Journal of Global Economic Analysis, Volume 8 (2023), No. 1, pp. 1-20.

GEMPACK simulations in R: A demonstration of running the GTAP model and processing its results entirely in R using packages `HARr` and `tabloToR`

By MAROS IVANIC [a]

With the newly available R packages `tabloToR` and `HARr`, it is now possible to run General Equilibrium Modelling PACKage (GEMPACK) models expressed in the TABLO language entirely in R without the need for a Fortran compiler, the GEMPACK software or any licenses. Working through a simple Global Trade Analysis Project (GTAP) simulation, we demonstrate how the simulation may be done in R and show that its results are virtually identical to those obtained in GEMPACK. With `tabloToR` and `HARr` offering a working replacement for GEMPACK, the packages could benefit those CGE modelers who are more comfortable working in R than in GEMPACK, and bring about benefits to the Computable General Equilibrium (CGE) modeling community through eliminating licensing costs and introducing additional efficiencies through easier integration and development of new features.

JEL codes: C68, D58

Keywords: GEMPACK; R; tabloToR; HARr; CGE Analysis;

1. Introduction

The newly available packages `tabloToR`[1] and `harR`[2] allow Global Trade Analysis Project (GTAP) modelers to perform a General Equilibrium Modelling PACKage (GEMPACK)-style (Harrison and Pearson 1996) analysis wholly within R, without the use of GEMPACK code or a GEMPACK license, and without the need to run executable files. The introduction of these open-source packages in an open-source

[a] United States Department of Agriculture, Economic Research Service, 1400 Independence Ave, SW, Washington DC, 20250 (maros.ivanic@usda.gov)

[1] https://github.com/mivanic/tabloToR; version used in this work: 2817fcaa7fcd911a20da41f924873d856f1bf9c2

[2] https://github.com/mivanic/HARr; version used in this work: a97dd9a3ccc8b742c74ac68448f1f64c34b9e8f5

programming language, which is available for multiple operating systems, makes it possible to perform GEMPACK-style economic analysis on non-Windows computers and for a lower cost (or essentially no cost when using open-source operating systems).

In addition to saving on licensing costs, being able to run GEMPACK models that are formulated in the TABLO language entirely in R removes an important cost associated with having to learn another software (e.g., GEMPACK or GAMS[3]). While many, if not most of, modern applied economists are familiar with R, very few have used Fortran and GEMPACK, which require additional expertise to be able to install a Fortran compiler that is compatible with the desired version of GEMPACK, build the GEMPACK system, and use it to compile executable files out of TABLO files. Some of these steps may even require administrative rights (depending on the IT policies in place), which may make the process unduly difficult if not impossible.

The ability to run TABLO-formulated models, such as GTAP, within R, opens many new possibilities to modelers. For example, in addition to removing the historical constraints on set names, and their length,[4] moving all work into R allows the model and the model results to be processed with other technologies that are available and widely used in R, such as saving the results to a database, generating automated visualizations of the results, or providing an interactive interface to a model in a browser. Perhaps even more importantly, it also allows the GTAP model to be integrated with other models that exist in R. Finally, the availability of the GEMPACK-models in R makes them accessible to the new generation of modelers who may not be familiar with Fortran or GEMPACK, or who may not be using the Windows operating system.

It is important to mention that running models formulated in the TABLO language in R through package `tabloToR` is quite different from the notable efforts (e.g., Rutherford and Harbor 2005; Britz and Mensbrugghe 2018) that have ported models, such as GTAP, to GAMS manually based on the TABLO code, essentially building parallel models that after porting no longer depend on the original TABLO code. Unlike such systems, package `tabloToR` maintains the TABLO code as the definition of the model, meaning that any further development of the TABLO code will be picked up and implemented by the package. Also, an important goal of `tabloToR` is to provide for fully open-source and modern way to interface models with other software and technologies (e.g., databases), which is something that is not possible in GAMS, which is commercial software representing fairly dated technology that is difficult to interface with modern systems.

Another important point to highlight is that running TABLO models in R implies that the solution represents a perturbation from the initial state (comparative static)

[3] GAMS is another commonly used commercial software system capable of running Computable General Equilibrium (CGE) models, such as GTAP (Rutherford and Harbor 2005)
[4] The names of objects in R are virtually unlimited in length and with few restrictions on their composition; for details see Venables, Smith, and the R Core Team (2022)

because the model assumes that the initial state of the data is in equilibrium (i.e., a vector of zeros represents a solution to the model). `tabloToR` replicates the solution approach of GEMPACK that allows it to calculate a close approximation of the solution by splitting the shocks to a series of smaller shocks and tracing the non-linear solution by using multiple updates of the data and Richardson extrapolation. This is very different from the implementations of models in GAMS that are expressed as levels, which means that they need to be first calibrated, but then they can be solved exactly without any extrapolation efforts. In principle, there is nothing preventing `tabloToR` from turning TABLO models into levels and solving them exactly using open-source solvers with the future development.

Switching technologies usually involves additional costs of having to learn the new technology. To make it easier for the modeling community to use the R packages—which operate a bit differently from the usual GEMPACK and require a slightly modified work flow—in this paper, we first explain the purpose of both packages and provide a few examples on their use. Secondly, we demonstrate how they can be used by performing a simple Computable General Equilibrium (CGE) analysis using the GTAP model. We then compare the results obtained in R to those obtained from a GEMPACK-compiled model to show that they are virtually the same, providing the assurance to the interested users that the packages are capable of producing solutions of a similar quality as GEMPACK.

Finally, we would like to discuss who could benefit from the use of `tabloToR`. The first obvious target audience involves those who would like to learn about GEMPACK-style models (e.g., GTAP) and to play with those models, without having to pay for and learn several additional technologies (e.g., Fortran, GEMPACK). This could include students or independent researchers, who understand R and are able to install a package and run it. The second group of researchers who might be interested in making modifications to the model (in the TABLO file) but would like to avoid the cost of compiling the model in GEMPACK (this can take considerable time). Finally, those researchers who are interested in linking GEMPACK models to other models could benefit from being able to avoid the need of executing their models at the system level (e.g., DOS) which comes with numerous technical issues.[5]

2. Accessing GEMPACK data in R: package `HARr`

One of the key features of CGE models is that they are computable, meaning that they can provide numeric solutions to shocks, based on the underlying data and parameters. In case of GEMPACK, data are generally provided in the format of `.har` files (header-array files), which are capable of holding numerous numeric arrays and/or string arrays (up to seven dimensions), inclusive of dimension labels. This is the case of the GTAP data, that is, by default provided as a set of HAR files (data,

[5] For example, GEMPACK compiled models cannot be run in parallel in the same directory without a danger of leakages of data between parallel runs.

parameters, sets).

Data in the form of `.har` files are read by the models as inputs. Models also generate a wealth of `.har` files with the results (e.g., updated `.har` files, `.slc` or `.sl4` files). Typically, researchers do not modify the input data, as the data are delicately balanced to assure that the initial state of the world is balance. However, researchers often have the need to modify the input parameters and to process the outputs of the models.

Working with GEMPACK data in `.har` format is generally not easy, owing to the fact that the format is specific to very old versions of Fortran (e.g., designed to work with magnetic tapes) and it has been further customized for the use in GEMPACK models. As a result, there are no readily available third-party data connectors that would be able to read from or write to `.har` files. The most commonly used way of manipulating `.har` files has been the use of GEMPACK-provided executable programs. Unfortunately, these programs (e.g., `seehar.exe` or `modhar.exe`) are often difficult to use, because they require the ability to be executed by the user, which is often disabled by administrators for security reasons. Additionally, these programs only run interactively, making it impossible to create a data connection to modern software (e.g., ODBC, R, etc.) as they have to be executed interactively. Faced with those difficulties, many researchers choose to move the data between `.har` files and their favorite data editors (e.g., Excel) using a simple clipboard, which is highly manual and prone to error.

`HARr` is a package that allows reading and writing of `.har` files directly into and from R. Unlike the existing solutions which act as wrappers around GEMPACK `.exe` files, this package is written in R and does not depend on any other software. It was developed by reverse-engineering the available `.har` files. The main benefit of the package is in allowing researchers to easily read in the data (into R or convert them into any other popular format), modify them and write them back into another `.har` file.

Another important and useful feature of the package is an option allowing to read all data in lower-case (the default option), allowing it to bridge the gap between case-insensitive GEMPACK and most modern programming languages that are case sensitive, including R. Without this option, it is possible for the sets to be defined inconsistently,[6] resulting in likely problems.

Please see the appendix of this paper or the GitHub documentation on this package for more important details on its use with examples.

[6] For example, a HAR file could specify a particular element of a set in upper case, e.g., 'Capital', but at the same time use this element as a data label in lower-case, e.g., 'capital', resulting in two different labels and likely errors in processing.

3. Executing GEMPACK models in R: package `tabloToR`

The purpose of this package is to interpret and solve a model written in the TABLO language into a model in R. The package essentially replicates the steps that a GEMPACK compiled model would take to generate a solution for a set of shocks to the model. This include interpreting a model written in the TABLO language into a system of equations, filling the coefficients of the system with the actual data, solving the system, and reporting the results.

Even though the steps that need to be taken to work through the solution of an economic model are the same, there are significant differences in the actual workflow between doing a CGE simulation in GEMPACK and `tabloToR`: in GEMPACK, the work is split into small steps which are executed by a set of executable programs, with each program typically receiving inputs as a separate file along with a command file, and writing the results to another file, which may become an input in another step in the process. For example, a modeler might run `ltg.exe` to turn a TAB file into an executable file and then run the executable file to turn the shock file into a file with the simulation results.

The workflow in R is very different because all of the steps are executed in the computer memory, within R's environment. This means that no files are produced in the course of compiling and running a model: instead, the results are added to the existing object, which is modified in the process. The benefits of this approach include the fact that the modeler is no longer required to keep track of a set of files associated with a simulation (e.g., `.tab`, `.cmf`, `.exe`, `.sl4`, `.slc`)—because the entire simulation with its results is a single object in R, it is easy to archive all model results in a single file, and revisit them in the future. The second benefit comes from the fact that with no potential conflicts in file names, the modeler can run the model (or many models) in parallel (e.g., using R's `parallel` package) without any danger of data corruption. This is of special value when using the package to run a Monte Carlo analysis, which require changing only some input parameters or shocks, and running the model many times, preferably in parallel.

To help the users familiar with conducting CGE analysis in GEMPACK, in Table 1 we show brief overview of the steps that they normally go through and their comparison with the steps taken when using `tabloToR`.

Table 1. Overview and comparison of the steps taken in GEMPACK and ‘tabloToR‘

Step in GEMPACK	Corresponding implementation in tabloToR
Compile a ‘.tab‘ file into an ‘.exe‘ file	function ‘loadTablo()‘, e.g., ‘model$loadTablo('GTAP.tab')‘
Specify which variables are exogenous using ‘exogenous‘ statement (CMF file)	Element ‘variableValues‘ implies which variables are exogenous (NA=endogenous, otherwise exogenous (shock) value)
Swap variables from endogenous to exogenous (or vice versa) using ‘swap‘ statement (CMF file)	‘variableValues‘ element implies which variables are exogenous and swapping only requires changing the values in the element
Specify shocks using ‘shock‘ statements (CMF file)	All definite values in element ‘variableValues‘ in the model represent the shocks
Run the model (e.g., ‘gtap.exe‘)	function ‘solveModel()‘, e.g., ‘model$solveModel()‘
Secify the solution method (e.g., Euler, 1,3,5 steps specified in the CMF file)	As an argument of the ‘solveModel()‘ function, e.g., ‘model$solveModel(method = 'euler', steps = c(1,4,5))‘
Specify the number of subintervals (in CMF file)	As an argument of the ‘solveModel()‘ function
Retrieve the solution (‘.slc‘ and ‘.sl4‘ file) in ViewSol or AnalyzeGE	All results are available in the model object in element ‘data‘, e.g., ‘model$data$wev

Please see the appendix of this paper or the GitHub documentation on this package for more important details on its use with examples.

4. An example simulation done entirely in R using packages `HARr` and `tabloToR`

To illustrate how it is possible to run a GTAP model in R, we define and solve a relatively simple GTAP simulation using the classic GTAP model (6.2) (Hertel

1997) [7] [8]. We use the latest GTAP v 11 (pre-release 4) data (Aguiar et al. 2019) aggregated into twelve regions (shown in Table 2), thirteen commodities (Table 3) and five factors. The simulation involves increasing productivity of agriculture in North America by one percent while reducing its import tariffs on agriculture and processed food from all other countries to zero.

Table 2. Aggregation of regions in the model

	Included countries and regions
aunz	aus, nzl, xoc
eastasia	chn, hkg, jpn, kor, mng, twn, xea
seasia	brn, khm, idn, lao, mys, phl, sgp, tha, vnm, xse
sasia	afg, bgd, ind, npl, pak, lka, xsa
northam	can, usa, mex, xna
southam	arg, bol, bra, chl, col, ecu, pry, per, ury, ven, xsm
centralam	cri, gtm, hnd, nic, pan, slv, xca, dom, hti, jam, pri, tto, xcb
eu	aut, bel, bgr, hrv, cyp, cze, dnk, est, fin, fra, deu, grc, hun, irl, ita, lva, ltu, lux, mlt, nld, pol, prt, rou, svk, svn, esp, swe, gbr
resteur	che, nor, xef, alb, srb, blr, rus, ukr, xee, xer
oasia	kaz, kgz, tjk, uzb, xsu, arm, aze, geo, bhr, irn, irq, isr, jor, kwt, lbn, omn, pse, qat, sau, syr, tur, are, xws
nafrica	dza, egy, mar, tun, xnf
ssafrica	ben, bfa, cmr, civ, gha, gin, mli, ner, nga, sen, tgo, xwf, caf, tcd, cog, cod, gnq, gab, xac, com, eth, ken, mdg, mwi, mus, moz, rwa, sdn, tza, uga, zmb, zwe, xec, bwa, swz, nam, zaf, xsc, xtw

[7] https://www.gtap.agecon.purdue.edu/resources/res_display.asp?RecordID=2458

[8] Even though version 7 of the GTAP model is available, this model does not appear to be properly described (e.g., closure) on the GTAP website; the version described by Corong et al. (2017) unfortunately does not work with the standard GTAP database

Table 3. Aggregation of commodities in the model

	Included commodities
grains	pdr, wht, gro
v_f	v_f
osd	osd
c_b	c_b
pfb	pfb
ocr	ocr
ctl	ctl
oap	oap
rmk	rmk
wol	wol
mnfc	frs, fsh, coa, oil, gas, oxt, tex, wap, lea, lum, ppp, p_c, chm, bph, rpp, nmm, i_s, nfm, fmp, ele, eeq, ome, mvh, otn, omf
food	cmt, omt, vol, mil, pcr, sgr, ofd, b_t
serv	ely, gdt, wtr, cns, trd, afs, otp, wtp, atp, whs, cmn, ofi, ins, rsa, obs, ros, osg, edu, hht, dwe

Step 1: Setting up an empty model

The first step in running a TABLO model in R involves setting up a new object of type `GEModel` found in package `tabloToR`. Unlike in GEMPACK where a simulation involves generating numerous files and running a number of programs, all of the analysis in R is done by performing operations on the single object of type `GEModel`, which contains a number of useful method and properties. For example, this object is capable of loading the data, setting up the model closure, and, after solving, it will contain the model solutions.

The following line sets up an empty model:

```
model = tabloToR::GEModel$new()
```

By initializing object `model` of type `GEModel`, we have loaded the required solution framework. Please note that there are no equations, variables, or data loaded at this point.

Step 2: Loading the model from the TABLO file

The following line loads the TABLO file and breaks it down into variables, equations, updates, and coefficients:

```
model$loadTablo('gtap.tab')
```

In this case, we have loaded the classic GTAP model exactly as it appears on the website of the GTAP Center.

Please note that loading a model without any supporting data means that all variables, coefficients and equations only exist as definitions at this point.

Step 3: Loading the external files that are required in the TABLO file

The following few lines prepare a data object filled with the actual data retrieved from the `.har` files distributed by the GTAP Center. Please note that the file names may be different, depending on whether how the data are aggregated (e.g., `FlexAgg` or `GTAPAgg`, etc.):

```
data = list(
  gtapsets = HARr::read_har('gsdgset.har'),
  gtapparm = HARr::read_har('gsdgpar.har'),
  gtapdata = HARr::read_har('gsdgdat.har')
)
```

Object `data` will be later inputted into object `model` to serve as the basis for all run-time data in the model. This object needs to contain all of the files that are mentioned in the TABLO file. In our example, these are `gtapsets`, `gtapparm` and `gtapdata`.

Note that this is the only time that package `HARr` is required in order to turn the `.har` files into an R data object.

Step 4: Loading the data into the model

The following line loads the data into the model:

```
model$loadData(data)
```

With this step we have initialized all sets and coefficients with actual values. Using these values, and variable and equation definitions, the model is able to generate the actual system of equations to be solved.

Step 5: Setting up exogenous variables and shocks

Unlike the closure in GEMPACK which is done in a separate `cmf` file where a command exists to declare variables as exogenous, in `tabloToR` it is only necessary to specify values for those variables that are exogenous. For convenience, all of the variables of the model are listed in the element `variableValues` of the model object. Initially, all values are uninitialized (set to `NA`), indicating that they are to be solved for (i.e., they are endogenous to the model). Only those values that are set to actual values by the modeler are considered exogenous by the model.

The following lines of code set up the variables in the standard closure as exogenous. If the entire variable is exogenous, we specify that (using R's notation []); alternatively, we can specify only those elements that are exogenous (such is the case in variable `qo`):

```
# These variables are set to 0 entirely
for(var in c("afall", "afcom", "afeall", "afecom", "afereg",
      "afesec", "afreg", "afsec", "ams", "aoall", "aoreg",
      "aosec", "atall", "atd", "atf", "atm", "ats", "au",
      "avaall", "avareg", "avasec", "cgdslack", "dpgov", "dppriv",
      "dpsave", "endwslack", "incomeslack", "pfactwld", "pop",
      "profitslack", "psaveslack", "tf", "tfd", "tfm", "tgd",
      "tgm", "tm", "tms", "to", "tpd", "tpm", "tp", "tradslack",
      "tx","txs")){
  model$variableValues[[var]][]=0
}
# only some qo's are exogenous--notice that we can use the defined
# sets, e.g., model$data$endw_comm
model$variableValues$qo[model$data$endw_comm,] = 0
```

We may now specify our shocks, taking advantage of the fact that we can use the standard R language, and that all of the set definitions and data are available for us to use in object `model`.

The following lines demonstrate the calculation and implementation of shocks for a subset of commodities and regions:

```
# Set up a vector of agricultural commodities
ag = c("grains", "v_f", "osd", "c_b", "pfb", "ocr", "ctl", "oap"
, "rmk", "wol")
# Set up a vector of agricultural commodities
targetCtries = c("northam")
# Set the values to 1, representing a one percent increase in overall
# productivity
model$variableValues$aoall[ag,targetCtries] = 1

# Set up a vector of agricultural and food commodities
agFood = c(ag, "food")

# specify the change in tariffs (tms) based on the values of imports
# CIF and FOB
model$variableValues$tms[agFood,,targetCtries] =
  (model$data$viws[agFood,,targetCtries] / model$data$vims[agFood,,
targetCtries] -1)*100
```

The ability of the modeler to use the actual R language to specify the shocks is of great value. In GEMPACK, it is often required to calculate shock values (e.g., the percentage change in the power of the tax to bring to power to a desired level) on the side or to take advantage of special modules for commonly preformed shocks. For example, to bring the tariff level to zero for a set of commodities and trade

partners, it is necessary either to calculate each shock before running the simulation outside the simulation workflow (i.e., outside the CMF file), or to change the model by introducing a new coefficient representing the power of tax and instructing the variable `tms` to trace it, which then allows the use of a special command `final_level` in the `cmf` file.

With the method of determining which variables are exogenous to the model that we described above, it is easy to toggle the type of a variable between exogenous and endogenous. Because all exogenous variables are indicated by having definite values, there is no need for the GEMPACK's swap statement. Instead of swapping variables, all that is needed is replacing the value of an exogenous variable with a missing value (NA), an assigning a definite value to the currently exogenous variable in order for its value to be come fixed. For example if one wanted to swap the quantity of exports (qxs) with the tariff rate (tms), all that is required is to set the appropriate value of tms to NA and set the value of exports to the desired level.

The final step is to initiate the solution. In the example, we specify three iterations and two sets of steps (1 and 3)—this solution method for such small shocks should closely replicate the commonly selected solution methods in GEMPACK—as in the following line of code. The default solution method is Euler:[9]

```
# Solve the model with the existing shocks using three subintervals
# and 1 and 3 steps for extrapolation
model$solveModel(iter = 3, steps = c(1,3))
```

It is important to note that once the model has been solved, the model's data has been updated. This means that running the same `solveModel()` function again, will apply the chocks to the updated model data.

Step 6: Review of the results

Once the model solves, all of the solutions are available in the `data` element of the model. Therefore, if we would like to view the results for domestic prices (pm), we can obtain them through the following command:

```
# Display the value of variable pm at the conclusion of the simulation
model$data$pm
```

5. Comparison of the results between `tabloInR` and `GEMPACK`

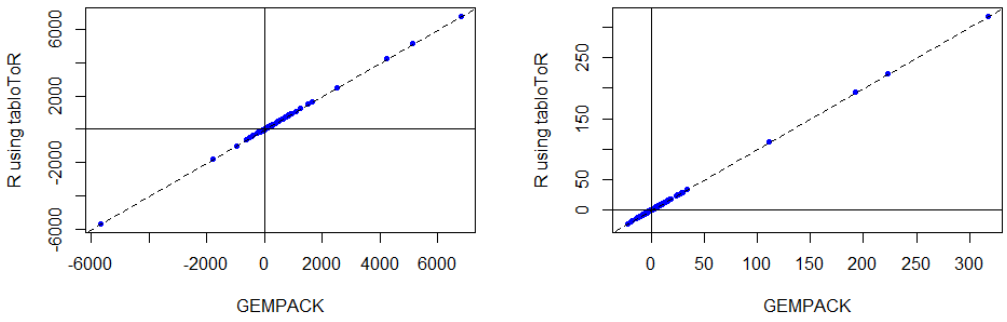
For researchers to accept and use `tabloToR` in their work, it is important to demonstrate that the results obtained using either system are identical beyond any rounding differences. To show that this is the case in our sample simulation, we present a comparison of the results obtained by `tabloInR` with the results obtained from a GEMPACK simulation with the same solution method (Euler 1-3 steps, three

[9] For an excellent description of the solution methods, please consult the GEMPACK documentation at https://www.copsmodels.com/gpmanual.htm

subintervals) in Figure 1 for all variables. In the figure, we draw a scatter plot of all change and percentage change variables obtained in the simulation. Clearly, the plot values lie on the diagonal, showing that the results between the two simulations are virtually identical.

Next, we provide a comparison form some of the key variables that are frequently of interest following a productivity and trade policy simulation. Specifically, we include the implications for domestic prices, output, trade and welfare. We present the results for the domestic prices, output imports and exports in North America (notham), and consumer price indices and welfare for all of the regions in Figure 2. Again, the set of the figures makes it clear, the results obtained from `tabloToR` are virtually identical to those obtained from GEMPACK.

At this point, it is important to reiterate the fact that while unlike GEMPACK, R is case sensitive. For this reason, the variables presented in the comparison have been put into the same lower-case (e.g., EV→ev).



(a) Change variables (18,170 observations)

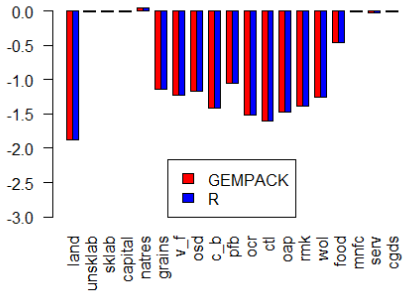
(b) Percentage change variables (56,550 observations)

Figure 1. Comparison of all variable values obtained from the simulation conducted in R using tabloToR and GEMPACK

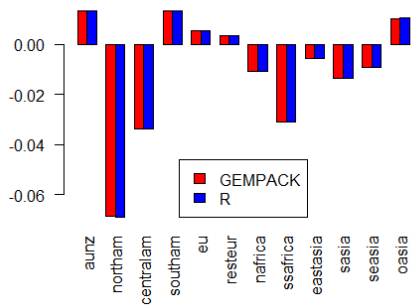
6. Conclusion

Running GEMPACK models expressed in the TABLO language, such as the GTAP model, wholly in R is now possible thanks to the newly available packages `tabloToR` and `HARr`. We have demonstrated that even though the workflows of the Fortran-based GEMPACK and R are somewhat different, a modeler is able to run the model and review the results without the need for a Fortran compiler, GEMPACK license or running any executable files. This means that the use of the R packages can expand the use of CGE modeling by making it possible to run the models free of license charges at a much wider set of operating systems.

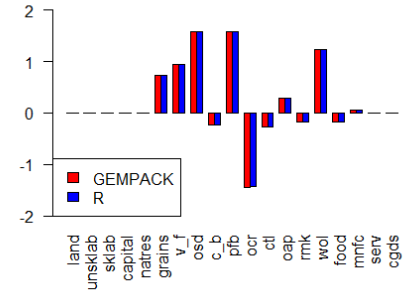
We have also shown that performing CGE analysis in R rather than in GEMPACK



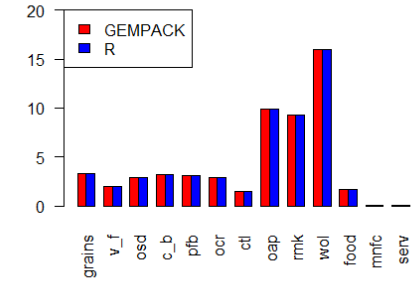
(a) Price changes (pm) in North America



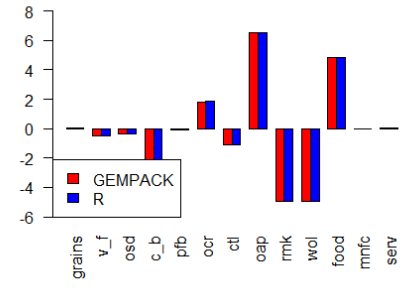
(b) CPI (ppriv)



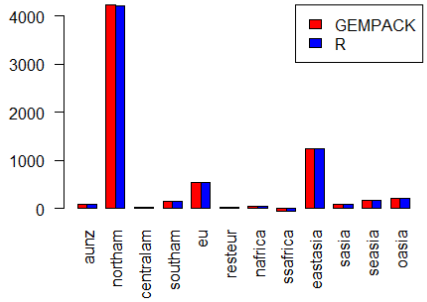
(c) Output change (qo) in North America



(d) Export changes (qxw) in North America



(e) Import changes (qiw) in North America



(f) Welfare (ev)

Figure 2. Comparison of the results obtained in GEMPACK and 'tabloToR'

offers several additional benefits to the modeler. First, the preparation of the simulations can be scripted in R, which makes it possible to set up, execute and interpret the results in a single file using the standard R language. The fact that the results of the simulation are also produced in R opens up a lot of opportunities to integrate the modeling work with the vast ecosystem of the solutions developed in R already. For example, with the model, such as GTAP, expressed and solved in R, it would be possible to link the model to other models in R, to present the results interactively in the browser (e.g., by using R Shiny) locally or on the web, link the outputs to external databases, etc.

An important feature of the `tabloInR` is that the results that it produces are essentially the same as produced by GEMPACK. This should give the modelers the assurance that switching the technologies will not impact negatively the results. After all, a well-defined model is just a system of equations which has one and one only solution and it should not matter which technology is used as long as the solution is correct.

Even though `tabloInR` is a solid replacement for GEMPACK, its development should continue. Fortunately, with the package being open-source and available for improvements on GitHub, the research community now has the opportunity to contribute directly to its development. This is quite different from the development of commercial software (e.g., GEMPACK) where the development is limited by the available resources. In practical terms, any research is able to make improvements to the package and propose those improvements to be implemented in the product (subject to standard tests to assure non-breaking changes).

Some of the obvious areas where `tabloInR` should be improved is increasing its speed. Even though it is possible to run a model of about dozen regions, dozen commodities and five factors in a few minutes (at the time of writing this paper), it would be desirable to reduce this time to seconds (as is the case when using GEMPACK), perhaps by importing the already existing open-source libraries in Fortran or C to R[10] to invert large matrices more efficiently. Additionally, it would be useful to bring all of the existing features from GEMPACK, such as extending the repertoire of the system's solution methods, for example in addition to `Euler` introduce `Midpoint` and `Gragg` and also automatic accuracy options. With `tabloToR`[11] being functional, it is now up to the research community to join efforts in improving the package further.

Disclaimer:
The findings and conclusions in this article are those of the authors and should not be construed to represent any official USDA or U.S. Government determination or policy. This research was supported in part by the USDA, Economic Research

[10] R is capable of using compiled Fortran or C functions natively

[11] Please follow the documentation of the package at https://github.com/mivanic/tabloToR to see which feature have been developed or are being developed

Service.

Acknowledgements

I thank Jayson Beckman and Noe Nava for their helpful comments on the earlier drafts of the paper, and Mark Horridge for his helpful technical assistance when I got stuck. I also thank Roberto Roson for his enthusiastic review of this work and many wonderful suggestions, and two anonymous reviewers for their excellent comments that greatly improved the final version of this paper.

7. References

Aguiar, Angel, Maksym Chepeliev, Erwin Corong, Robert McDougall, and Dominique van der Mensbrugghe. 2019. "The GTAP Data Base: Version 10." *Journal of Global Economic Analysis* 4 (1): 1–27. https://doi.org/10.21642/JGEA.040101AF.

Britz, Wolfgang, and Dominique van der Mensbrugghe. 2018. "CGEBox: A Flexible, Modular and Extendable Framework for CGE Analysis in GAMS." *Journal of Global Economic Analysis* 3 (2): 106–77. https://doi.org/10.21642/JGEA.030203AF.

Corong, Erwin L., Thomas W. Hertel, Robert McDougall, Marinos E. Tsigas, and Dominique van der Mensbrugghe. 2017. "The Standard GTAP Model, Version 7." *Journal of Global Economic Analysis* 2 (1): 1–119. https://doi.org/10.21642/JGEA.020101AF.

Harrison, W. J., and K. R. Pearson. 1996. "Computing Solutions for Large General Equilibrium Models Using GEMPACK." *Computational Economics* 9.

Hertel, T. W., ed. 1997. *The Standard GTAP Framework Is Documented in Global Trade Analysis: Modeling and Applications.* Cambridge University Press.

Rutherford, T., and A. Harbor. 2005. "GTAP6inGAMS: The Dataset and Static Model. Prepared for the Workshop: "Applied General Equilibrium Modeling for Trade Policy Analysis in Russia and the CIS", the World Bank Resident Mission, Moscow, December 1-9, 2005." http://www.mpsge.org/gtap6/gtap6gams.pdf.

Venables, W. N., D. M. Smith, and the R Core Team. 2022. *An Introduction to R.* https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf.

Appendix A. Package `HARr`

Package `HARr` offers three functions that are useful for performing GEMPACK based simulations: function `read_har` that reads the contents of a `.har` file as a list of data frames with the header names as the names of the elements, function `write_har` that writes a list of data frames into a `.har` file, and function `read_sl4` which reads the contents of `.sl4` files, which contain solutions of GEMPACK simulations.

The package is available at GitHub[12] and may be installed using package `devtools`:

```
devtools::install_github('mivanic/HARr')
```

Function `read_har`

Function `read_har()` is probably the most important and most commonly used function in the package. It allows reading the contents of `.har` files, which is the format that the GTAP database is distributed in, directly by R into R. Unfortunately, the `.har` format, which is well suited for reading and writing data on paper or magnetic tapes, is not commonly used by modern software. The current approach to reading these files (e.g., to GAMS or R) is to use the existing GEMPACK routines (`.exe` files distributed with the GEMPACK software) to export the contents of the files into other formats (such as csv) or to create a `.har` file from other text files; however, the need to use an executable file makes this approach difficult as it requires the installation of GEMPACK and the permission to run executable files within the operating system. Also, this approach is not available to many operating systems, as most of GEMPACK has been developed for Windows.

The function `read_har` included in package `HARr` reads the contents of `.har` files directly from the file. The function reads all headers contained in the file and reconstructs them into R object, such as string vectors, integer vectors, and real arrays into a list of objects. Each object in the list has the same name as its corresponding header. Therefore reading a typical GTAP database `.har` file will produce a single list with elements such as `vims` or `viws`, containing the data in the form of a numerical array.

Function `read_har` contains additional switches to allow a more useful conversion of `.har` files into R. The first one is switch `useCoefficientsAsNames` which tells the function to use the coefficient names (which are the secondary identifiers of the data contained in `.har` files, in addition to header names). This feature is extremely useful when the `.har` file contains the constructed data of a simulation (e.g., the `.slc` file) and where header names are meaningless.

The second important switch in function `read_har()` is `toLowerCase` which is set to TRUE by default. This switch assures that the string data read from a `.har` file are converted into lower case. This is important because GEMPACK is case

[12] https://github.com/mivanic/HARr

insensitive, which means that there may be inconsistency between set names found in the `.har` file, or additionally between `.har` files and the model TABLO file. To illustrate, the GTAP model references to capital as "Capital" in the TABLO file, while the set element is usually "capital." Because this difference in case is meaningful in R, it is important that the set element be the same between the data files and the model files.

Examples

A simple example of reading a `.har` file in R may be illustrated by the following line of code, which reads the specified `.har` file and turns it into a list of data objects with the names of the objects coming from the header names.

```
# Read a .har file into R
gtapData = HARr::read_har('gtapdata.har')
# Returns a list of arrays
```

To use this function to read the data part of the solution of a GTAP simulation (the `.slc` file), it is important to name the data objects using the coefficient names instead of the meaningless header names by using switch `useCoefficientsAsNames` set to true:

```
# Read a .har file with coefficient names as the data identifier
gtapData = HARr::read_har('solution.har', useCoefficientsAsNames=
TRUE)
# Returns a list of arrays where each array's name is the
# corresponding coefficient name
```

Function `read_sl4`

Even though an `.sl4` file is just a header array file, the data contained in it cannot be directly interpreted, because the solutions to the model variables are stacked into just a few headers, and it is therefore difficult to find the solution to a given variable or a given subtotal by looking at the raw headers. To provide the solved variables with their solutions and subtotals, it is necessary to process the header array file `.sl4` differently, using function `read_sl4()`. This function reads the solution file and converts it into a list of arrays, where each array represents the results for a single variable. Because SL4 files may contain subtotal in addition to the total values, `read_sl4()` adds the subtotals as layers in the outputted arrays. The final (total) solution is always labeled as `TOTAL` as is included as the last element in each array.

Example

This line of code shows how to read the variable solutions from an `.sl4` file into a list of numeric arrays:

```
# Read a solution file
solution = HARr::read_sl4('mySolution.sl4)
# Returns a list of arrays where each array corresponds to a variable
# in the model
```

Function `write_har()`

Function `write_har()` in package `HARr` can be used to save data into `.har` files. The required arguments are the data to be written (as a list of arrays with each name being at most four characters long) and a the file name.

Example

The following line of code writes a list of arrays into a `.har` file:

```
# Write a list of arrays into a .har file
HARr::write_har(myData, "myData.har")
```

Function `write_har()` is probably the least used one for CGE analysis in R as it is normally not needed to pass the results from R back to a `.har` file. However, it is included in the package for its completeness.

Appendix B. Package `tabloToR`

The main purpose of package `tabloToR` is to translate a TABLO file (e.g., GTAP model) into R code, and to build an object that allows the user to execute the model with actual data, parameters, closure, shocks, and a solution method, and to read the results of the simulation.

The package is available at GitHub[13] and may be installed using package `devtools`:

```
devtools::install_github('mivanic/tabloToR')
```

Because `tabloToR` creates an object with some important methods and properties, we discuss them here as well.

Function `GEModel$new()`

This is the initial function that sets up an empty object representing the model and its state (variable results, updated data, etc.)

```
model = tabloToR::GEModel$new()
```

Method `loadTablo`

The created model object has various properties and methods. The key method is `loadTablo(path_to_tablo_file)` that interprets a TABLO model and turns it into an R code inside the model object

```
model$loadTablo('gtap.tab')
```

Method `loadData`

Method load data initializes the model with actual data. The model representation in TABLO is general (e.g., set elements are not specified); running this method initializes all sets and coefficients.

Example

```
data = list(
  gtapsets = HARr::read_har('gsdgset.har'),
  gtapparm = HARr::read_har('gsdgpar.har'),
  gtapdata = HARr::read_har('gsdgdat.har')
)

model$loadData(data)
```

[13] https://github.com/mivanic/tabloToR

Method ***solveModel***

Method `solveModel` solves the model. It allows the specification of the solution method (e.g., Euler) and the number of subintervals.

Example

```
model$solveModel(iter = 3,steps = c(1,3))
```

Property ***variableValues***

Property `variableValues` allows the setting of the initial values of the variables contained in the model. It therefore allows the specification of the closure: the exogenous values (i.e., shocks) are indicated by setting their value; the endogenous variables are set to NA, indicating that their value will be determined by the model.

Example

```
model$variableValues$tms[] = 10
```

Property ***data***

Property `data` contains the solved variable values after the model has been solved.

Example

```
model$dta$qgdp[]
```