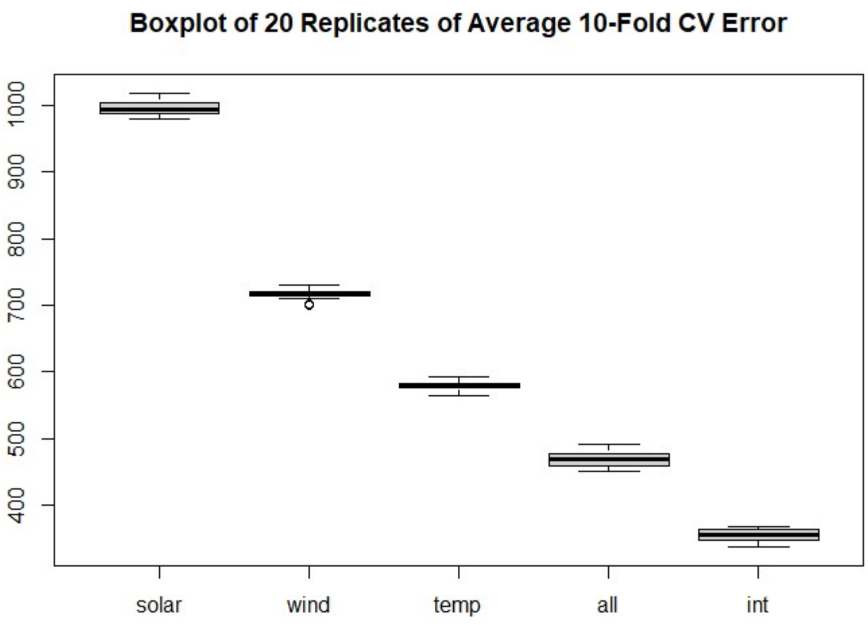
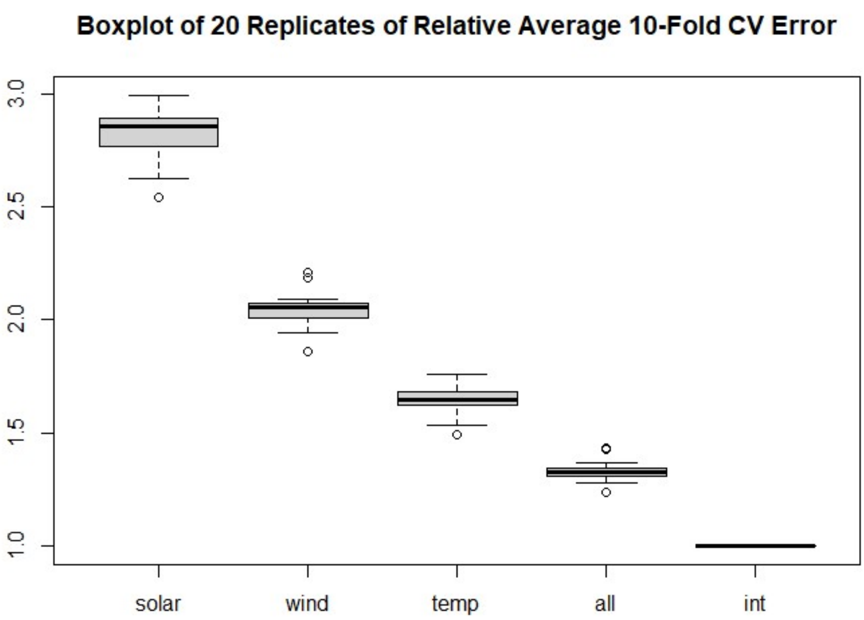
```r
#4.Now use 10-fold CV to estimate the MSPEs for the 5 models. Report the mean and
#95% confidence intervals for the 5 models
n.fold = n/10
n.fold = ceiling(n.fold)
ordered.ids = rep(1:10, times= n.fold)
ordered.ids = ordered.ids[1:n] # Remove excess labels(s)
shuffle = sample.int(n)
shuffled.ids = ordered.ids[shuffle]

data.CV = AQ
data.CV$fold = shuffled.ids

CV.MSPEs = array(0, dim= c(10, 5))
colnames(CV.MSPEs) = c("solar", "wind", "temp", "all", "int")

for(i in 1:10){
  #Use fold i for validation and the rest for training
  data.train = filter(data.CV, fold != i)
  data.valid = filter(data.CV, fold==i)

  #Remove fold from training and validation sets since it isn't a real predictor
  data.train = select(data.train, -fold)
  data.valid = select(data.valid, -fold)

  fit.solar = lm(Ozone ~ Solar.R, data = data.train)
  fit.wind = lm(Ozone ~ Wind, data = data.train)
  fit.temp = lm(Ozone ~ Temp, data = data.train)
  fit.all = lm(Ozone ~ Temp + Wind + Solar.R, data = data.train)
  fit.int = lm(Ozone ~ Temp + Wind + Solar.R + I(Temp^2) + I(Wind^2) + I(Solar.R^2) + Temp*Wind
               + Temp*Solar.R + Wind*Solar.R, data = data.train)

  pred.solar = predict(fit.solar, data.valid)
  pred.wind = predict(fit.wind, data.valid)
  pred.temp = predict(fit.temp, data.valid)
  pred.all = predict(fit.all, data.valid)
  pred.int = predict(fit.int, data.valid)

  Y.valid = data.valid$Ozone
  MSPE.solar = get.MSPE(Y.valid, pred.solar)
  MSPE.wind = get.MSPE(Y.valid, pred.wind)
  MSPE.temp = get.MSPE(Y.valid, pred.temp)
  MSPE.all = get.MSPE(Y.valid, pred.all)
  MSPE.int = get.MSPE(Y.valid, pred.int)

  CV.MSPEs[i, 1] = MSPE.solar
  CV.MSPEs[i, 2] = MSPE.wind
  CV.MSPEs[i, 3] = MSPE.temp
  CV.MSPEs[i, 4] = MSPE.all
  CV.MSPEs[i, 5] = MSPE.int

}

boxplot(CV.MSPEs,
        main = "Boxplot of CV Error With 10 Folds")

solar_cv = CV.MSPEs[, 1]
wind_cv = CV.MSPEs[, 2]
temp_cv = CV.MSPEs[, 3]
all_cv = CV.MSPEs[, 4]
int_cv = CV.MSPEs[, 5]

t.test(solar_cv)
#mean: 995.7716 95 percent confidence interval: 621.4722 1370.0710

t.test(wind_cv)
#mean: 732.6388  95 percent confidence interval: 494.8018 970.4759

t.test(temp_cv)
#mean: 583.9782   95 percent confidence interval: 192.4096 975.5469

t.test(all_cv)
#mean: 477.9924   95 percent confidence interval: 219.1922 736.7927

t.test(int_cv)
#mean: 379.6109   95 percent confidence interval: 171.3066 587.9151

#The model that allows curvature and interactions might be clearly good and solar model is clearly bad
```

```r
#5. Finally, repeat CV 20 times.
n.rep = 20 # Number of times to repeat CV/boostrap

### Start with CV. First, we need a container to store the average CV
### errors
ave.CV.MSPEs = array(0, dim = c(n.rep, 5))
colnames(ave.CV.MSPEs) = c("solar", "wind", "temp", "all", "int")

### We will put the entire CV section from above inside another
### for loop. This will repeat the entire CV process
### Note: we need to use a different loop variable for the outer
### for loop. It's common to use j when you have already used i
for (j in 1:n.rep) {
  n.fold = n / 10
  n.fold = ceiling(n.fold)

  ordered.ids = rep(1:10, times = n.fold)
  ordered.ids = ordered.ids[1:n]
  shuffle = sample.int(n)
  shuffled.ids = ordered.ids[shuffle]

  data.CV = AQ
  data.CV$fold = shuffled.ids

  CV.MSPEs = array(0, dim = c(10, 5))
  colnames(CV.MSPEs) = c("solar", "wind", "temp", "all", "int")

  for (i in 1:10) {
    data.train = filter(data.CV, fold != i)
    data.valid = filter(data.CV, fold == i)

    data.train = select(data.train, -fold)
    data.valid = select(data.valid, -fold)

    fit.solar = lm(Ozone ~ Solar.R, data = data.train)
    fit.wind = lm(Ozone ~ Wind, data = data.train)
    fit.temp = lm(Ozone ~ Temp, data = data.train)
    fit.all = lm(Ozone ~ Temp + Wind + Solar.R, data = data.train)
    fit.int = lm(Ozone ~ Temp + Wind + Solar.R + I(Temp^2) + I(Wind^2) + I(Solar.R^2)
                 + Temp*Wind + Temp*Solar.R + Wind*Solar.R, data = data.train)

    pred.solar = predict(fit.solar, data.valid)
    pred.wind = predict(fit.wind, data.valid)
    pred.temp = predict(fit.temp, data.valid)
    pred.all = predict(fit.all, data.valid)
    pred.int = predict(fit.int, data.valid)

    Y.valid = data.valid$Ozone
    MSPE.solar = get.MSPE(Y.valid, pred.solar)
    MSPE.wind = get.MSPE(Y.valid, pred.wind)
    MSPE.temp = get.MSPE(Y.valid, pred.temp)
    MSPE.all = get.MSPE(Y.valid, pred.all)
    MSPE.int = get.MSPE(Y.valid, pred.int)

    CV.MSPEs[i, 1] = MSPE.solar
    CV.MSPEs[i, 2] = MSPE.wind
    CV.MSPEs[i, 3] = MSPE.temp
    CV.MSPEs[i, 4] = MSPE.all
    CV.MSPEs[i, 5] = MSPE.int
  }

  ### We now have MSPEs for each fold of one iteration of CV. Let's
  ### get the average error across these folds (think of each fold
  ### as a data split), and store the result in ave.CV.MSPEs
  this.ave.MSPEs = apply(CV.MSPEs, 2, mean)
  ave.CV.MSPEs[j,] = this.ave.MSPEs # We are replacing a whole
  # row at once
}

boxplot(ave.CV.MSPEs,
        main = "Boxplot of 20 Replicates of Average 10-Fold CV Error")
```



The boxplot is showing that clearly the model that allows curvature and interactions is better than others and the model that only using solar variable is worse.

```
#(b) Repeat for RMSPE, and narrow focus if necessary to see best models better. Which model(s) give the best results most often?
rel.ave.CV.MSPEs = apply(ave.CV.MSPEs, 1, function(W){
  best = min(W)
  return(W / best)
})
rel.ave.CV.MSPEs = t(rel.ave.CV.MSPEs)

boxplot(rel.ave.CV.MSPEs,
        main = "Boxplot of 20 Replicates of Relative Average 10-Fold CV Error")
```



The model that allows curvature and interactions shows best result most often.

6. Based on what you've done, and considering practical concerns described in the preamble for the problem, which one model would you suggest using?

-> I would use the model that allows curvature and interactions.