2. Add three new models fitted to all 5 variables to the 10-fold CV comparison that has been used for LASSO, Ridge, and other methods: Full tree (CP=0), min-cv-pruned tree, and 1-se-pruned tree. Use the same folds as before.

```r
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)
}

# i=1
### 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", "Full-tree", "Min-cv tree", "1-se tree")
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]
```

```r
### 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


# Full-tree
fit.tree = rpart(Ozone ~ ., data = data, cp=0)
```

```r
fit.tree.pred = predict(fit.tree, data.valid)
MSPE.fit.tree = get.MSPE(Y.valid, fit.tree.pred)
all.MSPEs[i, "Full-tree"] = MSPE.fit.tree


# Min-cv tree
info.tree = fit.tree$cptable
info.tree
ind.min = which.min(info.tree[,"xerror"])
CP.min.raw = info.tree[ind.min, "CP"]
if(ind.min == 1){
  ### If minimum CP is in row 1, store this value
  CP.min = CP.min.raw
} else{
  ### If minimum CP is not in row 1, average this with the value from the
  ### row above it.

  ### Value from row above
  CP.above = info.tree[ind.min-1, "CP"]

  ### (Geometric) average
  CP.min = sqrt(CP.min.raw * CP.above)
}
fit.tree.min = prune(fit.tree, cp = CP.min)
fit.tree.min.pred = predict(fit.tree.min, data.valid)
MSPE.fit.tree.min = get.MSPE(Y.valid, fit.tree.min.pred)
all.MSPEs[i, "Min-cv tree"] = MSPE.fit.tree.min


#"1-se tree"
err.min = info.tree[ind.min, "xerror"]
se.min = info.tree[ind.min, "xstd"]
threshold = err.min + se.min
ind.1se = min(which(info.tree[1:ind.min,"xerror"] < threshold))

### Get the corresponding CP value, averaging if necessary
CP.1se.raw = info.tree[ind.1se, "xerror"]
if(ind.1se == 1){
  ### If best CP is in row 1, store this value
  CP.1se = CP.1se.raw
} else{
  ### If best CP is not in row 1, average this with the value from the
  ### row above it.
```

```r

    ### Value from row above
    CP.above = info.tree[ind.1se-1, "CP"]

    ### (Geometric) average
    CP.1se = sqrt(CP.1se.raw * CP.above)
  }

  fit.tree.1se = prune(fit.tree, cp = CP.1se)
  fit.tree.1se.pred = predict(fit.tree.1se, data.valid)
  MSPE.fit.tree.1se = get.MSPE(Y.valid, fit.tree.1se.pred)
  all.MSPEs[i, "1-se tree"] = MSPE.fit.tree.1se
}

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"))



### Calculate RMSPEs
all.RMSPEs = apply(all.MSPEs, 1, function(W){
  best = min(W)
  return(W / best)
})
all.RMSPEs = t(all.RMSPEs)

### Make a boxplot of RMSPEs
boxplot(all.RMSPEs, main = paste0("CV RMSPEs over ", K, " folds"))
```

```
> all.MSPEs
             LS   Hybrid     Ridge LASSO-Min LASSO-1se        GAM Full-tree Min-cv tree 1-se tree
 [1,] 1037.2843 935.5934 1026.2063 1031.8530 1081.5330  135.46137 521.42750   531.63564 1103.9618
 [2,]  419.5374 520.7976  462.0808  463.0394  426.9839 1166.67426 129.56077   122.76918  598.1703
 [3,]  561.5317 606.2170  569.6294  559.5286  427.4034  674.21375 209.19630   209.19630  284.9489
 [4,]  117.0370 140.0399  119.3903  119.7927  194.3578  140.95548  98.93844    98.93844  199.0212
 [5,]  200.1188 195.1213  201.8795  230.7676  285.8414   42.07896 115.41266   112.12023  458.6532
 [6,]  208.1940 180.5834  208.5333  209.9717  220.0076  297.97684 386.37579   386.37579  752.4572
 [7,]  507.6231 555.5441  515.8408  654.4310 1062.9010  231.24568 158.20856   163.34328  305.6825
 [8,]  151.2711 150.9220  149.6126  150.8763  319.8533  199.88708 234.91171   254.13553  336.9521
 [9,]  681.4810 731.9910  687.8407  749.9537  928.2086  854.82298 418.22831   421.05592  434.6720
[10,]  358.1365 256.2422  365.3990  398.0162  281.7134  262.95811 218.79173   218.79173  242.9717
```

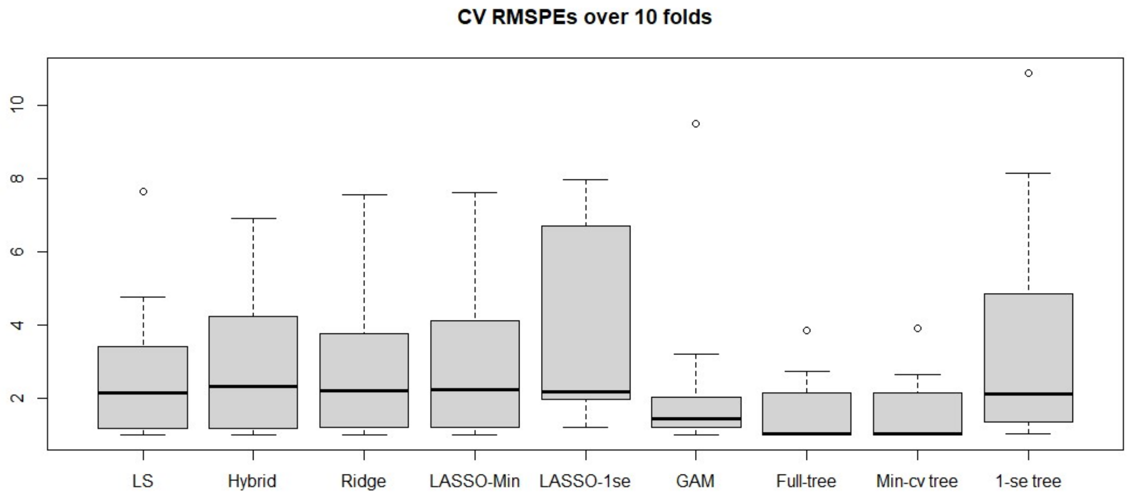
```r
mean(all.MSPEs[,7])
mean(all.MSPEs[,8])
mean(all.MSPEs[,9])
```

(a) **Compute the mean MSPE for each tree model and comment on the comparison.**

```
> mean(all.MSPEs[,7])
[1] 249.1052
> mean(all.MSPEs[,8])
[1] 251.8362
> mean(all.MSPEs[,9])
[1] 471.7491
```

Full-tree and Min-cv tree has similar small MSPEs and 1-se tree has quite larger MSPE.

(b) **ADD the three trees to the relative MSPE boxplots made previously. Comment on how well they perform compared to other methods.**



So far, it looks like Full-tree and Min-cv tree and GAM show the best result. However, 1-se tree doesn't do well on it.