Econometrics Week 5 Test

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.

Importing necessary modules and setting up the data set and running a model to match the information in the exam portion.

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\\Wk5\\'
df = pd.read_excel(path+'TrainExer5-5.xls')
```

In [3]:
```
df.head(5)
```

Out[3]:

	response	male	activity	age
0	1	0	0	58
1	1	1	0	50
2	1	1	0	40
3	1	1	0	36
4	1	1	0	28

In [4]:
```
xa = df.drop('response',axis=1)
xa['(Age/10)^2'] = (xa.age/10.0)**2.0
ya = df['response']

moda = sma.Logit(endog=ya, exog=sma.add_constant(xa))
fita = moda.fit(use_t=True)
print fita.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.650662
         Iterations 5
                         Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.061
Dependent Variable: response         AIC:              1213.7247
Date:               2017-02-24 09:17 BIC:              1237.8737
No. Observations:   925              Log-Likelihood:   -601.86
Df Model:           4                LL-Null:          -641.04
Df Residuals:       920              LLR p-value:      3.8865e-16
Converged:          1.0000           Scale:            1.0000
No. Iterations:     5.0000
------------------------------------------------------------------
               Coef.   Std.Err.     t     P>|t|    [0.025   0.975]
------------------------------------------------------------------
const         -2.4884    0.8900  -2.7959  0.0053  -4.2350  -0.7417
male           0.9537    0.1582   6.0291  0.0000   0.6433   1.2641
activity       0.9137    0.1848   4.9451  0.0000   0.5511   1.2764
age            0.0699    0.0356   1.9645  0.0498   0.0001   0.1398
(Age/10)^2    -0.0687    0.0341  -2.0146  0.0442  -0.1356  -0.0018
=================================================================
```

In [5]:
```
beta = fita.params
print beta

xAM50 = pd.DataFrame(np.array([1,1,50,(50/10)**2.0])).T
xAM50.columns = xa.columns
print "\n", xAM50

xIM50 = pd.DataFrame(np.array([1,0,50,(50/10)**2.0])).T
xIM50.columns = xa.columns
print "\n", xIM50
```

```
const        -2.488358
male          0.953694
activity      0.913748
age           0.069945
(Age/10)^2   -0.068692
dtype: float64

   male  activity   age  (Age/10)^2
0   1.0       1.0  50.0        25.0

   male  activity   age  (Age/10)^2
0   1.0       0.0  50.0        25.0
```

In [6]:
```
pr1_am50 = np.exp(np.dot(sma.add_constant(xAM50),beta.T)) / (1.0 + np.exp(np.d
ot(sma.add_constant(xAM50),beta.T)))
pr1_im50 = np.exp(np.dot(sma.add_constant(xIM50),beta.T)) / (1.0 + np.exp(np.d
ot(sma.add_constant(xIM50),beta.T)))
print "Probability of Response=1 for Active Male Age 50 ", pr1_am50, "\n"
print "Probability of Response=1 for Inactive Male Age 50 ", pr1_im50
```

```
Probability of Response=1 for Active Male Age 50  [ 0.76115956]

Probability of Response=1 for Inactive Male Age 50  [ 0.56101919]
```

The method of calculating the marginal effect and elasticity have been by recreating the slides from lecture 5.5 which can be found in the Appendix attached with this test.

In [7]:
```
b2 = beta['activity']
active_i = np.array([0,1])
b2i = b2*active_i
pr0_am50 = 1.0-pr1_am50
pr0_im50 = 1.0-pr1_im50
## This is the method I used to recreate the information given in the lecture
 5.5.
mrgeff_active = pd.DataFrame(pr1_am50*pr0_am50*b2i, columns = ['Active Male at
 50'])
mrgeff_inactive = pd.DataFrame(pr1_im50*pr0_im50*b2i, columns = ['Inactive Mal
e at 50'])
elasticity_active = pd.DataFrame(pr0_am50*b2i, columns = ['Active Male at
50'])
elasticity_inactive = pd.DataFrame(pr0_im50*b2i, columns = ['Inactive Male at
 50'])
```

The marginal effect of active status for a 50 year old, male, active customer is:

In [9]:
```
print mrgeff_active.iloc[1]
```

```
Active Male at 50    0.166115
Name: 1, dtype: float64
```

The marginal effect of active status for a 50 year old, male, *inactive* customer is:

In [10]:
```
print mrgeff_inactive.iloc[1]
```

```
Inactive Male at 50    0.225035
Name: 1, dtype: float64
```

The elasticity effect of active status for a 50 year old active male customer:

In [11]:
```
print elasticity_active.iloc[1]
```

```
Active Male at 50    0.21824
Name: 1, dtype: float64
```

The elasticity effect of active status for a 50 year old *inactive* male customer:

In [12]:
```
print elasticity_inactive.iloc[0]
```

```
Inactive Male at 50    0.0
Name: 0, dtype: float64
```

(a):

Using the designated information given in the exam we can create the elasticity effect of active status for a customer that is male and 50 years old.

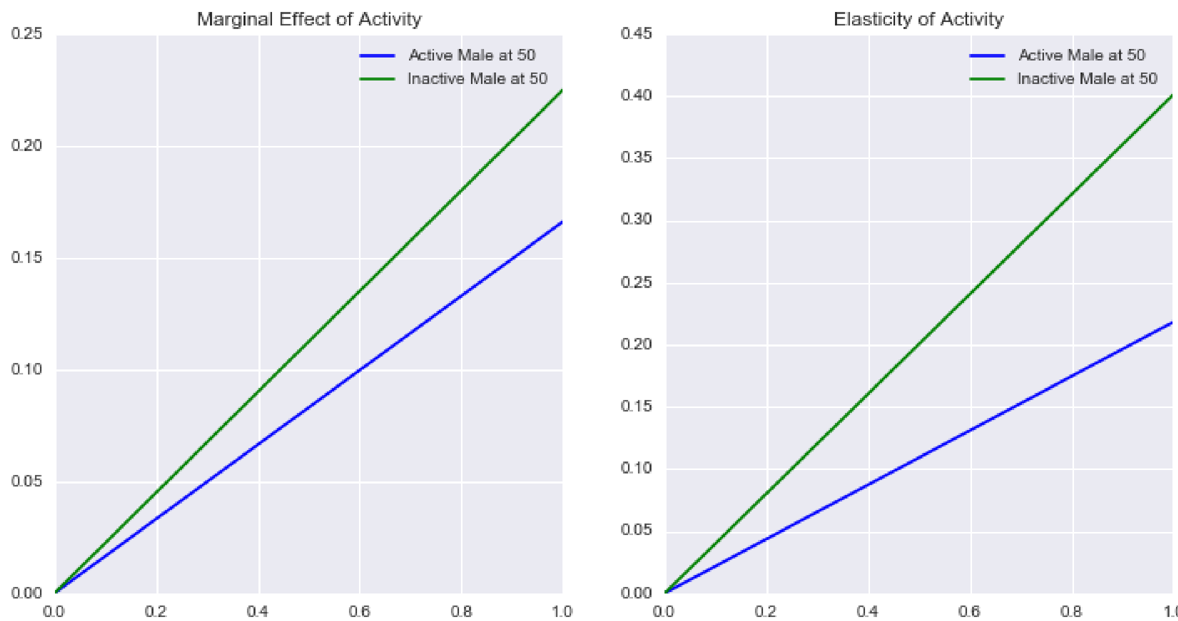
Through the following code, we create a variable xAM50 for active male at 50 to determine the probability of the outcome. Check it and then work out the marginal effects.

The elasticity effect of active status for a 50 year old male customer = 0.21824

The elasticity effect of inactive status for a 50 year old male customer = 0.0

In [13]:
```
fig, ax=subplots(1, 2, figsize=(12,6))
ax=ax.ravel()
mrgeff_active.plot(ax=ax[0], color='b', legend=True, title='Marginal Effect of
 Activity')
mrgeff_inactive.plot(ax=ax[0], color='g', legend=True)
elasticity_active.plot(ax=ax[1], color='b', legend=True, title='Elasticity of
 Activity')
elasticity_inactive.plot(ax=ax[1], color='g', legend=True)
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0xad7a4a8>



(b):

Elasticity can be defined and simplified as:

$$\frac{Pr[resp_i=1|active_i=1]-Pr[resp_i=1|active_1=0]}{Pr[resp_i=1|active_i=0]} = \ldots$$

$$\ldots = \frac{Pr[resp_i=1|active_i=1]}{Pr[resp_i=1|active_i=0]} - 1 = \ldots$$

$$\ldots = \frac{\frac{\exp(\beta_2)}{1+\exp(\beta_2)}}{\frac{exp(0)}{1+exp(0)}} - 1 = \ldots$$

$$\ldots = \frac{exp(\beta_2)*(1+exp(0)}{(1+exp(\beta_2))*(exp(0))} - 1 = \ldots$$

$$\ldots = \frac{2*exp(\beta_2)}{1+exp(\beta_2)} - 1 = \ldots$$

$$\ldots = \frac{2exp(\beta_2)-(1+exp(\beta_2))}{1+exp(\beta_2)} = \ldots$$

$$\ldots = \frac{exp(\beta_2)-1}{1+exp(\beta_2)} = \ldots$$

$$\ldots = (exp(\beta_2)-1)*(1+exp(\beta_2))^{-1} = \ldots$$

$$\ldots = (exp(\beta_2)-1)*Pr[resp_i=0|active_i=1]$$

This works as we can say that probability with active_i=1 and response_i=0 we would have:

$$(1+exp(\beta_2))^{-1} = 1 - \frac{exp(\beta_2)}{1+exp(\beta_2)} = \ldots$$

$$= 1 - Pr[resp_i=1|active_1=1] = Pr[resp_i=0|active_i=1]$$

(c):

Use the formula in part B to compute the elasticity of 50 years old male active customer.

The formula is :

$$(exp(\beta_2)-1)*P[resp_i=0|active_i=1]$$

We know that:

$$P[resp_i=0|active_i=1] = 1 - P[resp_i=1|active_i=1] = 1 - \frac{exp(\beta_2)}{1+exp(\beta_2)}$$

In [14]:
```
xb2 = beta['activity']
am50_2 = (np.exp(xb2)-1.0) / (1+np.exp(xb2))
am50_2
```

Out[14]: `0.42753297609121427`

As was mentioned above, the method I used to calculate the elasticity in part **(a)** was found by replicating the plots and information from the slides in lecture 5.5 and I have the code to show the exact results if necessary. With that being said, I cannot understand why this calculation is coming out at about twice that of the elasticity calculated in the corresponding part of **(a)**.

With this in mind, I have added an Appendix below for the recreation of lecture 5.5.

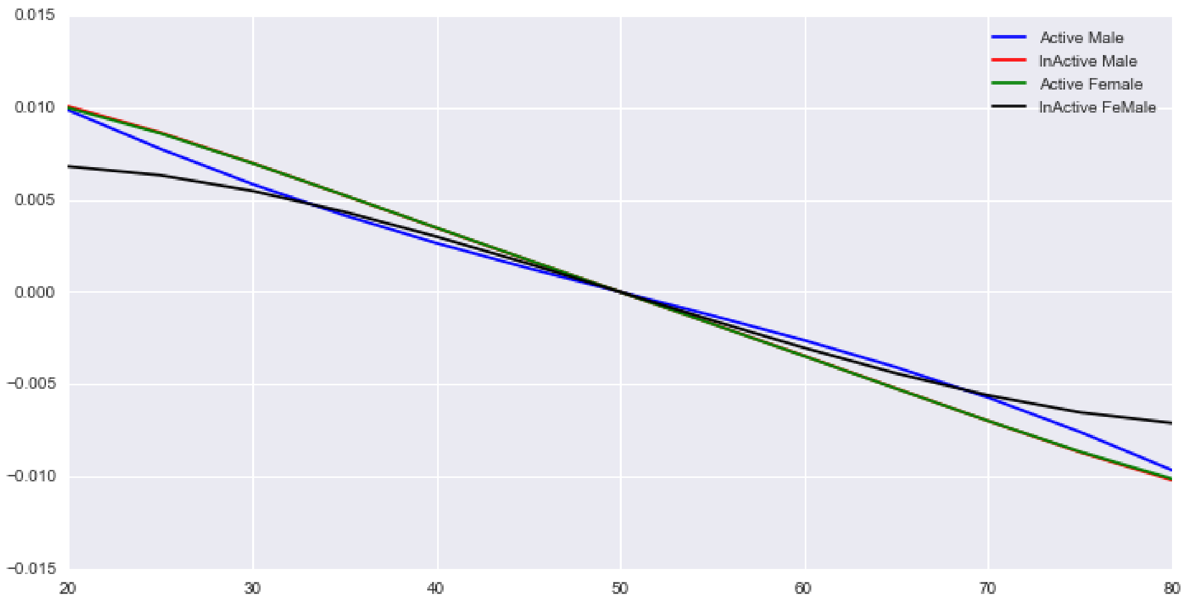
APPENDIX CONCERNING LECT 5.5

In [15]:
```
path2 = 'C:\\Users\\SchillW\\Documents\\Econ_Coursera\\Wk5\\'
df2 = pd.read_excel(path2+'TrainExer5-5.xls')
print df2.head(3)

## Build Model
xapp = df.drop('response',axis=1)
xapp['(Age/10)^2'] = (xapp.age/10.0)**2.0
yapp = df2['response']
modapp = sma.Logit(endog=yapp, exog=sma.add_constant(xapp))
fitapp = modapp.fit(use_t=True)
print "\n Summary: \n", fitapp.summary2()

betapp = fitapp.params
print betapp
xAM50_app = pd.DataFrame(np.array([1,1,50,(50/10)**2.0])).T
xAM50_app.columns = xapp.columns
print "\n x Active Male at 50 :\n", xAM50_app

ageApp = np.array(range(20,85,5))
print "\n Ages : \n", ageApp

betaApp = np.array([0.069945,-0.068692,0.953694,0.913748,-2.488358])

df3 = pd.DataFrame(ageApp, columns=['age'])
df3['age2'] = (df3['age']/10.0)**2.0

df1a=df3.copy(); df2a=df3.copy(); df3a=df3.copy(); df4a=df3.copy();
df1a['male'] = 1; df1a['activity']=1; df1a['const']=1;
df2a['male'] = 1; df2a['activity']=0; df2a['const']=1;
df3a['male'] = 0; df3a['activity']=1; df3a['const']=1;
df4a['male'] = 0; df4a['activity']=0; df4a['const']=1;

pr11 = np.exp(np.dot(df1a,betaApp))/(1+np.exp(np.dot(df1a,betaApp)))
pr12 = np.exp(np.dot(df2a,betaApp))/(1+np.exp(np.dot(df2a,betaApp)))
pr13 = np.exp(np.dot(df3a,betaApp))/(1+np.exp(np.dot(df3a,betaApp)))
pr14 = np.exp(np.dot(df4a,betaApp))/(1+np.exp(np.dot(df4a,betaApp)))

p3 = betapp['age']-2.0*betapp['age']*ageApp/100.0
c1 = pd.DataFrame(pr11*(1-pr11)*p3, columns = ['Active Male'], index=ageApp)
c2 = pd.DataFrame(pr12*(1-pr12)*p3, columns = ['InActive Male'], index=ageApp)
c3 = pd.DataFrame(pr13*(1-pr13)*p3, columns = ['Active Female'], index=ageApp)
c4 = pd.DataFrame(pr14*(1-pr14)*p3, columns = ['InActive FeMale'], index=ageAp
p)

print "\n betapp[age] = ", betapp['age']
```

```
   response  male  activity  age
0         1     0         0   58
1         1     1         0   50
2         1     1         0   40
Optimization terminated successfully.
         Current function value: 0.650662
         Iterations 5

 Summary:
                         Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.061
Dependent Variable: response         AIC:              1213.7247
Date:               2017-02-24 09:19 BIC:              1237.8737
No. Observations:   925              Log-Likelihood:   -601.86
Df Model:           4                LL-Null:          -641.04
Df Residuals:       920              LLR p-value:      3.8865e-16
Converged:          1.0000           Scale:            1.0000
No. Iterations:     5.0000
------------------------------------------------------------------
               Coef.  Std.Err.    t     P>|t|    [0.025   0.975]
------------------------------------------------------------------
const         -2.4884   0.8900 -2.7959 0.0053  -4.2350  -0.7417
male           0.9537   0.1582  6.0291 0.0000   0.6433   1.2641
activity       0.9137   0.1848  4.9451 0.0000   0.5511   1.2764
age            0.0699   0.0356  1.9645 0.0498   0.0001   0.1398
(Age/10)^2    -0.0687   0.0341 -2.0146 0.0442  -0.1356  -0.0018
=================================================================

const        -2.488358
male          0.953694
activity      0.913748
age           0.069945
(Age/10)^2   -0.068692
dtype: float64

 x Active Male at 50 :
   male  activity   age  (Age/10)^2
0   1.0       1.0  50.0        25.0

 Ages :
[20 25 30 35 40 45 50 55 60 65 70 75 80]

 betapp[age] =  0.0699452531304
```

In [16]:
```
fig, ax=subplots(1,1, figsize = (12,6))
c1.plot(ax=ax, color='b')
c2.plot(ax=ax, color='r')
c3.plot(ax=ax, color='g')
c4.plot(ax=ax, color='k')
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0xb1f5208>



In [ ]:

In [ ]: