HW_Example_Badly_Done

April 12, 2021

[1]:
```
import numpy as np
import time
import matplotlib.pyplot as plt
```

[2]:
```
def full_regression(y,x):
    eb=x*x*y
    return(eb)
```

[3]:
```
def regression_dispersion(x):
    ecov=2*x
    return(ecov)
```

[13]:
```
def regression_update(beta_previous,c, y_new, x_new, y_old, x_old):
    eb=dispersion_previous*(beta_previous/x_new.T)*(y_new-x_new*beta_previous)/
 ↪(1+(x_new*c*(x_new.T))[0,0])
    pn=(c.I)/2
    eb=eb-(y_old-x_old*eb)/(1-(x_old[0,0])
    pn=(pn.I)- x_old.T
    return eb,pn
```

[5]:
```
m=2**31-1
a = 1103515245
c =12345
```

[6]:
```
def lcg(s,m,a,c,n):
    lcgl=['']*(n+1)
    lcgl[0]=s
    for i in range(1,n+1):
        s= (s*a+c) % m
        lcgl[i]=s

    return (lcgl)


def lcgunif(s,m,a,c,n):
    lul=lcg(s,m,a,c,n)
    for i in range(n+1):
        lul[i]=1.0*lul[i]/m-0.5
    return (lul)
```

[7]:
```
lcgunif(5,m,a,c,5000)[4998:]
```

[7]:
```
[-0.07691774730427087, 0.23868520359447465, -0.07756264162136367]
```

[8]:
```
%matplotlib nbagg
import matplotlib.pyplot as plt
plt.plot(lcgunif(5,m,a,c,500),'ro')
plt.title('first 500 generated numbers, seed=5')
plt.show()
```

```
<IPython.core.display.Javascript object>

<IPython.core.display.HTML object>
```

[9]:
```
yl=np.matrix(lcgunif(0,m,a,c,(400000+5*1000))).T
xl=np.matrix([lcgunif(k,m,a,c,(400000+5*1000))for k in range(1,9)]).T
```

[71]:
```
%statsmodels nbagg
import statsmodels.api as sm
model = sm.OLS(yl[1:101],xl[1:101])
results = model.fit()
results.summary()
```

[71]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.107
Model:                            OLS   Adj. R-squared:                  0.029
Method:                 Least Squares   F-statistic:                     1.373
Date:                Sat, 22 Apr 2017   Prob (F-statistic):              0.219
Time:                        04:37:13   Log-Likelihood:                -12.259
No. Observations:                 100   AIC:                             40.52
Df Residuals:                      92   BIC:                             61.36
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
x1             0.1642      0.101      1.624      0.108        -0.037     0.365
x2            -0.0120      0.098     -0.122      0.903        -0.207     0.183
x3             0.1483      0.112      1.319      0.190        -0.075     0.372
x4            -0.1313      0.101     -1.297      0.198        -0.332     0.070
x5            -0.1849      0.100     -1.845      0.068        -0.384     0.014
x6             0.0093      0.102      0.091      0.928        -0.194     0.213
x7             0.1016      0.097      1.048      0.298        -0.091     0.294
```

```
x8             0.1225      0.093      1.314      0.192        -0.063      0.308
==============================================================================
Omnibus:                        6.668   Durbin-Watson:                   1.882
Prob(Omnibus):                  0.036   Jarque-Bera (JB):                3.541
Skew:                          -0.235   Prob(JB):                        0.170
Kurtosis:                       2.207   Cond. No.                         1.67
==============================================================================

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

Use built-in function to check my function.

[72]:

```
ebf=full_regression(yl[1:101],xl[1:101])
print ebf
```

```
[[ 0.16416022]
 [-0.0119616 ]
 [ 0.14833042]
 [-0.13126348]
 [-0.18489287]
 [ 0.00931188]
 [ 0.10160505]
 [ 0.12253977]]
```

Regression result using my full_regression function, which is same as built-in function.

[73]:

```
%statsmodels nbagg
import statsmodels.api as sm
model = sm.OLS(yl[11:31],xl[11:31])
results = model.fit()
results.summary()
```

[73]:

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.318
Model:                            OLS   Adj. R-squared:                 -0.137
Method:                 Least Squares   F-statistic:                    0.6992
Date:                Sat, 22 Apr 2017   Prob (F-statistic):              0.688
Time:                        04:37:22   Log-Likelihood:                -1.7580
No. Observations:                  20   AIC:                             19.52
Df Residuals:                      12   BIC:                             27.48
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
x1             0.0801      0.315      0.254      0.804        -0.607     0.767
x2            -0.2095      0.299     -0.701      0.496        -0.860     0.441
x3             0.1378      0.272      0.506      0.622        -0.456     0.731
x4             0.1326      0.283      0.469      0.648        -0.484     0.749
x5            -0.1966      0.302     -0.652      0.527        -0.854     0.461
x6             0.2707      0.338      0.801      0.439        -0.466     1.007
x7             0.5221      0.319      1.637      0.127        -0.173     1.217
x8             0.2255      0.262      0.860      0.407        -0.346     0.797
==============================================================================
Omnibus:                        0.492   Durbin-Watson:                   1.555
Prob(Omnibus):                  0.782   Jarque-Bera (JB):                0.571
Skew:                          -0.098   Prob(JB):                        0.752
Kurtosis:                       2.196   Cond. No.                         2.32
==============================================================================

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

[77]:

```
ebf=full_regression(yl[1:21],xl[1:21])
dp=regression_dispersion(xl[1:21])
for i in range(21,31):
    ebf,dp=regression_update(ebf,dp,yl[i],xl[i],yl[i-20],xl[i-20])
print ebf
```

```
[[ 0.08008111]
 [-0.20948436]
 [ 0.13784187]
 [ 0.13264206]
 [-0.19663574]
 [ 0.27071247]
 [ 0.52207175]
 [ 0.22548283]]
```

[ ]:

```

```

Regression result using my regression_update function, versus package function. Window size is 20.

[ ]:

```
st=time.time()
for i in range(801,200000):
    ebp=full_regression(yl[(i-599):(i+1)],xl[(i-599):(i+1)])
    dp=regression_dispersion(xl[(i-599):(i+1)])
print time.time()-st
```

```
86.0953099728
```

[20]:

```
ws=60000
st=time.time()
ebp=full_regression(yl[1:(ws+1)],xl[1:(ws+1)])
dp=regression_dispersion(xl[1:(ws+1)])
for i in range((ws+1),100000):
    ebp,dp=regression_update(ebp,dp,yl[i],xl[i],yl[i-100],xl[i-100])
print time.time()-st
```

```
41.4314110279
```

[61]:

```
betal=[[0 for x in range(1000)] for y in range(8)]
for i in range((400000+4*1000),(400000+5*1000)):
    ebp=full_regression(yl[(i-99):(i+1)],xl[(i-99):(i+1)])
    for j in range(8):
        betal[j][i-(400000+4*1000)]=ebp[j,0]
for k in range(8):
        print np.median(betal[k])
```

```
0.0209417264367
-0.0152007817198
0.0740453953213
0.0424843657507
0.016364936045
-0.0359080846089
-0.027549578199
0.0169411817901
```

[ ]:

```

```