**Data and Code Readme for "Approximating the Cost-of-Living Index for a Storable Good"**

Matthew Osborne

August 2017

# Overview

This document describes how to access the data used in the paper, and how to set up and run the code which replicates all the empirical output and simulations in the paper. To do this there are a number of steps that need to be followed in order, which I describe briefly, and then give details about in the relevant subsections below. To get started, the raw data used by the code needs to be downloaded from the Kilts Center at the University of Chicago, and properly set up. In particular, it will be necessary to run the SAS files that Chicago supplies in order to generate the daily price files. For running the code, there are a few additional programs that may need to be installed and set up, depending on your system. The code to process the data, to run the post-estimation analysis (such as constructing the indexes), and to run the simulations used in the identification and final discussion of the paper are written in a combination of R, C and C++. It will be necessary to have R installed, and if you are on windows you will need to install Rtools in order to allow compilation of the C files. The estimation of the structural model uses Fortran 90 with OpenMP. The supplied code can be compiled and run on a Linux/Unix workstation with the standard set of compilers (which should include OpenMP by default), or on Windows with any version of the MinGW compiler suite that includes OpenMP. On Windows this compiler can be obtained from the website SourceForge, and I used SeZero's build, which is available from https://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win64/Personal%20Builds/. The Fortran 90 code uses some LAPACK libraries that are available for download from Netlib, which I have supplied. To make the setup and instructions easier to follow I describe the details in 5 steps below, which should be followed in order. Instructions for compiling and running the code are given for Windows, but compiling on Unix will be similar. In addition to giving detailed instructions in each step, I also provide descriptions of each included file.

# 1 Step 1: Set up the raw data files

I've included a zip file called input data which contains the raw data. Create a directory to contain the contents of this file. Note that this data may be downloaded from the Chicago Kilts Center website (https://research.chicagobooth.edu/kilts/marketing-databases/erim). The files downloaded from the ERIM site do not contain pf1.csv and pf2.csv, which are the daily price files. The ERIM website supplies SAS code that is used to construct these files. I additionally include the file dateweektranslation.csv, which is a mapping of ERIM weeks to calendar dates that I copied from the instructions supplied on the Kilts website. Note the ssoup directory has a file called "sssoup_upcs.RData". This RData file is a list of the UPCs for Cambpells and the store brand described in Appendix 15.[1] I also note that I deleted the very last line of the raw data files sssoup_f1.dat and sssoup_f2.dat, as the last lines of the original versions of those files appeared to be timestamps that interfered with the data creation step.

# 2 Step 2: Create the estimation data

In the supplied zip file is a folder called "R code for data creation". Copy the contents of this code into your code folder. There are two subfolders here, titled "Step 1 Code" and "Step 1 Code". Run the contents of each in order:

1. **Step 1 code**: Compile the file functions.c using "R CMD SHLIB functions.c" from the command line. Create a new directory that will contain the ouput files for this step. In the load_erim_data_tuna.r file, set the inputpath to the location of the raw data and the outputpath variable to where the output files should for step 1 should go. Run the file in R. Repeat the process for load_erim_data_soup.r. Finally, run the file "load_demogs.r", which will create a file of demographic data used for some of the supplementary analysis (the path variables in this file should be the same as the load_erim_data_*.r files). Then, edit the inputpath and outputpath variables in the file "analyze_prices_tuna.r" to be the same as the prior files. Also, set the tablepath and figpath variables to point to where you want tables and figures to be placed. Run this file, and it will load the price files and process them.

2. **Step 2 code**: Copy the functions.dll file (or functions.do if you are using Unix)

[1] In this file, there are two UPCs listed as 1 and 2, which are placeholders for the composite products used in the soup category estimation.

from Step 1 Code into Step 2 Code. Compile the soup.c file using "R CMD SHLIB soup.". Create a new directory for the estimation data. Inside this directory, create two subdirectories, one called "tuna" and one call "sssoup". Then, inside the tuna and sssoup subdirectories, create a directory called "hb". Now, edit the pathnames in createestimationdata_tuna.r: Set the inputpath variable to point to the location of the output files from the previous step, and set the outputpath to point to the estimation directory. Run this file in R. Repeat this step for the createestimationdata_soup.r file.

## 2.1 Included Files with Descriptions

- **load_erim_data_tuna.r**: This file loads in the raw ERIM data from the Chicago Kilts website, and produces R Data files containing household purchases, household trips, store price information from the store price files, and product information. This applies to the tuna category.

- **load_erim_data_soup.r**: This file performs the same functions as load_erim_data_tuna.r, but for the soup category.

- **load_demogs.r**: This file loads in the raw demographic data from the ERIM files and saves it into an R Data file. This files loads in data for both tuna and soup.

- **analyze_prices_tuna.r**: This file loads in the R Data files created by load_erim_data_tuna.r, as well as the pf1.csv and pf2.csv files created using the ERIM SAS code. It outputs R Data files containing weekly prices and product information. It produces Figure 1, some initial summary statistics that are presented in the Data section of the paper, and the numbers presented in online Appendix Tables A5 through A7.

- **analyze_prices_soup.r**: This file does the same function as analyze_prices_tuna.r, except for the soup category (this also produces similar summary statistics as the tuna file, as well as the numbers presented in online Appendix Tables A8 through A9).

- **functions.c**: This contains two routines which loop through data and are called by analyze_prices_soup.r as well as the createestimationdata_*.r files. In particular, there is a routine that imputes missing prices, and a routine that calculates an imputation for inventory used in the regressions in online Appendix Table A1.

- **createestimationdata_tuna.r**: This file creates all the estimation files needed to run the fortran code, and also runs some supplementary analysis that goes into the

appendix tables. In particular, it runs the regression (for tuna) in online Appendix Table A1, as well as the analysis in online Appendix B related to nonlinear pricing (including creating the numbers in online Appendix Table A2).

- **createestimationdata_soup.r**: This is the same file as createestimationdata_tuna.r but for the soup data.

- **mcmc_tuna.r**: This file is called by createestimationdata_tuna.r. It runs a preliminary, static estimation of tuna demand that can be used to compute an importance distribution for the inclusive values.[2] The importance distributions are loaded into the Fortran code. Note that the output of this file is not used to construct the estimates in the current version of the paper (the importance distribution is taken over prices rather than an inclusive value distribution as described in online Appendix E.6), but if the variable pdrawtype in params.f90 is set to 2 the code will switch to using these. Regardless, the output file produced from this estimation routine needs to be available for the Fortran code to read, as it will crash if the resulting files don't exist.

- **mcmc_soup.r**: This is the same as mcmc_tuna.r but for the soup data.

- **soup.c**: Estimating a static demand model for soup is harder than for tuna due to the larger choice set, so I recoded some of the routines to compute the likelihood in C. These routines are called by mcmc_soup.r. Again, the output of this routine is not used in the estimates presented in the latest version of the paper.

# 3 Step 3: Compile and run the estimation code

The files for this step are location in the "Fortran code for estimation" subdirectory. Inside this directory are two subdirectories called tuna and soup - these contain the code to estimate the model on each category. First, inside the tuna directory navigate to the directory "lapack files", and if on Windows run compile.bat, which will create lapack1.dll from files in "lapack files", "lapack_routine", and "blas_routine" (if on Unix, a similar script can be written to create lapack1.so). Then, copy lapack1.dll to the tuna directory. In the tuna directory, edit the path variable in the file params.f90 to point to the estimation data directory from the previous step. Then, compile all the .f90 files into a single executable (I've supplied a

[2]The procedure estimates a standard random coefficients demand model and then computes the resulting inclusive values for each individual. Then for each household it computes the household level average inclusive value as well as the standard deviation. These are stored in text files that the Fortran code will load in.

compile.bat script that does this in Windows. A similar script can be written to compile in Unix). Run the resulting executable (I call the executable inv in my compile script). Note that the estimation may take two or three days to run. Repeat the same steps for the files in the soup directory (note that you can just copy the lapack1.dll file from the tuna to the soup directory - they are the same for each program). The soup estimation usually takes about twice as long as the tuna estimation.

The estimation output will be put into the estimation data directory, in the hb folder of the corresponding product category.

## 3.1 Included Files with Descriptions

Below I describe the files in the tuna directory. The files in the soup directory are structured in the same way. As described above, to replicate the paper it's sufficient to simply edit the path variable in params.f90.

- **params.f90**: This file contains the global variables needed for the estimation routine, as well as some work routines (such as random number generators).
- **main.f90**: This file is the main program file for the estimation routine. It calls routines to load in the data and run the estimation.
- **setvars.f90**: This file sets up the choice set, which is used in likelihood construction.
- **process.f90**: The main routine in this file (createhhdata) will load in the estimation files created in the prior steps.
- **hbstruct.f90**: This file contains the routines which run the estimation. In particular, the main mcmc loop is in hbstruct.
- **compile.bat**: Sample batch file that shows how to compile the main executable on Windows.
- **files in lapack files, lapack_routines, and blas_routines directories**: These files were downloaded from http://www.netlib.org/lapack/. They contain work routines to invert and construct Cholesky factorizations of matrices. I also include a sample compile batch file called compile_lapack.bat.

# 4 Step 4: Run the post-estimation code

The files in the "Post estimation code" directory will create the tables and figures in the paper. To create the main tables of price indexes, run the file make_indexes_big_tuna.r (or make_indexes_big_soup.r for soup). Note that you will need to set the estpath variable to point to the category's estimation data, and step1path to be the output directory you specified in part 1 of Step 2. Note that these files create some intermediate data from the estimates that is stored in the estimation data directory. This intermediate data is created if the flag createhhdata is TRUE. Creating the intermediate data is slow, so if you have already created the data and only wish to run the code to make the indexes, you can set this flag to FALSE. The file "create_tables_of_estimates.r" will create tables of parameter estimates, time plots, and will run the regressions of storage cost type on demographics. For this file, you need to set the path variables to the same values as the index creation scripts.

## 4.1 Included Files with Descriptions

- **make_indexes_big_tuna.r**: This file creates the cost-of-living indexes presented in the main paper (Table 4), as well as a number of supplementary tables and figures for the online Appendix: Figures A1 and A8, Table A3, the first 3 columns of A14, and the regressions for Tables A15 and A16. It also creates a file of id variables called tuna_hhids.RData and puts it in the estimates folder. This file is loaded by the create_tables_of_estimates.r file.

- **make_indexes_big_soup.r**: This file does the same analysis as make_indexes_big_tuna.r, but for soup. It also produces online Appendix Table A4 and the second 3 columns of A14, and the regressions for soup in Tables A15 and A16, and Figure A8.

- **create_tables_of_estimates.r**: This file creates the tables and figures of estimates (the numbers in Table 3, as well as online Appendix Tables A10, A11, A12 and A13, and online Appendix Figures A3 through A7).

- **index_code.cpp**: This code contains some supplementary routines to compute utilities and inclusive values for make_indexes_big_tuna.r and make_indexes_big_soup.r. Since the matrices of estimates are extremely large it is more computationally efficient to do some of these calculations in C++ rather than R.

# 5 Step 5: Run the simulation code

The directory "Simulation Code" has all the code necessary to run the simulations used in Section IV (Parameter Identification) and at the end of Section VII C. (Comparison of the Indexes). To set up the code, it is necessary to first compile the simulation_functions.dll file. To do this type "R CMD SHLIB simulation_functions.c funcs.c". The file "identification_simulation.r" will run the identification simulation. You will need to set the path variable to point to the location of the code, and to set tablepath to point to where you want the output tables to be. Note that I include a subdirectory called figuresandtables in the code directory. The files will create a lot of supplementary plots that will go here (such as plots of the purchase hazard, which are verbally described in the identification section). To run the simulation at the end of the paper, run "index_simulation.r". Again, the path variables should be set up the same way as the "identification_simulation.r" file.

## 5.1 Included Files with Descriptions

- **identification_simulation.r**: This file runs the identification simulation and creates the numbers in Table 2.
- **index_simulation.r**: This file creates the numbers that are in Table 5. The table has 2 sets of columns, the first 3 correspond to simulation 1, and the second three to simulation 2. To reproduce simulation 1, set the variable pregime to 1. To do simulation 2, set it to 2.
- **simulation_functions.r**: This code contains work functions for the two simulations. In particular, it contains code to simulate prices, and to simulate household choices (the underlying simulation is done in C, and this code calls the C code).
- **simulation_functions.c**: This code contains C functions to solve for individual value functions and simulate choices.
- **funcs.c**: This code contains some work functions that are used by simulation_functions.c.
- **funcs.h**: This is a header file for funcs.c.