Lecture 5: Application

Yufei Yin

Application

1.

```
library(leaps)
```

```
data("airquality")
AQ = na.omit(airquality[,1:4])
AQ$TWcp = with(AQ, Temp * Wind)
AQ$TWrat = with(AQ, Temp / Wind)
```

(a)

```
data.matrix = model.matrix(Ozone ~ ., data = AQ)

### We also need the response variable
Y = AQ$Ozone

fit.subsets = regsubsets(x = data.matrix, y = Y,
  intercept = F)
info.subsets = summary(fit.subsets)
seq.subsets = info.subsets$which
vars.seq.subsets.raw = apply(seq.subsets, 1, function(W){
  vars.list = names(W)[W]
  out = paste0(vars.list, collapse = ", ")
})

print(vars.seq.subsets.raw)
```

```
##                                             1
##                                        "TWrat"
##                                             2
##                                 "Solar.R, TWrat"
##                                             3
##                       "(Intercept), Temp, TWrat"
##                                             4
##             "(Intercept), Solar.R, Temp, TWrat"
##                                             5
##        "(Intercept), Solar.R, Wind, Temp, TWcp"
##                                             6
## "(Intercept), Solar.R, Wind, Temp, TWcp, TWrat"
```

The output shows the variables in the best model of each size.

(b)

```
info.subsets$bic
```

```
## [1] -185.2244 -189.0768 -204.1878 -207.1195 -204.6274 -202.8590
```

These are the BIC values of these best models in question(a), from variable size 1 to 6 respectively.

(c)

```
which.min(info.subsets$bic)
```

```
## [1] 4
```

The smallest BIC value gives the best model, so the best model have 4 variables. These 4 variables are (Intercept), Solar.R, Temp, and TWrat.

2.

```
n = nrow(AQ)

fit.start = lm(Ozone ~ 1, data = AQ)
fit.end = lm(Ozone ~ ., data = AQ)

step.BIC = step(fit.start, list(upper = fit.end),
                direction = "both",
                k = log(n),
                trace = 0)
summary(step.BIC)
```

```
##
## Call:
## lm(formula = Ozone ~ TWrat + Temp + Solar.R, data = AQ)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -56.168 -12.102  -4.424  11.403  77.471
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -93.30421   17.28283  -5.399 4.08e-07 ***
## TWrat         2.86326    0.42026   6.813 5.82e-10 ***
## Temp          1.25231    0.25551   4.901 3.41e-06 ***
## Solar.R       0.05960    0.02158   2.761  0.00678 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.72 on 107 degrees of freedom
## Multiple R-squared:  0.6585, Adjusted R-squared:  0.6489
## F-statistic: 68.77 on 3 and 107 DF,  p-value: < 2.2e-16
```

hybrid stepwise algorithm selected variables temperature, solar radiation, and the ratio of temperature over wind speed (and also the intercept, but this shouldn't have been included in the selection process).

3.

```
### Create function to compute MSPEs
get.MSPE = function(Y, Y.hat){
  return(mean((Y - Y.hat)^2))
}

### Create function which constructs folds for CV
### n is the number of observations, K is the number of folds
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)
  fold.ids = fold.ids.raw[1:n]

  ### Shuffle the fold labels
  folds.rand = fold.ids[sample.int(n)]

  return(folds.rand)
}
```

```
K = 10 #Number of folds

set.seed(2928893)

### Container for CV MSPEs
CV.MSPEs = array(0, dim = c(1, K))
rownames(CV.MSPEs) = "Step"
colnames(CV.MSPEs) = 1:K

### Get CV fold labels
n = nrow(AQ)
folds = get.folds(n, K)
```

```
### Perform cross-validation
for (i in 1:K) {
  ### Get training and validation sets
  data.train = AQ[folds != i, ]
  data.valid = AQ[folds == i, ]
  Y.train = data.train$Ozone
  Y.valid = data.valid$Ozone

  #############
  ### Step ###
  #############
  fit.start = lm(Ozone ~ 1, data = data.train)
  fit.end = lm(Ozone ~ ., data = data.train)

  fit.step = step(fit.start, list(upper = fit.end), trace = 0)

  pred.step = predict(fit.step, data.valid)
  MSPE.step = get.MSPE(Y.valid, pred.step)

  CV.MSPEs["Step", i] = MSPE.step
}
```

```r
### Get full-data MSPEs
full.MSPEs = mean(CV.MSPEs)

### Combine and print foldwise/full MSPEs
MSPEs = cbind(CV.MSPEs, full.MSPEs)
colnames(MSPEs) = c(1:K, "Full")
print(signif(MSPEs, 3))
```

```
##        1   2   3   4   5   6   7   8    9  10 Full
## Step 328 404 577 107 178 369 233 635 1070 787  469
```