Consumption-based CAPM and GMM

Table of Contents

Introduction

This script analyses the asset pricing ability of the conventional Consumption-based CAPM (CCAPM) on a specific set of test assets. Specifically, we ask whether the CCAPM can explain the cross-sectional variation in excess returns among decile-sorted, value-weighted momentum portfolios. Those returns will be analysed more in detail in Week 7, and you will see their construction threre.

The CCAPM reads in its SDF representation, using excess returns,

$$E_t\left[\delta\left(\frac{c_{t+1}}{c_t}\right)^{-\rho}\left(r_{\mathrm{it}+1}-r_{\mathrm{ft}+1}\right)\right)\Bigg] = 0$$

for all $i = 1, \ldots, n$ assets. Nota that $R_{it} = 1 + r_{it}$ is the gross return such that excess gross returns imply $R_{it} - R_{ft} = 1 + r_{it} - (1 + r_{ft}) = r_{it} - r_{ft}$. Its unconditional implications are summarized as

$$E\left[\left(\delta\left(\frac{c_{t+1}}{c_t}\right)^{-\rho}\left(r_{\mathrm{it}+1}-r_{\mathrm{ft}+1}\right)\right)\otimes Z_t\right] = 0,$$

where Z_t is a vector of instruments, including a constant as the first element. We will here consider two applications. One that uses no instruments in addition to the constant and another that uses four instruments in addition to the constant. We will use lagged consumption-growth as instruments for illustrative purposes. This choise is also common in the literature, though also dividend-price ratios, size and value spreads, and other variables have been used.

Data on stock returns and the risk-free rate is obtained from Kenneth French's data library, and we use in this example quarterly data. Data on consumption is the real nondurables plus services personal expenditure per capita series taken from St. Louis Federal Reserve database.

Since consumption inherently is a flow variable, and there is no point-in-time information as for returns, we need an assumption on the timing of actual consumption. The beginning of-period (BOP) timing convention assumes that consumption during period *t* takes place at the beginning of period *t*, while the end-of-period (EOP) timing convention assumes that it takes place at the end of period *t*. Most existing cross-sectional studies have adopted the EOP timing convention, although there are no definite theoretical reasons for choosing the

EOP convention over the BOP convention. When we use BOP, we match $\frac{c_t}{c_{t-1}}$ with $r_{\text{it}+1}$, whereas EOP matches $\frac{c_{t+1}}{c_t}$ with $r_{\text{it}+1}$. We employ EOP below, which is most often adopted in the literature.

Loading and structuring data

We begin by loading pre-processed data on raw returns, the risk-free rate and $\frac{c_{t+1}}{c_t}$. The file contains data from 1950 to 2018. Note that returns and the risk-free rate are expressed in percentages when downloaded (i.e. multiplied by 100).

```
% Housekeeping
clear;
clc;

% Loading CRSP data
load('momentumConsDataQ.mat');

% Define data variables and get dimensions
rf       = table2array(momentumConsData(:,12));
retMom   = table2array(momentumConsData(:,2:11));
c        = table2array(momentumConsData(:,1));

[nObs,nAss] = size(retMom);
```

Moment conditions

Now we set uo the (unconditional) moment conditions implied by the CCAPM. We will construct a new function that contains the moment conditions. It is called fMoments_CCAPM().

```
% The function has two outputs: The sample average of moments and their time series

% Purely to see it workings, let us specify some input variable
ret      = retMom;
z        = ones(nObs,1);
param    = [0.95 5];
cons     = c;

% Apply the function
[gT,GT] = fMoments_CCAPM(param,ret,rf,cons,z);

% Show gT
gT
```

```
gT = 1×11
   -0.0340    0.0083    0.0072    0.0066    0.0066    0.0058    0.0060    0.0054 ···
```

Object function

The object function we now want to minimize is given by

$$Q_T = g_T(\theta)' A_T g_T(\theta)$$

for some weighting matrix A_T. The function fGMM_obj() computes Q_T.

```
% To see its workings, apply the function with identity matrix as weighting
AT      = eye(size(gT,2));
QT      = fGMM_obj(param,ret,rf,cons,z,AT)
```

```
QT = 0.0015
```

Long-run covariance matrix (S) and gradient (D)

Now we set up the functions for computing the long-run covariance matrix (S) of the sample moments, the gradient (D) for computing standard errors, and the optimal weighting matrix in the second-stage GMM. They are called fLongRunHAC() and fGradient().

```
% To see its workings, apply the estimator for the long-run covariance of
% the sample moments

% Set lags of bandwidth (this should be done thoroughly)
flagAndrews = 1;
nLags        = 4; % Irellevant when flagAndrews == 1
S            = fLongRunHac(GT,flagAndrews,nLags);

% The gradient obtains as
D         = fGradient(param,ret,rf,cons,z);
```

GMM estimation using only a constant as instrument

We are now ready to conduct our GMM estimation. This can be done as first stage GMM (using the identity matrix as weighting matrix), second stage (using, subsequently, the inverse of the long-run covariance matrix), and iterated GMM (continuing updating the estimates and the long-run covariance matrix). Estimation is done with the function fGMM(). Recall that the GMM estimator is defined as

$$\hat{\theta} = \text{argmin}_\theta\, g_T(\theta)' A_T g_T(\theta).$$

There exists a variety of numerical optimizers in Matlab. We will use fmincon() here, but one might also use fminsearch(), fminunc(), or another, and one may add a pertubation of starting values to appropriately search for global optima, using e.g. multistart() or globalsearch().

```
% Set starting values of parameters
delta0   = 0.95;
rho0     = 5;
param0   = [delta0,rho0];
flagAndrews = 1;
nLags    = 4; % irrelevant when flagAndrews == 1

% Specify instruments
z        = ones(nObs,1);

% Specify nIter that determines whether we use first, second or iterated GMM
nIter    = 6;
```

```
% Apply fGMM() to obtain GMM estimates and output
res      = fGMM(param0,ret,rf,cons,z,flagAndrews,nLags,nIter);
```

```
Parameters after optimization stage:1
param = 1×2
    0.6996   91.4125
Parameters after optimization stage:2
param = 1×2
    0.8179   64.7944
Parameters after optimization stage:3
param = 1×2
    0.8335   59.5574
Parameters after optimization stage:4
param = 1×2
    0.8355   58.5818
Parameters after optimization stage:5
param = 1×2
    0.8352   58.5826
Parameters after optimization stage:6
param = 1×2
    0.8352   58.5826
```

```
% Have a look at the output
res.theta
```

```
ans = 1×2
    0.8352   58.5826
```

```
res.stdErr
```

```
ans = 1×2
    0.1122   33.6986
```

```
res.tStat
```

```
ans = 1×2
    7.4459    1.7384
```

```
res.J
```

```
ans = 7.1764
```

```
res.Jpval
```

```
ans = 0.7087
```

Try to experiment with choice of iterations in the GMM estimator to understand its impact on the standard errors.

No matter the number of iterations, we observe that the parameter estimates are $\hat{\theta} = (\hat{\delta}, \hat{\rho})'$ such that the value of the subjective discount factor is to the low side, yet the estimate of the relative risk aversion parameter is very high (much in excess of the reasonable range of $(0, 10])$. This is, yet another, sign of the equity premium puzzle...

GMM estimation using additional instrument(s)

We now consider the case where we use first, second, third, and fourth, lag of consumption growth as instruments, in addition to the constant. We keep everything else constant and there is, therefore, no need for changing any other inputs to the function other than z.

```
% Specify instruments
z1       = [0;cons(1:end-1)];  % we set first element equal to zero for convienece
z2       = [zeros(2,1);cons(1:end-2)]; % we set first two elements equal to zero for cor
z3       = [zeros(3,1);cons(1:end-3)]; % ...
z4       = [zeros(4,1);cons(1:end-4)]; % ...
z        = [ones(nObs,1) z1 z2 z3 z4]; % collect

% Specify nIter that determines whether we use first, second or iterated
% GMM
nIter    = 125;

% Apply fGMM() to obtain GMM estimates and output
res      = fGMM(param0,ret,rf,cons,z,flagAndrews,nLags,nIter);
```

```
Parameters after optimization stage:1
param = 1×2
    0.7018   90.6482
Parameters after optimization stage:2
param = 1×2
    0.7223   98.5390
Parameters after optimization stage:3
param = 1×2
    0.7390  101.2647
Parameters after optimization stage:4
param = 1×2
    0.7500  101.3769
Parameters after optimization stage:5
param = 1×2
    0.7577  100.3381
Parameters after optimization stage:6
param = 1×2
    0.7635   98.8964
Parameters after optimization stage:7
param = 1×2
    0.7682   97.3862
Parameters after optimization stage:8
param = 1×2
    0.7721   95.9444
Parameters after optimization stage:9
param = 1×2
    0.7756   94.6157
Parameters after optimization stage:10
param = 1×2
    0.7786   93.4114
Parameters after optimization stage:11
param = 1×2
    0.7813   92.3252
Parameters after optimization stage:12
param = 1×2
    0.7837   91.3475
Parameters after optimization stage:13
param = 1×2
    0.7859   90.4649
Parameters after optimization stage:14
param = 1×2
```

```
    0.7874   89.9157
Parameters after optimization stage:15
param = 1×2
    0.7888   89.3868
Parameters after optimization stage:16
param = 1×2
    0.7901   88.8886
Parameters after optimization stage:17
param = 1×2
    0.7912   88.4243
Parameters after optimization stage:18
param = 1×2
    0.7922   87.9953
Parameters after optimization stage:19
param = 1×2
    0.7932   87.5987
Parameters after optimization stage:20
param = 1×2
    0.7941   87.2319
Parameters after optimization stage:21
param = 1×2
    0.7949   86.8961
Parameters after optimization stage:22
param = 1×2
    0.7956   86.5850
Parameters after optimization stage:23
param = 1×2
    0.7963   86.2961
Parameters after optimization stage:24
param = 1×2
    0.7970   86.0302
Parameters after optimization stage:25
param = 1×2
    0.7976   85.7854
Parameters after optimization stage:26
param = 1×2
    0.7981   85.5595
Parameters after optimization stage:27
param = 1×2
    0.7986   85.3497
Parameters after optimization stage:28
param = 1×2
    0.7991   85.1565
Parameters after optimization stage:29
param = 1×2
    0.7995   84.9767
Parameters after optimization stage:30
param = 1×2
    0.7999   84.8102
Parameters after optimization stage:31
param = 1×2
    0.8003   84.6567
Parameters after optimization stage:32
param = 1×2
    0.8006   84.5155
Parameters after optimization stage:33
param = 1×2
    0.8009   84.3845
Parameters after optimization stage:34
param = 1×2
    0.8012   84.2620
Parameters after optimization stage:35
param = 1×2
```

```
    0.8015   84.1474
Parameters after optimization stage:36
param = 1×2
    0.8017   84.0440
Parameters after optimization stage:37
param = 1×2
    0.8019   83.9472
Parameters after optimization stage:38
param = 1×2
    0.8022   83.8564
Parameters after optimization stage:39
param = 1×2
    0.8024   83.7736
Parameters after optimization stage:40
param = 1×2
    0.8025   83.6961
Parameters after optimization stage:41
param = 1×2
    0.8027   83.6255
Parameters after optimization stage:42
param = 1×2
    0.8029   83.5581
Parameters after optimization stage:43
param = 1×2
    0.8030   83.4927
Parameters after optimization stage:44
param = 1×2
    0.8032   83.4356
Parameters after optimization stage:45
param = 1×2
    0.8033   83.3829
Parameters after optimization stage:46
param = 1×2
    0.8034   83.3330
Parameters after optimization stage:47
param = 1×2
    0.8035   83.2831
Parameters after optimization stage:48
param = 1×2
    0.8036   83.2419
Parameters after optimization stage:49
param = 1×2
    0.8037   83.2029
Parameters after optimization stage:50
param = 1×2
    0.8038   83.1660
Parameters after optimization stage:51
param = 1×2
    0.8039   83.1333
Parameters after optimization stage:52
param = 1×2
    0.8040   83.1014
Parameters after optimization stage:53
param = 1×2
    0.8040   83.0723
Parameters after optimization stage:54
param = 1×2
    0.8041   83.0456
Parameters after optimization stage:55
param = 1×2
    0.8042   83.0187
Parameters after optimization stage:56
param = 1×2
    0.8042   82.9971
Parameters after optimization stage:57
```

```
param = 1×2
    0.8043   82.9749
Parameters after optimization stage:58
param = 1×2
    0.8043   82.9547
Parameters after optimization stage:59
param = 1×2
    0.8044   82.9351
Parameters after optimization stage:60
param = 1×2
    0.8044   82.9176
Parameters after optimization stage:61
param = 1×2
    0.8044   82.9006
Parameters after optimization stage:62
param = 1×2
    0.8045   82.9006
Parameters after optimization stage:63
param = 1×2
    0.8045   82.8837
Parameters after optimization stage:64
param = 1×2
    0.8045   82.8703
Parameters after optimization stage:65
param = 1×2
    0.8046   82.8550
Parameters after optimization stage:66
param = 1×2
    0.8046   82.8438
Parameters after optimization stage:67
param = 1×2
    0.8046   82.8351
Parameters after optimization stage:68
param = 1×2
    0.8046   82.8260
Parameters after optimization stage:69
param = 1×2
    0.8046   82.8260
Parameters after optimization stage:70
param = 1×2
    0.8046   82.8260
Parameters after optimization stage:71
param = 1×2
    0.8047   82.8127
Parameters after optimization stage:72
param = 1×2
    0.8047   82.8029
Parameters after optimization stage:73
param = 1×2
    0.8047   82.7933
Parameters after optimization stage:74
param = 1×2
    0.8047   82.7875
Parameters after optimization stage:75
param = 1×2
    0.8047   82.7875
Parameters after optimization stage:76
param = 1×2
    0.8047   82.7796
Parameters after optimization stage:77
param = 1×2
    0.8048   82.7716
Parameters after optimization stage:78
param = 1×2
    0.8048   82.7711
```

```
Parameters after optimization stage:79
param = 1×2
    0.8048   82.7677
Parameters after optimization stage:80
param = 1×2
    0.8048   82.7677
Parameters after optimization stage:81
param = 1×2
    0.8048   82.7599
Parameters after optimization stage:82
param = 1×2
    0.8048   82.7554
Parameters after optimization stage:83
param = 1×2
    0.8048   82.7554
Parameters after optimization stage:84
param = 1×2
    0.8048   82.7533
Parameters after optimization stage:85
param = 1×2
    0.8048   82.7527
Parameters after optimization stage:86
param = 1×2
    0.8048   82.7483
Parameters after optimization stage:87
param = 1×2
    0.8048   82.7483
Parameters after optimization stage:88
param = 1×2
    0.8048   82.7443
Parameters after optimization stage:89
param = 1×2
    0.8048   82.7398
Parameters after optimization stage:90
param = 1×2
    0.8048   82.7386
Parameters after optimization stage:91
param = 1×2
    0.8048   82.7386
Parameters after optimization stage:92
param = 1×2
    0.8048   82.7373
Parameters after optimization stage:93
param = 1×2
    0.8048   82.7373
Parameters after optimization stage:94
param = 1×2
    0.8048   82.7373
Parameters after optimization stage:95
param = 1×2
    0.8048   82.7373
Parameters after optimization stage:96
param = 1×2
    0.8048   82.7373
Parameters after optimization stage:97
param = 1×2
    0.8048   82.7373
Parameters after optimization stage:98
param = 1×2
    0.8048   82.7373
Parameters after optimization stage:99
param = 1×2
    0.8048   82.7324
Parameters after optimization stage:100
param = 1×2
```

```
    0.8049   82.7312
Parameters after optimization stage:101
param = 1×2
    0.8049   82.7311
Parameters after optimization stage:102
param = 1×2
    0.8049   82.7310
Parameters after optimization stage:103
param = 1×2
    0.8049   82.7310
Parameters after optimization stage:104
param = 1×2
    0.8049   82.7310
Parameters after optimization stage:105
param = 1×2
    0.8049   82.7289
Parameters after optimization stage:106
param = 1×2
    0.8049   82.7289
Parameters after optimization stage:107
param = 1×2
    0.8049   82.7289
Parameters after optimization stage:108
param = 1×2
    0.8049   82.7289
Parameters after optimization stage:109
param = 1×2
    0.8049   82.7289
Parameters after optimization stage:110
param = 1×2
    0.8049   82.7274
Parameters after optimization stage:111
param = 1×2
    0.8049   82.7263
Parameters after optimization stage:112
param = 1×2
    0.8049   82.7241
Parameters after optimization stage:113
param = 1×2
    0.8049   82.7241
Parameters after optimization stage:114
param = 1×2
    0.8049   82.7241
Parameters after optimization stage:115
param = 1×2
    0.8049   82.7218
Parameters after optimization stage:116
param = 1×2
    0.8049   82.7208
Parameters after optimization stage:117
param = 1×2
    0.8049   82.7208
Parameters after optimization stage:118
param = 1×2
    0.8049   82.7208
Parameters after optimization stage:119
param = 1×2
    0.8049   82.7187
Parameters after optimization stage:120
param = 1×2
    0.8049   82.7185
Parameters after optimization stage:121
param = 1×2
    0.8049   82.7185
Parameters after optimization stage:122
```

```
param = 1×2
    0.8049   82.7185
Parameters after optimization stage:123
param = 1×2
    0.8049   82.7184
Parameters after optimization stage:124
param = 1×2
    0.8049   82.7184
Parameters after optimization stage:125
param = 1×2
    0.8049   82.7184
```

```
% Have a look at the output
res.theta
```

```
ans = 1×2
    0.8049   82.7184
```

```
res.stdErr
```

```
ans = 1×2
    0.0120    3.7839
```

```
res.tStat
```

```
ans = 1×2
   67.1675   21.8604
```

```
res.J
```

```
ans = 79.8524
```

```
res.Jpval
```

```
ans = 0.0127
```

We now reject the CCAPM based on the test of over-identifying restrictions, suggesting that there is information in lagged consumption grwoth that informs about future pricing errors of the model.