SPRING 2023 FINC B 9325 Financial Econometrics – Time Series
Homework 3 Sample Solution

Question 1

Preliminaries

The orthogonality conditions are given by, for $i = 1, \dots 10$,

$$\mathbb{E}\left[\beta\left(\frac{C_{t+1}}{C_t}\right)^{-\gamma} R^R_{i,t+1} - 1\right] = 0$$

Define

$$f_t(\theta) = \begin{bmatrix} \beta\left(\frac{C_{t+1}}{C_t}\right)^{-\gamma} R^R_{1,t+1} \\ \vdots \\ \beta\left(\frac{C_{t+1}}{C_t}\right)^{-\gamma} R^R_{10,t+1} \end{bmatrix}$$

Define the sample average

$$g_T(\theta) = \frac{1}{T}\sum_{t=1}^{T} f_t(\theta)$$

GMM implementation:

- <u>First Stage</u>:
$$\hat{\theta}_1 = \arg\min_{\theta} [g_T(\theta)]' I [g_T(\theta)]$$
where I is the identity matrix.

- <u>Second Stage</u>:
$$\hat{\theta}_2 = \arg\min_{\theta} [g_T(\theta)]' \hat{S}^{-1} [g_T(\theta)]$$
with
$$\hat{S} = \frac{1}{T}\sum_{t=1}^{T} [f_t(\hat{\theta}_1)][f_t(\hat{\theta}_1)]'$$

Estimator of the asymptotic variance:

$$\widehat{\text{asymp var}}(\hat{\theta}_2) = \frac{1}{T}(\hat{D}'\hat{S}^{-1}\hat{D})^{-1}$$

with

$$\hat{D} = \frac{\partial}{\partial \theta'} \frac{1}{T} \sum_{t=1}^{T} f_t(\hat{\theta}_2)$$

Hansen J-test:

$$J = T \cdot [g_T(\hat{\theta}_2)]' \hat{S}^{-1} [g_T(\hat{\theta}_2)], \qquad J \xrightarrow{d} \chi^2(10-2)$$

Results

	β (se)	γ (se)	J-stat	p-value	df
Full Sample (1951 - 2021)					
First Stage	1.005	7.243			
Second Stage	0.986 (0.165)	0.498 (36.683)	15.869	0.044	8
Short Sample (1951 - 2019)					
First Stage	1.343	88.196			
Second Stage	1.252 (0.217)	62.86 (50.826)	7.776	0.45	8

Other Estimations
We repeat the above replacing $\hat{S}$ with $\hat{S}_{NW}$, the Newey-West estimator of S with de-meaned moments and 4 lags:

$$\hat{S}_{NW} = \hat{\Gamma}_{0,T} + \sum_{v=1}^{q} \left(1 - \frac{v}{q+1}\right) (\hat{\Gamma}_{v,T} + \hat{\Gamma}'_{v,T})$$

where

$$\hat{\Gamma}'_{v,T} = \frac{1}{T} \sum_{t=v+1}^{T} [\tilde{f}_t(\hat{\theta}_1)][\tilde{f}_{t-v}(\hat{\theta}_1)]'$$

$$\tilde{f}_t(\hat{\theta}_1) = f_t(\hat{\theta}_1) - \frac{1}{T} \sum_{t=1}^{T} f_t(\hat{\theta}_1)$$

	β (se)	γ (se)	J-stat	p-value	df
Full Sample (1951 - 2021)					
First Stage	1.005	7.243			
Second Stage	0.985 (0.230)	1.583 (50.492)	14.792	0.063	8
Short Sample (1951 - 2019)					
First Stage	1.343	88.196			
Second Stage	1.191 (0.232)	51.160 (50.798)	8.217	0.412	8

Repeat above, with a NW estimator, and a just-identified system with two different moments conditions. Let $R^R_{m,t}$ be the equal weighted average of the 10 real portfolio returns, a proxy for the general market return. Let $R^R_{e,t}$ be the real return on the 10th portfolio minus the real return on the 1st portfolio. This moment reflects the so-called value premium. The two moment conditions are

$$\mathbb{E}\begin{bmatrix} \beta\left(\frac{C_{t+1}}{C_t}\right)^{-\gamma} R^R_{m,t+1} - 1 \\ \beta\left(\frac{C_{t+1}}{C_t}\right)^{-\gamma} R^R_{e,t+1} \end{bmatrix} = 0$$

	β (se)	γ (se)	J-stat	p-value	df
Full Sample (1951 - 2021)					
First Stage	0.771	40.780			
Second Stage	0.771 (0.213)	40.780 (12.368)			
Short Sample (1951 - 2019)					
First Stage	1.484	187.912			
Second Stage	1.484 (0.184)	187.912 (148.05)			

We now repeat the full sample estimation, with a NW estimator, and replace the consumption growth outlier in 2020 Q2 (0.89) with 2 times the minimum net consumption growth in the sample until the end of 2019.

	β (se)	γ (se)	J-stat	p-value	df
Full Sample (1951 - 2021)					
First Stage	1.219	59.144			
Second Stage	1.172 (0.166)	45.992 (39.15)	10.327	0.242	8

Clearly, it was the outlier observation that completely drove γ to be much smaller.

Sample Code for base estimation and with Newey West Weighting Matrix:

```
import pandas as pd
import os
import numpy as np
from scipy.optimize import minimize
from autograd import jacobian
from scipy import stats
import seaborn as sn
import matplotlib.pyplot as plt

folder_path = '/Users/yz3937/Desktop/hw3_new/data/'

# Read data-----------------------------------------------------------

df = pd.read_csv(f'{folder_path}homework3data.csv')

for i in range(1, 10+1):
    df[f'R_dec{i}'] = df[f'R_dec{i}']/(1 + df['inflation'])

R = np.array(df.iloc[:,3:13]) #284 by 10
c_growth = np.array(df.iloc[:, 13]).T #284 by 1

# Functions-----------------------------------------------------------

def mom(theta):

    beta, alpha = theta
    T = R.shape[0]

    f = (beta*np.multiply(R, np.matrix(((c_growth)**(-alpha))).T) - 1)

    return f

def Q(theta,W):

    '''
    GMM objective function

    '''

    beta, alpha = theta
    T = R.shape[0]
    f = mom(theta)
    g = f.mean(axis=0)
```

```
    return (g@W@g.T)[0,0]

def Shat(theta):

    beta, alpha = theta
    T = R.shape[0]
    f = mom(theta)

    return (1/T)*f.T@f

def corr_(theta, name):

    beta, alpha = theta
    T = R.shape[0]
    f = mom(theta)
    f_df = pd.DataFrame(f)
    f_df.columns = range(1, 10+1)
    corr_matrix = f_df.corr()
    sn.heatmap(corr_matrix, annot=True,cmap=sn.cm.rocket_r)
    plt.savefig(f'{folder_path}corr_{name}.png', dpi=1000)
    plt.clf()

    return None

def Shat_NW(theta, q=4):

    beta, alpha = theta
    T = R.shape[0]

    S = 0

    f = mom(theta)
    f = f - f.mean(axis=0)

    for v in range(1, q+1):
        G = (1/T)*f[v:].T@f[:-v]
        S = S + (1 - (v/(q+1)))*(G + G.T)

    G = (1/T)*f.T@f
    S = S + G

    return S

def GMM(NW=False, Andrews=False):

    '''
```

```python
    2-stage GMM

    '''

    M = R.shape[1]
    T = R.shape[0]

    # Stage 1
    theta0 = [1,1]
    W0 = np.identity(M)
    model = minimize(Q, x0=theta0, args=(W0), method = 'Nelder-Mead',\
                     options={'fatol':10**(-30), 'xatol': 0.0001})
    theta1 = model.x

    print(f'First Stage Estimates: {theta1}')

    # Stage 2
    if NW == True:
        W1 = np.linalg.inv(Shat_NW(theta1))
    else:
        W1 = np.linalg.inv(Shat(theta1))
    model = minimize(Q, x0=theta1, args=(W1), method = 'Nelder-Mead',\
                     options={'fatol':10**(-30), 'xatol': 0.0001})
    theta2 = model.x

    print(f'Second Stage Estimates: {theta2}')

    J = T*model.fun

    return theta2, W1, J


def calc_se(theta, W):

    beta, alpha = theta
    T = R.shape[0]
    D_list = []

    for i in range(R.shape[1]):

        def mom(theta):
            beta, alpha = theta
            return (beta*R[:, i]*(c_growth)**(-alpha) - 1).mean()

        jac = jacobian(mom)
```

```
        D_list.append(jac(theta))

    D = np.vstack(D_list)

    sandwich = np.linalg.inv(D.T@W@D)

    se = np.diag((sandwich*(1/T)))**(1/2)

    return se


# Full Sample

theta2, W2, J = GMM()
se = calc_se(theta2, W2)
p_value = 1 - stats.chi2.cdf(J, df=8)

theta2_NW, W2_NW, J_NW = GMM(NW=True)
se_NW = calc_se(theta2_NW, W2_NW)
p_value_NW = 1 - stats.chi2.cdf(J_NW, df=8)


# Up to and including 2019

c_growth = np.array(df[df['year'] <= 2019].iloc[:, 13]).T
R = np.array(df[df['year'] <= 2019].iloc[:,3:13])

theta2_2019, W2_2019, J_2019 = GMM()
se_2019 = calc_se(theta2_2019, W2_2019)
p_value_2019 = 1 - stats.chi2.cdf(J_2019, df=8)

theta2_2019_NW, W2_2019_NW, J_2019_NW = GMM(NW=True)
se_2019_NW = calc_se(theta2_2019_NW, W2_2019_NW)
p_value_2019_NW = 1 - stats.chi2.cdf(J_2019_NW, df=8)

# Correlation matrix
corr_(theta2, 'longsample')
corr_(theta2_2019, 'shortsample')
```