Econometrics - Week 2

William Schill

This file was originally created using Jupyter with Python and HTML and has been saved as a pdf rather than exported to a pdf due to errors in exporting and latex, because of this formatting is a little wonky.

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 [20]:
```
xa = df['SATV']
ya = df['FGPA']
moda = sma.OLS(ya,sma.add_constant(xa))
fita = moda.fit()
print fita.summary2()
```

```
                 Results: Ordinary least squares
=================================================================
Model:              OLS              Adj. R-squared:     0.007
Dependent Variable: FGPA             AIC:                780.8876
Date:               2017-02-03 10:40 BIC:                789.7112
No. Observations:   609              Log-Likelihood:     -388.44
Df Model:           1                F-statistic:        5.201
Df Residuals:       607              Prob (F-statistic): 0.0229
R-squared:          0.008            Scale:              0.21037
-------------------------------------------------------------------
         Coef.     Std.Err.      t       P>|t|     [0.025    0.975]
-------------------------------------------------------------------
const    2.4417      0.1551   15.7468    0.0000    2.1372    2.7463
SATV     0.0631      0.0277    2.2805    0.0229    0.0088    0.1174
-----------------------------------------------------------------
Omnibus:              11.335       Durbin-Watson:          1.949
Prob(Omnibus):        0.003        Jarque-Bera (JB):       7.694
Skew:                 0.138        Prob(JB):               0.021
Kurtosis:             2.524        Condition No.:          48
=================================================================
```

(a)(i)

The coefficient for SATV 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 : (2.8525132602634113, 2.85925120
55333299)
```

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. The confidence intervals in the summary are based on the p-value and the ones listed above are based on the z-score:

Pvalue confidence interval: 0.009 < t < 0.117 Confidence Interval for FGPA is : (2.7894274148855658, 2.7961653601554843) Confidence Interval for FGPA with SATV+1 is : (2.8525132602634113, 2.8592512055333299)

In [21]:
```
xb = df.drop(['FGPA','Observation'],axis=1)
yb = ya
modb = sma.OLS(yb,sma.add_constant(xb))
fitb = modb.fit()
print fitb.summary2()
```

```
                 Results: Ordinary least squares
=================================================================
Model:              OLS              Adj. R-squared:     0.078
Dependent Variable: FGPA             AIC:                737.3379
Date:               2017-02-03 10:40 BIC:                754.9852
No. Observations:   609              Log-Likelihood:     -364.67
Df Model:           3                F-statistic:        18.24
Df Residuals:       605              Prob (F-statistic): 2.41e-11
R-squared:          0.083            Scale:              0.19521
--------------------------------------------------------------------
         Coef.     Std.Err.      t       P>|t|      [0.025    0.975]
--------------------------------------------------------------------
const    1.5570      0.2161    7.2054    0.0000     1.1327    1.9814
SATM     0.1727      0.0319    5.4104    0.0000     0.1100    0.2354
SATV     0.0142      0.0279    0.5071    0.6123    -0.0407    0.0690
FEM      0.2003      0.0374    5.3576    0.0000     0.1269    0.2737
-----------------------------------------------------------------
Omnibus:               7.757       Durbin-Watson:          1.912
Prob(Omnibus):         0.021       Jarque-Bera (JB):       5.727
Skew:                  0.118       Prob(JB):               0.057
Kurtosis:              2.588       Condition No.:          103
=================================================================
```

(b)(i)

The coefficients 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 intervals for all of the variables p-values can be found in the table above. 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.

Confidence Interval for FGPA is : (2.7822678481821517, 2.8033249268589073) Confidence Interval for FGPA with SATV+1 is : (2.7964297447224977, 2.8174868233992534)

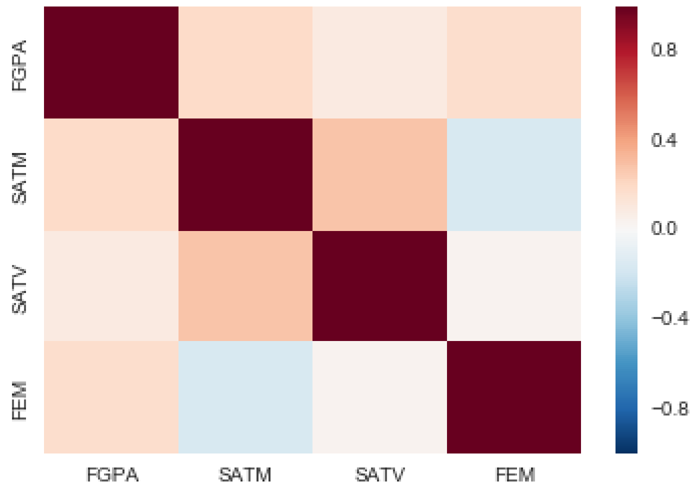
In [10]:
```
corchk = df.drop(['Observation'], axis=1)
corchk.corr()
```

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 0xa675f98>



(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 [13]:
```
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 [14]:
```
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 [15]:
```
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
(24.565193999316588, 5.5276752534973362e-11, 2.0)
4.95632868153
```

d(i):

I read the question as regarding the outcome of the model and not the SATV variable itself. 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.

We know that:

$t = (y_{bar1} - y_{bar0})/(s_p * \sqrt(1/n_0 + 1/n_1))$

where s_p = *s pooled* = $s_p = ((n_0 - 1) * s_0^2 + (n_1 - 1) * s_1^2)/(n_0 + n_1 - 2)$

so $t^2 = (y_{bar1} - y_{bar0})^2/(s_p^2 * (1/n_0 + 1/n_1))$

For a single outcome, and large n, we know that $s/\sqrt(n-k) \approx s/\sqrt(n)$

And $(s/\sqrt(n))^2 \approx e'e/n$

We can substitute this into our equation for t which gives us:

$1/s_p^2 * (1/n_0 + 1/n_1) \approx e_p' * e_p * (1/n_0 + 1/n_1)$

And using the same method in squaring the numerator of the t-statistic we have:

$(y_{bar1} - y_{bar0})^2 \approx e_1'e_1 - e_0'e_0$

In [19]:
```
n0 = len(FGPA1); n1 = len(FGPA3)
t_stat = np.sqrt( (fitb.ess - fita.ess)/(fitb.centered_tss - fitb.ess) /
(1.0/n0 + 1.0/n1) )
print t_stat
```

```
4.9726862464
```

The above result is approximately equal to the square root of the F-test.