240a2_ps3nb

November 18, 2015

1 Problem Set 3, Linear Regression, Preston Mui

```
In [1]: # Direct Python to plot all figures inline (i.e., not in a separate window)
        %matplotlib inline

        # Load libraries
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from __future__ import division
        import statsmodels.api as sm
        np.random.seed(813558889)

        # Read in Data
        nlsy79 = pd.read_csv('NLSY79_TeachingExtract.csv')
        nlsy79.set_index(['HHID_79','PID_79'], drop=False)
        nlsy79.rename(columns = {'AFQT_Adj':'AFQT'}, inplace=True) # Renaming AFQT

        # Calculate log earnings
        nlsy79['LogEarn'] = np.log(nlsy79[["real_earnings_1997", "real_earnings_1999", \
                                    "real_earnings_2001", "real_earnings_2003"]].mean(axis=1))

        # Convert Pandas Dataframe Columns into Numpy Arrays
        # Drop all observations with missing values in Log Earnings, AFQT, and HGC_Age28
        nlsy79 = nlsy79[np.isfinite(nlsy79['AFQT']) & np.isfinite(nlsy79['HGC_Age28']) & np.isfinite(nl
                        & (nlsy79['male']!=0) & (nlsy79.hispanic!=1) & (nlsy79.black!=1) & (nlsy79.core
        LogEarn = nlsy79.LogEarn.values
        AFQT = nlsy79.AFQT.values
        HGC_Age28 = nlsy79.HGC_Age28.values
```

1.1 Compute the Least Squares Fit

```
In [86]: # Define function to calculate least squares, and return variance-covariance matrix
         def leastsquares(Y,X):
             beta_hat = np.linalg.inv(X.T.dot(X)).dot(X.T.dot(Y))
             ui2 = np.diag((Y - beta_hat.dot(X.T))**2)
             Gamma = np.linalg.inv(X.T.dot(X))
             Omega = X.T.dot(ui2).dot(X)
             vcov_hat = Gamma.dot(Omega).dot(Gamma.T)
             return beta_hat, vcov_hat

         # Create X = regressors for question 1, run OLS
         X = np.column_stack((np.ones_like(HGC_Age28),HGC_Age28))
```

```
(betaX_hat, vcovX_hat) = leastsquares(LogEarn,X)
smolsX = sm.OLS(LogEarn,X).fit(cov_type='HC0')

print("Here are the results using my code\n")
print("Coefficient estimates:")
print(betaX_hat)
print("Variance-Covariance Matrix:")
print(vcovX_hat)

print("\n Here are the results using StatsModels\n")
print("Coefficient estimates:")
print(smolsX.params)
print("Variance-Covariance Matrix:")
print(smolsX.cov_params())

print("\nThe results with my code and the StatsModels OLS code are the same.")
```

```
Here are the results using my code

Coefficient estimates:
[ 8.62443117  0.16098318]
Variance-Covariance Matrix:
[[  1.24180225e-02  -8.80119399e-04]
 [ -8.80119399e-04   6.41755392e-05]]

 Here are the results using StatsModels

Coefficient estimates:
[ 8.62443117  0.16098318]
Variance-Covariance Matrix:
[[  1.24180225e-02  -8.80119399e-04]
 [ -8.80119399e-04   6.41755392e-05]]

The results with my code and the StatsModels OLS code are the same.
```

1.2 Long/short and Auxilliary Regression

In [87]:
```
# Run Least Squares on HGC_Age28 and AFQT:
XW = np.column_stack((np.ones_like(HGC_Age28),HGC_Age28,AFQT))
(betaXW_hat, vcovXW_hat) = leastsquares(LogEarn,XW)

print "Coefficients of Least Squares of Log earn on HGC_Age28 and AFQT:\n"
print "Constant:", betaXW_hat[0]
print "HGC_Age28:", betaXW_hat[1]
print "AFQT:", betaXW_hat[2]

print "\n Variance-Covariance Matrix:"
print(vcovXW_hat)
```

```
Coefficients of Least Squares of Log earn on HGC_Age28 and AFQT:

Constant: 8.90148094127
HGC_Age28: 0.114065246045
AFQT: 0.00620478098098
```

```
 Variance-Covariance Matrix:
[[  1.15461109e-02  -8.94670563e-04   1.57320920e-05]
 [ -8.94670563e-04   8.98564884e-05  -5.35199697e-06]
 [  1.57320920e-05  -5.35199697e-06   9.30095621e-07]]
```

Auxillary Regression:

```
In [88]: (betaX_hat - betaXW_hat[0:2]) / betaXW_hat[2]
         print "Auxillary Regression Results of AFQT on Constant and HGC_Age28:"
         print "Constant: ", ((betaX_hat - betaXW_hat[0:2]) / betaXW_hat[2])[0]
         print "HGC_Age28: ", ((betaX_hat - betaXW_hat[0:2]) / betaXW_hat[2])[1]
```

```
Auxillary Regression Results of AFQT on Constant and HGC_Age28:
Constant:  -44.6510148915
HGC_Age28:  7.56157767652
```

```
In [89]: (gamma) = leastsquares(AFQT,X)[0]
         print "Direct Least Squares Regression of AFQT of Constant and HGC_Age28"
         print "Constant: ", gamma[0]
         print "HGC_Age28: ", gamma[1]
         print "The estimates from direct OLS are identical to those of the Auxillary Regression"
```

```
Direct Least Squares Regression of AFQT of Constant and HGC_Age28
Constant:  -44.6510148915
HGC_Age28:  7.56157767652
The estimates from direct OLS are identical to those of the Auxillary Regression
```

1.3 Finding the coefficient on HGC_Age28 by regressing Log Earnings

Define $V = AFQT - X \cdot \gamma'$, where X is the ones vector concatenated with the HGC_Age28 vector, and γ are the coefficients from the least squares estimate of AFQT on X. Then, the coefficient on HGC_Age28 in the regression of Log Earnings on Constant, AFQT, and HGC_Age28 is given by a regression of Log Earnings on V.

```
In [90]: V = AFQT - gamma.dot(X.T)
         print "Least Squares Regression of Log Earnings on V:"
         print "Coefficient on V: ", leastsquares(LogEarn,V.reshape(len(V),1))[0]
         print "This is the same coefficient as that on HGC_Age28 in the Log Earnings Regression"
```

```
Least Squares Regression of Log Earnings on V:
Coefficient on V:  [ 0.00620478]
This is the same coefficient as that on HGC_Age28 in the Log Earnings Regression
```

1.4 Bayesian Bootstrapping

```
In [91]: # Estimate OLS on a constant, HGC_Age28, HGCAge28 x (AFQT - 50), AFQT

         # Create XZW, the matrix of regressors
         XZW = np.column_stack((np.ones_like(HGC_Age28),HGC_Age28,HGC_Age28*(AFQT - 50),AFQT))

         # Run OLS
         coefficients = leastsquares(LogEarn,XZW)[0]
         alpha0 = coefficients[0]
         beta0 = coefficients[1]
         gamma0 = coefficients[2]
         delta0 = coefficients[3]
```

```
print "Estimated Coefficients: "
print u"\u03B1", "= ", alpha0
print u"\u03B2", "= ", beta0
print u"\u03B3", "= ", gamma0
print u"\u03B4", "= ", delta0
```

```
Estimated Coefficients:
α =  8.59935782333
β =  0.120254979024
γ =  -0.0003793989497
δ =  0.0109839662074
```

(a) **Provide a semi-elasticity interpretation of β_0:** β_0 is the overall average elasticity of earnings with respect to grade completion, across the population.

(b) **Provide a semi-elasticity interpretation of $\beta_0 + \gamma_0(AFQT - 50)$:** is the elasticity of earnings with respect to grade completion, given some level of AFQT.

(c) **Interpret the Null Hypothesis $H_0 : \gamma_0 = 0$:** Under the null hypothesis, the elasticity of earnings with respect to grade completition does not change with AFQT scores.

```
In [92]: # Set up Bayesian Bootstrapping (10000 pulls)
         B = 10000
         M = np.empty((B,4))
         N = len(LogEarn)

         # Do Bayesian Bootstrapping
         for b in range (0,B):
             wgts = np.random.gamma(1.,1.,N) # Random draws of Gamma(1,1) variables
             wgts = wgts/np.sum(wgts)              # Converting draws to Dirichlet

             result = sm.WLS(LogEarn,XZW,weights=wgts).fit()
             M[b,:] = np.matrix(result.params) # Linear regression with Dirichlet wgts
```

```
In [93]: # Calculate confidence intervals for beta + gamma(AFQT - 50)
         # For each AFQT score 1 - 100, and each pull of beta, gamma, calculate beta + gamma(AFQT - 50)

         beta_gammaAFQT50_estimates = np.zeros((100,B))
         for a in range(100):
             for b in range(B):
                 beta_gammaAFQT50_estimates[a,b] = M[b,1] + M[b,2] * (a+1-50)

         # For each AFQT score, pull the 0.025 and 0.975 quantiles
         lowerbound = np.percentile(beta_gammaAFQT50_estimates,2.5,axis=1)
         upperbound = np.percentile(beta_gammaAFQT50_estimates,97.5,axis=1)
         estimate = np.zeros(100)
         for a in range(100):
             estimate[a] = beta0 + gamma0*(a + 1 - 50)

         # Plot point line and interval
         afqtvalues = np.linspace(1,100,100)
         fig, ax = plt.subplots(1)
         ax.plot(afqtvalues, upperbound, label='Upper Bound', color='blue')
         ax.plot(afqtvalues, estimate, label='Point Estimate', color='red')
         ax.plot(afqtvalues, lowerbound, label='Lower Bound', color='blue')
```

```
ax.fill_between(afqtvalues, lowerbound, upperbound, facecolor='grey', alpha=0.5)
ax.legend()
ax.set_xlabel('AFQT')
ax.set_ylabel('Elasticity of Earnings to Schooling')
```

Out[93]: <matplotlib.text.Text at 0x7fefd10a47d0>