Classical Minimum Distance (CMD) Estimation Code for "Top Wealth in America: New Estimates under Heterogeneous Returns"

September 16, 2022

The .tex file describes the Matlab code that performs the Classical Minimum Distance (CMD) estimation of the parameters entering a model of fixed income returns with factor exposure for "Top Wealth in America: New Estimates under Heterogeneous Returns". For more information on the CMD model, please refer to section K in the appendix.

0 Files

This section will provide a brief overview of the folders/files.

- **build**: Contains the code that was used to construct the inputs for the CMD parameter estimation.
  - **build_data_for_cmd_top01_bot99pt9.do**: Constructs **cmdinputs_top01bot99pt9_20210615.csv**, the inputs for our analysis where we partition the groups into the top 0.1% and the bottom 99.9%.
  - **build_data_for_cmd_top1_bot99.do**: Similarly, this do-file constructs **cmdinputs_top1bot99_20210615.csv**, the inputs for our analysis where we partition the groups into the top 1% and the bottom 99%.
- **figs**: Contains figures generated by **cmd_estimation_prgm.m**.
- **tex**: Contains subfolders with .tex files with the estimated parameters.
- **cmd_estimation_prgm.m**: This is the main program that is used to construct the CMD estimates. It is called in **run_all_top01bot99pt9.m & run_all_top1bot99.m**.

Any other programs not specifically listed in this section are helper functions that are called within **cmd_estimation_prgm.m**.

- **run_all_top01bot99pt9.m & run_all_top1bot99.m**: These files can be run to produce the final CMD estimates.

Each subsequent section of this note corresponds to a section of the main script, **cmd_estimation_prgm.m**.

1 Environment

Again, the main scripts to run are **run_all_top01bot99pt9.m & run_all_top1bot99.m**, which primarily call **cmd_estimation_prgm.m**. This program solves the CMD minimization problem using IPOPT (**I**nterior **P**oint **Opt**imizer), an open source software package for large-scale nonlinear optimization. IPOPT can be easily installed on Matlab as part of the OPTI Toolbox. The OPTI Toolbox can be downloaded from https://github.com/jonathancurrie/OPTI. The statistics and machine learning toolbox in MATLAB will also need to be installed if it has not already been. After installing IPOPT, the main root directory at the top of this section can be changed to any local directory.

2 Compute Sample Moments

In this section, the code computes a number of sample moments:

(1) The function **mhat.m** computes the following 20 sample moments from the main estimation dataset: $\widehat{\mu}_{y_1}$, $\widehat{\mu}_{y_2}$, $\widehat{\mu}_a$, $\widehat{\mu}_{r_1}$, $\widehat{\mu}_{r_2}$, $\widehat{\sigma}^2_{y_1}$, $\widehat{c}_{y_1,y_2}$, $\widehat{c}_{y_1,a}$, $\widehat{c}_{y_1,r_I}$, $\widehat{c}_{y_1,r_C}$, $\widehat{\sigma}^2_{y_2}$, $\widehat{c}_{y_2,a}$, $\widehat{c}_{y_2,r_I}$, $\widehat{c}_{y_2,r_C}$, $\widehat{\sigma}^2_a$, $\widehat{c}_{a,r_I}$, $\widehat{c}_{a,r_C}$, $\widehat{\sigma}^2_{r_I}$, $\widehat{c}_{r_I,r_C}$, $\widehat{\sigma}^2_{r_C}$. These sample moments are subsequently stacked into a vector.

(2) The function **mhat_scf.m** computes the following 6 sample moments from the SCF dataset: $\widehat{c}_{r_1,r_2}$, $\widehat{c}_{r_1,a_1}$, $\widehat{c}_{r_1,a_2}$, $\widehat{c}_{r_2,a_1}$, $\widehat{c}_{r_2,a_2}$, $\widehat{c}_{a_1,a_2}$. These sample moments are subsequently stacked into a vector.

(3) The function **mhat_scf_param4.** computes the following 14 sample moments: $\widehat{\mu}_{r1}$, $\widehat{\mu}_{r2}$, $\widehat{\mu}_{a1}$, $\widehat{\mu}_{a2}$, $\widehat{\sigma}^2_{r1}$, $\widehat{\sigma}^2_{r2}$, $\widehat{\sigma}^2_{a1}$, $\widehat{\sigma}^2_{a2}$, $\widehat{c}_{r_1,r_2}$, $\widehat{c}_{r_1,a_1}$, $\widehat{c}_{r_1,a_2}$, $\widehat{c}_{r_2,a_1}$, $\widehat{c}_{r_2,a_2}$, $\widehat{c}_{a_1,a_2}$. These sample moments are subsequently stacked into a vector.

3 Solve CMD Minimization Problem with IPOPT

In this section, the code uses IPOPT to solve the CMD minimization problem and compute estimated parameters in four versions of the model:
The function **cmd_ipopt4.m** computes estimated parameters in the 4-parameter version of the model. The functions **cmd_objective4.m** and **cmd_gradient4.m** respectively compute the objective function and gradient of the CMD minimization problem in the 4-parameter model.

4 Evaluate Model Moments at Estimated Parameters

In this section, the code evaluates model moments at the estimated parameter vector. The function **estmoms.m** takes as input the estimated parameter vector and computes model-implied moments.

5 Back Out Parameters of Interest

In this section, the code computes group-specific rates of return for the model. The function **computerates.m** computes model-implied log rates of return for both groups and subsequently exponentiates log returns to back out returns in levels. The output is two time series of group-specific rates of return spanning the time period in the main estimation sample.

6 Compute Standard Errors with Block Bootstrap

In this section, the code computes standard errors for both the estimated parameter vector and the time series of group-specific rates of return. Due to the difficulty in computing analytical standard errors, the function **stderr.m** computes standard errors with bootstrap. Because the main estimation sample consists of time series data and simple bootstrap hinges on independent and identically distributed sampling, block bootstrap is the most appropriate choice. If, for instance, standard errors were clustered at the county level, block bootstrap would sample (with replacement) all observations within counties to preserve the within-cluster correlation of residuals. When working with time series data, sampling with replacement is less straightforward because in principle an empiricist could assume autocovariances not to be zero over the entire sample period. This would effectively imply using only one block. A more sensible approach is to construct blocks of pre-specified length (i.e., a given number of time periods) and sample them with replacement. Blocks could be overlapping

(e.g. $b_1 = \{2007, 2008, 2009\}$, $b_2 = \{2008, 2009, 2010\}$, $b_3 = \{2009, 2010, 2011\}$) or non-overlapping (e.g. $b_1 = \{2007, 2008, 2009\}$, $b_2 = \{2010, 2011, 2012\}$, $b_3 = \{2013, 2014, 2015\}$). The current version of the code uses overlapping blocks. An additional choice a researcher must make is block length, i.e., the number of time periods contained in each block. A rule of thumb reported in a few papers and textbooks is to choose $T^{\frac{1}{3}}$, where T is the number of time periods in the sample. Because the main estimation sample spans 52 years and $28^{\frac{1}{3}} \approx 3.04$, the current version of the code sets block length to 3.

7 Plot Group-Specific Rates of Return

In this section, the code computes 95% confidence intervals around the time series of group-specific rates of return for each of the four versions of the model. Specifically, the function **computeci95** adjusts the t-statistic to the number of parameters (and thus degrees of freedom) in each model and computes bounds for the corresponding 95% confidence intervals. The time series of group-specific rates of return are subsequently plotted. The resulting graphs are saved in the *figs* folder.