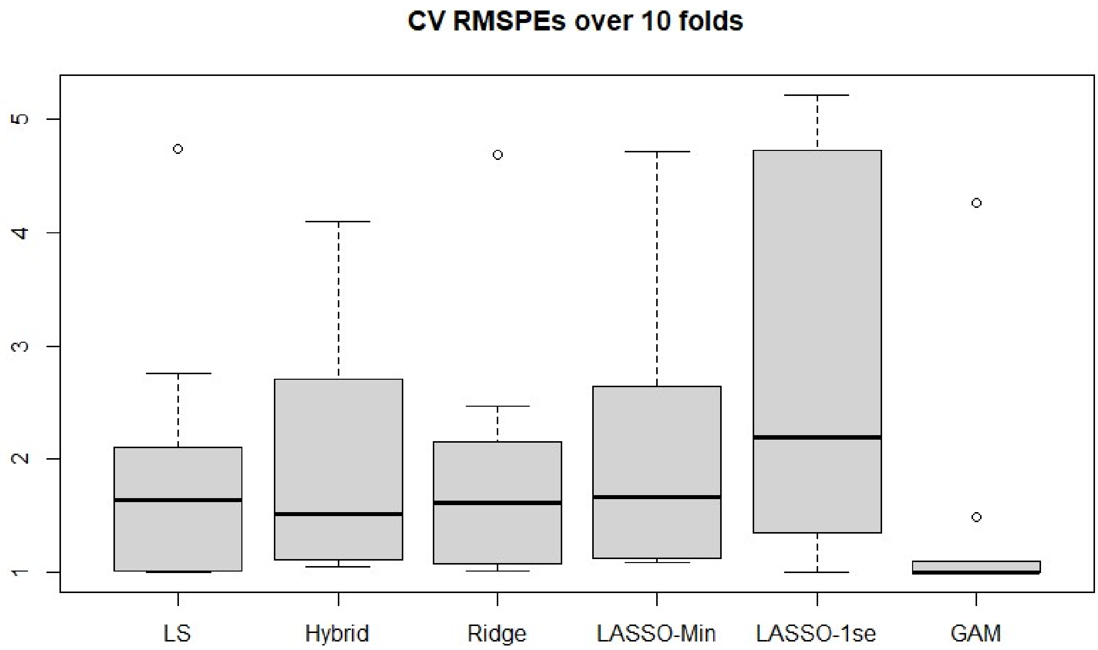
(a) **Report the separate MSPEs from each fold,** *MSPEv, v* = 1, . . . , 10 **and the MSPE for the full data.**

```
      GAM
4938.2488
  80.6323
 192.6688
 571.8521
 207.6464
 274.0896
  80.5867
 207.3574
 488.7937
 109.5156
```

(b) **Starting with boxplots the plots made earlier for least squares, hybrid stepwise, ridge, and LASSO, ADD a boxplot of the 10 CV error estimates for GAM as the last box on the right. Comment on how GAM compares to other methods**



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



(d) Using the knowledge gained from the analysis you did in Question 1, give a 1-sentence explanation for why GAM performs as it does. (If it is better than other methods, why? If it is no better than other methods, why?)

- ➔ Gam may not be the best model if the variables are interactive, however we include two interactions so it covers it.