HW6

```r
#1a
set.seed(3301)
dat1 = read.table("trees.txt")


dat= dat1[sample(1:nrow(dat1)),]
dat
```

```
##       D  H    V
## 10 11.2 75 19.9
## 24 16.0 72 38.3
## 9  11.1 80 22.6
## 25 16.3 77 42.6
## 14 11.7 69 21.3
## 29 18.0 80 51.5
## 16 12.9 74 22.2
## 3   8.8 63 10.2
## 22 14.2 80 31.7
## 26 17.3 81 55.4
## 1   8.3 70 10.3
## 23 14.5 74 36.3
## 2   8.6 65 10.3
## 21 14.0 78 34.5
## 20 13.8 64 24.9
## 5  10.7 81 18.8
## 8  11.0 75 18.2
## 11 11.3 79 24.2
## 31 20.6 87 77.0
## 18 13.3 86 27.4
## 4  10.5 72 16.4
## 28 17.9 80 58.3
## 12 11.4 76 21.0
## 19 13.7 71 25.7
## 13 11.4 76 21.4
## 17 12.9 85 33.8
## 6  10.8 83 19.7
## 27 17.5 82 55.7
## 30 18.0 80 51.0
## 7  11.0 66 15.6
## 15 12.0 75 19.1
```

```r
X1 = cbind(1, dat$D,(dat$D)^2, dat$D*dat$H, dat$H, (dat$H)^2)
y1 = log(dat$V)


olscv=function(X, y, K=5, permute=FALSE)
{
```

```r
n=length(y)
if(permute)
{
ind=sample(n)
} else
{
ind=1:n
}
total.sq.err=0
for(k in 1:K)
{
leave.out=ind[ (1+floor((k-1)*n/K)):floor(k*n/K) ]
X.tr=X[-leave.out,,drop=FALSE]
y.tr=y[-leave.out]
X.va=X[leave.out,,drop=FALSE]
y.va=y[leave.out]
bhat.tr=lm.fit(x=X.tr, y=y.tr)$coefficients
total.sq.err=total.sq.err + sum((y.va - X.va%*%bhat.tr)^2 )
}
return(total.sq.err/n)
}
get.ic=function(X, y, K=5)
{
n=dim(X)[1]
p=dim(X)[2]
beta.hat=lm.fit(x=X, y=y)$coefficients
rss=sum((y-X%*%beta.hat)^2)
common=n*log(2*pi)+n*log(rss/n) + n
aic=common + 2*(p+1)
bic=common + (p+1)*log(n)
est.mspe=olscv(X=X,y=y, K=K)
return(c(aic, bic, est.mspe))
}
keep.status=as.matrix(expand.grid( replicate(5, c(0,1), simplify=FALSE) ))
scores=matrix(NA, nrow=nrow(keep.status), ncol=3)
for(j in 1:nrow(keep.status))
{
keep=as.logical(c(1, keep.status[j,]))
scores[j,] = get.ic(X=X1[,keep, drop=FALSE], y=y1, K=5)
}
result=cbind(scores, keep.status)
colnames(result)=c("AIC", "BIC", "Est. MSPE", "D", "D^2", "D*H", "H", "H^2")
result
```

```
##              AIC       BIC   Est. MSPE D D^2 D*H H H^2
##  [1,]  51.15694  54.02492 0.283139819 0   0   0 0   0
##  [2,] -33.90673 -29.60477 0.019028369 1   0   0 0   0
##  [3,] -18.80140 -14.49943 0.032889688 0   1   0 0   0
##  [4,] -38.88103 -33.14508 0.015980573 1   1   0 0   0
##  [5,] -38.92627 -34.62431 0.018611391 0   0   1 0   0
##  [6,] -43.10662 -37.37067 0.016620345 1   0   1 0   0
##  [7,] -37.12530 -31.38935 0.019329150 0   1   1 0   0
##  [8,] -59.96195 -52.79202 0.007443704 1   1   1 0   0
##  [9,]  36.25653  40.55849 0.171288264 0   0   0 1   0
```

```
## [10,] -51.98466 -46.24871 0.011119744 1   0   0 1   0
## [11,] -31.42541 -25.68946 0.023214911 0   1   0 1   0
## [12,] -61.31366 -54.14373 0.006938839 1   1   0 1   0
## [13,] -38.02440 -32.28845 0.018886283 0   0   1 1   0
## [14,] -59.59979 -52.42985 0.007137994 1   0   1 1   0
## [15,] -42.02996 -34.86003 0.015004631 0   1   1 1   0
## [16,] -59.46716 -50.86324 0.007396804 1   1   1 1   0
## [17,]  36.38881  40.69077 0.171552384 0   0   0 0   1
## [18,] -50.86370 -45.12775 0.011854996 1   0   0 0   1
## [19,] -30.34839 -24.61244 0.024502576 0   1   0 0   1
## [20,] -60.82499 -53.65505 0.007112102 1   1   0 0   1
## [21,] -38.38371 -32.64776 0.018856739 0   0   1 0   1
## [22,] -58.40831 -51.23838 0.007673977 1   0   1 0   1
## [23,] -45.57566 -38.40573 0.013220505 0   1   1 0   1
## [24,] -58.86482 -50.26090 0.007337706 1   1   1 0   1
## [25,]  38.22342  43.95937 0.184654972 0   0   0 1   1
## [26,] -53.42026 -46.25033 0.011878035 1   0   0 1   1
## [27,] -34.78626 -27.61633 0.022726751 0   1   0 1   1
## [28,] -59.74039 -51.13647 0.008193491 1   1   0 1   1
## [29,] -42.46605 -35.29612 0.018378160 0   0   1 1   1
## [30,] -57.67335 -49.06943 0.008251693 1   0   1 1   1
## [31,] -46.99104 -38.38711 0.014417017 0   1   1 1   1
## [32,] -57.75516 -47.71725 0.008321643 1   1   1 1   1
```

-61.31366 -54.14373 0.006938839 1 1 0 1 0 subset("D","D^2",H)

```
#2a
df1 = read.table("divorce.txt")
dfs= df1[sample(1:nrow(df1)),]
X =cbind(1, dfs$year, dfs$unemployed, dfs$femlab, dfs$marriage, dfs$birth, dfs$military)
y=dfs$divorce
keep.status=as.matrix(expand.grid(replicate(6, c(0,1), simplify=FALSE) ))
scores=matrix(NA, nrow=nrow(keep.status), ncol=3)
for(j in 1:nrow(keep.status))
{
keep=as.logical(c(1, keep.status[j,]))
scores[j,] = get.ic(X=X[,keep,drop=FALSE], y=y, K=5)
}
result=cbind(scores, keep.status)
colnames(result)=c("AIC", "BIC", "Est. MSPE", "year", "unemployed", "femlab", "marriage", "birth", "mil
result
```

```
##            AIC      BIC Est. MSPE year unemployed femlab marriage birth
##  [1,] 488.7042 493.3918 32.683423    0          0      0        0     0
##  [2,] 376.5083 383.5397  7.301734    1          0      0        0     0
##  [3,] 487.2109 494.2423 31.469088    0          1      0        0     0
##  [4,] 378.5012 387.8764  7.317050    1          1      0        0     0
##  [5,] 354.7949 361.8263  5.505102    0          0      1        0     0
##  [6,] 351.1220 360.4972  5.189291    1          0      1        0     0
##  [7,] 356.5218 365.8970  5.507413    0          1      1        0     0
##  [8,] 352.4142 364.1332  5.206033    1          1      1        0     0
##  [9,] 464.8266 471.8580 25.301632    0          0      0        1     0
## [10,] 378.4685 387.8437  7.545192    1          0      0        1     0
## [11,] 450.5445 459.9198 20.511985    0          1      0        1     0
## [12,] 380.4676 392.1867  7.708269    1          1      0        1     0
```

```
## [13,] 354.2976 363.6729  5.605174     0           0       1         1      0
## [14,] 348.6569 360.3759  5.264765     1           0       1         1      0
## [15,] 352.8331 364.5521  5.425902     0           1       1         1      0
## [16,] 342.0435 356.1063  4.786245     1           1       1         1      0
## [17,] 433.9999 441.0313 15.913895     0           0       0         0      1
## [18,] 350.4289 359.8041  5.300500     1           0       0         0      1
## [19,] 390.9631 400.3383  8.875145     0           1       0         0      1
## [20,] 333.3542 345.0732  4.120880     1           1       0         0      1
## [21,] 332.3504 341.7256  4.178502     0           0       1         0      1
## [22,] 330.3148 342.0338  4.036506     1           0       1         0      1
## [23,] 322.1993 333.9183  3.587583     0           1       1         0      1
## [24,] 322.6855 336.7483  3.627496     1           1       1         0      1
## [25,] 435.3220 444.6972 17.112431     0           0       0         1      1
## [26,] 338.3458 350.0648  4.724621     1           0       0         1      1
## [27,] 389.7720 401.4910  9.375392     0           1       0         1      1
## [28,] 328.6932 342.7560  4.144707     1           1       0         1      1
## [29,] 305.7125 317.4315  3.144683     0           0       1         1      1
## [30,] 297.2075 311.2704  2.860362     1           0       1         1      1
## [31,] 305.1557 319.2186  3.080686     0           1       1         1      1
## [32,] 298.9332 315.3399  2.883705     1           1       1         1      1
## [33,] 490.6776 497.7090 32.940527     0           0       0         0      0
## [34,] 378.4579 387.8331  7.406861     1           0       0         0      0
## [35,] 488.7953 498.1705 31.613396     0           1       0         0      0
## [36,] 380.4578 392.1768  7.468437     1           1       0         0      0
## [37,] 356.4381 365.8133  5.593542     0           0       1         0      0
## [38,] 351.4301 363.1492  5.292228     1           0       1         0      0
## [39,] 358.3422 370.0613  5.646839     0           1       1         0      0
## [40,] 353.2910 367.3538  5.365377     1           1       1         0      0
## [41,] 463.9449 473.3201 25.097068     0           0       0         1      0
## [42,] 380.4404 392.1594  7.756714     1           0       0         1      0
## [43,] 452.3988 464.1179 21.152696     0           1       0         1      0
## [44,] 382.4339 396.4967  7.889235     1           1       0         1      0
## [45,] 354.5405 366.2596  5.674388     0           0       1         1      0
## [46,] 343.7198 357.7826  5.086314     1           0       1         1      0
## [47,] 353.9535 368.0163  5.534376     0           1       1         1      0
## [48,] 338.5314 354.9380  4.679825     1           1       1         1      0
## [49,] 433.5884 442.9636 15.829247     0           0       0         0      1
## [50,] 350.6562 362.3752  5.204449     1           0       0         0      1
## [51,] 392.0134 403.7324  8.981115     0           1       0         0      1
## [52,] 335.2648 349.3276  4.200108     1           1       0         0      1
## [53,] 334.1290 345.8481  4.240597     0           0       1         0      1
## [54,] 332.2960 346.3588  4.189880     1           0       1         0      1
## [55,] 323.5488 337.6116  3.670322     0           1       1         0      1
## [56,] 323.4006 339.8073  3.715725     1           1       1         0      1
## [57,] 434.0992 445.8183 17.114144     0           0       0         1      1
## [58,] 340.2344 354.2973  4.790949     1           0       0         1      1
## [59,] 391.3723 405.4351  9.576873     0           1       0         1      1
## [60,] 330.0888 346.4954  4.180049     1           1       0         1      1
## [61,] 305.6053 319.6682  3.122805     0           0       1         1      1
## [62,] 289.8468 306.2534  2.599032     1           0       1         1      1
## [63,] 303.4288 319.8355  3.010537     0           1       1         1      1
## [64,] 290.9269 309.6773  2.614563     1           1       1         1      1
##       military
##  [1,]        0
```

```
##  [2,]         0
##  [3,]         0
##  [4,]         0
##  [5,]         0
##  [6,]         0
##  [7,]         0
##  [8,]         0
##  [9,]         0
## [10,]         0
## [11,]         0
## [12,]         0
## [13,]         0
## [14,]         0
## [15,]         0
## [16,]         0
## [17,]         0
## [18,]         0
## [19,]         0
## [20,]         0
## [21,]         0
## [22,]         0
## [23,]         0
## [24,]         0
## [25,]         0
## [26,]         0
## [27,]         0
## [28,]         0
## [29,]         0
## [30,]         0
## [31,]         0
## [32,]         0
## [33,]         1
## [34,]         1
## [35,]         1
## [36,]         1
## [37,]         1
## [38,]         1
## [39,]         1
## [40,]         1
## [41,]         1
## [42,]         1
## [43,]         1
## [44,]         1
## [45,]         1
## [46,]         1
## [47,]         1
## [48,]         1
## [49,]         1
## [50,]         1
## [51,]         1
## [52,]         1
## [53,]         1
## [54,]         1
## [55,]         1
```

```
## [56,]        1
## [57,]        1
## [58,]        1
## [59,]        1
## [60,]        1
## [61,]        1
## [62,]        1
## [63,]        1
## [64,]        1
```

subset(year, femlab, marriage, birth, military) 289.8468 306.2534 2.496206 1 0 1 1 1 1

```
#2b
#model with predictors that generated the lowest AIC, BIC, MSPE
train=1:70
test= 71:77
tdata= df1[sample(1:70),]
ttdata = df1[sample(test),]
y=tdata$divorce
#{year, unemployed, femlab, marriage, birth, military}
X = cbind(1, tdata$year, tdata$unemployed, tdata$femlab, tdata$marriage, tdata$birth, tdata$military)
X1 =cbind(1, ttdata$year, ttdata$unemployed, ttdata$femlab, ttdata$marriage,
          ttdata$birth, ttdata$military)
y1= ttdata$divorce
get.ic(X=X[train,], y=y[train], K=5)
```

```
## [1] 262.112699 280.100661   2.875636
```

```
#{year, femlab, marriage, birth, military}
X2 = cbind(1, tdata$year, tdata$femlab, tdata$marriage, tdata$birth, tdata$military)
X3 =cbind(1, ttdata$year, ttdata$femlab, ttdata$marriage,
          ttdata$birth, ttdata$military)
get.ic(X=X2[train,], y=y[train], K=5)
```

```
## [1] 261.231001 276.970468   2.900803
```

According to the output the subset of predictors with the best output are {year, femlab, marriage, birth, military}

```
#2c
#FULL model
beta.hat1=qr.coef(qr(X[train,]), y=y[train])
beta.hat1
```

```
## [1] 428.16156307  -0.22860248  -0.05325023   0.87878246   0.13802516
## [6]  -0.11343592  -0.04909694
```

```
fitted1 = X1[1:7,]%*%beta.hat1 #X1 is the test set
mean((y1[1:7] - fitted1)^2)
```

```
## [1] 5.112417
```

```
#{year, femlab, marriage, birth, military}
beta.hat2=qr.coef(qr(X2[train,]), y=y[train])
beta.hat2
```

```
## [1] 456.54703307  -0.24504472   0.93050659   0.14834382  -0.10601865
## [6]  -0.04747228
```

```r
fitted2 = X3[1:7,]%*%beta.hat2 #X3 is the test set
residuals2 = y[test] - fitted2
mean((y1[1:7] - fitted2)^2)
```

```
## [1] 5.095095
```

yes, the model does overestimate the response for subjects in the test set because the values of the mean squared error is greater than 0

```r
#3a
data = read.table("paper.txt")
data
```

```
##    bright operator
## 1    59.8        a
## 2    60.0        a
## 3    60.8        a
## 4    60.8        a
## 5    59.8        a
## 6    59.8        b
## 7    60.2        b
## 8    60.4        b
## 9    59.9        b
## 10   60.0        b
## 11   60.7        c
## 12   60.7        c
## 13   60.5        c
## 14   60.9        c
## 15   60.3        c
## 16   61.0        d
## 17   60.8        d
## 18   60.6        d
## 19   60.5        d
## 20   60.5        d
```

```r
X = cbind(rep(1,20), 1*(data$operator=='a'), 1*(data$operator=='b'), 1*(data$operator=='c')) #full mode
X0 = cbind(rep(1,20))#null model with just the intercept
print("assumptions: intercept, predictor variable, coefficients, and the response as bright")
```

```
## [1] "assumptions: intercept, predictor variable, coefficients, and the response as bright"
```

```r
y = data$bright
beta.hat = qr.coef(qr(X), y=y)
beta.hat
```

```
## [1] 60.68 -0.44 -0.62 -0.06
```

```r
y
```

```
##  [1] 59.8 60.0 60.8 60.8 59.8 59.8 60.2 60.4 59.9 60.0 60.7 60.7 60.5 60.9 60.3
## [16] 61.0 60.8 60.6 60.5 60.5
```

```r
rssf= sum((y-X%*%beta.hat)^2)
beta.hat0= qr.coef(qr(X0), y=y)
rssf0 = sum((y-X0%*%beta.hat0)^2)
n = length(y); p = length(beta.hat); d=4
f=((rssf0-rssf)/3)/(rssf/(n-p))
1-pf(f,d,n-p)
```

```
## [1] 0.01628656
```

H0:B2=B3=B4=0 H1:H0 is false B1 is the intercept pvalue >.01 we do not reject the null hypothesis at .01 signficance level and conclude that the operator is not relevant and significant at .01 in the linear regression model with response bright.

```
#3b
X = cbind(1,data$operator)
y = data$bright
get.ic(X=X, y=y)
```

```
## [1] 18.2116991 21.1988960  0.1748935
```

```
X1= cbind(rep(1,20))
get.ic(X=X1, y=y)
```

```
## [1] 23.0800462 25.0715107  0.1797031
```

disregard the last 3rd column, which is est.mse

According to the values of the AIC and BIC, the model with the predictors are more accurate than the model with only the intercept. However, the values of the AIC and BIC also reveals that the difference is between the two models are not that significant. With the AIC model being 18.2117 for the model with predictor and 23.08005 for the model with only the intercept. And the BIC model 21.1989 and 25.07151. This demostrates that the conclusion we had in part a is feasible due to the lack of significance of the predictor variable, operator.

```
#4
set.seed(3301)

get.bic=function(X, y)
{
n=dim(X)[1]
p=dim(X)[2]
beta.hat=lm.fit(x=X, y=y)$coefficients
rss=sum((y-X%*%beta.hat)^2)
bic=n*log(2*pi)+n*log(rss/n) + n + log(n)*(p+1)
return(bic)
}

reps=5e4
beta=c(1, 0, 0)
sigma=0.5
n=c(5,10,20,50,100)
m.list = numeric(5)
n.list=numeric(reps)

X1 = cbind(rep(1,100),0,0)
for(k in 1:100){
  for(s in 2:3){
    X1[k,s]= rnorm(1)
  }
}
for(i in 1:5){
  X = X1[1:n[i],]

for(r in 1:reps){
```

```
    y=X%*%beta + sigma*rnorm(n=n[i])
    bic.list=numeric(4)
    bic.list[1]=get.bic(X=X, y=y)
    bic.list[2]=get.bic(X=X[,-3],y=y)
    bic.list[3]=get.bic(X=X[,-2],y=y)
    bic.list[4]= get.bic(X=cbind(rep(1,n[i])), y=y)
    picked.index= which.min(bic.list)
    n.list[r]=1*(picked.index==1)
}
 m.list[i] = mean(n.list)
}
m.list
```

```
## [1] 0.27006 0.06098 0.01608 0.00404 0.00152
```