**Application**

1. First, in order to use this set for modeling, we need to deal with its missing data. Since this is not a class on imputation, we deal with them in the easiest but most biased way: case deletion. When you have missing data in real life, *learn how to deal with them better than this!!!* **Run the code below to clean the data and report the results.**

AQ = na . omit ( a i r q u a l i t y [ , 1 : 4 ] )
dim(AQ)

```
#1
AQ = na.omit(airquality[, 1:4])
dim(AQ)
# 111 4 which means that there are 111 rows and 4 columns
```

```r
#2
set.seed(4099183)
n = nrow(AQ)
reorder = sample.int(n)

size_train = floor(n*0.75)
ind_train = reorder[1:size_train]
ind_valid = reorder[(size_train+1):n]

data_train = AQ[ind_train, ]
data_valid = AQ[ind_valid, ]

#print(data_valid)
# Ozone Solar.R Wind Temp
# 94       9      24 13.8   81
# 124     96     167  6.9   91
# 99     122     255  4.0   89
# 140     18     224 13.8   67
# 24      32      92 12.0   61
# 17      34     307 12.0   66
# 30     115     223  5.7   79
# 20      11      44  9.7   62
# 106     65     157  9.7   80
# 63      49     248  9.2   85
# 31      37     279  7.4   76
# 82      16       7  6.9   74
# 146     36     139 10.3   81
# 4       18     313 11.5   62
# 116     45     212  9.7   79
# 14      14     274 10.9   68
# 110     23     115  7.4   76
# 29      45     252 14.9   81
# 44      23     148  8.0   82
# 120     76     203  9.7   97
# 8       19      99 13.8   59
# 66      64     175  4.6   83
# 109     59      51  6.3   79
# 113     21     259 15.5   77
# 22      11     320 16.6   73
# 38      29     127  9.7   82
# 92      59     254  9.2   81
# 149     30     193  6.9
```

```r
#3
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)

get.MSPE = function(Y, Y.hat){
  residuals = Y-Y.hat
  resid.sq = residuals^2
  SSPE = sum(resid.sq)
  MSPE = SSPE / length(Y)
  return (MSPE)
}

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)

print(MSPE.solar)
print(MSPE.wind)
print(MSPE.temp)
print(MSPE.all)
print(MSPE.int)
```

```
# > print(MSPE.solar)
# [1] 934.5455
# > print(MSPE.wind)
# [1] 574.8944
# > print(MSPE.temp)
# [1] 586.7952
# > print(MSPE.all)
# [1] 387.3352
# > print(MSPE.int)
# [1] 327.3287
# The model that allows curvature and interactions wins this competition
```