Documentation for:

Growth-Rate and Uncertainty Shocks in Consumption:

Cross-Country Evidence

Emi Nakamura
Columbia University

Dmitriy Sergeyev
Bocconi University

Jón Steinsson
Columbia University

July 31, 2016

1 Overview

The estimation of the empirical model is carried out using the JAGS software package.[1] We call JAGS from R and do post estimation analysis in R. We also perform a few calculations using Stata. When you unzip the `LRRDataAndPrograms.zip` file, you will find this readme file and two folders called `bugsForDist` and `stataForDist`. `bugsForDist` contains data and programs to replicate the part of our analysis that we performed in JAGS and R. `stataForDist` contains files to perform the analysis that we did in Stata.

2 Needed Software

You will need to install R and JAGS. These programs are freely available on the Internet. You will also need Stata.

3 Directory Structure of bugsForDist

The directory `bugsForDist` contains two directories: `data` and `sandbox`. The folder `sandbox` contains our code and the data. The folder `data` is where output is saved. Within each of sandbox and data there is a directory called `v11`. Within `v11`, there are three directories `v11.32d27`,

[1] JAGS is an alternative to software called WinBugs. The advantage of JAGS is that it can work both on Windows and Unix operation systems.

v11.33d3, v11.35d3. v11.32d27 directory contains files for the full version of the model. Directory v11.33d3 contains files for the full model that is estimated using post WWII data. Directory v11.35d3 contains files for the simple model. The directory /bugsForDist/sandbox/v11 contains the file cdata20111116.txt with the consumption data used in the estimation. The directories /bugsForDist/sandbox/v11/v11.32d27, /bugsForDist/sandbox/v11/v11.33d3, /bugsForDist/sandbox/v11/v11.35d3 contain the file disInd.Rda. The disaster dummies estimated in Barro, Nakamura, Steinsson, and Ursa (2013) are stored in this file.

4 Files in /bugsForDist/sandbox/v11/v11.32d27

This folder contains the files that estimate the *full* version of the model and calculate asset prices for the resulting consumption process. Here is a brief description of how to perform different parts of the analysis in our paper for this version of the model. Similar steps should be taken for estimation of the full model on the post WWII data (v11.33d3) and the simple model (v11.35d3).

4.1 Estimate the model

Estimating the model involves three files: lrrmodel.bug, runModel.R, funcs.R. File lrrmodel.bug contains a description of the model in JAGS notation. runModel.R is the R program you run to estimate the model. runModel.R makes use of R functions contained in funcs.R. In addition, if you run the estimation on a unix platform, file runModel.sh is a shell script that can be used to call runModel.R.

The core of runModel.R is a loop. Each iteration of the loop calls JAGS and asks it to return Niter draws from the posterior distribution of the unknown parameters, where Niter is specified at the top of funcs.R. The loop runs for NRuns iterations, where NRuns is also specified towards the top of funcs.R. After each iteration of the loop, runModel.R saves the output of that iteration in a file named something like bugsSimSave.XX1.1.Rda. We typically set Niter to $50,000$ and NRuns to 100 (this takes about a week on a standard desktop, so start with something smaller). We have adopted this setup to be able to stop the estimation before it finishes without loosing too much output. You should also specify the prior type towards the top of funcs.R by setting priorType to 'XX' or 'YY'. The results in the paper are based on two chains with prior type 'XX' and two chains with prior type 'YY'. If you are interested in running more than one chain with one run of runModel.R and saving the output from these chains in the same folder for aggregation, you should

use `nOfChain` to give each chain a number. Otherwise, you can run the program several times in different folders to create separate chains. In addition to changing these variables in `getControlVar` (the first function in `funcs.R`), you will need to change the directories in `getDirInfo` (the second function in `funcs.R`).

4.2 Aggregating Output and Calculating Results

Suppose you have run `runModel.R` with `nIter` = 50,000, `NRuns` = 100, `priorType` = `'XX'`, and `nOfChain` = 1. Then `/bugsForDist/data/v11/v11.32d27` will contain one hundred files with names `bugsSimSave.XX1.1.Rda`, ..., `bugsSimSave.XX1.100.Rda`. The next step is to aggregate these files. This is done using the files `runAgg.R` and `funcsAgg.R`. File `runAgg.R` is the file you run and `funcsAgg.R` is a support file with a bunch of functions that perform the aggregation and produce various results using the aggregated output. `runAgg.sh` is a shell script that calls `runAgg.R`.

The first step of this process is to open `funcsAgg.R`, and in the function `getRunsToUse` (the second function in `funcsAgg.R`), change `XX1ToUse` to $1:100$ and `XX1Dir` to the folder where the output was saved if the folder was different from `outDir`. If more than one chain was produced with one run of `runModel.R`, you may want to combine the chains by setting `combineChains = 1` in function `getControlAgg`. You also need to specify if the posterior distribution from these combined chains will be used or only from chain 1. This can be done by setting `usePooledChains = 1` in function `getControlAgg`.

Then you should be able to run `runAgg.R`. If you want to drop some output from the beginning of the Markov chain as burnin, you can do this by not using the first several batches and thus set `XX1ToUse` to $21:100$.

The rest of `runAgg.R` calls various functions in `funcsAgg.R` that produce various results. Some results are printed on the screen. But all results are saved into `/bugsForDist/data/v11/v11.32d27`. Here is a brief description of what these functions do:

- `summaryCommonParameters` produces results reported in table 1 of the paper and saves them in a file named `Parameters.Common.PooledChains.csv` (if pooled chains are used) or `Parameters.Common.Chain.1.csv` (if only chain 1 is used).

- `summaryCountryParameters` produces the results reported in tables 3 and A.2 of the paper and saves them in `Parameters.Country.PooledChains.csv` (if pooled chains are used).

- `simPlotMany` produces a large number of plots. These plots are saved as pdf files with names such as `bigData.Argentina.pdf`, `bigData.CommonParameters.pdf`, and `dC_XX_sigmaSq.pdf`.

- `saveWorldComponents` produces the posterior average of estimated world components $x_{W,t}$ and $\sigma^2_{W,t}$ and saves them in files `xxW.PooledChains.csv` and `sigmaSqW.PooledChains.csv` (if pooled chains are used). They are plotted in figures 2 and 3 of the paper.

- `modelStatesStats` computes statistics by simulating long data and saves the results in `C.stats.simData.csv`. In particular, this function computes the frequency of hitting the lower bound for volatility that is reported in footnote 8 in the paper.

- `modelConsumptionSDandCorr` computes consumption growth rate standard deviations and correlations using Monte Carlo simulations of the model. The results are saved in files `C.stats.simData.Autocorr.Dis.ByCountry.csv` (when consumption includes disasters), `C.stats.simData.Autocorr.noDis.ByCountry.csv` (when consumption excludes disasters), `C.stats.simData.Autocorr.noDis.MedCountry.csv`, `C.stats.simData.CrossCorrMed.csv`, `C.stats.simData.CrossCorrUS.csv`. The results are reported in table 4 of the paper.

- `modelConsumptionStats` computes consumption growth rate statistics using Monte Carlo simulations of the model. The results are save in `C.stats.simData.Dis.ByCountry.csv`, `C.stats.simData.noDis.ByCountry.csv`, `C.stats.simData.postWWII.ByCountry.csv`. The results are reported in table 4 of the table.

- `VRsimMonteCarloConsumption` creates variance ratios for consumption growth for all countries at 15-year horizon using Monte Carlo simulations. The results are saved in files with names like `VR.C.simData.Dis.MonteCarlo.15.ByCountry.csv`. The results are reported in table 4 of the paper.

- `realConsumptionStats` calculates standard deviation, auto-correlation, and cross-correlation of actual consumption growth and saves it in files with names like `C.stats.realData.CrossCorrUS.csv`. The results are reported in table 4 of the paper.

- `realConsumptionLevel` saves consumption series for selected countries in excel file `C.levelPath.realData.PooledChains.csv`. Figure 1 of the paper plots these data for France.

- `VRrealCmany` creates variance ratios for actual consumption growth for all horizons upto 30 years and saves them in `VR.C.realData.PooledChains.csv`, `VR.C.realDataNoDis.PooledChains.csv`. The results are reported in table 4.

- `summarySV` creates series of average estimated stochastic volatility for all countries and saves in `SV.realData.PooledChains.csv`. The results are plotted in figure 4.

- `summarySVsomeCountries` creates series of average and quantiles of estimated stochastic volatility for certain countries. The results are saved in files like `SV.realData.United.Kingdom.PooledChains.csv`.

- `summarySVWorld` computes quantiles of the world stochastic volatility. The results are saved in `SV.realData.World.PooledChains.csv`.

- `compareLRRinNSSvsBY` compares the amount of long-run risks (both growth rate and volatility shocks) across our estimated model, Bansal and Yaron (2004), and Bansal, Kiku, and Yaron (2012). The results are saved in `fractionXX.csv` (these results are presented in table 2 of the paper) and `StatsSVmeasure1.csv`.

- `compareSVinNSSvsBY` compares the amount of stochastic volatility in our estimated model, Bansal and Yaron (2004), and Bansal, Kiku, and Yaron (2012) using the quantile method. The results are saved in `StatsSVmeasure2.csv` and presented in the paper in table 2.

- `VRofSVreal` creates variance ratios for stochastic volatility using actual data at horizon of 15 years. The results are saved in `VR.SV.realData.15.PooledChains.csv` and presented in table 4 of the paper.

- `VRofSVrealMany` creates variance ratios for stochastic volatility using real data for horizons upto 15 and saves them in `VR.SV.realData.Dis.PooledChains.csv` and `VR.SV.realData.noDis.PooledChains.csv`. The results are presented in table 4.

- `VRofSVsimDataMonteCarlo` computes variance ratios for stochastic volatility using simulated data at horizon 15. The results are saved in files like `VR.SV.simData.Dis.MonteCarlo.15.ByCountry.csv`.

4.3 Asset Pricing

The basic asset pricing results in the paper are produced using files `runNAPe.YYY.R` (where `YYY` denotes the name of a particular country, for example, `Australia`, or `manyCountries` when the program is run for all of the country) and `funcsNAPe.R` (or `funcsNAP.R` in case of the simple model). As with `runModel.R`, the file to run is `runNAPe.YYY.R` and the key variables that you need to vary are at the top of `funcsNAPe.R` (in the function `getControlVarNAPi`) and at the top of `runNAPe.YYY.R` (several global variables). The core of the asset pricing code is the function `getPDivs_EZW` in `funcsNAPe.R`, which calculates the price-dividend ratio over the state space for the assets that you are interested in doing asset pricing for. Most of the code above this function is a series of functions that are used in `getPDivs_EZW`. The output from `getPDivs_EZW` is saved in a file named `outputEZW.YYY.Rda`. This file is then used to simulate the model and calculate average asset returns.

We add the possibility to use parallel computing when solving for equilibrium asset prices. This requires an R library called *snow*. To turn on parallel computing, one has to set `GlobalUseSnow = 1` at the top of `runNAPe.R` and set the variable `nprocs` in function `getControlVarNAPi` to the number of processors to be used for computation. If the program is run on a unix operation system, the script files `runNAPe.YYY.sh` must indicate the same number of processors as in `nprocs` by modifying the line: `#PBS -l nodes=2:ppn=16,walltime=11:59:59,mem=23900mb`.

To compute asset prices with constant volatility, one has to set parameters `nCoarseStateGridSigmaSq`, `nCoarseStateGridSigmaSqW`, `nFineStateGridSigmaSq`, `nFineStateGridSigmaSqW` in function `getControlVarNAPi` to 1. To compute asset prices with constant volatility and without persistent growth rate components $x_{i,t}$ and $x_{W,t}$, one has to, in addition, set parameters `nCoarseStateGridXX`, `nCoarseStateGridXXW`, `nFineStateGridXX`, `nFineStateGridXXW` in function `getControlVarNAPi` to 1.

Each program `runNAPe.YYY.R` (where `YYY` is a country name) produces asset prices for individual countries. File `runNAPe.manyCountries.R` puts together the results for individual countries and saves them in `summaryStatsBigFile.Rda`. In addition, `runNAPe.manyCountries.R` generates data for all of the countries using common realization of world growth rate and volatility shocks.

Figures 5-9 and tables 5-8 are produced using `runNAPe.YYY.R` and `funcsNAPe.R`.

5 Statistics on Asset Returns

In the paper, we report a few statistics based on data on the real return on stocks and bills. The Stata code to produce these statistics is in `stataForDist`. The file `clean_barro_ursua.do` reads the historical asset return data, used in Barro and Ursua (2009), into Stata and constructs a data file for analysis. The file `clean_volatility.do` reads estimated stochastic volatility, which is saved by program `runAgg.R` in files like `SV.realData.United.States.PooledChains.csv` in folder `/bugsForDist/data/v11/v11.32d27`, into Stata. The file `clean_gfd.do` reads asset returns obtained from the Global Financial Data into Stata. The file `build.do` calculates several statistics about the asset prices and runs predictability regressions.

The data on asset returns that we use are the data on returns used in Barro and Ursua (2009). We use data on the total return on equity and government bills, and we use data on inflation to deflate these returns. The primary source of these data is Global Financial Data (described in Taylor (2005)). The data on total returns on equity can be found in GFD's "Equity Database," the data on yields of government bills can be found in their "Fixed Income Database," while information on consumer price indices is located with their Economic Database. The series are ordered by country and are available at yearly frequencies with varying starting dates. Barro and Ursua supplement the data from Global Financial Data with data from Dimson, Marsh, and Staunton (2008)—available through a special licence of the EnCorr international data module in Morningstar—for Canada 1900-13, Denmark 1900-14, Italy 1900-05, Netherlands 1900-19, Sweden 1900-01, Switzerland 1900-10, and South Africa 1900-10. Information in the EnCorr module is organized by series, including total equity returns, total returns on government bills, and inflation.[2] Care should be taken in using the Dimson, Marsh, and Staunton (2008) data for later periods, usually wars, with missing entries in Global Financial Data. These Dimson, Marsh, and Staunton (2008) data appear to be generated (for periods such as France 1940 and Portugal 1974-77 when stock-return data seem to be unavailable) by interpolation. Barro and Ursua do not use any of this information. They use stock-price data for Argentina 1900-35 from Nakamura and Zarazaga (2003), for Japan 1893-1914 from Fujino and Akiyama (1977), and for Mexico 1902-29 (missing 1915-18) from Haber, Razo, and Maurer (2003). For Brazil 1900-1942 they used data available from Aldo Musacchio, and for 1945-1953 they used data from Goldsmith (1986). See Barro and Ursua (2009) and Barro and Ursua (2008) for more detailed discussion of the construction of these data. These data are available upon

[2] See further description at http://datalab.morningstar.com/knowledgebase/aspx/files/DMS.doc

request from the authors to researchers who have access to Global Financial Data and Morningstar.

References

Bansal, R., D. Kiku, and A. Yaron (2012): "An Empirical Evaluation of the Long-Run Risks Model for Asset Prices," *Critical Finance Review*, 1, 183–221.

Bansal, R., and A. Yaron (2004): "Risks for the Long Run: A Potential Resolution of Asset Pricing Puzzles," *Journal of Finance*, 59(4), 1481–1509.

Barro, R., E. Nakamura, J. Steinsson, and J. Ursa (2013): "Crises and Recoveries in an Empirical Model of Consumption Disasters," *American Economic Journal: Macroeconomics*, 5(3), 35–74.

Barro, R. J., and J. F. Ursua (2008): "Macroeconomic Crises since 1870," *Brookings Papers on Economic Activity*, 2008, 255–350.

——— (2009): "Stock-Market Crashes and Depressions," NBER Working Paper No. 14760.

Dimson, E., P. Marsh, and M. Staunton (2008): "The Worldwide Equity Premium: A Smaller Puzzle," in *Handbook of the Equity Risk Premium*, ed. by R. Mehra, pp. 467–514, Amsterdam, Holland. Elsevier.

Fujino, S., and R. Akiyama (1977): *Security Prices and Rates of Interest in Japan: 1874-1975*. Hitotsubashi University, Tokyo, Japan.

Goldsmith, R. W. (1986): *Brasil 1850-1984: desenvolvimento financeiro sob um século do inflação*. Harper and Row do Brasil, São Paulo, Brasil.

Haber, S., A. Razo, and N. Maurer (2003): *The Politics of Property Rights: Political Instability, Credible Commitments, and Economic Growth in Mexico, 1876-1929*. Cambridge University Press, New York, NY.

Nakamura, L. I., and C. E. Zarazaga (2003): "Banking and Finance in Argentina in the Period 1900-35," in *A New Economic History of Argentina*, ed. by G. della Paolera, and A. M. Taylor, Cambridge, England. Cambridge University Press.

Taylor, B. (2005): "GFD Guide to Total Returns on Stocks, Bonds and Bills," Available on the Internet from Global Financial Data at www.globalfindata.com.