Problem 1.

Refer to Patient satisfaction Problem 6.15. The hospital administrator wishes to determine the best subset of predictor variables for predicting patient satisfaction

a. Indicate which subset of predictor variables you would recommend as best for predicting patient satisfaction according to each of the following criteria: (1) Ra,p 2 , (2) $AICp$, (3) Cp, (4) $BICp$. Support your recommendations with appropriate graphs.

b. Do the four criteria in part (a) identify the same best subset? Does this always happen?

(Option) c. Would forward stepwise regression have any advantages here as a screening procedure over the all-possible-regressions procedure?

In [28]:
```
import pandas as pd, numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math
```

In [2]:
```
df = pd.read_csv('CH06PR15.txt', sep = '\s+', header =None, names=['Y','X1','X2','X3']
df.head()
```

Out[2]:

	Y	X1	X2	X3
0	48	50	51	2.3
1	57	36	46	2.3
2	66	40	48	2.2
3	70	41	44	1.8
4	89	28	43	1.8

In [3]:
```
x1= df['X1']
x2= df['X2']
x3= df['X3']
y= df['Y']
```

a. Indicate which subset of predictor variables you would recommend as best for predicting patient satisfaction according to each of the following criteria: (1) Ra,p 2 , (2) $AICp$, (3) Cp, (4) $BICp$. Support your recommendations with appropriate graphs.

In [26]:
```
import statsmodels.api as sm
import statsmodels.formula.api as smf
model123 = smf.ols('y ~ x1+x2+x3', data=df)
results123 = model123.fit()
sse123 = np.sum((results123.fittedvalues - df.Y)**2)
mse123 = sse123/(n-4)
```

Regression of Y on X1

In [43]:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
model1 = smf.ols('y ~ x1', data=df)
results1 = model1.fit()
sse1 = np.sum((results1.fittedvalues - df.Y)**2)
ssr1 = np.sum((results1.fittedvalues - df.Y.mean())**2)
sstoX1 = ssr1+sse1
R2_X1 = ssr1/sstoX1
print('R^2 =',R2_X1)

n=len(y)
p1=2
R2a_X1 = 1 - (sse1/(n-p1))/(sstoX1/(n-1))
print('R^2a =',R2a_X1)

Cp1 = sse1/mse123 - (n-2*p1)
print('Cp=',Cp1)
aic1 = n * math.log(sse1/n)+ 2*p1
print('AICp=',aic1)
bic1 = n * math.log(sse1/n)+ p1*math.log(n)
print('BICp=',bic1)
```

```
R^2 = 0.6189842519960211
R^2a = 0.6103248031777488
Cp= 8.353606281990459
AICp= 220.52939082271948
BICp= 224.18667361569766
```

Regression of Y on X2

In [44]:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
model2 = smf.ols('y ~ x2', data=df)
results2 = model2.fit()
sse2 = np.sum((results2.fittedvalues - df.Y)**2)
ssr2 = np.sum((results2.fittedvalues - df.Y.mean())**2)
sstoX2 = ssr2+sse2
R2_X2 = ssr2/sstoX2
print('R^2 =',R2_X2)

n=len(y)
p1=2
R2a_X2 = 1 - (sse2/(n-p1))/(sstoX2/(n-1))
print('R^2a =',R2a_X2)

Cp2 = sse2/mse123 - (n-2*p1)
print('Cp=',Cp2)

aic2 = n * math.log(sse2/n)+ 2*p1
print('AICp=',aic2)
bic2 = n * math.log(sse2/n)+ p1*math.log(n)
print('BICp=',bic2)
```

```
R^2 = 0.3635387359110576
R^2a = 0.34907370718176345
Cp= 42.112323633767204
AICp= 244.1312019619498
BICp= 247.788484754928
```

Regression of Y on X3

In [45]:
```python
import statsmodels.api as sm
import statsmodels.formula.api as smf
model3 = smf.ols('y ~ x3', data=df)
results3 = model3.fit()
sse3 = np.sum((results3.fittedvalues - df.Y)**2)
ssr3 = np.sum((results3.fittedvalues - df.Y.mean())**2)
sstoX3 = ssr3+sse3
R2_X3 = ssr3/sstoX3
print('R^2 =',R2_X3)

n=len(y)
p1=2
R2a_X3 = 1 - (sse3/(n-p1))/(sstoX3/(n-1))
print('R^2a =',R2a_X3)

Cp3 = sse3/mse123 - (n-2*p1)
print('Cp=',Cp3)

aic3 = n * math.log(sse3/n)+ 2*p1
print('AICp=',aic3)
bic3 = n * math.log(sse3/n)+ p1*math.log(n)
print('BICp=',bic3)
```

```
R^2 = 0.41549754587804466
R^2a = 0.40221339919345467
Cp= 35.24564299480552
AICp= 240.21372333269096
BICp= 243.87100612566914
```

Regression of Y on X1 and X2

In [46]:
```python
import statsmodels.api as sm
import statsmodels.formula.api as smf
model12 = smf.ols('y ~ x1+x2', data=df)
results12 = model12.fit()
sse12 = np.sum((results12.fittedvalues - df.Y)**2)
ssr12 = np.sum((results12.fittedvalues - df.Y.mean())**2)
sstoX12 = ssr12+sse12
R2_X12 = ssr12/sstoX12
print('R^2 =',R2_X12)

n=len(y)
p2=3
R2a_X12 = 1 - (sse12/(n-p2))/(sstoX12/(n-1))
print('R^2a =',R2a_X12)

Cp12 = sse12/mse123 - (n-2*p2)
print('Cp=',Cp12)
aic12 = n * math.log(sse12/n)+ 2*p2
print('AICp=',aic12)
```

```
bic12 = n * math.log(sse12/n)+ p2*math.log(n)
print('BICp=',bic12)
```

```
R^2 = 0.6549558538884385
R^2a = 0.6389072889530168
Cp= 5.59973485144706
AICp= 217.96764722745866
BICp= 223.45357141692594
```

Regression of Y on X1 and X3

In [47]:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
model13 = smf.ols('y ~ x1+x3', data=df)
results13 = model13.fit()
sse13 = np.sum((results13.fittedvalues - df.Y)**2)
ssr13 = np.sum((results13.fittedvalues - df.Y.mean())**2)
sstoX13 = ssr13+sse13
R2_X13 = ssr13/sstoX13
print('R^2 =',R2_X13)

n=len(y)
p2=3
R2a_X13 = 1 - (sse13/(n-p2))/(sstoX13/(n-1))
print('R^2a =',R2a_X13)

Cp13 = sse13/mse123 - (n-2*p2)
print('Cp=',Cp13)
aic13 = n * math.log(sse13/n)+ 2*p2
print('AICp=',aic13)
bic13 = n * math.log(sse13/n)+ p2*math.log(n)
print('BICp=',bic13)
```

```
R^2 = 0.6760863825317273
R^2a = 0.6610206328820403
Cp= 2.807203767352547
AICp= 215.06065417704067
BICp= 220.54657836650796
```

Regression of Y on X2 and X3

In [48]:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
model23 = smf.ols('y ~ x2+x3', data=df)
results23 = model23.fit()
sse23 = np.sum((results23.fittedvalues - df.Y)**2)
ssr23 = np.sum((results23.fittedvalues - df.Y.mean())**2)
sstoX23 = ssr23+sse23
R2_X23 = ssr23/sstoX23
print('R^2 =',R2_X23)

n=len(y)
p2=3
R2a_X23 = 1 - (sse23/(n-p2))/(sstoX23/(n-1))
print('R^2a =',R2a_X23)

Cp23 = sse23/mse123 - (n-2*p2)
print('Cp=',Cp23)
```

```
aic23 = n * math.log(sse23/n)+ 2*p2
print('AICp=',aic23)
bic23 = n * math.log(sse23/n)+ p2*math.log(n)
print('BICp=',bic23)
```

```
R^2 = 0.4684544629858883
R^2a = 0.44373141475267386
Cp= 30.247056275166514
AICp= 237.8450063165764
BICp= 243.33093050604367
```

Regression of Y on X1, X2 and X3

In [49]:

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
model123 = smf.ols('y ~ x1+x2+x3', data=df)
results123 = model123.fit()
sse123 = np.sum((results123.fittedvalues - df.Y)**2)
ssr123 = np.sum((results123.fittedvalues - df.Y.mean())**2)
sstoX123 = ssr123+sse123
R2_X123 = ssr123/sstoX123
print('R^2 =',R2_X123)

n=len(y)
p3=4
R2a_X123 = 1 - (sse123/(n-p3))/(sstoX123/(n-1))
print('R^2a =',R2a_X123)

Cp123 = sse123/mse123 - (n-2*p3)
print('Cp=',Cp123)
aic123 = n * math.log(sse123/n)+ 2*p3
print('AICp=',aic123)
bic123 = n * math.log(sse123/n)+ p3*math.log(n)
print('BICp=',bic123)
```

```
R^2 = 0.682194333280746
R^2a = 0.6594939285150851
Cp= 4.0
AICp= 216.18496218375304
BICp= 223.49952776970943
```

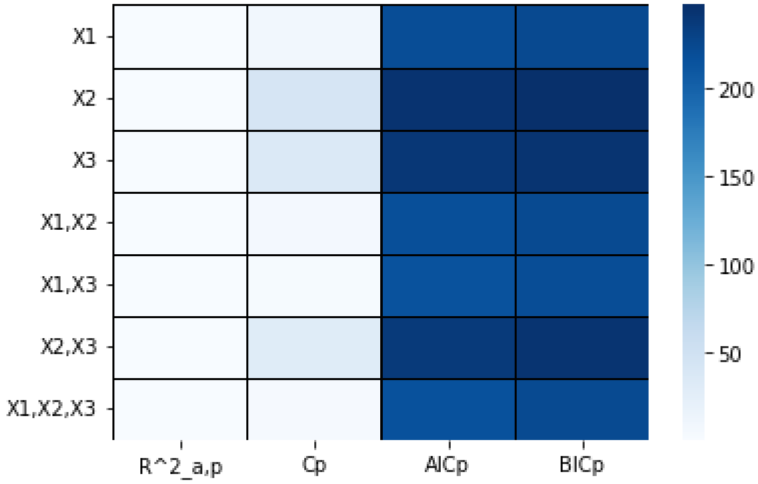
In [53]:

```
table = {'R^2_a,p':[R2a_X1,R2a_X2,R2a_X3,R2a_X12,R2a_X13,R2a_X23,R2a_X123],
        'Cp':[Cp1,Cp2,Cp3,Cp12,Cp13,Cp23,Cp123],
        'AICp':[aic1,aic2,aic3,aic12,aic13,aic23,aic123],
        'BICp':[bic1,bic2,bic3,bic12,bic13,bic23,bic123]}
t = pd.DataFrame(table)
t.index = ['X1','X2','X3','X1,X2','X1,X3','X2,X3','X1,X2,X3']
print(t)
```

```
          R^2_a,p         Cp        AICp        BICp
X1       0.610325   8.353606  220.529391  224.186674
X2       0.349074  42.112324  244.131202  247.788485
X3       0.402213  35.245643  240.213723  243.871006
X1,X2    0.638907   5.599735  217.967647  223.453571
X1,X3    0.661021   2.807204  215.060654  220.546578
X2,X3    0.443731  30.247056  237.845006  243.330931
X1,X2,X3 0.659494   4.000000  216.184962  223.499528
```

In [59]:

```
sns.heatmap(data=t, cmap= 'Blues', linecolor='black', linewidths=1);
```



=> indicates that the variables x1 and x3 should be included in the model while the variable x2 should be discarded. This is confirmed by the highest value of R2a,p=0.6610206, and lowest values for AICp, Cp, BICp

b. Do the four criteria in part (a) identify the same best subset? Does this always happen?

Yes, the four criteria in part (a) identify the same best subset. This does not always happen.