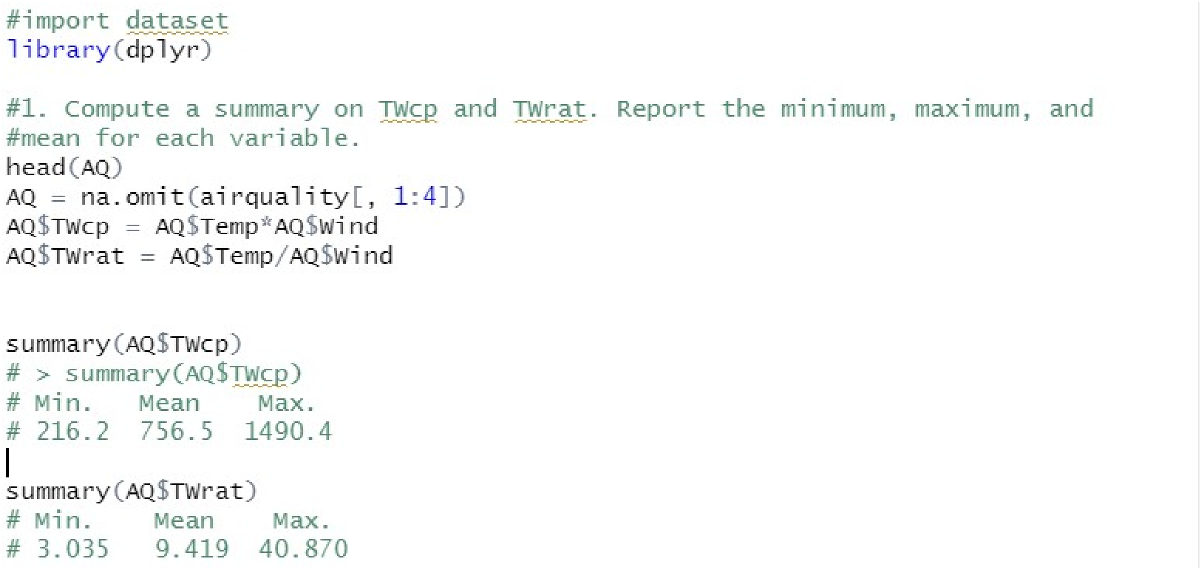
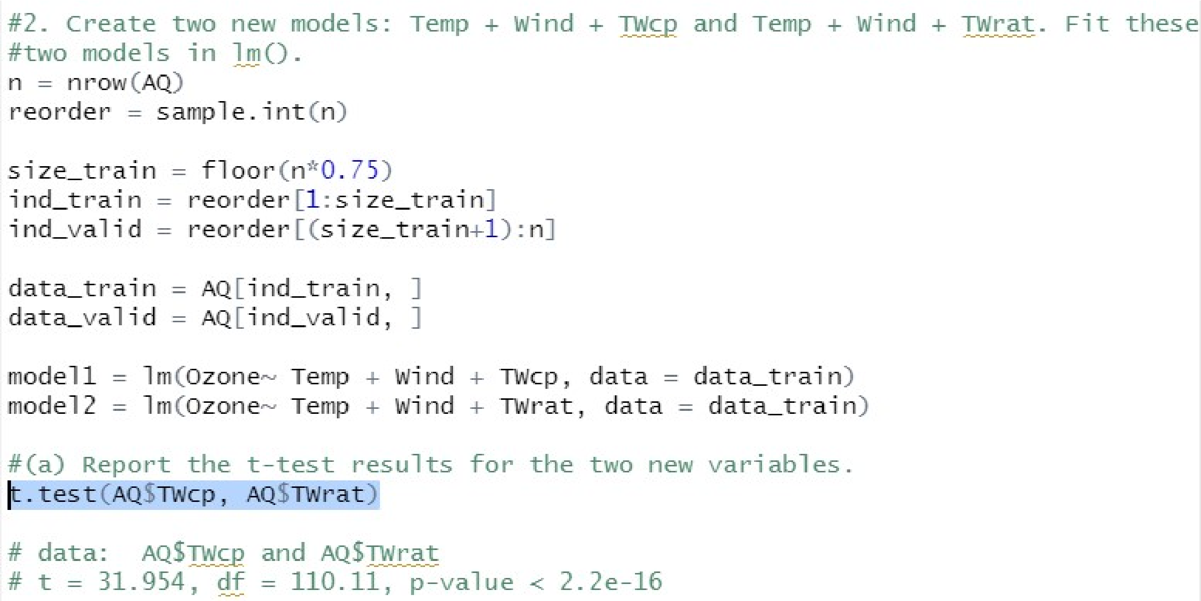
1. Compute a summary on TWcp and TWrat. **Report the minimum, maximum, and mean for each variable**



```
#import dataset
library(dplyr)

#1. Compute a summary on TWcp and TWrat. Report the minimum, maximum, and
#mean for each variable.
head(AQ)
AQ = na.omit(airquality[, 1:4])
AQ$TWcp = AQ$Temp*AQ$Wind
AQ$TWrat = AQ$Temp/AQ$Wind


summary(AQ$TWcp)
# > summary(AQ$TWcp)
# Min.   Mean    Max.
# 216.2  756.5  1490.4

summary(AQ$TWrat)
# Min.    Mean    Max.
# 3.035   9.419  40.870
```

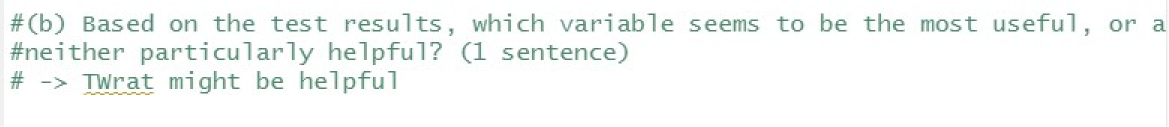
2. Create two new models: Temp + Wind + TWcp and Temp + Wind + TWrat. Fit these two models in lm().

(a) **Report the t-test results for the two new variables**.



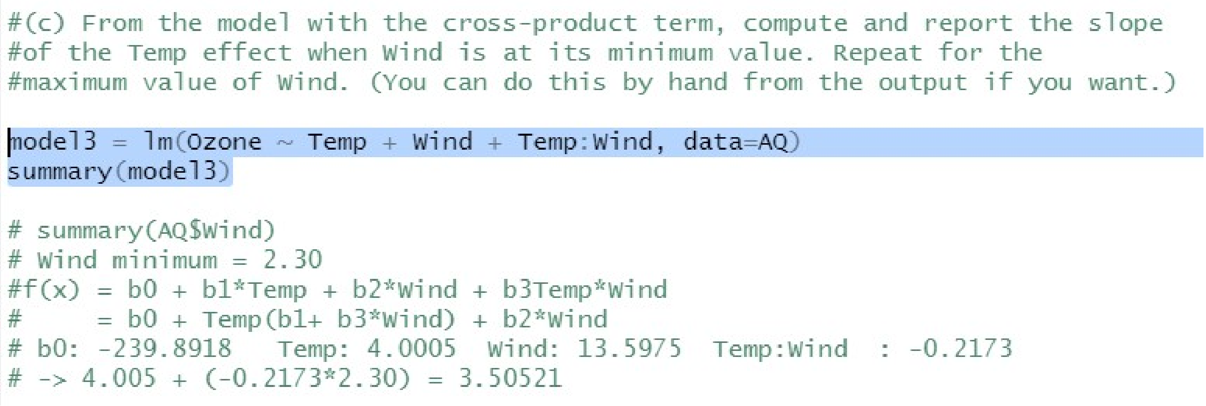
```
#2. Create two new models: Temp + Wind + TWcp and Temp + Wind + TWrat. Fit these
#two models in lm().
n = nrow(AQ)
reorder = sample.int(n)

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, ]

model1 = lm(Ozone~ Temp + Wind + TWcp, data = data_train)
model2 = lm(Ozone~ Temp + Wind + TWrat, data = data_train)

#(a) Report the t-test results for the two new variables.
t.test(AQ$TWcp, AQ$TWrat)

# data:  AQ$TWcp and AQ$TWrat
# t = 31.954, df = 110.11, p-value < 2.2e-16
```

(b) Based on the test results, which variable seems to be the most useful, or are neither particularly helpful? **(1 sentence)**



```
#(b) Based on the test results, which variable seems to be the most useful, or are
#neither particularly helpful? (1 sentence)
# -> TWrat might be helpful
```

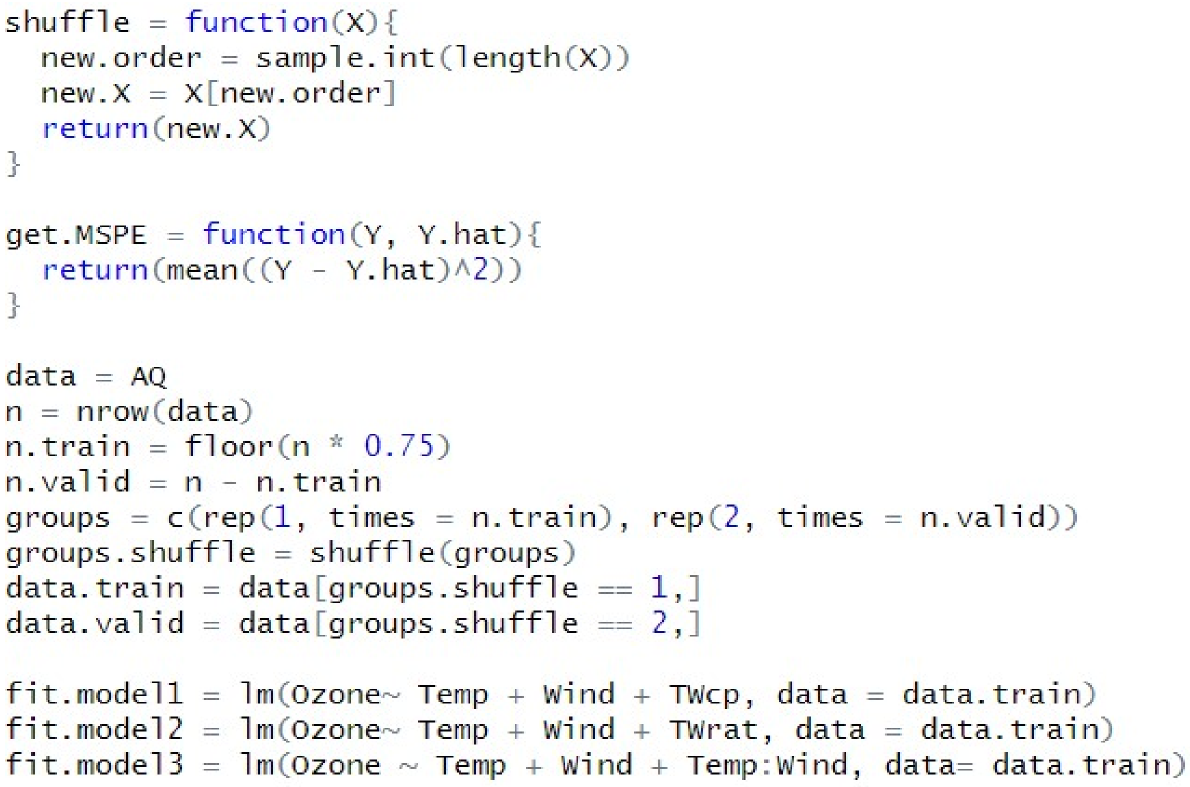
(c) From the model with the cross-product term, compute and **report the slope of the Temp effect when Wind is at its minimum value. Repeat for the maximum value of Wind.** (You can do this by hand from the output if you want.)



```
#(c) From the model with the cross-product term, compute and report the slope
#of the Temp effect when Wind is at its minimum value. Repeat for the
#maximum value of Wind. (You can do this by hand from the output if you want.)

model3 = lm(Ozone ~ Temp + Wind + Temp:Wind, data=AQ)
summary(model3)

# summary(AQ$Wind)
# Wind minimum = 2.30
#f(x) = b0 + b1*Temp + b2*Wind + b3Temp*Wind
#      = b0 + Temp(b1+ b3*Wind) + b2*Wind
# b0: -239.8918   Temp: 4.0005  Wind: 13.5975  Temp:Wind  : -0.2173
# -> 4.005 + (-0.2173*2.30) = 3.50521
```

3. Fit each model on the training data and **report the MSPEs from the validation data**.

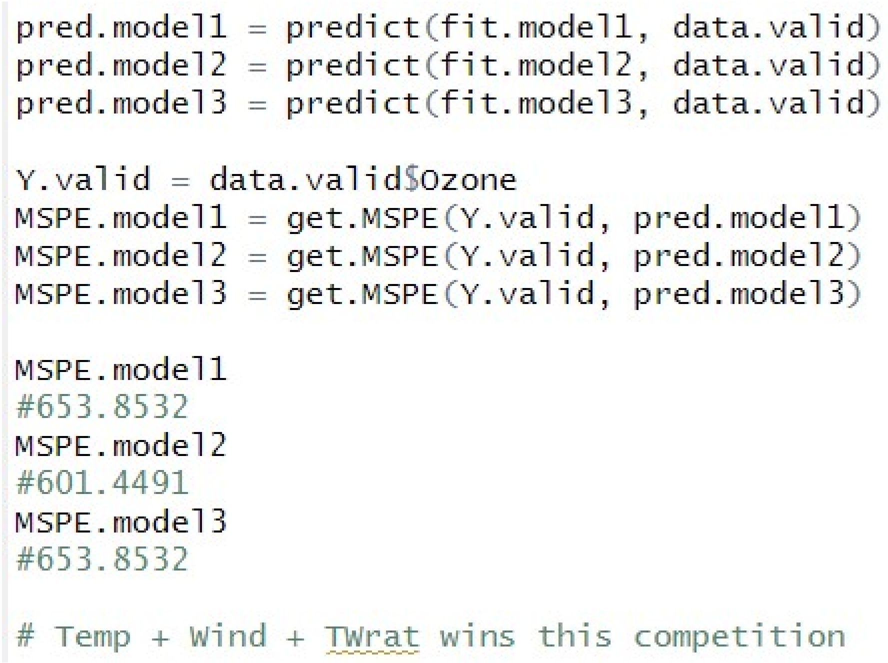
(a) **Which model wins this competition?**



```
shuffle = function(X){
  new.order = sample.int(length(X))
  new.X = X[new.order]
  return(new.X)
}

get.MSPE = function(Y, Y.hat){
  return(mean((Y - Y.hat)^2))
}

data = AQ
n = nrow(data)
n.train = floor(n * 0.75)
n.valid = n - n.train
groups = c(rep(1, times = n.train), rep(2, times = n.valid))
groups.shuffle = shuffle(groups)
data.train = data[groups.shuffle == 1,]
data.valid = data[groups.shuffle == 2,]

fit.model1 = lm(Ozone~ Temp + Wind + TWcp, data = data.train)
fit.model2 = lm(Ozone~ Temp + Wind + TWrat, data = data.train)
fit.model3 = lm(Ozone ~ Temp + Wind + Temp:Wind, data= data.train)
```

(b)

(c)



```
pred.model1 = predict(fit.model1, data.valid)
pred.model2 = predict(fit.model2, data.valid)
pred.model3 = predict(fit.model3, data.valid)

Y.valid = data.valid$Ozone
MSPE.model1 = get.MSPE(Y.valid, pred.model1)
MSPE.model2 = get.MSPE(Y.valid, pred.model2)
MSPE.model3 = get.MSPE(Y.valid, pred.model3)

MSPE.model1
#653.8532
MSPE.model2
#601.4491
MSPE.model3
#653.8532

# Temp + Wind + TWrat wins this competition
```