Econometrics - Week 2

William Schill

In [1]:
```
import numpy as np
import pandas as pd
from matplotlib.pyplot import *
import statsmodels.api as sma
import statsmodels.stats as sms
import seaborn as sns
%matplotlib inline
```

In [2]:
```
path = 'C:\\Users\\SchillW\\Documents\\Econ_Coursera\\Wk2\\'
df = pd.read_excel(path+'TestExer2-GPA-round2.xls')
```

In [3]:
```
df.head()
```

Out[3]:

	Observation	FGPA	SATM	SATV	FEM
0	1	2.518	4.0	4.0	1
1	2	2.326	4.9	3.1	0
2	3	3.003	4.4	4.0	1
3	4	2.111	4.9	3.9	0
4	5	2.145	4.3	4.7	0

In [4]:
```
xa = df['SATV']
ya = df['FGPA']
moda = sma.OLS(ya,sma.add_constant(xa))
fita = moda.fit()
print fita.summary()
```

```
                            OLS Regression Results                            
==============================================================================
=
Dep. Variable:                   FGPA   R-squared:                       0.00
8
Model:                            OLS   Adj. R-squared:                  0.00
7
Method:                 Least Squares   F-statistic:                     5.20
1
Date:                Fri, 27 Jan 2017   Prob (F-statistic):             0.022
9
Time:                        10:27:38   Log-Likelihood:                -388.4
4
No. Observations:                 609   AIC:                             780.
9
Df Residuals:                     607   BIC:                             789.
7
Df Model:                           1                                         

Covariance Type:            nonrobust                                         

==============================================================================
=
                 coef    std err          t      P>|t|      [95.0% Conf. In
t.]
------------------------------------------------------------------------------
-
const          2.4417      0.155     15.747      0.000         2.137     2.74
6
SATV           0.0631      0.028      2.280      0.023         0.009     0.11
7
==============================================================================
=
Omnibus:                       11.335   Durbin-Watson:                   1.94
9
Prob(Omnibus):                  0.003   Jarque-Bera (JB):                7.69
4
Skew:                           0.138   Prob(JB):                       0.021
3
Kurtosis:                       2.524   Cond. No.                         48.
2
==============================================================================
=

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correc
tly specified.
```

(a)(i) The coefficient is 0.0631 amd the p-value is 0.023. Assuming H0 is that SATV has a significant effect on FGPA, we cannot reject the Null to the 95% confidence interval.

In [5]:
```
b = fita.params
FGPA1 = np.dot(sma.add_constant(xa),b.T)
FGPA2 = np.dot(sma.add_constant((xa+1)),b.T)

ciFGPA1 = sms.weightstats.zconfint(FGPA1, alpha=0.05)
ciFGPA2 = sms.weightstats.zconfint(FGPA2, alpha=0.05)

print "Confidence Interval for FGPA is :", ciFGPA1
print "Confidence Interval for FGPA with SATV+1 is :", ciFGPA2
```

```
Confidence Interval for FGPA is : (2.7894274148855658, 2.7961653601554843)
Confidence Interval for FGPA with SATV+1 is : (2.8525132602634118, 2.85925120
55333303)
```

In [6]:
```
# print "Mean , Stand Dev of FGPA1 = ", np.mean(FGPA1), " , ", np.std(FGPA1)
# print "Mean , Stand Dev of FGPA2 = ", np.mean(FGPA2), " , ", np.std(FGPA2)
print "\n==================================\n"
print "Increase of 1 Point in SATV has ",
(np.mean(FGPA2)/np.mean(FGPA1)-1.0)*100.0, '% effect on mean FGPA'
```

```
==================================

Increase of 1 Point in SATV has  2.25887736248 % effect on mean FGPA
```

(a)(ii) The confidence interval for the 1 point increase can be seen above. A 1 point change to SATV is approximately a 2.26% incraese in FGPA.

In [7]:
```
xb = df.drop(['FGPA','Observation'],axis=1)
yb = ya
modb = sma.OLS(yb,sma.add_constant(xb))
fitb = modb.fit()
print fitb.summary()
```

```
                            OLS Regression Results
==============================================================================
=
Dep. Variable:                   FGPA   R-squared:                       0.08
3
Model:                            OLS   Adj. R-squared:                  0.07
8
Method:                 Least Squares   F-statistic:                     18.2
4
Date:                Fri, 27 Jan 2017   Prob (F-statistic):           2.41e-1
1
Time:                        10:27:38   Log-Likelihood:                -364.6
7
No. Observations:                 609   AIC:                             737.
3
Df Residuals:                     605   BIC:                             755.
0
Df Model:                           3

Covariance Type:            nonrobust

==============================================================================
=
                 coef    std err          t      P>|t|      [95.0% Conf. In
t.]
------------------------------------------------------------------------------
-
const          1.5570      0.216      7.205      0.000         1.133     1.98
1
SATM           0.1727      0.032      5.410      0.000         0.110     0.23
5
SATV           0.0142      0.028      0.507      0.612        -0.041     0.06
9
FEM            0.2003      0.037      5.358      0.000         0.127     0.27
4
==============================================================================
=
Omnibus:                        7.757   Durbin-Watson:                   1.91
2
Prob(Omnibus):                  0.021   Jarque-Bera (JB):                5.72
7
Skew:                           0.118   Prob(JB):                       0.057
1
Kurtosis:                       2.588   Cond. No.                         10
3.
==============================================================================
=

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correc
tly specified.
```

(b)(i) The coefficient are listed above and assuming the H0 is that the variables have a significant impact of FGPA, we can reject the Null for SATV when SATM and FEM are included.

In [8]:
```
b2 = fitb.params
FGPA3 = np.dot(sma.add_constant(xb),b2.T) #prediction

xb2 = xb.copy()
xb2['SATV'] = xb2['SATV']+1.0
FGPA4 = np.dot(sma.add_constant(xb2),b2.T) #prediction

ciFGPA3 = sms.weightstats.zconfint(FGPA3, alpha=0.05)
ciFGPA4 = sms.weightstats.zconfint(FGPA4, alpha=0.05)

print "Confidence Interval for FGPA is :", ciFGPA3
print "Confidence Interval for FGPA with SATV+1 is :", ciFGPA4
```

```
Confidence Interval for FGPA is : (2.7822678481821517, 2.8033249268589073)
Confidence Interval for FGPA with SATV+1 is : (2.7964297447224977, 2.81748682
33992534)
```

In [9]:
```
print "\n==================================\n"
print "Increase of 1 Point in SATV has ",
(np.mean(FGPA4)/np.mean(FGPA3)-1.0)*100.0, '% effect on mean FGPA'
```

```
==================================

Increase of 1 Point in SATV has  0.507086610525 % effect on mean FGPA
```

(b)(ii) : The confidence interval for FGPA with a 1 point increase in SATV for the model with more variables is seen above. The effect of a 1 point increase here is 0.51% which makes sense as the variable has a far less significant impact when accompanied by the other variables.

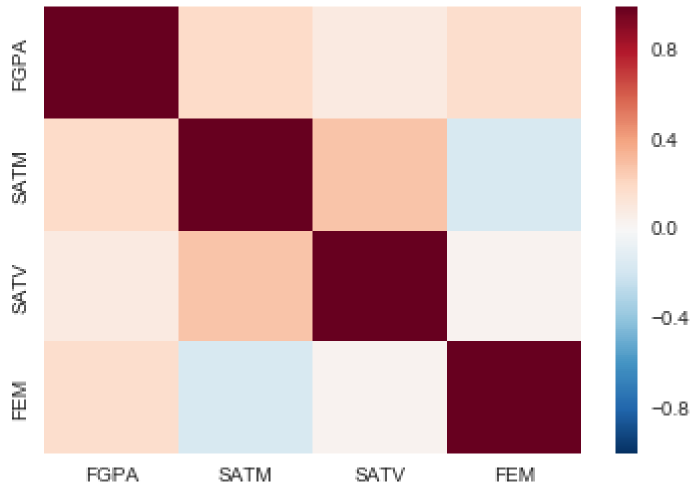
In [10]:
```
corchk = df.drop(['Observation'], axis=1)
corchk.corr()
```

Out[10]:

	FGPA	SATM	SATV	FEM
FGPA	1.000000	0.195040	0.092167	0.176491
SATM	0.195040	1.000000	0.287801	-0.162680
SATV	0.092167	0.287801	1.000000	0.033577
FEM	0.176491	-0.162680	0.033577	1.000000

In [11]:
```
sns.heatmap( corchk.corr())
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0xa60fa90>



(c) SATM and SATV have a 0.288 correlation which is lowering the significance of SATV in part *b(ii)*. SATV could be dropped with out any significant effect on the model.

In [12]:
```
print fitb.fvalue
print fitb.f_pvalue
```

```
18.2448959522
2.41149998676e-11
```

In [19]:
```
R = np.array([[0,1,0,0],
             [0,0,1,0],
             [0,0,0,1]])
print fitb.f_test(R)
print np.shape(xb)
```

```
<F test: F=array([[ 18.24489595]]), p=2.41149998676e-11, df_denom=605, df_num
=3>
(609, 3)
```

Comparing F tests for regression a and regression b:

In [98]:
```
g = 3.0-1.0 ##difference in number of parameters so 3 versus 1
n = np.shape(xb)[0]
k = np.shape(xb)[1] + 1 #adding in the constant
F = ((fitb.rsquared - fita.rsquared) / g) / ( (1-fitb.rsquared)/(n-k) )
Fcheck = (fitb.ess - fita.ess)/(fitb.centered_tss - fitb.ess) * (n-k)/g
print "\n=====================================================\n"
print "Manual calculation of F test"
print F
print Fcheck
print "g=",g, "  n=",n, "  k=",k
print "\n=====================================================\n"
print "Sqaure root of F = ", np.sqrt(F)
```

```
=====================================================

Manual calculation of F test
24.5651939993
24.5651939993
g= 2.0   n= 609   k= 4

=====================================================

Sqaure root of F =  4.95632868153
```

In [99]:
```
print "\n=====================================================\n"
print "statsmodels F test calculation for comaprison"
F2 = fitb.compare_f_test(fita)
print F2
print np.sqrt(F2[0])
```

```
=====================================================

statsmodels F test calculation for comaprison
(24.565193999316627, 5.5276752534971204e-11, 2.0)
4.95632868153
```

d(i):

The calculation for the F test (using part against part b) shows that the the F value exceeds the critical value of *3.9* and we can reject the Null hypothesis. We can confirm this using statsmodels method built into the fit as shown above.

d(ii):

Analytically, it can be shown that the F-test is approximately equal to the t-test squared. However in each calculation, this was unachievable. The statsmodels method derived a value of 8.66E-13 is close to the numerical calculation seen below that of 7.88E-13. In no way was I able to get a value of 4.956 for the t-test.

In [97]:
```
sms.weightstats.ttest_ind(FGPA3,FGPA1)
```

Out[97]:
```
(8.6611425410076572e-13, 0.99999999999930911, 1216.0)
```

In [93]:
```
top = np.mean(FGPA3) - np.mean(FGPA1)

s = np.sqrt( (np.std(FGPA3)**2 + np.std(FGPA1)**2)/2 )
t = top / (s * np.sqrt( 2.0/len(FGPA3) ))

t2 = top / np.sqrt( np.std(FGPA3)**2/len(FGPA3) + np.std(FGPA1)**2/len(FGPA1)
)
```

In [94]:
```
print t, t2
```

```
7.88023842255e-13 7.88023842255e-13
```

In [ ]: