Journal of Global Economic Analysis, Volume 2 (2017), No. 2, pp. 128-154.

R Meets GEMPACK for a Monte Carlo Walk

BY NELSON VILLORIA[a]

This paper discusses two R functions for reading the output of GEMPACK-based CGE models into R. We highlight the potential of coupling GEMPACK and R by conducting systematic sensitivity analysis of model results using Monte Carlo experiments. We also show how R can enhance the analysis of CGE results by allowing for formal hypothesis testing of the effects of policies which outcome depends on stochastic shocks.

JEL codes: C6, C15

Keywords: Monte Carlo, Systematic Sensitivity Analysis, Gaussian Quadratures, GEMPACK, R.

1. Introduction

The statistical programming language R (R Core Team, 2017) is a strong tool for data manipulation and analysis. The General Equilibrium Modelling PACKage, GEMPACK (Harrison et al., 2016) is a powerful and efficient language to use for solving Computable General Equilibrium (CGE) Models. This article discusses two functions that facilitate reading solutions from GEMPACK into R. As discussed below, coupling R with GEMPACK allows using R's capabilities for data handling, visualization, and statistical analysis to enhance the CGE analysis performed in GEMPACK.

To bring this point home, the use of `rgtap` is illustrated in a series of examples of increasing complexity. In particular, emphasis is put on coupling GEMPACK and R for conducting systematic sensitivity analysis of model results using Monte Carlo experiments. Of course, other areas of application are also possible, from streamlining R-GEMPACK work flows to keeping track of recursive simulations. Nevertheless, the focus on systematic sensitivity analysis using Monte Carlo experiments is motivated by the possibility of using increasingly accessible multi-core computers and computer clusters to conduct more sophisticated statistical analysis of CGE results. This has the potential to produce deeper insights than the use of Gaussian Quadrature implemented in GEMPACK (Pearson and Arndt, 2000), especially in modeling exercises where non-linearities are important. Examples of

[a] Department of Agricultural Economics, Kansas State University, 222 Waters Hall, Manhattan, KS (nvilloria@ksu.edu).

these applications are found often and range from the evaluation of policies that can abruptly change price regimes (Hertel, Martin, and Leister, 2010) to agricultural productivity shocks whose distribution may be asymmetrical (Ramirez, Misra, and Field, 2003).

The functions in `rgtap` are probably most useful for users of both R and GEMPACK. Indeed, throughout the paper, it is assumed that the reader has working familiarity with R data types (e.g., lists) and functions, particularly the `lapply` family used to vectorize repetitive operations. It is also assumed that the user has some working knowledge of running GEMPACK programs from the DOS command line, which probably requires familiarity with COPS (2017a) and COPS (2017b).

2. Software requirements and installation of `rgtap`

`rgtap` uses two programs provided by the GEMPACK developers at the Centre of Policy Studies at Victoria University (COPS)[1]: `sltoht` and `har2csv`. The program `sltoht` converts GEMPACK SL4 files into header array files[2]. `sltoht` is part of the standard GEMPACK installation; however, the program does not require a GEMPACK license and can be freely downloaded from http://www.copsmodels.com/gpf90.htm. Whether stand-alone or as part of a GEMPACK distribution, `sltoht` must be in the Windows path for the scripts below to work. We strongly suggest that the reader take a moment to verify that this is the case by typing `sltoht` in the DOS command prompt to ensure that the program can be found.

The other program, `har2csv`, is a free add-on created by Prof. Mark Horridge for advanced GEMPACK users[3]. `har2csv` reads GEMPACK Header Array Files (HAR) files into text, comma separated value files, which we refer to as CSV files, that can be manipulated in any spreadsheet or any other programs with capabilities of reading structured data from plain text files. For the examples below to run, `har2csv` must also be on Windows path. As before, the reader is encouraged to verify that this is the case by typing `har2csv` in the DOS command prompt to ensure that the program is found. While there are versions of GEMPACK that run on Linux systems, `har2csv` is a Windows program; therefore, `rgtap` can run only on Windows OS. `rgtap` is hosted at GitHub and can be installed by using the command `install_github` from the R package `devtools`[4]:

---

[1] See http://www.copsmodels.com/index.htm.

[2] SL4 files can be visualized in ViewSOL, AnalyseGE, or ViewHAR, but they cannot be read or directly manipulated using other programs. The purpose of `stloht` (see chapter 39 in http://www.copsmodels.com/gpmanual.htm#gpd4.8 for details in usage and potential applications) is to convert SL4 files into header array (HAR) file that can be read by other GEMPACK programs.

[3] As of this writing, bundel16.exe can be downloaded from http://www.copsmodels.com/gpmark9.htm

[4] See https://cran.r-project.org/web/packages/devtools/index.html. Some users have reported difficulties using `install_github` in network drives. These users

```
install_github("nvilloria/rgtap")
```

3. Usage

`rgtap` has two functions, `extractvar` and `readsol`; `extractvar` takes variables from a solution or data file (e.g., sl4 or upd) and puts them in a new GEMPACK solution file with extension .sol (http://www.copsmodels.com/gpmanual.htm#gpd3.8), which can be converted to CSV using `har2csv` so that it can be read into R. The function `extractvar` is just a wrapper for `stloht` and as such requires a list of variables to extract from a solution file. This list is a "mapping file," which is described within the help for `sltoht` in the GEMPACK manual (http://www.copsmodels.com/gpmanual.htm#gpd4.8.3).

The second function, `readsol` is a wrapper for `har2csv`; `har2csv` extracts specific headers from GEMPACK Header Array Files (har) and solution files (sol). It is important to keep in mind that the process of using `stloht` to translate SL4 files into HAR files, and then `har2csv` to convert the HAR files into CSV files, can be realized directly, by using, for example, the DOS command prompt (see COPS (2017a) and COPS (2017b) for examples). Other programs with scripting capabilities such as GAMS (https://www.gams.com/) or Python (https://www.python.org/) are well equipped to interface with GEMPACK using `stloht` and `har2csv`. Likewise the R code discussed below is general enough to conduct stochastic simulations using different modeling programs (such as GAMS) as long as such programs offer ways to communicate with R.

4. Examples

The four examples below are based on the "GTAP Course 3x3 aggregation SSA-EUR," ACRS3X3 for short, developed in 1998 and still used today in GTAP short courses. ACRS3X3 is part of the standard RunGTAP distribution and is also freely available from the GTAP website; however, for replicating the examples below, it is necessary to use the zipped version available in the supplementary files published with this paper[5], which has already modified GEMPACK command files (CMF). As in the GTAP Short Course, the first three examples are developed around the "TMSFE" experiment, in which the EU cuts tariffs applied to exports from sub-Saharan Africa by 10%. In this triad of examples, we will read and plot some simulation results, study the sensitivity of some key results to uncertainty around the true value of the Armington trade elasticity using Monte Carlo simulations,

---

may prefer installing the binary files that can be downloaded from https://github.com/nvilloria/rgtap/raw/master/rgtap_0.0.0.9000.zip, and running the command `install_packages(''rgtap_0.0.0.9000.zip'')` in the folder containing the zip file.

[5] The file can also be downloaded from https://github.com/nvilloria/rgtap/raw/master/ACRS3X3.zip.

and then demonstrate how to use R's robust and convenient facilities for parallel computing to speed up the Monte Carlo experiments. A fourth and final example illustrates how to conduct statistical analysis of model results following random exogenous productivity shocks.

Preliminaries

The starting point for following the exercises below is to download the file, ACRS3X3.zip, from https://github.com/nvilloria/rgtap/ACRS3X3.zip and unzip the contents of ACRS3X3 somewhere in the user's computer.

To develop these examples, the original GTAP model file (gtap.tab) was modified to read the region's generic elasticities of substitution between domestic and imported goods (ESUBD), as well as among imported goods (ESUBM) from a file identified as SIGMA (see lines 117-118, 831-832, and 1838-1839 of gtap.tab). The need for these modifications will become apparent in Examples 2 and 3 below. This change in the model requires instructing the CMF file to read SIGMA from sigma.har (see line 17 in `tmsfse_ex1.cmf`). The reader is encouraged to inspect sigma.har, which contains two headers, ESBD and ESBM, with the elasticities used in the original model. We also find it convenient to instruct GEMPACK to collect the simulation results in a sub-folder named "results" (see lines 22-24 on tmsfse.cmf). Although this is not strictly necessary for single runs of the model, it is an imperative for the Monte Carlo experiments in Examples 2 to 4.

Before working on the examples below, we recommend taking a look inside ACRS3X3. The user should verify the existence of six files: `ACRS3X3_ex1.R` to `ACRS3X3_ex4.R`, `ACRS3X3_ex2_comp.R` and `ACRS3X3_ex4_comp.R`. These files contain the R scripts used in the examples. These scripts are reproduced in listings 1 to 5 below. There are also five CMF files: `tmsfse_ex1.cmf`, ..., `tmsfse_ex4a.cmf`, and `tmsfse_ex4b.cmf`.

4.1 Example 1: Importing GEMPACK output and plotting the changes in food output in sub-Saharan Africa after the EU liberalizes its markets (ACRS3X3)

To reproduce this example, open the file, `ACRS3X3_ex1.R`, within the ACRS3X3 folder in the R user interface of choice. Execute the script (reproduced in listing 1) step by step to produce figure 1.

Listing 1. `ACRS3X3_ex1.R`: Output changes in sub-Saharan Africa after tariff cuts in the EU

```
## Example 1: Plotting the changes in food output in sub-Saharan
## Africa after the EU liberalizes its markets (ACRS3X3)
## install.packages("devtools")
## Install and load rgtap library:
## library(devtools)
## install_github("nvilloria/rgtap")
library(rgtap)

## Run the experiment tmsfse from R:
```

```
a <- system("gtap -cmf tmsfse_ex1.cmf", intern = FALSE, ignore.stdout
= TRUE)
#quit(save = "no", status = 0, runLast = TRUE)
## Extract variables defined in example1.map to a solution file
## named tmsfse_ex1.sol:
b <- extractvar(solution.dir = "./results/",
        solution.name = "tmsfse_ex1",
        var.map = "ACRS3X3_rgtap.map",
        solution.out = "./results/tmsfse_ex1.sol")

## Read qo (%-change in output):
qo <- readsol(solution.dir = "./results/",
          solution.out = "tmsfse_ex1.sol",
          csv.out = "./results/tmsfse_ex1.csv",
          header = "0002")

## Plot qo (%-change in output) for the non-saving commodities in
## sub-Saharan Africa::
pdf( file = "acrs3X3qo.pdf", width = 12, height = 7 )
barplot( as.matrix( qo$Value[qo$REG == "SSA"] ), beside = T,
      names.arg = qo$NSAV_COMM[qo$REG == "SSA"],
      ylab = "% Change in Output",
      main = "Output changes in sub-Saharan Africa")
dev.off()
```

Under the hood: The first step in listing 1 is to install (if this has not been done before) and load the `rgtap` library. The second step uses `system`, R's interface to Windows OS, to execute `gtap.exe` using the instructions in the command file, `tmsfse_ex1.cmf`. This is equivalent to typing "`gtap -cmf tmsfse_ex1.cmf`" directly into the command prompt[6]. The third instruction is a call to `extractvar`, which needs to know where the solutions are stored (`solution.dir`); the name of the solution file (without the extension) we have chosen in the command file `tmsfse_ex1.cmf`. `extractvar` also requires a GEMPACK mapping file for use with GEMPACK's `sltoht` (`var.map`). The mapping file we are using is `ACRS3X3_rgtap.map`, which is in the ACRS3X3 folder downloaded above. The reader should open this file in a text editor and verify that we are interested in extracting only three variables, changes in market prices (pm), changes in output (qo), and changes in bilateral exports (qxs), from the standard GTAP model. To these variables, we have assigned the headers, 0001, 0002, and 0003. `ACRS3X3_rgtap.map` should look like this (see section 39.3 of the online GEMPACK manual at http://www.copsmodels.com/gpmanual.htm#gpd4.8.3) :

```
0001 pm
0002 qo
```

[6] See "Using BAT files to run GEMPACK Programs automatically" at https://www.copsmodels.com/gp-bat.htm for an in-depth introduction.

```
0003 qxs
```

Finally, we need to tell `extractvar` the name of the solution file (with extension *.sol) which will store the values of the variables, pm, qo, and qxs, defined in the GEMPACK mapping file, `ACRS3X3_rgtap.map`. In listing 1 we name this file, `tmsfse_ex1.sol`. As mentioned before, `extractvar` is a wrapper for `sltoht`. This is transparent to the user by simply typing `extractvar` in R's command line:

```
> extractvar
function (solution.dir, solution.name, var.map, solution.out)
{
    system(paste("sltoht", paste(solution.dir, solution.name,
        sep = ""), paste("-map=", var.map, sep = ""), solution.out),
        ignore.stdout = TRUE)
}
```

After the variables of interest have been extracted to the file `tmsfse_ex1.sol`, the function `readsol` reads them into R. The arguments, `solution.dir` and `solution.out`, are as indicated for extractvar. The argument `csv.out` is the comma separated values (CSV) file that will contain the values in a specific `header` (0002 in our case) of the solution file, `tmsfse_ex1.sol`. As before, by typing `readsol` in R's command line, we can see that the function is a wrapper of `har2csv`, so we can convert GEMPACK har files to comma separated values:

```
> readsol
function (solution.dir, solution.out, csv.out, header)
{
    har2csv.status <- system(paste("har2csv", paste(solution.dir,
        solution.out, sep = ""), csv.out, header, sep = " "),
        ignore.stdout = TRUE)
    if (har2csv.status == 0) {
        y <- read.csv(csv.out)
    }
    else {
        y <- NA
    }
}
```

At this point, we have the GTAP produced percentage changes in output, qo, converted to an R data frame, which allows for a potentially limitless set of possibilities for manipulation and analysis. The following code snippet examines some of the key features of qo (class, structure, and the first six entries). We encourage the reader to verify that qo has been properly read into R by typing the following commands

and comparing the messages in the R console:

```
  > class(qo)
[1] "data.frame"

> str(qo)
'data.frame': 12 obs. of 3 variables:
 $ NSAV_COMM: Factor w/ 4 levels "CGDS","Food",..: 2 3 4 1 2 3 4 1 2 3 ...
 $ REG : Factor w/ 3 levels "EU","ROW","SSA": 3 3 3 3 1 1 1 1 2 2 ...
 $ Value : num 3.23474 -1.82701 -0.31287 -0.00121 -0.29498 ...

> head(qo)
  NSAV_COMM REG Value
1 Food SSA 3.858260870
2 Mnfcs SSA -2.196391106
3 Svces SSA -0.366095155
4 CGDS SSA -0.001352218
5 Food EU -0.366300881
6 Mnfcs EU 0.081489623
```

We close this example by plotting the changes in output in three economic sectors within sub-Saharan Africa after the EU cut its tariffs on sub-Saharan African food imports. This is shown in figure 1 produced by the last bit of code in listing 1. Notice that we are using the pdf output option of R, which produces high-quality, publication-ready figures, one of the many features explaining the popularity of using R for quantitative research. This concludes our first example. In the next example, we focus on using R's scripting capabilities as well as the functions `extractvar` and `readsol` to perform Monte Carlo experiments using the GTAP model. We start with serial simulations in example 2, and then in examples 3 and 4, we demonstrate the abilities of R to harness the power of parallel computing in a way that is portable across platforms.

4.2 Example 2: R for Serial Monte Carlo Experiments with the GTAP model

This example shows how to conduct systematic sensitivity analysis of model results using Monte Carlo experiments in R. Here we focus on serial experiments. In the next two examples, we discuss parallel simulations.

Our goal is to use R in estimating the complete distribution of changes in output, market prices, and bilateral exports from sub-Saharan Africa to the EU, given the uncertainty surrounding the true value of the Armington elasticities of substitution among foreign suppliers, ESUBM. As explained above, we modified the GTAP model in the ACRS3X3 folder to read the set of values of SIGMA from an individual file. The reason for this modification is that in the standard implementation of the GTAP model, all the parameters, including ESUBD and ESUBM, are read from a

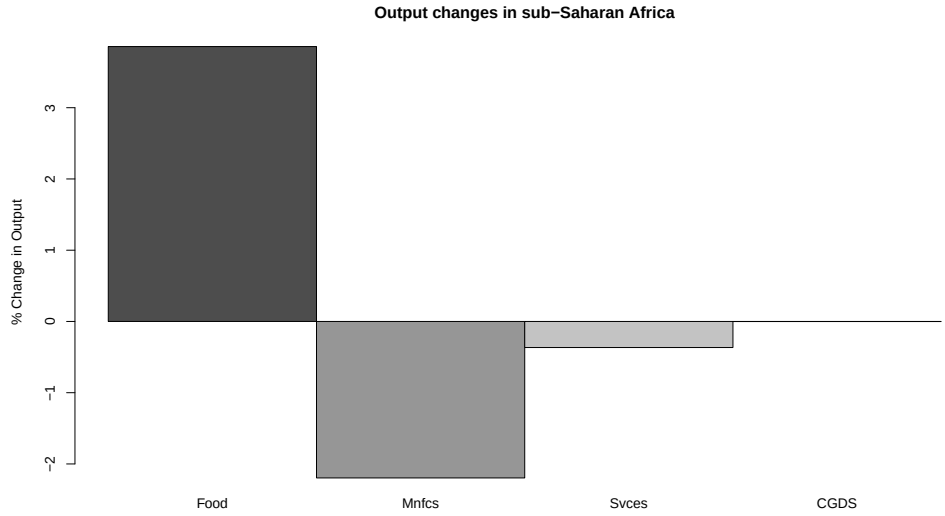

Figure 1. Percentage change in sectoral output in sub-Saharan Africa after a 10% cut in EU's food tariffs.

single parameter file. As we will vary the values of ESUBM (and consequently ESUBD) many times, creating a new parameter file for each simulation is rather impractical, particularly as the model (and parameter files) grows in size.

For this example, we use a handy feature of GEMPACK whereby parameters are passed from the command line to the CMF file (Horridge and Jerie, 2013). To implement this feature, we provide a modified CMF file, `tmsfse_ex2.cmf`, from which we have extracted the following fragment focusing on the statements addressing model inputs and outputs:

```
! Original Data files
! ----------------------
!
File GTAPSETS = SETS.HAR ; ! file with set specification
File GTAPDATA = Basedata.har ; ! file containing all base data
File GTAPPARM = Default.prm ; ! file containing behavioral parameters
File SIGMA = .\results\sigma.<p1>.har ; ! file containing Armington elasticities
!
!
! Output files
! ------------
!
Solution file = .\results\tmsfse_ex2.<p1> ;
Updated file GTAPDATA = .\results\tmsfse_ex2.<p1>.upd ;
```

Notice that SIGMA is addressed to a file named sigma.$< p1 >$.har, within the sub-folder results. The expression $< p1 >$ in sigma.$< p1 >$.har takes the value of a parameter $< p1 >$ that is passed from the command line to the CMF file. For example, if p1 equals 1, the CMF file will look for the file sigma.1.har. The beauty of this syntax is that we can let p1 take the values 1 to, say, 10,000, and use the CMF to solve the GTAP model 10,000 times, each run with a different set of values of ESUBM and ESUBD. Moreover, the output files, "Solution file" and "Updated file GTAPDATA," also have the suffix ¡p1¿, so the results of using sigma.1.har will be stored in `tmsfse_ex2.1.*` (where * equals sl4, slc, and xac) and the updated database in `tmsfse_ex2.1.upd`.

As with GEMPACK, R can be run in batch mode. Opening the DOS command prompt in the ACRS3X3 folder and typing the following command—which assumes that R is in the Windows path[7]—will run the scripts in the file `ACRS3X3_ex2.R` in batch mode, while saving a log of the simulation steps in the file `ACRS3X3_ex2.Rout`.

```
R CMD BATCH --no-restore --no-save ACRS3X3_ex2.R ACRS3X3_ex2.Rout
```

Under the hood: The command above executes the code in listing 2. In order to simplify the explanations we have divided the code into blocks and sub-blocks. The first block (BLOCK 1) of code in this file creates and assigns values to the elasticities of substitution among imports (ESUBM), by dividing the domestic-import elasticities (ESUBD) by two—the values in the file correspond to those in the standard GTAP parameter file for this particular aggregation. This step is necessary to initialize these variables. BLOCK 2 consists of a vectorized loop using the `lapply` function. This vectorized loop will execute the inner code from BLOCKS 2.1 to 2.4 for each integer i in the set 1:N, which is set equal to 10 in the provided script.

Listing 2. `ACRS3X3_ex2.R`: Serial Monte Carlo Experiments using ACRS3X3

```
## Example 2: R for Serial Monte Carlo Experiments with the GTAP model

## Load the rgtap library:
library(rgtap)

## BLOCK 1 (In the Standard GTAP model ESUBM = 2*ESUBD, see
## header ESUBD in default.prm to verify original values):
ESUBM.FOOD <- 2.39*2
ESUBM.MNFCS <- 2.86*2
ESUBM.SVCS <- 1.95*2
```

[7] If R is not in the Windows path, the user must specify the directory containing the R executable program, e.g., C:/Program Files/R/R-3.3.1/bin/R.exe. Also, the user should check that the R script is in the directory in which the command is run. The flexibility of DOS commands would permit running the command over a script in another directory, but for expository purposes, we find simpler to have all the scripts and log files (*.Rout) in the same location.

```
N <- 10

## BLOCK 2: SERIAL SOLUTIONS:
mc.serial.time <- system.time(
   mc.serial <- lapply( c(1:N), function(i){
      ## BLOCK 2.1: Write sigma.<p1>.har
      ESUBM.FOOD.i <- rnorm(n =1, mean = ESUBM.FOOD, sd = 2)
      lines <- c(
      paste( '3 Real SpreadSheet Header "ESBD";' ),
      paste( ESUBM.FOOD.i/2),
      paste( ESUBM.MNFCS/2 ),
      paste( ESUBM.SVCS/2 ),
      paste( '3 Real SpreadSheet Header "ESBM";' ),
      paste( ESUBM.FOOD.i ),
      paste( ESUBM.MNFCS ),
      paste( ESUBM.SVCS ))
      writeLines(lines, con = paste('./results/sigma.', i, '.txt', sep
       ="") )
      system( paste('txt2har ./results/sigma.', i, '.txt ./results/
      sigma.', i,'.har', sep = "") )
      ## BLOCK 2.2 Run the GTAP model:
      exp <- paste('gtap -cmf tmsfse_ex2.cmf -p1=', i, sep = "")
      gtap.status <- system(exp, ignore.stdout = TRUE)
      if( gtap.status == 0){
         ## BLOCK 2.3: Extract varaiables in map to a solution file:
         ext.status <- extractvar(solution.dir = "./results/",
                           solution.name = paste("tmsfse_ex2.",i, sep
                            =""),
                           var.map = "ACRS3X3_rgtap.map",
                           solution.out = paste("./results/tmsfse_ex2
                           .",i,".sol", sep =""))
         if( ext.status == 0){
            ## BLOCK 2.4: Read results: pm, qo, qxs(SSA, EU) and sigma
            :
            qo <- readsol( solution.dir = "./results/",
                      solution.out = paste("tmsfse_ex2.",i,".sol",
                      sep =""),
                      csv.out = "qo.csv",
                      header = "0002" )
            pm <- readsol( solution.dir = "./results/",
                      solution.out = paste("tmsfse_ex2.",i,".sol",
                      sep =""),
                      csv.out = "pm.csv",
                      header = "0001" )
            qxs <- readsol( solution.dir = "./results/",
                       solution.out = paste("tmsfse_ex2.",i,".sol",
                       sep =""),
                       csv.out = "qxs.csv",
                       header = "0003" )
            sigma <- readsol( solution.dir = "./results/",
                        solution.out = paste('sigma.', i, '.har', sep
```

```
                          =""),
                         csv.out = "sigma.csv",
                         header = "ESBM" )
            list(qo = qo, pm = pm, qxs = qxs, sigma = sigma)
         }else{ ## Default to this if extractvar fails
            list(qo = NA, pm = NA, qxs = NA, sigma = sigma, status =
            ext.status)}
      }else{ ## Default to this if gtap fails
         list(qo = NA, pm = NA, qxs = NA, sigma = sigma, status = gtap.
         status)}
   }))
## BLOCK 3: Save results and delete unwanted files
save(mc.serial.time, mc.serial, file = "./results/ACRS3X3_ex2.RData")
file.remove(
   paste("./results/", setdiff(list.files("./results"),
                        c("ACRS3X3_ex2.RData")), sep ="")
         )
```

BLOCK 2.1 writes a file with the values of ESUBD and ESUBM in which the the elasticity for the food sector is a random draw from a normal distribution using R's function `rnorm()` which means ESUBM.FOOD (set equal to 2.39*2 in BLOCK 1). To create the sigma.$< p1 >$,har file, we create an object line which basically strings together the instructions in the `paste` commands, as well as lines required by the GEMPACK program `text2har` to create a har file from a GEMPACK text file. Once these lines are strung together, the command `writeLines` saves them to the computer disk, and the command `system` calls `text2har`, another GEMPACK program that needs to be in Windows path, so a har file is created (see Horridge and Jerie, 2013, for an example using DOS). Notice that the name of the har file depends on the specific value of i, which is the i^{th} iteration of the vectorized loop bracketed within the lapply functions. For instance, for i = 1000, the file would be named sigma.1000.har.

BLOCK 2.2 in listing 2 runs the GTAP model using a CMF in which the values p1 within the CMF file discussed above are set equal to the value taken by index i. For instance, if $i = 1000$, the CMF file will look for sigma.1000.har, solve the GTAP model for the random draw of ESUBM obtained using `rnorm()`, and save the results as tmsfsem.1000.* where * equals the GEMPACK file types sl4, slc, upd, and axt. Note that the `system` command is pointed to `gtap.status`. Under the hood, `gtap.status` captures GTAP's errorlevel value (see COPS, 2017a,b, for more details), which equals 0 if GTAP runs successfully, and 1 otherwise. The `if` command at the end of BLOCK 2.2 will skip the rest of the script if GTAP fails, going directly to the last else command in the script, which will assign NA values to the different variables while storing the value of `gtap.status`. By introducing this procedure, we ensure that if GTAP fails in one or more simulations, we do not lose the results of the successful simulations. The if-else statements in listing 2 will appear in the next two examples.

BLOCK 2.3 calls `rgtap`'s `extractvar` function and produces a solution file—named `tmsfem.i.sol`, so for i=1000, the file is `tmsfem.1000.sol`— with the variables specified in the mapping file, `ACRS3X3_rgtap.map`. The set of instructions in BLOCK 2.4 reads the individual variables of interest from the i^{th} GEMPACK solution file and stores them in an R list. In the resulting list, we also save the specific value of ESUBM associated with each set of results in order to relate the distribution of the exogenous shocks to the shapes of the distributions of the endogenous variables, qo, pm, and qxs. As in BLOCK 2.2, we also introduce an if-else statement that will skip reading the variables into R (BLOCK 2.4, discussed next) and assign NA values if the errorlevel value of `extractvar`, stored in `ext.status`, equals 1 (i.e., a failure of `extractvar`).

Once the N simulations are completed, what we have is a list with N elements, one for each Monte Carlo iteration, which in turn stores a list with the specific values of qo, pm, and qxs as well as ESUBM for that iteration. This object, named `mc.serial`, as well as the clock time used to run the set of Monte Carlo simulations, `mc.serial.time`, are stored in the computer disk using R's compressed data format (BLOCK 3). We also remove the potentially large number of files that were created during the Monte Carlo simulations using R's `file.remove` function.

The densities of ESUBM, qo, pm, and qxs are displayed in figure 2. This can be readily produced with the script shown in listing 3, which is a set of basic R commands for organizing and plotting data. An advantage of having a distribution of outcomes is that departures from asymmetry, such as those characterizing the changes in food output and market prices in SSA, can be better appreciated. This also highlights the point that the distribution of model outcomes is unrelated to the assumed distribution of exogenous shocks, ESUBM in this case (see Villoria and Preckel, 2017, for a more extended discussion of this point).

Listing 3. `ACRS3X3_ex2_comp.R`: R scripts for plotting variables from Example 2

```
## Read values for plots (Change to ex3 for plot in example 3):
load("./results/ACRS3X3_ex2.RData")

## Use serial for example 2 and parallel for example 3. I.e. comment
## out "mc.serial.time" and uncomment mc.parallel.time:
mc.serial.time
## mc.parallel.time

## Use serial for example 2 and parallel for example 3:
.results <- mc.serial

status <- sapply(.results, function(i){
   d <- i[["status"]]
})

table(status)
```

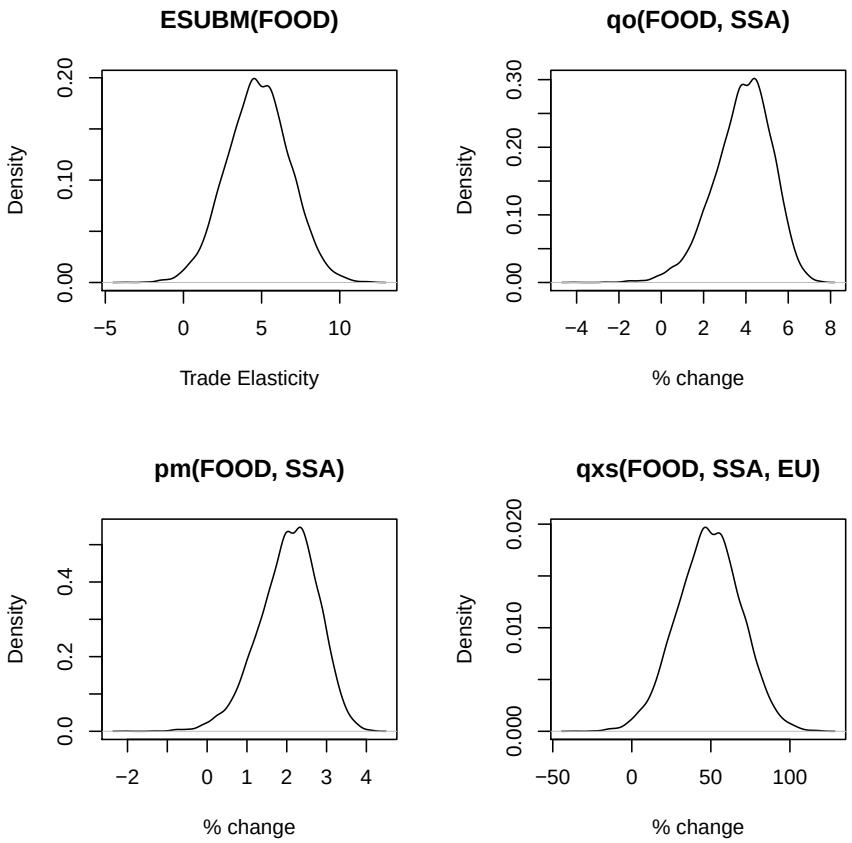

Figure 2. Systematic Sensitivity Analysis using 10,000 Monte Carlo simulations for different values of ESUBM. Displayed are the distribution of ESUBM as well as the percentage changes in food output, prices and EU-bound exports in sub-Saharan Africa after a 10% cut in the EU's food tariffs.

```
sigma.food <- sapply(.results, function(i){
   d <- i[["sigma"]]
   d$Value[1]
})

qo.food.ssa <- sapply(.results, function(i){
   d <- i[["qo"]]
   with(d, d$Value[ NSAV_COMM == "Food" & REG == "SSA"] )
})

pm.food.ssa <- sapply(.results, function(i){
   d <- i[["pm"]]
   with(d, d$Value[ NSAV_COMM == "Food" & REG == "SSA"] )
})

qxs.food.ssa.eu <- sapply(.results, function(i){
   d <- i[["qxs"]]
   with(d, d$Value[ TRAD_COMM == "Food" & REG == "SSA" & REG.1 == "EU"
    ] )
})

## Plots:
pdf( file = "ACRS3X3ex2.pdf", width = 6, height = 6)
par( mfrow = c(2,2) )
plot( density ( sigma.food ), main = "ESUBM(FOOD)", xlab = "Trade
Elasticity" )
plot( density ( qo.food.ssa ), main = "qo(FOOD, SSA)", xlab ="% change
" )
plot( density ( pm.food.ssa ), main = "pm(FOOD, SSA)", xlab ="% change
" )
plot( density ( qxs.food.ssa.eu ), main = "qxs(FOOD, SSA, EU)", xlab
="% change" )
dev.off()
```

4.3 Example 3: Harnessing the power of parallel computing in R

Monte Carlo experiments conducted in a serial fashion may be exceedingly time consuming. However, increasing the availability of computers with capacities for parallel computing, including computer clusters with hundred of cores and large amounts of memory, may considerably reduce computation time. R has powerful features for parallel computing that, together with the ability to act as a scripting language, can be used to parallelize GEMPACK runs[8]. In this example, we modified the code in listing 2 to run in parallel. The modified code is shown in listing 4. Listing 2 and listing 4 are identical, except for the code blocks identified

[8] GEMPACK allows the user to divide a single multi-step simulation between processors (See chapter 31 at http://www.copsmodels.com/gpmanual.html). In order to avoid potential conflicts as multi-step and parallel simulations compete for computing resources, we suggest using the GEMPACK default of no parallel simulations.

as PARALLEL 1 to PARALLEL 3, which are discussed below. Listing 4 is in the file `ACRS3X3_ex3.R`, with model inputs specified in `tmsfse_ex3.cmf`. As before, the code can be executed in batch mode using the following command[9]:

```
R CMD BATCH --no-restore --no-save ACRS3X3_ex3.R ACRS3X3_ex3.Rout
```

Listing 4. `ACRS3X3_ex3.R`: Parallel Monte Carlo Experiments with Random Parameters

```
## Example 3: Harnessing the power of parallel computing in R.

## Load the rgtap library:
library(rgtap)
## BLOCK 1:
ESUBM.FOOD <- 2.39*2
ESUBM.MNFCS <- 2.86*2
ESUBM.SVCS <- 1.95*2

N <- 12

## PARALLEL 1: SET-UP THE CLUSTER:
library(snowfall)
n.cpus <- 4
sfInit(parallel = TRUE, cpus = n.cpus )

sfExport(list= list("ESUBM.FOOD", "ESUBM.MNFCS", "ESUBM.SVCS"))
sfLibrary(rgtap)

## BLOCK 2: PARALLEL SOLUTIONS:
mc.parallel.time <- system.time(
   mc.parallel <- sfLapply(1:N, function(i){
      ## BLOCK 2.1: Write sigma.<p1>.har
      ESUBM.FOOD.i <- rnorm(n =1, mean = ESUBM.FOOD, sd = 2)
      lines <- c(
         paste( '3 Real SpreadSheet Header "ESBD";' ),
         paste( ESUBM.FOOD.i/2),
         paste( ESUBM.MNFCS/2 ),
         paste( ESUBM.SVCS/2 ),
         paste( '3 Real SpreadSheet Header "ESBM";' ),
         paste( ESUBM.FOOD.i ),
         paste( ESUBM.MNFCS ),
         paste( ESUBM.SVCS ))
      ## PARALLEL 2: SET-UP INDIVIDUAL SUB-FOLDERS
      soldir.i <- paste("./results.", i, sep = "")
      dir.create( soldir.i)
      writeLines(lines, con = paste(soldir.i, '/sigma.', i, '.txt',
      sep ="") )
      system( paste('txt2har ', soldir.i, '/sigma.', i, '.txt ',
      soldir.i,
```

[9] See footnote 7.

```
                ’/sigma.’, i,’.har’, sep = "") )
      file.copy( from = "tmsfse_ex3.cmf", to = soldir.i, overwrite =
      TRUE)
      file.copy( from = "standard.cls", to = soldir.i, overwrite =
      TRUE)
      ## BLOCK 2.2 Run the GTAP model:
      exp <- paste(’gtap -cmf ’, soldir.i, ’/tmsfse_ex3.cmf -p1=’, i,
      sep = "")
      gtap.status <- system(exp, ignore.stdout = TRUE)
      Sys.sleep(0.1)
      if( gtap.status == 0){
         ## BLOCK 2.3: Extract variables in map to a solution file:
         ext.status <- extractvar(solution.dir = soldir.i,
                 solution.name = paste("/tmsfse_ex3.",i, sep =""),
                 var.map = "ACRS3X3_rgtap.map",
                 solution.out = paste(soldir.i, "/tmsfse_ex3.",i,".sol
                 ", sep ="")
                 )
         if( ext.status == 0){
         ## BLOCK 2.4: Read results: pm, qo, qxs(SSA, EU) and sigma:
         qo <- readsol( solution.dir = soldir.i,
                    solution.out = paste("/tmsfse_ex3.",i,".sol", sep
                    =""),
                    csv.out = paste(soldir.i, "/qo.csv", sep = ""),
                    header = "0002" )
         pm <- readsol( solution.dir = soldir.i,
                    solution.out = paste("/tmsfse_ex3.",i,".sol", sep
                    =""),
                    csv.out = paste(soldir.i, "/pm.csv", sep = ""),
                    header = "0001" )
         qxs <- readsol( solution.dir = soldir.i,
                     solution.out = paste("/tmsfse_ex3.",i,".sol", sep
                     =""),
                     csv.out = paste(soldir.i, "/qxs.csv", sep = ""),
                     header = "0003" )
         sigma <- readsol( solution.dir = soldir.i,
                      solution.out = paste(’/sigma.’, i, ’.har’, sep
                      =""),
                      csv.out = paste(soldir.i, "/sigma.csv", sep =
                      ""),
                      header = "ESBM" )
         list(qo = qo, pm = pm, qxs = qxs, sigma = sigma, status =
         gtap.status)
      }else{ ## Default to this if extractvar fails
         list(qo = NA, pm = NA, qxs = NA, sigma = sigma, status = ext.
         status)}
   }else{ ## Default to this if gtap fails
      list(qo = NA, pm = NA, qxs = NA, sigma = sigma, status = gtap.
      status)}
   }))
sfStop()
save(mc.parallel.time, mc.parallel, file = "./results/ACRS3X3_ex3.
```

```
RData")
## PARALLEL 3: DELETE UNWANTED INTERMEDIATE RESULTS
unlink("results.*", recursive = TRUE)
```

Under the hood: For parallelizing the code in listing 2, we use R's `snowfall` package (Knaus, 2010), which allows setting up a cluster system that shields the user from complicated setups. This is shown in the code under PARALLEL 1 in listing 4, where we have chosen to use four computer cores. We also have replaced the vectorized loop function `lapply` with `sfLapply`, which distributes the vectorized loops across the available computer cores.

It is important to keep in mind that parallel computations demand increased awareness of how the computer manages its resources. A major consideration is memory. For instance, using four cores simultaneously will require four times the random access memory (RAM) needed to run a single model[10].

Provided RAM is not a limiting aspect, GEMPACK uses a variety of intermediate files whose names cannot be indexed by iterations. Therefore, if the simulations are conducted in the same folder, GEMPACK will try to rewrite files that may still be in use, which will produce an error. For this reason, for parallel runs, each simulation i is run in a directory that is created by R on the fly under the names results.i, where i is the value p1 passed from the command line to the CMF file. This is shown in the snippet of code labeled PARALLEL 2 in listing 4. As indicated in the R script, this step requires copying the closure file to the results.i directory, as well as a copy of the CMF file `ACRS3X3_ex3.cmf`. `ACRS3X3_ex3.cmf`, in turn, has been modified in the following way:

```
! Original Data files
! ----------------------
!
LOG FILE = ./results.<p1>/tmsfse_ex3.<p1>.log ;
File GTAPSETS = SETS.HAR ; ! file with set specification
File GTAPDATA = Basedata.har ; ! file containing all base data
File GTAPPARM = Default.prm ; ! file containing behavioral parameters
File SIGMA = ./results.<p1>/sigma.<p1>.har ; ! file containing behavioral
parameters
!
!
! Output files
! ------------
!
Solution file = .\results.<p1>\tmsfse_ex3.<p1> ;
```

[10] A very useful discussion about computer requirements for running GEMPACK in parallel at https://www.copsmodels.com/gp-fnewpc.htm.

```
Updated file GTAPDATA = .\results.<p1>\tmsfse_ex3.<p1>.upd ;
Extrapolation Accuracy File = yes ;
```

So the log files, the SIGMA file, and the solution files are written into the folder `results.i`. A related consideration is that GEMPACK may need a few extra moments between simulations to finish writing the data to the computer disk and recover resources. This is the reason we have included a command `Sys.sleep` set 0.1 seconds which introduces a small delay between critical operations when saving and passing results from GEMPACK to R. The final difference between listings 2 and 4 is that in listing 2 we removed the files from the `results` folder, while in listing 4 we use R's `unlink` to recursively delete all the directories with individual simulations (see code under PARALLEL 3). The structure of the results stored in `./results/ACRS3X3_ex3.RData` is identical to the results from Example 2. Therefore, we can use the code snippet in listing 3 to handle and visualize the results.

To get a perspective on the speed gains due to parallelization consider that 100 serial runs (as in Example 2) of ACRS3X3 took 51.71 seconds[11] ; using six computer cores of the same Windows PC, the 100 simulations to 19.92 seconds. When we scaled up to 10,000 simulations, the serial simulations took 85.66 minutes while the parallel simulations took 29.26 minutes.

4.4 Example 4: Simple Statistical Analysis of Differences in Model Outcomes

Thus far, we have used R mostly as a scripting language to pass instructions and model inputs onto GEMPACK to then read model outputs from GEMPACK files. As part of demonstrating this process, we have used many useful features of R, from drawing random shocks on the fly to setting up a parallel computing environment and also to taking advantage of R's strengths in creating high resolution graphics. In this example, we go one step further and use R to conduct a statistical analysis of the model results. The example also illustrates the use of GEMPACK replaceable parameters to pass shocks (as opposed to parameters) to the GTAP model.

The scenario is motivated by the large number of real world situations where the interest is on understanding the price volatility, often of an agricultural commodity, under different policy regimes, given historical short term fluctuations in productivity, such as agricultural yields. Examples of this problem can be found in Valenzuela et al. (2007), Hertel, Martin, and Leister (2010) and Villoria and Mghenyi (2016). In these works, the customary strategy is to run stochastic simulations with and without polices and use the GTAP implementation of Systematic Sensitivity Analysis via Gaussian Quadratures (GQ-SSA) (Pearson and Arndt, 2000). GQ-SSA is a powerful tool to elicit the mean and standard deviations of model outcomes at extremely low computational costs. However, as discussed by Villoria and Preckel

[11] In a computer running Microsoft Windows 10 Pro,Intel Core i7-4790, 3.60 GHz, with 4 physical cores and 8 logical cores, and 32GB of RAM memory.

(2017), GQ-SSA can be imprecise and, as implemented, inadequate to capture higher moments of the distribution of model outcomes, such as those in figure 2 where the price and outcome distributions are clearly asymmetric. It is only natural to expect fewer symmetric distributions when the policies involve discrete shifts in policy regimes, such as in the case of import safeguards in Hertel, Martin, and Leister (2010) or tariff-rate quotes. Moreover, in applications of climate change, extreme events are likely to be expressed as changes in the tail of the distributions, which even when assumed away by using the triangular distribution currently used in GTAP's GQ-SSA, may induce skewness in economic outcomes such as outputs and prices.

Even if the results of the GQ-based SSA were precise and there was no interest in the distributional properties of the model outcomes, GQ-based samples cannot be used to test whether the difference in, say, the price variance accruing to two different policy regimes is statistically significant. As a consequence, researchers rely on qualitative assessments that at best can be backed up using exceedingly large confidence intervals using Chebychev's inequality (Villoria and Preckel, 2017).

Against this background, in this example we demonstrate how to use `rgtap` to assess whether a variable export tax on sub-Saharan Africa food exports has a statistically significant effect on the volatility of regional food prices that are associated with worldwide food supply shocks. For the sake of example, let us assume that aggregated, normalized productivity shocks to the supply of food in the three regions of ACORS3X3 are normally distributed with mean zero and standard deviations 0.88 (sub-Saharan Africa), 0.12 (EU), and 0.10 (ROW). Furthermore, it is assumed that the productivity shocks have the cross-region correlation patterns in table 1 whereby productivity shocks in sub-Saharan Africa are negatively correlated with productivity shocks in Europe, and, to a lesser extent, positively correlated with productivity shocks in the rest of the world.

Table 1. Hypothetical cross-country correlations of supply shocks to the food sectors of the ACORS3X3 regions

	ssa	eu	row
ssa	1.00		
eu	-0.57	1.00	
row	0.21	-0.09	1.00

The research questions are: Does the implementation of an export tax on food produced in sub-Saharan Africa affect the volatility of regional prices (relative to an unobstructed equilibrium) stemming from domestic and foreign supply shocks? Does the answer to the preceding question depend on the patterns of correlation across the different regional food sectors? The empirical strategy for answering the first question consists of formally comparing the variances of prices under different policy regimes generated. For this, we simulate price outcomes for a large set of probable configurations of regional supply shocks under the two different policy regimes, and then we use an F test to determine whether any difference in simulated

variances is statistically significant. The answer to the second question is obtained by comparing simulations obtained using the correlation patterns in table 1 against simulations that assume uncorrelated shocks.

In order to model prices with and without stabilization policies, we saved a copy of `tmsfe_ex3.cmf`, renamed to `tmsfe_ex4a.cmf`, and modified the Shocks statement in the following way:

```
! Shocks
! ------
!
shock aoall("food","ssa") = <p2>;
shock aoall("food","eu") = <p3>;
shock aoall("food","row") = <p4>;
```

where p2-p4 are parameters that will be passed from the command line. As in the previous example, p1 is used to index directory folders and solution file names, a fact that the reader can verify by inspecting the statements regarding "Original Data files" and "Output files." The stabilization policy is implemented by "swapping", or making exogenous, the changes in food output in sub-Saharan Africa [qo("food", "SSA")] with destination-generic export taxes [tx("food", "SSA")] which will endogenously adjust to different configurations of the supply shocks. (See lines 1757-1767 of the gtap.tab file in the provided version of ACRS3X3 for additional motivation). The file with this modification is named `tmsfe_ex4b.cmf`, and its Shocks statement is:

```
! Shocks
! ------
!
swap tx("food","ssa") = qo("food","ssa");
shock aoall("food","ssa") = <p2>;
shock aoall("food","eu") = <p3>;
shock aoall("food","row") = <p4>;
```

Since we are running four experiments (with and without policies, and with and without correlations), we avoid the proliferation of R scripts by using R's BATCH mode ability to pass arguments into the R script file:

```
REM Correlated shocks without and with policies:
R CMD BATCH --no-restore --no-save "--args 10 2 Y I" acrs3x3_ex4.r
acrs3x3_ex4YI.rout
R CMD BATCH --no-restore --no-save "--args 10 2 Y A" acrs3x3_ex4.r
acrs3x3_ex4YA.rout
REM Uncorrelated shocks without and with policies:
R CMD BATCH --no-restore --no-save "--args 10 2 N I" acrs3x3_ex4.r
```

```
acrs3x3_ex4NI.rout
R CMD BATCH --no-restore --no-save "--args 10 2 N A" acrs3x3_ex4.r
acrs3x3_ex4NA.rout
```

The first argument after the keyword args is the number of simulations (we recommend using only 10 to ensure that the codes are running well). The second argument is the number of CPUs to use. The third and fourth arguments are for correlated shocks, Y(es) or N(ot), and stabilization policies, Inactive or Active; `acrs3x3_ex4.r` is the script used to run the simulations (reproduced in listing 5), and the files with extension rout are logs of the simulations that are helpful for record keeping.

Listing 5. ACRS3X3_EX4.R: Parallel Monte Carlo Experiments with Random Shocks

```
## Example 4: Simple Statistical Analysis of Differences in Model
## Outcomes.

## BLOCK 1: Assigns parameters for MC experiments based on
## command-line arguments and uses if-else assignments to parse the
## options defined by the user:
options(echo=TRUE) ## Echoes log of MC sims to the BATCH command log
              ## file (suffixed .Rout)
args <- commandArgs(trailingOnly = TRUE)
print(args) # Prints BATCH parameter values to log file

## Arguments are taken from the bat file, MonteCarlo.bat, which read
## this entire file:
N <- as.numeric(args[1]) ## Number of MC sims
n.cpus <- as.numeric(args[2]) ## Number of CPUs to be used
corr.shocks <- args[3] ## Correlated shocks, Y (for yes) or N(o)
policy <- args[4] ## A (for active) or I (for inactive)

## Variance-covariance matrix of hypothetical food productivity
## shocks: ssa, eu, row:

S <- matrix( c( 0.78873056, -0.06192094, 0.01917166,
           -0.06192094, 0.01496224, -0.00116997,
            0.01917166, -0.00116997, 0.01056702),
         nrow =3, ncol =3)

## Correlated Shocks:
if( corr.shocks == "Y"){
   ## If yes, the covariance matrix is used.
   .Sigma <- S
}else{
   ## Otherwise, use a square matrix with zero
   ## covariances:
   .Sigma <- diag(diag(S), nrow = 3, ncol = 3)
}
```

```
## Use variable levies to keep output constant?
if( policy == "I"){
   ## Chooses a CMF file without stabilization policy:
   cmf.file <- "tmsfse_ex4a.cmf"
}else{
   ## Chooses a CMF file that swaps qo(food,ssa) with the source
   ## generic export tax tx(food,ssa):
   cmf.file <- "tmsfse_ex4b.cmf"
}


## PARALLEL 1: SET-UP THE CLUSTER:
library(snowfall)
sfInit(parallel = TRUE, cpus = n.cpus )
sfLibrary(rgtap)
sfLibrary(MASS)
sfExport("N",".Sigma", "cmf.file")

## BLOCK 2: PARALLEL SOLUTIONS:
mc.parallel.time <- system.time(
   mc.parallel <- sfLapply(1:N, function(i){
      ## BLOCK 2.1: Draw random shocks from a multivariate normal
      ## and assign them to be passed onto gtap via the CMF:
      shocks <- mvrnorm(n =1 , mu = c(0,0,0), Sigma = .Sigma )
      ssa.ao <- shocks[1]
      eu.ao <- shocks[2]
      row.ao <- shocks[3]
      ## PARALLEL 2: SET-UP INDIVIDUAL SUB-FOLDERS
      soldir.i <- paste("./results.", i, sep = "")
      dir.create( soldir.i)
      file.copy( from = cmf.file, to = soldir.i, overwrite = TRUE)
      file.copy( from = "standard.cls", to = soldir.i, overwrite =
      TRUE)
      ## BLOCK 2.2 Run the GTAP model:
      exp <- paste("gtap -cmf ", paste( soldir.i, "/", cmf.file, sep =
       ""),
                paste( "-p", c(1:4), "=", c(i , ssa.ao, eu.ao, row.ao),

                     sep = "", collapse = " ") )
      gtap.status <- system(exp, ignore.stdout = TRUE)
      Sys.sleep(0.1)
      if( gtap.status == 0){
      ext.status <- extractvar(solution.dir = soldir.i,
              solution.name = paste("/tmsfse_ex4.",i, sep =""),
              var.map = "ACRS3X3_rgtap.map",
              solution.out = paste(soldir.i, "/tmsfse_ex4.",i,".sol",
              sep ="")
              )
      Sys.sleep(0.1)
      if( ext.status == 0){
      ## BLOCK 2.4: Read results: pm, qo, qxs(SSA, EU) and sigma:
      qo <- readsol( solution.dir = soldir.i,
```

```
                solution.out = paste("/tmsfse_ex4.",i,".sol", sep ="")
                ,
                csv.out = paste(soldir.i, "/qo.csv", sep = ""),
                header = "0002" )
      pm <- readsol( solution.dir = soldir.i,
                solution.out = paste("/tmsfse_ex4.",i,".sol", sep ="")
                ,
                csv.out = paste(soldir.i, "/pm.csv", sep = ""),
                header = "0001" )
      qxs <- readsol( solution.dir = soldir.i,
                 solution.out = paste("/tmsfse_ex4.",i,".sol", sep
                 =""),
                 csv.out = paste(soldir.i, "/qxs.csv", sep = ""),
                 header = "0003" )
      list(qo = qo, pm = pm, qxs = qxs, shocks = shocks, status = gtap.
      status)
   }else{ ## Default to this if extractvar fails
      list(qo = NA, pm = NA, qxs = NA, shocks = shocks, status = ext.
      status)}
   }else{ ## Default to this if gtap fails
      list(qo = NA, pm = NA, qxs = NA, shocks = shocks, status = gtap.
      status)}
   })
)
sfStop()
save(mc.parallel.time, mc.parallel,
    file = paste("./results/ACRS3X3_ex4.",corr.shocks,policy,
                                        ".RData", sep = "") )
## PARALLEL 3: DELETE UNWANTED INTERMEDIATE RESULTS
unlink("results.*", recursive = TRUE)
```

Under the hood: Most of the features in 5 have already been explained. The main innovation is BLOCK 1 to parse the arguments passed from the command line. This is achieved with the use of if-else statements under "Correlated Shocks" and "Use variable levies to keep output constant?". Note that the way we implement the uncorrelated shocks is by using R matrix manipulation tools to set to zero the off-diagonal terms of the covariance matrix S. Another difference from other scripts is in BLOCK 2.1, where GEMPACK's replaceable parameters are used to pass shocks (instead of parameter values) onto the CMF file. A noteworthy feature in BLOCK 2.1 is the use of the function `mvrnorm()` from the R package MASS (Venables and Ripley, 2003) for drawing shocks from a multivariate normal distribution with the dependency among variables specified in the covariance matrix.

After running the script in listing 5, we run a series of F tests to compare the variances of prices among regimes with and without policies and with and without correlations (see listing 6 for an example of doing this comparison as well as other ex-post analysis.). The results, shown in table 2, demonstrate that (regardless of whether the correlation of shocks is used) the variable export tax significantly increases the variance of food prices in sub-Saharan Africa.

Table 2. Comparison of variances and p-value from F-test for ratio of variances across policy regimes and correlation assumptions for Example 4

Policy Regime/Correlation	Correlated	Uncorrelated	p-value
Standard Model	1.038	1.043	0.957
Variable Export Tax	1.353	1.361	0.951
p-value	0.009	0.009	

Note: The variances of the correlated and uncorrelated experiments are identical up to two decimal values. We decided to show them here not because they are inherently interesting, but to drive the point that with Monte Carlo simulations we can apply formal statistical tests to CGE results.

Listing 6. `ACRS3X3_ex4.R`: Parallel Monte Carlo Experiments with Random Shocks

```
load("./results/ACRS3X3_ex4.NI.RData")
.results <- mc.parallel
status <- sapply(.results, function(i){
   d <- i[["status"]]
})
table(status)
summary(.results[[1]][["pm"]])
pm.NI <- lapply(.results, function(i){
   d <- as.data.frame(i[["pm"]])
   d$Value[ d$NSAV_COMM == "Food" & d$REG == "SSA" ]
})
pm.NI <- do.call(c, pm.NI)

load("./results/ACRS3X3_ex4.NA.RData")
.results <- mc.parallel
status <- sapply(.results, function(i){
   d <- i[["status"]]
})
table(status)
summary(.results[[1]][["pm"]])
pm.NA <- lapply(.results, function(i){
   d <- as.data.frame(i[["pm"]])
   d$Value[ d$NSAV_COMM == "Food" & d$REG == "SSA" ]
})
pm.NA <- do.call(c, pm.NA)

load("./results/ACRS3X3_ex4.YI.RData")
.results <- mc.parallel
status <- sapply(.results, function(i){
   d <- i[["status"]]
})
table(status)
summary(.results[[1]][["pm"]])
pm.YI <- lapply(.results, function(i){
   d <- as.data.frame(i[["pm"]])
```

```
   d$Value[ d$NSAV_COMM == "Food" & d$REG == "SSA" ]
})
pm.YI <- do.call(c, pm.YI)

load("./results/ACRS3X3_ex4.YA.RData")
.results <- mc.parallel
status <- sapply(.results, function(i){
   d <- i[["status"]]
})
table(status)
summary(.results[[1]][["pm"]])
pm.YA <- lapply(.results, function(i){
   d <- as.data.frame(i[["pm"]])
   d$Value[ d$NSAV_COMM == "Food" & d$REG == "SSA" ]
})
pm.YA <- do.call(c, pm.YA)


## Table comparing standard deviations of prices with and without
## considering correlations or policies. Included are the p-values of
## ratio test variances where the null hypothesys is that the
## variances are equal:
.t <- rbind(
   c( sd(pm.NI), sd(pm.YI), var.test( pm.NI, pm.YI)$p.value ),
   c( sd(pm.NA), sd(pm.YA), var.test( pm.NA, pm.YA)$p.value ),
   c( var.test( pm.NI, pm.NA)$p.value, var.test( pm.YI, pm.YA)$p.value
   , 0 )
)
colnames(.t) <- c("no.corr", "yes.corr", "p-value")
rownames(.t) <- c("Standard Model", "Variable Export Tax", "p-value")

library(xtable)
xtable(.t, digits = 3)

quantile( pm.YI, prons = c(0.025, 0.975))
quantile( pm.YA, prons = c(0.025, 0.975))

t.test( pm.YI, pm.YA )

d.pm.YI <- density( pm.YI)
d.pm.YA <- density( pm.YA)
xmin <- min( d.pm.YI$x, d.pm.YA$x )
xmax <- max( d.pm.YI$x, d.pm.YA$x )
plot(d.pm.YI, xlim = c(xmin, xmax), main = "", xlab = "%-change in
prices")
lines(d.pm.YA, col = "red")
```

5. Concluding Remarks

The statistical programming language R is a powerful tool to use for data manipulation and analysis. GEMPACK is a powerful and efficient language for solving Computable General Equilibrium Models. This paper discusses two functions that allow reading the output of CGE models into R for further analysis. The paper also demonstrates R's scripting abilities to run GEMPACK. We demonstrate the usefulness of coupling GEMPACK and R for conducting systematic sensitivity analysis of model results using Monte Carlo experiments. Of course, other areas of application are also possible, from streamlining R-GEMPACK work flows to keeping track of recursive simulations. Nevertheless, the focus on systematic sensitivity analysis using Monte Carlo experiments was motivated by the possibility that increasingly accessible multi-core computers and computer clusters offer the possibility of conducting more sophisticated statistical analysis of CGE results. This has the potential to produce deeper insights than with the currently used tools, especially in modeling exercises where non-linearities are important. Examples of these applications are found often and range from the evaluation of policies that can abruptly change price regimes to climate-driven productivity shocks whose distribution may be highly asymmetrical.

Acknowledgements

This project is supported by Agriculture and Food Research Initiative Competitive Grant no. 2015-67023-25258 from the USDA National Institute of Food and Agriculture.

References

COPS. 2017a. "Running GEMPACK Programs from another program." http://www.copsmodels.com/gp-runprog.htm.

COPS. 2017b. "Using BAT files to run GEMPACK Programs automatically." http://www.copsmodels.com/gp-bat.htm.

Harrison, J., J.M. Horridge, M. Jerie, and K. Pearson. 2016. "GEMPACk Manual." http://www.copsmodels.com/gpmanual.htm#gpd4.2.5.3.

Hertel, T.W., W. Martin, and A.M. Leister. 2010. "Potential Implications of a Special Safeguard Mechanism in the World Trade Organization: the Case of Wheat." *The World Bank Economic Review*, 24(2): 330–359. doi:10.1093/wber/lhq010.

Horridge, M., and M. Jerie. 2013. "Performing multiple simulations using CMF parameters with GEMPACK 11.2: the mystery of Horridge's kink." In *Presented at the 16th Annual Conference on Global Economic Analysis, Shanghai, China*. Shanghai, China. http://www.gtap.agecon.purdue.edu/resources/res_display.asp?RecordID=4268.

Knaus, J. 2010. "snowfall: Easier cluster computing (based on snow)." *R package version*, 1. http://cran.case.edu/web/packages/snowfall/.

Pearson, K., and C. Arndt. 2000. "Implementing Systematic Sensitivity Analysis Using GEMPACK." Purdue University, West Lafayette IN, Technical Paper No. 03.

R Core Team. 2017. "R: A Language and Environment for Statistical Computing." http://www.R-project.org.

Ramirez, O.A., S. Misra, and J. Field. 2003. "Crop-Yield Distributions Revisited." *American Journal of Agricultural Economics*, 85(1): 108–120. doi:10.1111/1467-8276.00106.

Valenzuela, E., T.W. Hertel, R. Keeney, and J.J. Reimer. 2007. "Assessing Global Computable General Equilibrium Model Validity Using Agricultural Price Volatility." *American Journal of Agricultural Economics*, 89(2): 383–397.

Venables, W.N., and B.D. Ripley. 2003. *Modern Applied Statistics with S*. Springer Science & Business Media.

Villoria, N.B., and E.W. Mghenyi. 2016. "The Impacts of India's Food Security Policies on South Asian Wheat and Rice Markets." *The World Bank Economic Review*, Feb., pp. 1–22. doi:10.1093/wber/lhw002.

Villoria, N.B., and P.V. Preckel. 2017. "Gaussian Quadratures vs. Monte Carlo Experiments for Systematic Sensitivity Analysis of Computable General Equilibrium Model Results." *Economics Bulletin*, 37(1): 480–487.