

The GTAP model in R

The benefits of involving the community in the development of
general equilibrium tools for the future

Maros Ivanic¹

¹Economic Research Service, United States Department of
Agriculture

June 1, 2020

The findings and conclusions in this paper are those of the author and should not be construed to represent any official USDA or U.S. Government determination or policy.

Abstract

Computable general equilibrium analysis has been greatly facilitated by dedicated computing frameworks, such as GEMPACK, that allow the specification of models and scenarios in a largely human-readable format and are also able to solve them efficiently. While GEMPACK does its job well, it remains privately developed software that follows standards and conventions that have departed significantly from the mainstream which make it hard to integrate it with most commonly used analytical frameworks, such as R.

In this paper we try to answer the question whether open-source, community-supported solutions may provide a viable substitute to GEMPACK's functionality. We perform a sample GTAP model simulation in R using several newly available R packages and compare the results and performance between the frameworks. We find that while the speed of the solvers currently available in R is lagging, especially for larger models, it is likely that the community of researchers would be able to eliminate this performance gap. We also expect that the benefits of being integrated in the R framework with its vast ecosystem of solutions more than compensates for the extra run time in R.

Contents

1	Introduction	3
1.1	The important role of GEMPACK in computable general equilibrium analysis	3
1.2	The break from mainstream technologies	3
1.3	The threats to the future of GEMPACK in CGE modeling in competition with open-source solutions	4
1.4	A case for community-driven, open-source CGE analysis	5
2	Bringing GEMPACK functionality to R	7
2.1	HARr: Direct access to HAR files in R	7
2.2	TabloToR: Interpret and solve a TABLO model in R	7
2.3	Solution outputs	8
3	Example: Running a GTAP model in R	8
3.1	Model and simulation definition	9
3.2	Executing the model in R	10
3.3	Viewing the results of the simulation	12
3.4	Comparison with GEMPACK results	12
3.4.1	One-iteration results versus Johansen	13
3.4.2	Six-iteration results versus six-iteration Gragg	13
3.5	Comparison of performance	15
4	Conclusions	15

1 Introduction

1.1 The important role of GEMPACK in computable general equilibrium analysis

The introduction of GEMPACK [1] as a comprehensive computing framework to define and solve general equilibrium models marked a beginning of an important and successful project. Even though computers have been able to complex mathematical systems for some time, they require high level of understanding of programming which represents a steep learning curve and a barrier to using them among many economists who are not specially trained in computer science. By creating a simpler, human-readable language that can define most commonly encountered equations found in general equilibrium analysis, GEMPACK has greatly reduced this barrier and made computable general equilibrium accessible to many economists.

In addition to making CGE analysis easier to perform, GEMPACK's TABLO language has brought a common language to researchers who often tended to develop their own special purpose models. [2] With one language to define most problems and an out-of-the-box installation of GEMPACK, the community of CGE researchers benefited greatly from being able to exchange and use each other's solutions without much effort.

Finally, with its efficient solver specialized in the type of models frequently encountered in CGE analysis, GEMPACK has also resolved an important technological constraint of CGE analysis where the size of the mathematical models often exceeded the available computing power. Allowing relatively large models to be solved on an average personal computer GEMAPACK has greatly increased the number of analyses that the researchers could perform.

1.2 The break from mainstream technologies

There should be no surprise that GEMPACK was developed in Fortran. When the development GEMPACK started in the late 1980's, Fortran was a popular computer language to solve problems in physics, including systems of equations which was precisely the problem that the CGE represents. The choice of Fortran appears to have helped initially for GEMPACK to take off by allowing it to use some of the existing solutions for file management and linear algebra libraries. Over time, however most developers have moved away from Fortran and embraced object-oriented computer languages and

functional programming that sped up software development through greater code reusability and more robust testing.

The origin of GEMPACK in Fortran has also left its mark in the types of input-output communication systems that it supports which are quite limited and fairly outdated. Virtually all of the communication of a GEMPACK system is reduced to physical files in a format that had been developed for tape data storage, i.e. header-array files that are constructed to allow for easy rewinding and fast-forwarding. Today's systems, however, are much more diversified in the communication channels that they support, such as API calls, and they generally use standard data formats to facilitate interoperability.

The focus of the GEMPACK framework has been on creating a series of executable images executed by the operating system, rather than an interpreted language independent of the operating system and executed in real-time. For example, GEMPACK models, such as GTAP, are frequently compiled into executable files on Windows computers and executed by the operating system. Also the various utilities made available to researchers are compiled as programs executed by the operating system. This approach represents a major departure from modern software development which is currently much more concerned with portability across operating systems. By creating solutions that do not depend on a small set of operating systems, the developers are free to focus on the problem at hand rather than idiosyncrasies of a specific operating system.

Finally, keeping GEMPACK as a privately developed system represents another departure from mainstream development method where development of highly specialized analytical software is now more frequently done as an open-source platform which allows for many more individuals interested in the project to contribute to it.

1.3 The threats to the future of GEMPACK in CGE modeling in competition with open-source solutions

Not being able to utilize community-supported development due to its proprietary nature is probably the greatest single threat to the future of GEMPACK. The need to pay for the entire cost of software development instead of plugging in existing open-source solutions means that there can only be limited resources that can be devoted to address user needs, reducing the

overall speed of development.

The departure of GEMPACK from most mainstream development approaches and technologies also means that most features have to be developed from scratch since the existing solutions cannot be readily adapted, further increasing the cost of development. For example, bringing GEMPACK to the cloud would require reworking significant parts of the system while this would be typically a small task to achieve in modernly designed systems.

In addition to missing on the community’s contribution to its core development, GEMPACK also misses on important lateral improvements, which may not be necessary to run a CGE model but which make the analysis easier to perform. These include an extension of the framework to other operation systems (Linux, iOS, Chrome, etc.), creating support additional file formats (e.g., JSON, XML, etc.), data connections (ODBC, SQL Server, etc.) or support for cloud computing and many others.

The final threat to GEMPACK comes from the fact that a new generation of researchers is entering the field with higher expectation of the software functionality, including its integrability with other tools. After all, CGE analysis is becoming much more data driven, relying on other data sources and it is therefore very important to be able to plug in a data connection definition rather than download the data and process them into a new HAR file each time something changes.

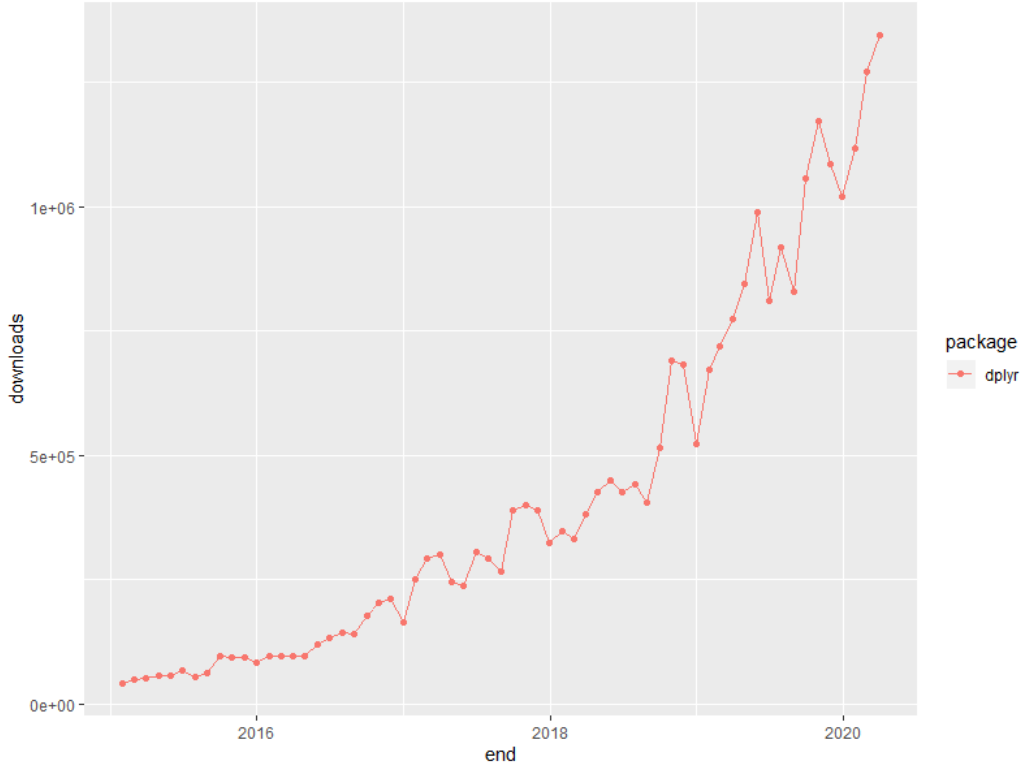
1.4 A case for community-driven, open-source CGE analysis

Even though there have been doubts about the sustainability and quality of community-driven software development, it is now clear that it is a viable alternative to private code development. Open-source tools such as R or Python are as mainstream as any of the commercially produced products, serving as a proof that open-source development can produce stable software solutions with high penetration among the users.

If we consider the main functions that the GEMPACK system performs—read data, read model, interpret the model into a system of equations/matrices, solve the model, generate outputs—most of these components, perhaps with the exception of an efficient solver for large matrices, have been largely addressed by the existing open-source solutions.

While there may not be an ultimate single measure of the success of

Figure 1: Downloads of package dplyr (the most popular R package)



the outcomes of open-source software development, some metrics are helpful. The first metric involves the number of solutions developed by the community for the use. CRAN currently lists over 10,000 official packages for download. Second, the numbers of monthly installations of those packages which give an indication of the activity of the software. This growth may be demonstrated by the number of downloads of R's most popular package (dplyr) in Figure 1 [3].

The increasing number of R packages and their active certainly confirm that this particular area of open-source analytical development is strong and growing, but does this mean that it could support a very specific framework of CGE analysis? To answer this question we proceed by looking critically at the required components that have been in use in each step of the GEMPACK-based CGE analysis.

2 Bringing GEMPACK functionality to R

In the previous section, we considered in general the components of the GEMPACK system that need to be provided by the open-source systems in order to allow for the entire GEMPACK framework functionality to be executed in an open-source framework. In this section we focus specifically on R as a representative of the most commonly used open-source framework used in data analysis. The components that we consider here are following: reading HAR files in R, interpreting a TABLO model in R, solving the model in R, producing the results.

2.1 HARr: Direct access to HAR files in R

GEMPACK uses data files that are in a custom-built format (HAR) based on a standard way for Fortran file access. HAR data formats are binary, which allows for their size to be smaller than a human-readable format would allow, but because of their custom definition they are not compatible with the existing data storage formats. Access to the HAR files is provided through GEMPACK built routines and no on-line access (e.g., through http requests) is currently available.

To allow R direct access to HAR files, we use an R package HARr [4], which reads a HAR file (currently only those headers which contain character, integer and dense real matrices) and outputs an object representing the data contained in the headers, including any information on sets.

2.2 TabloToR: Interpret and solve a TABLO model in R

The TABLO language and the associated editor were an extremely important innovation in GEMPACK as they provided a canonical language for CGE analysis that benefitted anyone wishing to conduct CGE analysis in GEMPACK. Because the language was well documented, it allowed the researchers to benefit from this shared language even without the need for internet connectivity.

While the TABLO language has been designed to support general equilibrium analysis, general-purpose languages, such as R, lack certain short-hand syntactical options (aka sugar coating) that are extremely useful for a particular problem. A few examples of such short-hand syntaxes which are

available in TABLO but unavailable or best avoided in R include the implicit creation of the change or percent change variables for updated coefficients, the implicit assumption that equations represent linearized equations, or the multiplication of percent change variables (but not linear variables) by 100 to support a particular way of outputting their results.

The package `TabloToR` [5] bridges the gap between the human-readable TABLO language with its implicit assumptions by allowing the TABLO model to be read in, interpreted and turned into a reference class object of class `GEModel` for further manipulation (e.g., loading of data, setting shocks, solving the model) in R.

The `GEModel` object provided by the package is able to read the TABLO file by mainly relying on the TABLO syntax which is very similar to that used in R. Some of the differences, e.g., the symbols used for equality or inequality, are fixed during the read. Based on the contents of the TABLO file, the object creates abstract generator functions that are able to generate the run-time coefficient levels, formulas and equation coefficients based on the actual data for the model. Based on the definition of exogenous variables and their levels, the object uses a sparse linear system [6] solver to solve the system iteratively and update the coefficients based on the solved values.

2.3 Solution outputs

Following the simulation, the `GEModel` object contains the resulting variable values and updated coefficients. Because all results exist as standard arrays, they can be effortlessly processed by R or exported to other formats.

3 Example: Running a GTAP model in R

To illustrate how the GTAP model may be executed in R, we perform a sample run of a fairly simple simulation using a moderately disaggregated GTAP database. Using various solution methods, we arrive at a set of results which we then compare with the results obtained from the `GEMPACK` framework for an identical simulation. Finally, we compare the run times between the two frameworks.

Table 1: Regional aggregation

Oceania	LatinAmer
EastAsia	EU_28
SEAsia	MENA
SouthAsia	SSA
NAmerica	RestofWorld

Table 2: Sectoral aggregation

Crops	i_s
Animals	nfm
Energy	fmp
Food	ele
LightMnf	eeq
p_c	ome
chm	mvh
bph	otn
rpp	omf
nmm	Svces

3.1 Model and simulation definition

The sample run that we perform involves the standard GTAP model version 6.2 [7]. Using the latest available GTAP database version 10 [8], we created an aggregation of ten regions, twenty commodities and five factors of production. The aggregation is shown in tables 1–3.

To demonstrate the executing of the GTAP model in R, We perform a simple simulation, in which we lower the supply of unskilled labor in three regions most hit by covid-19 (Latin America, North America and the EU) by 20 percent.

Table 3: Factor aggregation

Land
UnSkLab
SkLab
Capital
NatRes

3.2 Executing the model in R

In order to run the GTAP model in R, we require two packages: HARr to read the header-array files and tabloToR to interpret and execute a TABLO model.

```
1 require(HARr)
2 require(TabloToR)
```

We then may use tabloToR to set up a new instance of object GEModel:

```
1 # Initialize a new object of class GEModel
2 GTAP = tabloToR::GEModel$new()
```

Method loadTablo allows us to read the TABLO file into the object where it is processed:

```
1 # Load a TABLO file into the object (and interpret it)
2 GTAP$loadTablo('gtap.tab')
```

Reading in the entire data set from HAR files is conveniently done using HARr's function read_har:

```
1 # Read in the data from HAR files
2 data = list(
3     GTAPSETS = HARr::read_har('sets.har'),
4     GTAPPARAM = HARr::read_har('default.prm'),
5     GTAPDATA = HARr::read_har('basedata.har')
6 )
```

We then load the data into the GEModel object using method loadData:

```
1 # Load the data to the model
2 GTAP$loadData(data)
```

To provide the shocks to the system, we need to specify a vector of shock values with the names of the elements specifying the variable name to be shocked, e.g., c('pop["USA"]'=3). To specify the entire vector, we may filter out all variables that are mentioned as exogenous in the standard closure:

```
1 # Get all variables in the model
2 allVariables = GTAP$data$variables
3
4 # Specify the exogenous variables in the standard
  closure
```

```

5 exogenousVariables=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")
6
7 # Filter out all model variables that match the
    exogenous variable pattern
8 exogenousModelVariables = allVariables[ (Reduce(function
    (a,f)
9     c(a,grep(sprintf('~%s\\[',f), allVariables)),
        exogenousVariables,c()
10 ))]
11
12 # Only select some of the qo variables that are
    exogenous (for factors)
13 for(r in GTAP$data$REG)for(e in GTAP$data$ENDW_COMM)
    exogenousModelVariables=c(exogenousModelVariables,
    sprintf('qo["%s","%s"]',e,r))

```

Create a vector of shocks, initially empty (of zeros):

```

1 shocks= array(0, dim=length(exogenousModelVariables),
    dimnames=list(exogenousModelVariables))

```

We may now change some of the zero shocks to the desired levels:

```

1 # Specify shocks
2 shocks['qo["UnSkLab","LatinAmer"]']=-20
3 shocks['qo["UnSkLab","NAmerica"]']=-20
4 shocks['qo["UnSkLab","EU_28"]']=-20

```

Once the shock vector is specified, we can load it into the GEModel object:

```

1 # Specify shocks (a vector of values for all exogenous
    variables)
2 GTAP$setShocks(shocks)

```

In the final step, we execute method `solveModel` with an optional specification for the number of iterations. One iteration is equivalent to the Johansen method in GEMPACK:

```
1 # Solve the model in a single iteration (=Johansen)
2 GTAP$solveModel(iter = 1)
```

3.3 Viewing the results of the simulation

The results of the simulation are found in the property data of the model. For example, to obtain the calculated value of variable `EV`, we may execute the following:

```
1 > GTAP$data$EV
2      Oceania      EastAsia      SEAsia
3    -877.459    32765.119     117.912
4    SouthAsia    NAmerica    LatinAmer
5    6273.467  -577674.288  -151445.859
6      EU_28      MENA      SSA
7 -424023.541  -14477.480   -4209.668
8 RestofWorld
9    -9370.851
```

In addition to viewing the values of the variables, we may also view the updated values for the coefficients:

```
1 > GTAP$data$EVOA[, 'NAmerica']
2      Land      UnSkLab      SkLab
3 52387.03 1782174.78 7476622.18
4      Capital      NatRes
5 4937005.57 109949.38
```

3.4 Comparison with GEMPACK results

To see how the results from the GTAP simulation performed in R compare to that performed in GEMPACK, we run the simulation described above with two solution methods: first, as a single-iteration solution which is expected to be identical to the Johansen solution method offered by GEMPACK; second, we run the model as a ten-iteration solution (splitting the shock into

ten equally sized sub-shocks) and compare it to the Gragg method which is expected to provide the highest level of accuracy available in GEMPACK.

We look at the results for two variables: EV and pm for unskilled labor. The first one represents a change variable (in millions of US dollars) while second one represents a percent change variable.

3.4.1 One-iteration results versus Johansen

We first compare the results of a single-iteration run in R with a Johansen run in GEMPACK. We present the results for equivalent variations in Table 4 and domestic factor prices for unskilled labor in Table 5 obtained from solving the model in R and GEMPACK.

Inspecting both tables, we can confirm that the solutions are in fact equivalent, differing only on the sixth decimal place. This difference can be explained by higher default precision of R in recording and manipulating float numbers, and it is probably of no practical significance.

Table 4: Results for variable EV		
	GEMPACK	R
Oceania	-877.46	-877.46
EastAsia	32765.12	32765.12
SEAsia	117.91	117.91
SouthAsia	6273.46	6273.47
NAmerica	-577674.06	-577674.29
LatinAmer	-151445.88	-151445.86
EU_28	-424023.56	-424023.54
MENA	-14477.49	-14477.48
SSA	-4209.67	-4209.67
RestofWorld	-9370.85	-9370.85

3.4.2 Six-iteration results versus six-iteration Gragg

Having compared the single-iteration results in R with Johansen results in GEMPACK and finding the results essentially identical, we now proceed to comparing the results of a ten-iteration solution in R with 1-2-3 (six steps) step Gragg solution in GEMPACK. We again present the results obtained

Table 5: Results for variable pm["UnSkLab",]

	GEMPACK	R
Oceania	0.382161	0.382161
EastAsia	0.721118	0.721118
SEAsia	0.323598	0.323599
SouthAsia	0.885713	0.885714
NAmerica	12.528844	12.528842
LatinAmer	13.717290	13.717290
EU_28	13.516024	13.516023
MENA	-0.019788	-0.019788
SSA	-0.027744	-0.027743
RestofWorld	0.006760	0.006761

from both R and GEMPACK for equivalent variation and domestic factor price for unskilled labor in tables 6 and 7.

Clearly, the multistep solutions obtained from the two systems differ, even though not substantially. Most of the difference can be explained by the different methodology employed in Gragg that attempts to extrapolate the convergence and estimate the solution based on the increasing number of subintervals. The current solution system in R is a simple multi-step solution that traces the true solution using a fixed number of discrete steps.

Table 6: Results for variable EV solved using six iterations in R and 1-2-3 Gragg in GEMPACK

	GEMPACK	R
Oceania	-732.90	-961.02
EastAsia	35736.14	35885.19
SEAsia	192.99	129.14
SouthAsia	6769.05	6870.86
NAmerica	-620605.19	-632683.60
LatinAmer	-162656.00	-165867.36
EU_28	-456636.84	-464401.39
MENA	-14891.53	-15856.11
SSA	-4308.89	-4610.54
RestofWorld	-9581.44	-10263.19

Table 7: Results for variable pm[“UnSkLab”,] solved using six iterations in R and 1-2-3 Gragg in GEMPACK

	GEMPACK	R
Oceania	0.453670	0.419283
EastAsia	0.798568	0.792391
SEAsia	0.358550	0.354937
SouthAsia	0.979884	0.973986
NAmerica	15.022247	14.530789
LatinAmer	16.559319	15.995961
EU_28	16.316963	15.746737
MENA	0.012397	-0.021670
SSA	0.004999	-0.030381
RestofWorld	0.027524	0.007405

3.5 Comparison of performance

Running the same simulation in R and GEMPACK allows for some immediate comparison of performance, measured in the length of the run time. Using the same computer, we observe that six iterations took 804.46 seconds to run in R, but only 4.79 seconds in GEMPACK. While the difference in the run times clearly shows that solving a model in GEMPACK is significantly faster, there are a few important difference in what the the systems solve. First, the R system solves the full system during the run, while the GEMPACK system has removed the omitted and back-solved variables during its compilation.

4 Conclusions

As we have shown in this paper, there are open-source solutions already available in R that allow replication of GEMPACK’s functionality entirely in R. This means that for many applications, researchers no longer need to purchase and use GEMPACK in order to run GEMPACK-like models on GEMPACK-like data, such as the GTAP model using the GTAP database. The availability of packages HARr and TabloToR in R allows the researchers to use the data and the model exactly as they are typically distributed in the header-array and TABLO formats, respectively.

Allowing for GTAP modeling to be executed entirely outside of GEMPACK in a widely used and free open-source R framework is likely to make it accessible to a much wider audience of other researchers. This should be of great benefit to the GTAP community because it would allow GTAP-related analysis to be performed many more CGE analysts, who may not be able to run a model in GEMPACK if they never purchased and used it before.

In addition to saving money, running GTAP in R removes many of the integrability issues that GEMPACK imposes: the inputs to the model may be provided in any of the numerous formats that R is capable of handling, the model may be run entirely within R and its outputs come in the standard R formats as named lists and arrays. This makes it easy to use the GTAP model in any automated applications or to submit its results to any visualization tool available in R for easy presentation, including running an online version of the GTAP model available for real-time simulations, such as Shiny [9].

Longer run times, minutes instead of seconds, currently remain the main negative issue of running GTAP models in R. The observed history of open-source development gives us a lot of reason to be optimistic that this gap in performance would be soon bridged if the community of CGE modelers finds the move to R advantageous, embraces this approach and contributes to its development. If we assume the same vigor and enthusiasm of community would be applied to improving GEMPACK in R as it has been applied to many other applications, there is little doubt the framework will soon reach and exceed the performance of GEMPACK.

References

- [1] Centre of Policy Studies. <http://www.copsmodels.com/gempack.htm>.
- [2] Centre of Policy Studies. Brief History of Impact/CoPS: People, Dates and Places. <https://www.copsmodels.com/copshistory.htm>.
- [3] Guangchuang Yu. dlstats: Download Stats of R Packages. <https://cran.r-project.org/web/packages/dlstats/index.html>.
- [4] Maros Ivanic. HARR: an R package to read GEMPACK-style header-array files. available internally on the ERS network only at <http://bitbucket-ers:7990/scm/gpr/harr.git>.
- [5] Maros Ivanic. tabloToR: an R package to interpret and solve GEMPACK's TABLO models. available internally on the ERS network only at <http://bitbucket-ers:7990/scm/gpr/tablort.git>.
- [6] Douglas Bates, Martin Maechler, Timothy A. Davis, Jens Oehlschlägel, and Jason Riedy. Matrix: Sparse and dense matrix classes and methods. <https://cran.r-project.org/web/packages/Matrix/>.
- [7] Global Trade Policy Center. GTAP model version 6.2. https://www.gtap.agecon.purdue.edu/resources/res_display.asp?RecordID=2458.
- [8] Angel Aguiar, Maksym Chepeliev, Erwin Corong, Robert McDougall, and Dominique van der Mensbrugghe. The gtap data base: Version 10. *Journal of Global Economic Analysis*, 4(1):1–27, Jun 2019.
- [9] Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie, Jonathan McPherson, Mark Otto, Jacob Thornton, Alexander Farkas, Scott Jehl, Stefan Petre, Andrew Rowles, Dave Gandy, Brian Reavis, Kristopher Michael Kowal, Denis Ineshin, Sami Samhuri, John Fraser, John Gruber, and Ivan Sagalaev. shiny: Web Application Framework for R. <https://cran.r-project.org/web/packages/shiny/index.html>.