R09323036 李祖福 problem3

HW3

```python
[1]: import pandas as pd
     import numpy as np
     from scipy.optimize import minimize
     from scipy.stats import norm
```

```python
[42]: data = pd.read_excel("/Users/lee/Downloads/CilibertoTamerEconometrica.xlsx")
```

Q1

Q1

```python
[53]: n = len(data)
      data['mk'] = 1
      X = np.array(data[['mk', 'marketsize', 'marketdistance']])
      X
```

```
[53]: array([[1.        , 1.81654286, 0.69199997],
             [1.        , 1.00100732, 0.685     ],
             [1.        , 1.23256075, 0.50300002],
             ...,
             [1.        , 1.45095992, 1.72099996],
             [1.        , 1.23586535, 0.54699999],
             [1.        , 0.8648805 , 0.90600002]])
```

```python
[30]: N = np.array(data[['airlineAA', 'airlineDL', 'airlineUA', 'airlineAL', 'airlineLCC', 'airlineWN']].sum(axis=1))
      N
```

```
[30]: array([1, 1, 0, ..., 4, 2, 1])
```

```python
[39]: def nl(para):
          beta = np.array(para[:3])
          delta = np.array(para[3])

          nll = 0
          for i in range(n):
              if N[i] == 0:
                  nll += -np.log(norm.cdf(np.dot(-X[i], beta)))
              elif N[i] == 6:
                  nll += -np.log(1 - norm.cdf(np.dot(-X[i], beta) + delta*np.log(6)))
              else:
                  nll += -np.log(norm.cdf(np.dot(-X[i], beta) + delta*np.log(N[i] + 1)) - norm.cdf(np.dot(-X[i], beta) + de
          return nll
```

```python
[40]: para = [1.3, 1.3, 1.3, 1]
```

```python
[41]: ans = minimize(nl, para, options={'disp': True})
```

```
Warning: Desired error not necessarily achieved due to precision loss.
         Current function value: 4599.864692
         Iterations: 15
         Function evaluations: 472
         Gradient evaluations: 76
```

```python
[43]: para = ans.x
      beta, delta = para[:3], para[3]
      cvmt = ans.hess_inv
      se = [np.sqrt(cvmt[i,i]) for i in range(len(cvmt))]
```

```python
[46]: print('Beta:',beta,'\nBeta\'s standard error:',se[:3])
      print('Delta:',delta,'\nDelta\'s standard error:',se[3])
```

```
Beta: [0.98325407 0.06796466 0.47342029]
Beta's standard error: [0.016073392954031825, 0.017222813891410622, 0.012128213291040213]
Delta: 1.9211390416649874
Delta's standard error: 0.03196461970500831
```

Q2

Q2

```
[59]: data = np.array(data)
      data2 = []
      for i in range(n):
          imk = []
          for j in range(6):
              imk.append(np.append(X[i], (data[i][j+15], data[i][j+21])))
          data2.append(np.vstack(imk))
```

```
[60]: data2[0]
```

```
[60]: array([[1.        , 1.81654286, 0.69199997, 0.18000001, 0.81831986],
             [1.        , 1.81654286, 0.69199997, 0.55909091, 0.        ],
             [1.        , 1.81654286, 0.69199997, 0.24636364, 0.81831986],
             [1.        , 1.81654286, 0.69199997, 0.48030305, 0.02098957],
             [1.        , 1.81654286, 0.69199997, 0.10666667, 0.        ],
             [1.        , 1.81654286, 0.69199997, 0.        , 0.00691858]])
```

```
[64]: def of(para):
          beta_alpha = np.array(para[:5])
          delta = np.array(para[5])
          phi = np.array(para[6])

          E_N = np.zeros(n)
          np.random.seed(7323014)
          for i in range(n):
              N_list = np.zeros(T)
              for j in range(T):
                  u_i0 = np.random.randn(1)
                  u_ik = np.random.randn(6)
                  for k in range(6,0,-1):
                      profit = np.dot(data2[i], beta_alpha) - delta*np.log(k) + phi*u_i0 + np.sqrt(1-phi**2)*u_ik
                      profit_num_firm = sum(profit > 0)
                      if profit_num_firm >= k:
                          N_list[j] = k
                          break
              E_N[i] = np.mean(N_list)
          Error = N - E_N
          mu = np.dot(Error, X)
          return np.dot(mu, mu)
```

```
[68]: para = [0, 0, 0, 0, 0, 0.5, 0.5]
```

```
[69]: θ_list = []
      T = 30
      S = 30
      seed_list = np.random.randint(0,high=10000,size=S)
      for seed in range(S):
          ith_seed = seed_list[seed]
          def of(para):
              β_α = np.array(para[:5])
              δ = np.array(para[5])
              ρ = np.array(para[6])

              E_N = np.zeros(n)
              np.random.seed(ith_seed)
              for i in range(n):
                  N_list = np.zeros(T)
                  for j in range(T):
                      u_i0 = np.random.randn(1)
                      u_ik = np.random.randn(6)
                      for k in range(6,0,-1):
                          profit = np.dot(data2[i], β_α) - δ*np.log(k) + ρ*u_i0 + np.sqrt(1-ρ**2)*u_ik
                          profit_num_firm = sum(profit > 0)
                          if profit_num_firm >= k:
                              N_list[j] = k
                              break
                  E_N[i] = np.mean(N_list)
              Error = N - E_N
              μ = np.dot(Error, X)
              return np.dot(μ, μ)
          ans = minimize(of,
                         para,
                         method='Nelder-Mead',
                         options={'disp': True})
          θ_list.append(ans.x)
```

```
beta: [0.00047961 0.00116044 0.00014594]
beta's standard error: [1.35488485e-04 5.41428688e-05 1.26644489e-04]
alpha: [0.00022884 0.00045789]
alpha's standard error: [0.00021857 0.00022327]
delta: 0.24409692574675956
delta's standard error: 0.0030761758032818412
phi: 0.49324155613599996
phi's standard error: 0.004369318858032602
```