

```

#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

```

```

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

### 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")

```

A box plot illustrating the distribution of the number of features for five different feature sets: 'solar', 'wind', 'temp', 'all', and 'int'. The y-axis represents the number of features, ranging from 400 to 1000. The 'solar' set has the highest number of features, with a median around 1000. The 'wind' set has a median around 710 and a notable outlier at approximately 700. The 'temp' set has a median around 580. The 'all' set has a median around 470. The 'int' set has the lowest number of features, with a median around 350. The plot shows that the number of features generally decreases as the feature set becomes more specific or filtered.

Feature Set	Min	Q1	Median	Q3	Max	Outliers
solar	980	990	1000	1000	1020	None
wind	700	705	710	715	725	700
temp	560	570	580	590	600	None
all	450	460	470	480	490	None
int	340	350	355	360	370	None

```
#(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")
```

A box plot showing the distribution of the number of species per site for five categories: solar, wind, temp, all, and int. The y-axis represents the number of species, ranging from 1.0 to 3.0. The 'solar' category has the highest median (approx. 2.8) and a single outlier at 2.5. The 'wind' category has a median of approx. 2.05 and two outliers at 1.85 and 2.2. The 'temp' category has a median of approx. 1.65 and one outlier at 1.5. The 'all' category has a median of approx. 1.35 and two outliers at 1.25 and 1.45. The 'int' category has a single data point at 1.0.

Category	Min	Q1	Median	Q3	Max	Outliers
solar	2.6	2.75	2.8	2.9	3.0	2.5
wind	1.95	2.0	2.05	2.05	2.1	1.85, 2.2
temp	1.55	1.6	1.65	1.65	1.75	1.5
all	1.3	1.3	1.35	1.35	1.4	1.25, 1.45
int	1.0	1.0	1.0	1.0	1.0	

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.