BLP Assignment for Econ 517
Spring 2017

Jeremy Fox

Due February 13, 2017 on Canvas as a Zip archive

Please carefully read the instructions about group work on the syllabus.

A while back my co-authors (JP Dubé and the late Che-Lin Su) and I wrote some MATLAB code that called the KNITRO optimizer. Rice does not have a site license for KNITRO. However, there is a slightly worse performing but still very good open source optimizer called IPOPT. Recently, Gunnar Heins of the University of Florida rewrote our MATLAB code to use IPOPT instead of KNITRO.

Heins's IPOPT version of the code is on the Canvas site. You need MATLAB and IPOPT for his code to work. Compiling IPOPT by hand takes a while, in my experience. However, for Windows only (sorry Linux and Mac users) there is a pre-compiled IPOPT binary as part of the OPTI Toolbox at

- https://www.inverseproblem.co.nz/OPTI/

1. The **first** part of the assignment is to rewrite Heins's version of my code in Julia (version 0.5), still calling IPOPT. It is easy to install IPOPT for Julia. Use the wrapper package Ipopt.jl. One aspect that is new for 2017 is that you should first try to write the model in the Julia problem definition language JuMP. With JuMP, you do not need to code up the derivatives as JuMP will compute them using automatic differentiation in the background. This will in principle save you lots of time. You can search for documentation on JuMP; keep in mind that JuMP is not a solver. It might require tweaking the syntax in Heins a little to get JuMP to work. I am most concerned about memory (RAM) usage based on my experience with AMPL ten years ago. If you run into memory issues, work on a much smaller problem to debug the code on your desktop or laptop and then pay a few bucks for a cloud service like Amazon EC2 (I recommend an Ubuntu Linux instance as it has been easy to install Julia in the past) to run the full version in a larger memory run. But don't use a pay service to debug your code. You could also use another cloud provider, even Rice's cluster if you have access through whatever means. Ideally, your code should produce exactly the same run of IPOPT (the same starting value, the same fake data, the same sequence of iterations) as Heins's version of my code. You

can export the fake data from MATLAB to Julia, but only the fake data (the equivalent of what real data would look like) can be exported. If you suspect a bug in Heins's code, you need to comprehensively document that the Heins code is in error.

2. You should compute standard errors for the parameters, as my code does. All of the code should run without modification on my machine in Julia 0.5. Do NOT hard code directory paths that I need to change to get it to run on my machine.

3. If you cannot get JuMP to work and can document the precise problem giving you trouble (this is not an excuse to give up), then you can code up the first and second derivatives manually, like in Heins. This will take time so leave time to do this. In this case, the code should use the same IPOPT settings and provide the same inputs, such as gradients, Hessians, sparsity patterns and so forth. Your code should produce exactly the same run of IPOPT (the same starting value, the same fake data, the same sequence of iterations) as Heins's version of my code.

4. The **second** part of the assignment is to compute the own price demand elasticity for each product and then average the own price elasticity across products and markets, weighting each product the same. Include in your PDF document a derivation of the formula for the mean (across both markets and products) own price elasticity that your code calculates, for the BLP demand system specifically. Pay special attention to deriving the elasticity correctly; do not speculate. There is no reason to not be able to properly derive the BLP elasticity. You do not need to compute standard errors for the mean own price demand elasticity, although it is straightforward to do so using the Delta method.

5. For the **third** part of the assignment, increase the number of simulation draws from 100 to 10,000. How do the point estimates, standard errors, and mean own price elasticities change? This change could increase the memory requirement (particularly if you are using JuMP), so be prepared to use a cloud service. Save the fake data from this part in your Zip archive.

6. For the **fourth** part of the assignment, tweak the code to use a different set of 10,000 draws for each market. How do the point estimates, standard errors, and mean own price elasticities change? Save the fake data from this part in your Zip archive.

7. The **fifth** part of the assignment is to compute the 2SLS estimator for the logit model without random coefficients on the fake data from the fourth part. Use the same instruments as in the fourth part. Report point estimates, standard errors and the mean own price elasticity for the logit model without random coefficients. This is an example of estimating a misspecified model as the true data generating process has random coefficients in it. You can use 2SLS in Julia, R, Stata or MATLAB but I advise doing it in R or Stata (or calling an R package in Julia) to make sure your formulas are correct.

8. There is no excuse for confusing the terms "standard deviation" and "standard error." If you do not understand the difference between these terms in general and in a model with random coefficients specifically, please figure them out before undertaking this assignment as you will not be able to learn as much if you are mixed up about basic terminology.

Please upload to Canvas your Julia code, a file with any needed fake data exported from MATLAB, a single PDF document with instructions on how to run your code and all of: the formula for the own-price elasticities for both BLP and the logit without random coefficients as well as the BLP (100 draws and the two runs with 10,000 draws separately) and 2SLS point estimates, standard errors, and mean own price elasticity estimates. All files should be in a single Zip archive.

Note that I do not provide any technical assistance on how to use Windows, Linux, cloud services, shells, MATLAB, IPOPT, Julia and JuMP. It is your job to figure these pieces of software out.