EviewsR: An R Package for Dynamic and Reproducible Research Using EViews, R, R Markdown and Quarto

by Sagiru Mati, Irfan Civcir, and S. I. Abba

Abstract EViews is a software designed for conducting econometric data analysis. There exists a one-way communication between EViews and R, as the former can run the code of the latter, but the reverse is not the case. We describe **EviewsR**, an R package which allows users of R, R Markdown and Quarto to execute EViews code. In essence, **EviewsR** does not only provide functions for base R, but also adds EViews to the existing **knitr**'s knit-engines. We also show how EViews equation, graph, series, and table objects can be imported and customised dynamically and reproducibly in R, R Markdown and Quarto document. Therefore, **EviewsR** seeks to improve the accuracy, transparency and reproducibility of research conducted with EViews and R.

Introduction

EViews, which stands for Econometric Views, is software designed for econometric analysis. EViews provides powerful statistical, time series, forecasting, and modelling tools through an innovative, easy-to-use object-oriented interface (Startz 2019). The software can be operated via its built-in menu, code or scripts saved in a file with the `.prg` extension. Although EViews is programmable, it lacks some routines to estimate new econometric techniques. For this reason, the users of EViews have to resort to using R for routines that are available in R but not yet implemented in EViews. This process involves having to switch between R and EViews, which hinders reproducibility and can lead to errors. EViews can open and run R code, but the converse is not true. An R package **hexView** (Murrell 2019) has been created to facilitate importing EViews workfile data to R. The package is limited as it cannot run EViews code from within the R environment. Another R package **gets** (Pretis, Reade, and Sucarrat 2018) allows exporting R object to EViews program via the package's function `eviews(object, file=NULL, print=TRUE, return=FALSE)`. Still, **gets** package does not allow embedding EViews code in R Markdown and Quarto. Xie (2019) had created the R package **knitr**, which allows communication between R and other statistical applications such as Stata and Octave, or programming languages such as Julia and Python. The statistical application and programming languages supported by the **knitr** package are called knit-engines, and EViews is not one of them. Users can view the existing **knitr** languages by running the following code in R console: `names(knitr::knit_engines$get())`. Another major challenge is that EViews does not support markdown syntax. To address these problems, we created an R package **EviewsR** (Mati 2019b) which integrates EViews and R. Basically, **EviewsR** provides base R functions and additional **knitr**'s knit-engine for EViews.

The **EviewsR** package is designed to be useful for all users of EViews, R, R Markdown and Quarto. The package allows EViews users to comfortably work in R, R Markdown or Quarto and have access to the ecosystems of all the applications. For example, they can make their document dynamic. On the other hand, the users of R, R Markdown and Quarto can easily benefit from the unique EViews's statistical or econometric routines and appealing graphics. For example, they can easily plot a line graph from an undated R dataframe (`dataFrame`) by `EviewsR::eviews_graph(dataFrame,start_date=1990)`, which may need several lines of code in base R. EViews has some advantages over R: 1. code stability. EViews has standard syntax and documentation and offers backward compatibility of code so that code does not break; 2. it also generates corresponding code for each menu action; 3. better support for econometric models; 4. easier to customise graphs. EViews graphs can be edited using the graphical user interface (GUI) and get the corresponding code generated by EViews; 5. stronger support for timeseries and panel data. Thus, teachers of Econometrics can spend more time on teaching Econometrics than on teaching software. EviewsR package helps EViews users to use R Markdown or Quarto without having to learn R functions for estimation and graphing.

Research is reproducible if its scientific computations can be replicated by an independent researcher (Stodden, Leisch, and Peng 2014). On the other hand, interactive reports entail the ability of a computational output to reactively change with the changes in input(s). For example, the output of an input `2+2` is 4. This output is expected to change automatically to 8 when the input changes to `3+5`. Modification of figures, tables, bibliography, captions and other objects becomes very easy in dynamic documents. Xie (2014) provides a detailed explanation and implementation of dynamic documents with R.

Sandve (2013) discuss ten simple rules that will ensure reproducibility of computational research.

Some of these rules include avoiding manual data manipulation steps, use of version control and providing public access to scripts and results. Christensen and Miguel (2018) examines the transparency, reproducibility, and credibility of Economics research, revealing evidence of result non-replicability within the field. The interest in reproducibility of research has traversed various fields of STEM and social sciences (see for example Franco, Malhotra, and Simonovits 2014; Simmons, Nelson, and Simonsohn 2011; Gerber et al. 2014; Harvey, Liu, and Zhu 2015; Ioannidis 2005). The aforementioned studies emphasise on the need for guidelines and solid criteria to ensure reproducibility of research. Therefore this article can help ensure replicability and reproducibility of research in the fields that employ EViews and R for their computations.

We categorise the reproducibility of research into three: 1. sharing the data and providing an easy guide on how to implement the computations 2. sharing the data, text and software code in separate files 3. sharing the data, the text and code in a single file. This paper aims to implement the third aspect of reproducibility using EViews, R, R Markdown and Quarto.

We intend to contribute to the current theme of dynamic and reproducible research as follows. We have created an R package **EviewsR**, which does not only integrate EViews and R, but also adds `eviews` as a new knit-engine for the **knitr** package. We also show how to create and modify EViews `equation`, `graph`, `series` and `table` objects dynamically and reproducibly. EViews code can now be embedded in R Markdown and Quarto documents so that both R and EViews users can collaborate on a single document. The package also provides R functions that could be used to 1. graph EViews series objects 2. graph an R dataframe using EViews 3. import data from external sources such as csv, xlsx as a new EViews workfile or into an existing workfile 4. create an EViews workfile from an R dataframe 5. save an EViews workfile or page as a workfile or another file format 6. execute EViews code 7. export an R dataframe as a new EViews workfile or to an existing EViews workfile 8. import EViews table object as `kable` 9. import EViews series objects as a dataframe or xts object 10. import EViews equation data members, `graph`, `series` and `table` objects 11. simulate a random walk process using EViews. We finally show how to use existing EViews workfiles in a dynamic document in order to avoid repeating time-consuming computations.

The rest of the article is structured as follows. We provide an overview of EViews, R, R Markdown and Quarto in Section 2.2. The description of the **EviewsR** package is in Section 2.3. We briefly explain how to use the package along with R, R Markdown and Quarto in Sections 2.4 and 2.6. Section 2.5 is dedicated to the implementation of dynamic document, Section 2.7 to the package implementation, while Section 2.8 covers the summary and conclusion.

EViews, R, R Markdown and Quarto

EViews (Econometric Views) is a statistical tool that facilitates both time-series and panel data analyses[1]. It can be operated using GUI, command or a program containing a set of commands.

R, on the other hand, is a free and open-source statistical programming language developed and maintained by R Core Team (2019)[2]. Unlike the EViews, the R software is command-based, implying that every output is generated by executing a command or a set of commands. Thus, reproducing any outputs is as easy as running the code in the R console. Base R's functionality can be extended via custom-made functions and objects, that can be organized into R packages. The R packages are available for free to download at Comprehensive R Archive Network (CRAN)[3].

RStudio is an Integrated Development Environment (IDE) for the R. It simplifies the use of the R as some of the R code can be executed via the GUI drop-down menus in RStudio[4]. In addition to that, RStudio works as an efficient plain text editor; it is easy and straightforward to edit text files with extensions such as `bib`, `tex`, `Rmd`, `Rmarkdown`, `md`, `yaml` and several other extensions.

R Markdown provides an easy way to write a markdown document (Allaire et al. 2020). It is available in RStudio with two alternative extensions: Rmd and Rmarkdown. It facilitates the ability to combine Markdown syntax with the syntax of R and other programming languages supported by the **knitr** package. Users can easily create R Markdown documents in RStudio by clicking `File-> New File-> R Markdown`. R Markdown documents consist of three components: metadata, text and code (Xie 2015). Metadata, also known as YAML metadata or YAML frontmatter, is written in-between a pair of three dashes. It can contain the author name, output format, title and so on (see Xie 2014, 2015, 2019)

[1]Please visit `https://eviews.com` for details

[2]Please visit `https://www.r-project.org/` for details.

[3]The CRAN's homepage is `https://cran.r-project.org/`. The R software and its packages can be downloaded from the homepage.

[4]RStudio can be downloaded for free from `https://rstudio.com/`

```
---
title: 'An Example of YAML metadata'
author: "Author Name"
output: bookdown::pdf_document2
---
```

After the above YAML metadata, text and code follow until the end of the R Markdown document. Text syntax is just like plain Markdown syntax, but code has to be placed inside blocks delimited by backticks. Note that we use `bookdown::pdf_document2` as the output because it allows for cross-reference in an R Markdown document.

Quarto is the "next generation version of R Markdown" developed by RStudio team. It is an open-source scientific and technical publishing system built on Pandoc. Unlike R Markdown, Quarto can be used to create dynamic content with Python, R, Julia, and Observable and work with IDEs such as VS code, RStudio, Jupyter and Text Editor[5]. The extension of Quarto file is `.qmd`.

Code chunks

The ability to embed R code is the major difference between an R Markdown/Quarto document and a Markdown document. R code can easily be embedded in R Markdown and Quarto documents. The R **knitr** package extends this capability to allow users to embed the code of other programming languages such as Python and Go, or other statistical packages such as Stata and Octave. A minimal example of R code chunk looks like the following

````
```{r chunkLabel,eval=TRUE,echo=FALSE}
y=runif(100)
```
````

Or in YAML format:

````
```{r}
#| label: chunkLabel
#| eval: true
#| echo: false
y=runif(100)
```
````

R chunk starts with three back-ticks, followed by curly braces containing the knit-engine, chunk label, chunk options, R code and ends with three back-ticks. In the chunk above, `r` is the engine name (knit-engine), `chunkLabel` is an arbitrary but unique text that gives the chunk a name, `eval=TRUE` and `echo=FALSE` are some examples of chunk options, `y=runif(100)` is a typical R code. If `r` is replaced with `eviews` in the chunk above, we will refer to the chunk as an EViews code chunk, not an R code chunk. We will continue to use the YAML format since both R Markdown and Quarto accept it.

We use prompts like `"EViews >"` and `"R >"` to represent code written in EViews and R respectively. We use `"-"` and `"+"` to signify continuation of EViews and R code respectively. Therefore `"+ -"` stands for continuation of EViews code written within R function. Code chunk indicates code written in an R Markdown or Quarto document.

About EviewsR

EviewsR is an R package to integrate EViews and base R, and also built on top of **knitr** package (Xie 2014, 2015, 2019) to add new knit-engine. It allows users of base R to communicate with EViews via R functions and users of R Markdown to embed EViews code chunks in an R Markdown document.

How to configure EViews before using EviewsR

To run the package successfully, users need to do one of the following:

1. Do not change anything if the name of the EViews executable is one of the following: `EViews12_x64`, `EViews12_x86`, `EViews11_x64`, `EViews11_x86`, `EViews10_x64`, `EViews10_x86`, `EViews9_x64`, `EViews9_x86`. The package will find the executable automatically.

[5] Please visit https://quarto.org/ for details

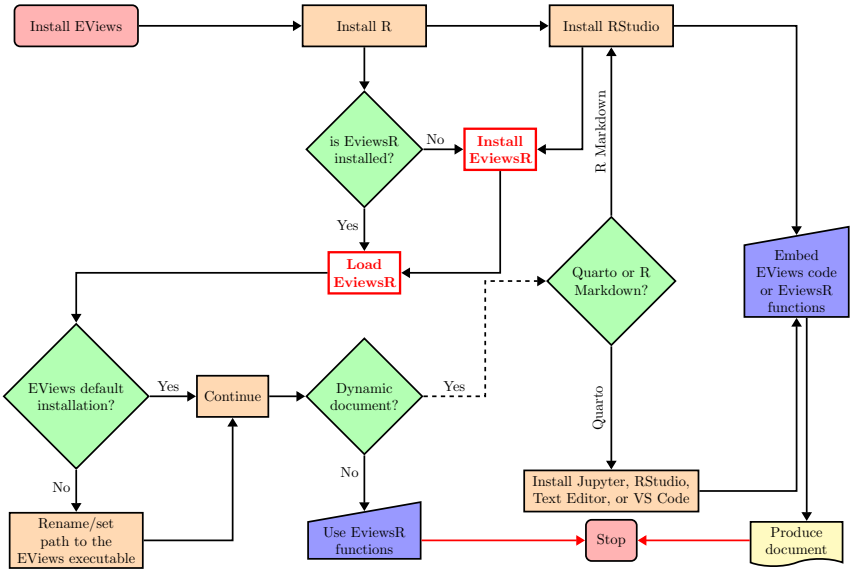

Figure 1: Setting up EviewsR, EViews, R, R Markdown and Quarto

2. Rename the EViews executable to `eviews` or one of the names above.
3. Alternatively, you can use `set_eviews_path()` function to set the path to the EViews executable as follows:

```
R> set_eviews_path("C:/Program Files (x86)/EViews 10/EViews10.exe")
```

Or

```
R> set_eviews_path("C:\\Program Files (x86)\\EViews 10\\EViews10.exe")
```

The backslash symbol (\) is used as a special escape symbol in R strings, so in order to write a literal backslash we need to precede it with another backslash.

How to use EviewsR

The package **EviewsR** can be used along with base R, R Markdown or Quarto document. Users should start by loading the **EviewsR** package via R console or by creating an R chunk in an R Markdown or Quarto document as shown below:

````
```{r EviewsRPackage,echo=FALSE}
if(!require("EviewsR")) install.packages("EviewsR")
library(EviewsR)
```
````

Figure 1 presents a chart showing the steps and requirements to use **EviewsR** package.

EviewsR: R Markdown and Quarto document

As mentioned earlier, the **EviewsR** package adds `eviews` as a knit-engine to **knitr** package. Therefore, it allows users to embed EViews code in an R Markdown or Quarto document. After loading the package, then create an EViews chunk as shown below:

````
```{eviews}
#| label: fig-EviewsR
#| graph: ""
#| eval: true
#| echo: false
````
````

```
  'This is some comment in EViews program, feel free to write anything

  wfcreate(wf=EviewsR,page=EviewsR) m 2000 2022

  for %y EviewsR1 EviewsR2
  pagecreate(page={%y}) m 2000 2022
  next

  for %y EviewsR EviewsR1 EviewsR2

  pageselect {%y}

  genr y=@cumsum(nrnd)
  genr x=@cumsum(nrnd)

  graph x_graph.line(o=eviews5) x
  graph y_graph.dot(o=bokeh) y

  table EviewsRTable

  for !j=1 to 7
  EviewsRTable(1,{!j})="Header"+" "+@str({!j})
  next

  for !i=1 to 10
  for !j=1 to 7
  EviewsRTable({!i}+1,{!j})=@str({!i})+","+@str({!j})
  next
  next

  next

  wfsave EviewsR_files/EviewsR
```

The above EViews chunk creates an EViews program with the chunk's content, then automatically opens EViews and runs the program, which will create an EViews workfile with pages (`EviewsR`, `EviewsR1`, and `EviewsR2`) each containing random walk series x and y from January, 2000 to December, 2022. The program will also save an EViews workfile named `EviewsR.wf1` in the current directory. We believe that elaboration on the chunk header and options is in order. The word `eviews` tells the chunk to execute using EViews code, `fig-EviewsR` is the chunk's label, `"eval: true"` asks the chunk to evaluate the content (code) of the chunk, `"echo: true"` allows the content of the chunk to appear in the final document output, `comment: NULL` eliminates the comment prefix in the chunk output. We set `'graph: """'` in order not show any EViews graph objects. Please refer to **knitr**'s documentation for details. Table 1 lists the specific chunk options for **EviewsR** package:

EviewsR: Dynamic Document

This section is about working with EViews's equation, graph, series and table objects dynamically in an R Markdown or Quarto document.

Accessing EViews objects from EViews chunk

The EViews chunk below (`label: fig-EviewsR1`) contains EViews code which generates a workfile named `EviewsR_workfile` along with a page `EviewsR_page` with monthly frequency from 2000 2022. The chunk also creates additional three EViews pages (`page1`, `page2`, and `EviewsR`) and generates three series objects x, y and z. It then runs the ordinary least square (OLS) method with y as the dependent variable and the rest as independent variables on each of the new pages. On each of the new pages, an equation, a table and three graph objects are created. The equation object is defined as `OLS`, the table object as `OLS_TABLE` and three graph objects as `graph1`, `graph2` and `graph3`.

Table 1: The chunk options available for EviewsR package in R Markdown and Quarto documents

Chunk option	Default value	Possible values	Example	Meaning
equation	*	Any valid name pattern for EViews @wlookup function	eq*, ??e, *e*	EViews equation object
graph	*	asis, first, last, asc, desc, or any valid name pattern for EViews @wlookup function	gr*, g??, *t*	EViews graph object
series	*	Any valid name pattern for EViews '@wlookup' function	se*, ?e?, *y*	EViews series object
table	*	Any valid name pattern for EViews @wlookup function	ta*, ?t, *s*	EViews table object
page	*	Any valid name pattern for EViews @wlookup function	page1, page2, page3, page4, page5	EViews page
graph_procs		Any valid EViews graph's procedure	align(2,1,1)	EViews graph's procedure
save_options		Any valid options for EViews save command	t=pdf, -c	EViews options for graph save command
class	df	xts		Class of R's object for imported EViews series objects
save_path		Any valid path	C:/Users/EviewsR	Path to save EViews graphs

The chunk automatically creates a new environment with the chunk's label as the environment's name, then returns all the EViews's equation, series and table objects into R as a dataframe saved within the new environment. Note that the chunk label (`fig-EviewsR1`) is not a valid R object, so **EviewsR** removes the prefix `fig-` before creating the new environment as `EviewsR1`. The prefix (`fig-`) is added to the chunk label because it is the Quarto's default way to wrap plots contained in a chunk in a figure environment for cross-reference. If you are working with R Markdown or the chunk does not contain EViews graph objects, you do not need to add the prefix.

The contents of the EViews `table` objects are imported into R exactly as they are on the EViews workfile, but the table name is changed to small letters. You can use the following format to get the imported table:

- `chunkLabel$pageName_tableName`

The `chunkLabel` stands for the chunk label and is the new environment, `pageName` for the EViews page name and `tableName` for the EViews table object. Therefore, `EviewsR1$eviewsr_ols_table` accesses the table `ols_table`, which is imported from `EviewsR` page of chunk `EviewsR1` into R as a dataframe.

````
```{eviews}
#| label: fig-EviewsR1
#| fig.cap: EViews graphs automatically imported by EViews chunk (fig-EviewsR1: default chunk options)
#| out.width: 32%
#| out.height: 15%
#| fig.ncol: 3
#| echo: false
#| eval: true

'This is some comment in EViews program, feel free to write anything

wfcreate(page=EviewsR_page,wf=EviewsR_workfile) m 2000 2022

!n=123

for %y page2 EviewsR  page1
pagecreate(page={%y}) q 2000 2022
pageselect {%y}
rndseed !n
!n=!n+100
````
````

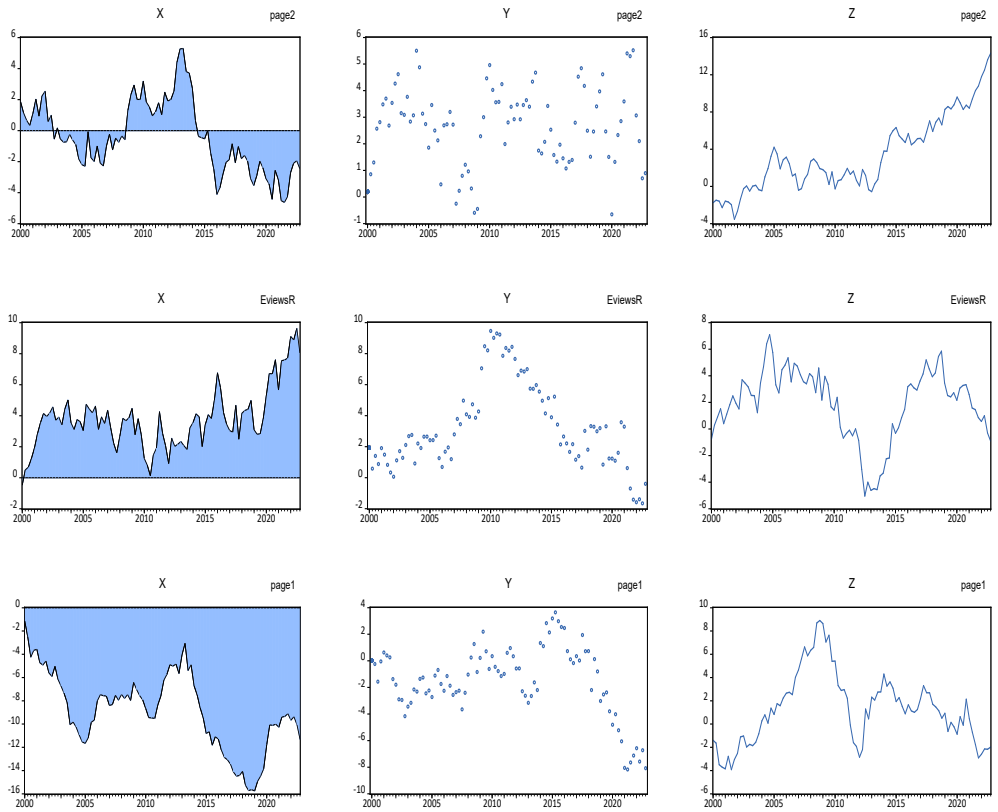

Figure 2: EViews graphs automatically imported by EViews chunk (fig-EviewsR1: default chunk options)

```
genr x=@cumsum(nrnd)
genr y=@cumsum(nrnd)
genr z=@cumsum(nrnd)
equation OLS.ls y c x z
freeze(OLS_TABLE,mode=overwrite) OLS
delete(noerr) GRAPH*
freeze(GRAPH3,mode=overwrite) z.line
graph GRAPH2.dot y
graph GRAPH1.area x
graph3.addtext(ar) %y
graph2.addtext(ar) %y
graph1.addtext(ar) %y
next
```

On the other hand, some data members of the EViews `equation` objects are extracted and imported into the new environment as a dataframe. Data member' is an EViews term for statistics, summaries, and information criteria related to EViews `equation` objects. The code below shows the general way to access the imported equation object, where `equationName` is the EViews equation object. The page and equation names are separated by underscore (_). The data members of the equation objects can be accessed via standard list element access syntax:

- `chunkLabel$pageName_equationName$dataMember`

For example, the R^2 value of the OLS equation object on `EviewsR` page is 0.402377, which can be accessed using the inline expression `` `r EviewsR1$eviewsr_ols$r2` ``.

Note that `@coefs`, `@pval`, `@stderrs` and `@tstats` return vectors with length equal to the number of estimated coefficients. The second value of each of these vectors can be accessed with typical square bracket notation, i.e. by appending `[2]`. The rest of the data members return scalar values. The number of data members reported by the EviewsR package is greater than that provided by `lm()`, `glm()` and `summary()` R functions. For example, R users need to use more functions or lines of code to get the

values of Schwarz and Hannan-Quinn Information Criteria of an estimated model. These values are automatically imported into R, R Markdown or Quarto environment by the EviewsR package.

The graph objects are saved on disk to the path defined by the chunk option `fig.path` and can be captioned and sub-captioned via the chunk options `fig.cap` (`fig-cap`) and `fig.subcap` (`fig-subcap`) in an R Markdown (Quarto) document. The graph file on the disk are named in the following format:

- `chunkLabel-pageName-graphName`

For example, `eviewsr1-page2-graph1` is the name of `graph1` object of `page2` created in `fig-EviewsR1` chunk.

The series objects of each page can be fetched through

- `chunkLabel$pageName`.

For example, `EviewsR1$page2` provides the dataframe of all the series objects on `page2`. Even if the page is not defined in the created or imported workfile, EViews names both the workfile and page as `untitled` by default. So `EviewsR1$untitled` is the right way to access the series object, if no page name is defined in the `fig-EviewsR1` chunk.

By default, **EviewsR** imports the series objects as a dataframe with date column formatted as `POSIXct` . We can import the series objects as `xts` object by setting chunk option `class="xts"`.

Note that the chunk label is case-sensitive, while the EViews equation, series and table objects are not. They can be in capital letters in the chunk, but they are in lowercase in the R object. This is because R variable names are case-sensitive, while EViews objects are not. For consistency, **EviewsR** is designed to convert EViews object names to lowercase. Therefore, we recommend naming EViews objects with lowercase names.

The graph objects in Figure 2 are imported with the default chunk options. We can use chunk options `page`, `graph` and `graph_procs` to change the behaviour of the graphical outputs. The chunk option `page` can take `page="*"`, or a space-delimited (or a vector of) string wtih the names of EViews pages. Note that we ask EViews to write the page name on the top right corner of each graph. The chunk option `graph` can have the values such as `*`, `first`, `last`, `asis`, `asc`, `desc` or a numeric vector.

Importing EViews graph asis

We set these chunk options to get Figure 3.

- `graph: "@asis"`
- `graph_procs: template newspaper`

The option `graph: "@asis"` makes sure that the graph objects are imported in the order they appear in the EViews chunk, that is `graph3` followed by `graph2` and `graph1`. This option is only useful in an R Markdown or Quarto document:

Use of graph_procs chunk option

To plot Figure 4, include the following arguments in the chunk options:

- `graph_procs: [template bokeh, setelem(1) lcolor(green) fillcolor(green)]`
- `graph: "@asis"`
- `page: eviewsr page1`

Note that only the graph objects from pages `EviewsR` and `page1` are included.

Importing only the first graph

To plot only the first graph object from each of the EViews pages, after all the graph objects on each page are arranged in ascending order. Therefore only a combination of `graph1` object from each of the pages is plotted as shown in Figure 5:

- `graph_procs: [template modern,datelabel format("YYYY"),setelem(1) fillcolor(red)]`
- `graph: "@first"`

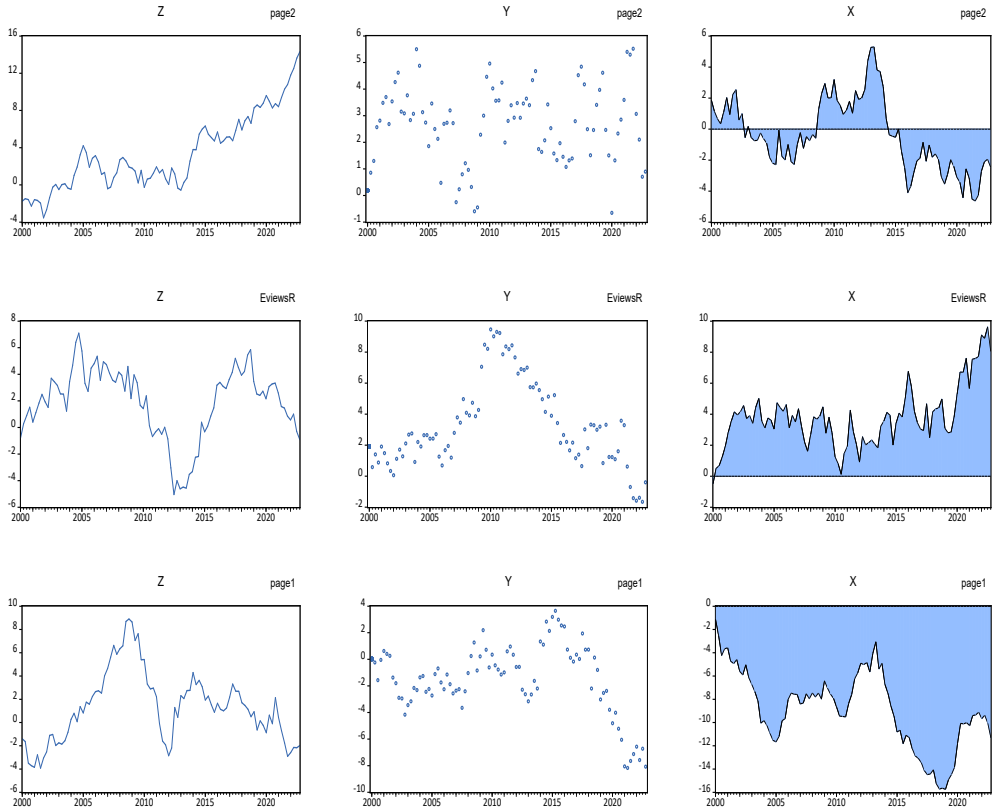

Figure 3: EViews graphs automatically imported by EViews chunk (fig-EviewsR2)

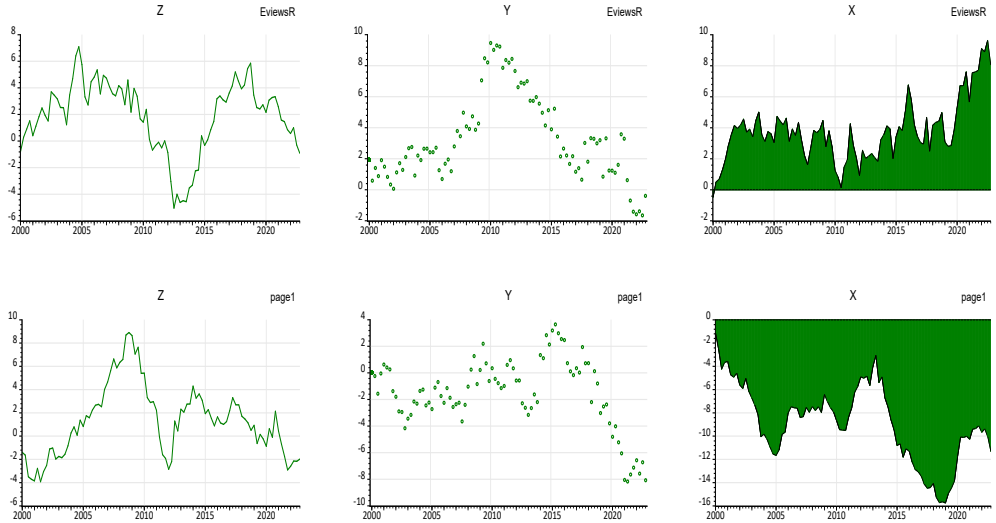

Figure 4: EViews graphs automatically imported by EViews chunk (fig-EviewsR3)

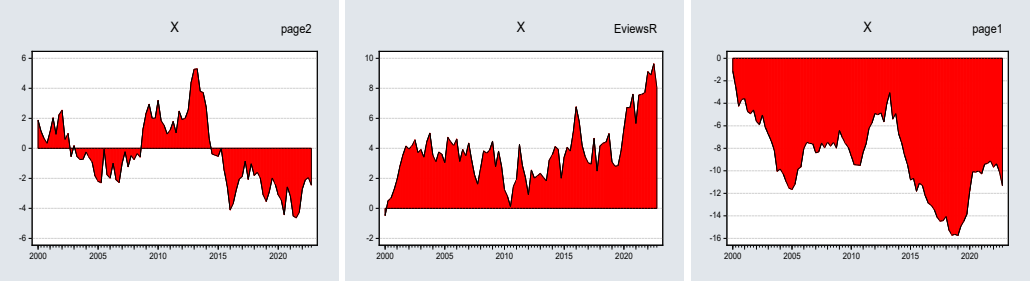

Figure 5: Importing only the first EViews graphs by EViews chunk (fig-EviewsR4)

Importing only the last graph

The chunk option `graph: "@last"` is the direct opposite of the `graph: " @first"` as it considers the last graph object only.

- `graph: "@last"`

Importing only graphs selected by index

We modify the chunk options as follows to ensure that only the second, fifth and eighth graph objects across all pages, are included in the output. Therefore only `graph2` object from each of the pages is considered.

- `graph: [2,5,8]`

Importing graphs in ascending/descending order

We arranges all the graph objects across all the pages in ascending order by setting:

- `graph: "@asc"`

We can also import the graphs in descending order by:

- `graph: "@desc"`

EviewsR: base R functions

We understand that not everyone uses R Markdown or Quarto. Moreover, some EViews computations may take long time to complete, in which case it will be better to work with the existing workfile. For these reasons, we create a number of R functions that can be used to communicate with EViews from R. The functions include:

To work with EViews, a workfile and a workfile page are required, therefore we add `wf` (workfile) and `page` (workfile page) arguments for each function. These functions are explained as follows:

The create_object() function.

The function `create_object()` can be used to create an EViews object in the existing EViews workfile. We can use the EViews workfile (`EviewsR.wf1`) created by the EViews chunk in Section 2.4. The complete EViews syntax for an object command has the form:

```
EViews> action(action_opt) object_name.view_or_proc(options_list) arg_list

EViews> object_type(options) object_name[=expression]
```

Where `action` can be one of the four EViews commands (`do`, `freeze`, `print`, `show`); `action_opt` modifies the default behaviour of the `action`; `object_name` is any arbitrary character string to represent the name of the EViews object to be acted upon; `view_or_proc` stands for the EViews object's view or procedure to be performed; `options_list` is the option for the `view_or_proc`; `arg_list` stands for the EViews view or procedure arguments.

To create an equation object eviews_equation as an ar(1) process on EviewR1 page of EviewsR.wf1 workfile, we can use the following EViews code:

```
EViews> wfopen EviewsR_files/EviewsR
EViews> pageselect EviewsR1
EViews> equation eviews_equation.ls y ar(1)
EViews> wfsave EviewsR_files/EviewsR
```

The above can be translated into base R function as:

```
R> create_object(wf = "EviewsR_files/EviewsR", page = "EviewsR1",
+     action = "equation", object_name = "eviews_equation", view_or_proc = "ls",
+     arg_list = "y ar(1)")
```

To create a series object series1 and assign it to the square of y series on EviewsR2 page of EviewsR.wf1 workfile:

```
EViews> wfopen EviewsR_files/EviewsR
EViews> pageselect EviewsR2
EViews> series series1=y^2
EViews> wfsave EviewsR_files/EviewsR
```

```
R> create_object(wf = "EviewsR_files/EviewsR", page = "EviewsR2",
+     object_type = "series", object_name = "series1", expression = "y^2")
```

The eviews_graph() function

EViews graph can be included in an R Markdown or Quarto document by eviews_graph() function. This function is a blend of EViews freeze, graph and save commands. The following are the EViews syntaxes:

```
EViews> freeze(options, name) object_name.view_command
EViews> graph graph_name.graph_command(options) arg1 [arg2 arg3 ...]
EViews> graph_name.save(options) [path\]file_name
```

The first and second syntaxes can be merged to produce another syntax which we use to create the eviews_graph() function. So the new syntax is:

```
EViews> freeze(options, name) object_name.graph_command(options)
```

Each of the freeze, graph_command and save commands has an options keyword. We use mode for freeze command options, graph_options for graph_command command options, and save_options for save command options. Check EViews manual for all the available options for these commands.

The series argument can be the names of EViews series objects or an R dataframe. If series is a set of EViews series objects, wf and page need to be specified. If series is an R dataframe, wf and page are optional.

To create figures of series objects x and y saved on EviewsR page of the EviewsR.wf1 workfile:

```
EViews> wfopen EviewsR_files/eviewsr
EViews> pageselect eviewsr
EViews> freeze(eviewsGraph_x,mode=overwrite) x.line
EViews> freeze(eviewsGraph_y,mode=overwrite) y.line
EViews> for %y eviewsGraph_x eviewsGraph_y
EViews> {%y}.axis(l) font(Calibri,14,-b,-i,-u,-s)
EViews> {%y}.axis(r) font(Calibri,14,-b,-i,-u,-s)
EViews> {%y}.axis(b) font(Calibri,14,-b,-i,-u,-s)
EViews> {%y}.axis(t) font(Calibri,14,-b,-i,-u,-s)
EViews> {%y}.legend columns(5) inbox position(BOTCENTER) font(Calibri,12,-b,-i,-u,-s)
EViews> {%y}.options antialias(on)
EViews> {%y}.options size(6,3)
EViews> {%y}.options -background frameaxes(all) framewidth(0.5)
EViews> {%y}.setelem(1) linecolor(@rgb(57,106,177)) linewidth(1.5)
```

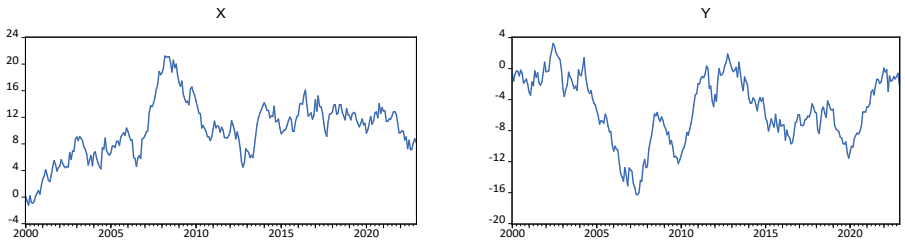

Figure 6: EviewsR example figure using LaTeX subfig package

```
EViews> {%y}.setfont legend(Calibri,12,-b,-i,-u,-s) text(Calibri,14,-b,-i,-u,-s)
EViews> {%y}.setfont obs(Calibri,14,-b,-i,-u,-s)
EViews> {%y}.textdefault font(Calibri,14,-b,-i,-u,-s)
EViews> {%y}.save(t=pdf) {%y}
EViews> next
```

Note that the EViews `for loop` in the code above contains the default values of EViews graph procedures provided by **EviewsR** package. It can be found in the `eviews_graph()` function's skeleton defined as `graphicsDefault`. We will avoid repeating the default values and use 'insert default values' instead. These default values can be modified or replaced entirely by using `graph_procs` argument. For example the line colour of the first graph element is blue by default. This can be changed to red by setting `graph_procs="setelem(1) linecolor(red)"`, which overwrites the default "`setelem(1) linecolor(@rgb(57,106,177))`". We can also overwrite the default behaviour by setting any available EViews templates, for example `graph_grocs="template defaults"`, which resets the graph objects to EViews global graphics defaults.

The EViews code above can be easily written as an R chunk to produce Figure 6 as follows:

````
```{r}
#| label: fig-eviewsGraph
#| fig.cap: EviewsR example figure using \LaTeX{} subfig package
#| fig.show: hold
#| out.width: 45%
#| out.height: 15%
#| eval: TRUE

eviews_graph(series="x y",wf="EviewsR_files/EviewsR",page = "EviewsR",save_options="t=pdf")

```
````

To produce EViews graphs aligned in two columns, as in Figure 7, from an R dataframe named `EviewsRDataFrame`:

- Use R's `write.csv()` function to write the dataframe as a CSV file.

```
R> write.csv(EviewsRDataFrame, "csvFile.csv", row.names = FALSE)
```

- Use EViews to import the `csvFile.csv`, then create and save the graphs on disk:

```
EViews> import csvFile.csv @freq m start_date=1990
EViews> group some_group x y
EViews> freeze(eviewsGraph1_xy,mode=overwrite) some_group.line(m)
                    insert default values
EViews> eviewsGraph1_xy.axis(b) angle(45) font(b)
EViews> eviewsGraph1_xy.save(t=png,d=300) eviewsGraph1_xy
```

- Use **knitr**'s `include_graphics()` function to import the graph into the R Markdown or Quarto document.

The values provided by `graph_procs` argument are appended to the graph's default values. Therefore, `graph_procs` overwrites the existing default value of the EViews graph procedures.

We use the following options to modify the figure:

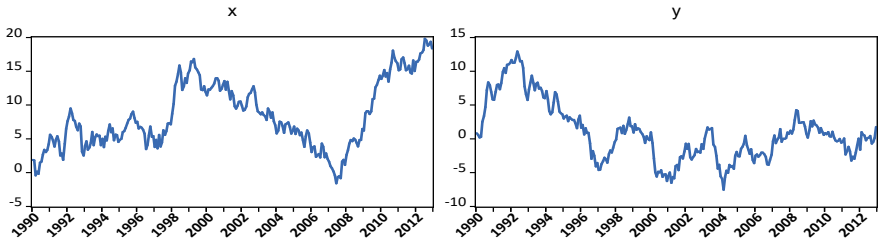

Figure 7: EViews graph from dataframe (chunk: fig-eviewsGraph1)

- `group=TRUE`
- `graph_options="m"`
- `graph_procs="axis(b) angle(45) font(b)"`

````
```{r}
#| label: fig-eviewsGraph1
#| fig.cap: "EViews graph from dataframe (chunk: fig-eviewsGraph1)"
#| out.width: 90%
#| out.height: 15%
#| eval: true

eviews_graph(series=EviewsRDataFrame,start_date = "1990",group=TRUE,
graph_options="m",graph_procs="axis(b) angle(45) font(b)")
```
````

To create a scatterplot along with histogram on each axis border using green colour (#008753 hex code):

```
EViews> wfopen EviewsR_files/eviewsr
EViews> pageselect eviewsr
EViews> group some_group x y
EViews> freeze(eviewsGraph2_xy,mode=overwrite) some_group.scat(ab=histogram) linefit()
                  insert default values
EViews> eviewsGraph2_xy.setelem(1) lcolor(@hex(008753))
EViews> eviewsGraph2_xy.save(t=png,d=300) eviewsGraph2_xy
```

The equivalent R chunk to produce Figure 8:

````
```{r}
#| label: fig-eviewsGraph2
#| fig.cap: EViews graph from dataframe
#| out.width: 90%
#| out.height: 40%
#| eval: TRUE

eviews_graph(series="x y",wf="EviewsR_files/EviewsR",page="EviewsR",
graph_command="scat(ab=histogram) linefit()",group=TRUE,
graph_procs='setelem(1) lcolor(@hex(008753))')
```
````

If we want to plot all the series objects contained in `EviewsR2` page of `EviewsR.wf1` workfile that lives in `EviewsR_files/` folder, we can simply use:

```
R> eviews_graph(wf = "EviewsR_files/EviewsR", page = "EviewsR2")
```

Note that we have not specified the `series` argument as `eviews_graph()` function is designed to plot all series objects by default. Similarly, we do not need to provide the page argument if we intend to include the graphs of all the series objects from all the pages of the workfile. We can also use any valid EViews wildcard expressions or pattern, such as `series="x*"`, `series="???x"` and so on.

To plot two or more line graphs with daily frequency on one frame from a dataframe:

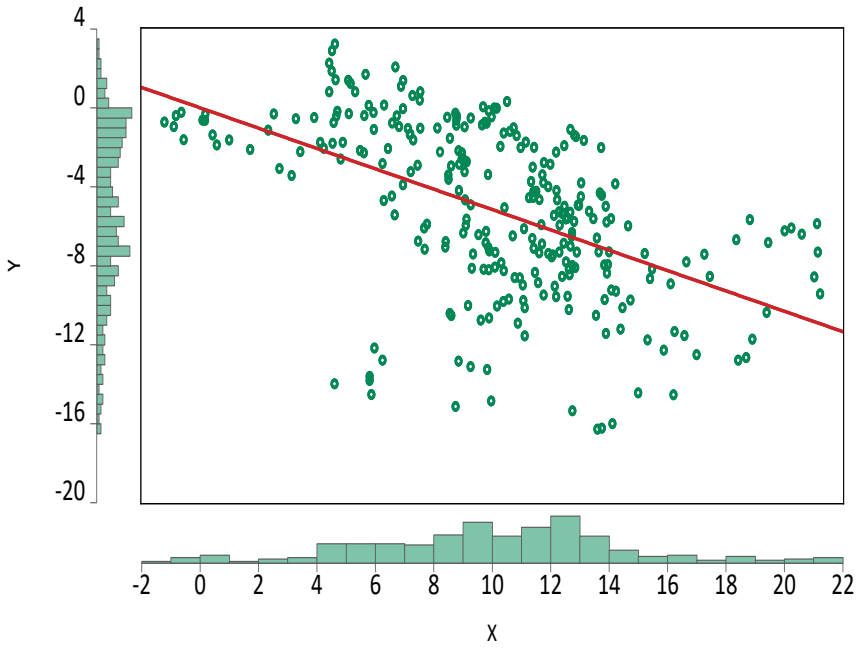

Figure 8: EViews graph from dataframe (chunk: fig-eviewsGraph2)

```
EViews> import csvFile.csv @freq 7 start_date=2000
EViews> group some_group x y
EViews> freeze(eviewsGraph3_xy,mode=overwrite) some_group.line
                    insert default values
EViews> eviewsGraph3_xy.datelabel format("dd Mon, yyyy")  interval(month,1)
EViews> eviewsGraph3_xy.save(t=png,d=300) eviewsGraph3_xy
```

Figure 9 uses the R equivalent of the above EViews code as follows:

````
```{r}
#| label: fig-eviewsGraph3
#| fig.cap: "EViews graph from dataframe (chunk: fig-eviewsGraph3)"
#| out.width: 90%
#| out.height: 40%
#| eval: TRUE

eviews_graph(series=EviewsRDataFrame,frequency="7",start_date = "2000",group = TRUE,
graph_procs='datelabel format("dd Mon, yyyy")  interval(month,1)')
```
````

To import EViews line graph of stacked values of the series objects as in Figure 10:

```
EViews> import csvFile.csv @freq a start_date=2010
EViews> group some_group x y
EViews> freeze(eviewsGraph4_xy,mode=overwrite) some_group.line(s)
                    insert default values
EViews> eviewsGraph4_xy.save(t=png,d=300) eviewsGraph4_xy
```

The R's syntax:

````
```{r}
#| label: fig-eviewsGraph4
#| fig.cap: "EViews graph from dataframe (chunk: fig-eviewsGraph4)"
#| out.width: 90%
````
````

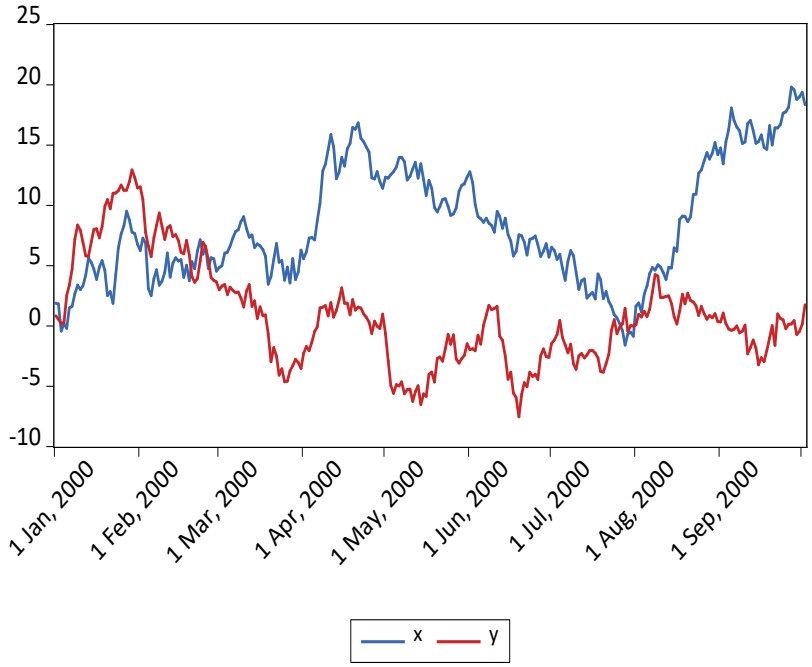

Figure 9: EViews graph from dataframe (chunk: fig-eviewsGraph3)

```
#| out.height: 40%
#| eval: TRUE
#| graph_procs: [template magazine, datelabel format("YYYY")]

eviews_graph(series=EviewsRDataFrame,frequency="m",start_date = "2010",
group = TRUE,graph_options="s")
```

Some journals require contributors to submit figures in black-and-white (see for example Mati, Civcir, and Ozdeser 2023; Mati, Civcir, and Ozdeser 2019). Setting `graph_procs='template monochrome'` or `save_options="t=png,-c"` gives the greyscale image as in Figure 11. The difference is that the former overwrites all the graph modifications to match the monochrome template, while the latter only saves the graphs without any modifications.

The eviews_import() function

Data can be imported from external sources by `eviews_import()` function. This function is a wrapper for EViews's `import` command. The EViews syntax is:

```
EViews> import([type=], options) source_description import_specification [@smpl smpl_string]
+   [@genr genr_string] [@rename rename_string]
```

To create a new workfile `eviews_import.wf1` from `eviews_import.csv`, which contains columns of x and y variables, using monthly series starting from 1990:

```
EViews> import EviewsR_files/eviews_import.csv @freq m 1990
EViews> wfsave EviewsR_files/eviews_import
```

The R's syntax:

```
R> eviews_import(source_description = "EviewsR_Files/eviews_import.csv",
+     frequency = "m", start_date = "1990")
```

To import `x` and `y`, with the former renamed to `x2`, from `eviews_import.csv` into existing workfile `eviews_import.wf1` within the sample of 1990m10 to 1992m11

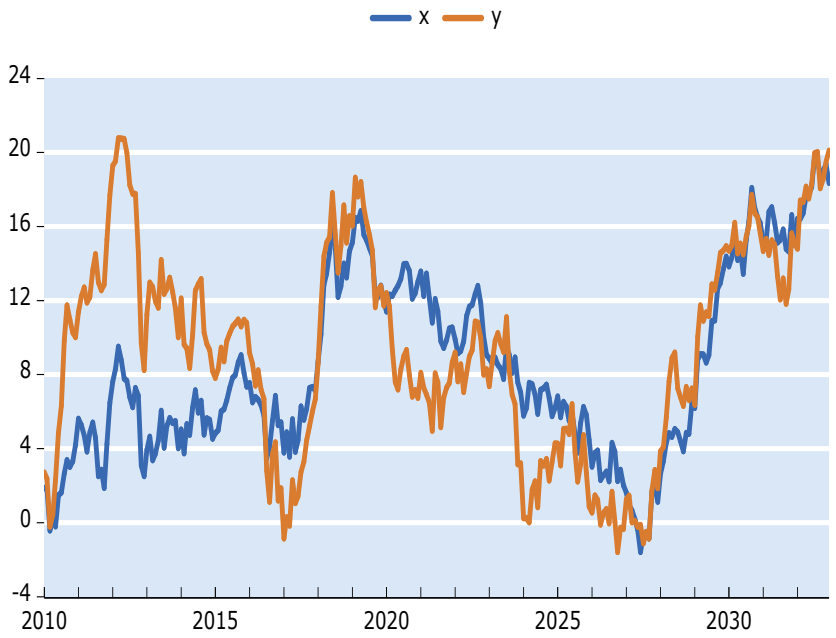

Figure 10: EViews graph from dataframe (chunk: fig-eviewsGraph4)

Figure 11: EViews greyscale figure (chunk: fig-eviewsGraph5)

```
EViews> wfopen EviewsR_files/eviews_import.wf1
EViews> import EviewsR_files/eviews_import.csv @smpl 1990m10 1992m11 @rename x x2
EViews> wfsave EviewsR_files/eviews_import

R> eviews_import(source_description = "EviewsR_files/eviews_import.csv",
+     wf = "EviewsR_files/eviews_import", smpl_string = "1990m10 1992m11",
+     rename_string = "x x2")
```

To import an R dataframe `EviewsRDataFrame` into `eviews_import.wf1` workfile and simultaneously generate another series object `z` as the sum of `x` and `y`:

The dataframe needs to be written as `eviews_import.csv` file using `write.csv()` function before executing the following code.

```
EViews> wfopen EviewsR_files/eviews_import.wf1
EViews> import EviewsR_files/eviews_import.csv @smpl 1990m10 1992m11 @genr z=x+y
EViews> wfsave EviewsR_files/eviews_import
```

The R's syntax below makes the process easier:

```
R> eviews_import(source_description = EviewsRDataFrame, wf = "EviewsR_files/eviews_import",
+     genr_string = "z =x+y")
```

To create a new workfile `eviews_import1.wf1` from an R dataframe `EviewsRDataFrame` with a quarterly series starting from September, 2000:

```
R> eviews_import(source_description = EviewsRDataFrame, wf = "EviewsR_Files/eviews_import1",
+     frequency = "m", start_date = "2000m9")
```

It is easier to plot a line graph of an undated dataframe with EviewsR than with the base R. For example, `eviews_graph(dataFrame,start_date=1990)` will graph a line plot labelled with dates starting from January 1990 on the x-axis. In base R, this requires creating a column for date in the dataframe or converting the dataframe to a timeseries object before plotting.

The eviews_pagesave() function

An EViews page can be saved in various formats by `eviews_pagesave()` function.

```
EViews> pagesave(options) source_description table_description [@keep keep_list] [@drop drop_list]
+ [@keepmap keepmap_list] [@dropmap dropmap_list] [@smpl smpl_spec]
```

To save `EviewsR` page from `EviewsR.wf1` workfile as a workfile named `eviews_pagesave.wf1`:

```
EViews> wfopen EviewsR_files/EviewsR
EViews> pageselect eviewsr
EViews> pagesave eviews_pagesave
```

The R's syntax:

```
R> eviews_pagesave(wf = "EviewsR_files/EviewsR", page = "EviewsR",
+     source_description = "EviewsR_files/eviews_pagesave")
```

To save the first ten observations of series `x` only from `EviewsR` page of `EviewsR.wf1` workfile in a `CSV` file named `eviews_pagesave.csv`:

```
EViews> wfopen EviewsR_files/EviewsR
EViews> pageselect eviewsr
EViews> pagesave eviews_pagesave.csv @keep x @smpl @first @first+9

R> eviews_pagesave(wf = "EviewsR_files/EviewsR", page = "EviewsR",
+     source_description = "EviewsR_files/eviews_pagesave.csv",
+     keep_list = "x", smpl_spec = "@first @first+9")
```

Similarly, the values of `keepmap_list` and `dropmap_list` can be set as `keepmap_list="y*"` and `dropmap_list="x?"` respectively.

The eviews_wfcreate() function

An EViews workfile can be created using `eviews_wfcreate()` function in R.

```
EViews> wfcreate(options) frequency[(subperiod_opts)] start_date end_date [num_cross_sections]

EViews> wfcreate(options) frequency[(subperiod_opts)] start_date +num_observations

EViews> wfcreate(options) u num_observations
```

To create an EViews workfile `eviews_wfcreate.wf1` along with a page named `EviewsR_page` in `EviewsR_files/` folder:

```
EViews> cd EviewsR_files
EViews> wfcreate(wf=eviews_wfcreate,page=EviewsR_page) m 2000 2022
EViews> wfsave eviews_wfcreate
```

The R's syntax:

```
R> eviews_wfcreate(wf = "eviews_wfcreate", page = "EviewsR_page",
+     frequency = "m", start_date = "2000", end_date = "2022",
+     save_path = "EviewsR_files")
```

The eviews_wfsave() function

An EViews workfile can be saved in various output formats using `eviews_wfsave()` function in R.

```
EViews> wfsave(options) [path\]filename

EViews> wfsave(options) source_description [@keep keep_list] [@drop drop_list]
+ [@keepmap keepmap_list] [@dropmap dropmap_list] [@smpl smpl_spec]

EViews> wfsave(options) source_description table_description [@keep keep_list]
+ [@drop drop_list] [@keepmap keepmap_list] [@dropmap dropmap_list] [@smpl smpl_spec]
```

To save all series objects in `EviewsR` page of `EviewsR.wf1` workfile except x in `eviews_wfsave.csv` file:

```
EViews> wfopen EviewsR_files/eviewsr
EViews> pageselect eviewsr
EViews> wfsave eviews_wfsave.csv @drop x

R> eviews_wfsave(wf = "EviewsR_files/EviewsR", page = "eviewsr",
+     source_description = "EviewsR_files/eviews_wfsave.csv", drop_list = "x")
```

The exec_commands() function

A set of EViews commands can be executed with the help of `exec_commands()` function in R.

The EViews chunk in Section 2.4 can be translated using this function as follows:

```
R> exec_commands(c("'This is some comment in EViews program, feel free to write anything",
+
+   'wfcreate(wf=EviewsR,page=EviewsR) m 2000 2022',
+
+   'for %y EviewsR1 EviewsR2',
+   'pagecreate(page={%y}) m 2000 2022',
+   'next',
+
+   'for %y EviewsR EviewsR1 EviewsR2',
+
+   'pageselect {%y}',
```

```
+
+   'genr y=@cumsum(nrnd)',
+   'genr x=@cumsum(nrnd)',
+
+   'graph x_graph.line(o=eviews5) x',
+   'graph y_graph.dot(o=bokeh) y',
+
+   'table EviewsRTable',
+
+   'for !j=1 to 7',
+   'EviewsRTable(1,{!j})="Header"+" "+@str({!j})',
+   'next',
+
+   'for !i=1 to 10',
+   'for !j=1 to 7',
+   'EviewsRTable({!i}+1,{!j})=@str({!i})+","+@str({!j})',
+   'next',
+   'next ',
+
+   'next',
+
+   'wfsave EviewsR_files/EviewsR'))
```

Alternatively, we can assign the EViews commands to an object before calling the function:

```
R> commands=r'('This is some comment in EViews program, feel free to write anything
+
+   wfcreate(wf=EviewsR,page=EviewsR) m 2000 2022
+
+   for %y EviewsR1 EviewsR2
+   pagecreate(page={%y}) m 2000 2022
+   next
+
+   for %y EviewsR EviewsR1 EviewsR2
+
+   pageselect {%y}
+
+   genr y=@cumsum(nrnd)
+   genr x=@cumsum(nrnd)
+
+   graph x_graph.line(o=eviews5) x
+   graph y_graph.dot(o=bokeh) y
+
+   table EviewsRTable
+
+   for !j=1 to 7
+   EviewsRTable(1,{!j})="Header"+" "+@str({!j})
+   next
+
+   for !i=1 to 10
+   for !j=1 to 7
+   EviewsRTable({!i}+1,{!j})=@str({!i})+","+@str({!j})
+   next
+   next
+
+   next
+
+   wfsave EviewsR_files/EviewsR)'
R> exec_commands(commands)
```

To create a workfile `exec_commands1.wf1` using a monthly frequency from November, 2000 to January 2022:

```
EViews> cd EviewsR_files
```

```
EViews> wfcreate(wf=exec_commands1,page=Page) m 2000m11 2022m1
EViews> %wf=@wfname
EViews> wfsave {%wf}
EViews> exit
```

The base R's syntax:

```
R> exec_commands(c("cd EviewsR_files", "wfcreate(wf=exec_commands1,page=Page) m 2000m11 2022m1"))
```

To execute EViews commands on an existing EViews workfile like the one created above:

```
EViews> wfopen EviewsR_files/exec_commands1
EViews> pageselect EviewsR
EViews> genr x=@cumsum(nrnd)
EViews> genr y=@cumsum(nrnd)
EViews> genr z=x+y
EViews> delete(noerr) grap
EViews> graph grap.line x y z
EViews> %wf=@wfname
EViews> wfsave {%wf}
EViews> exit
```

The R's syntax:

```
R> exec_commands(commands = c("genr x=@cumsum(nrnd)", "genr y=@cumsum(nrnd)",
+     "genr z=x+y", "delete(noerr) grap", "graph grap.line x y z"),
+     wf = "EviewsR_files/exec_commands1", page = "page")
```

The export_dataframe() function

Use `export_dataframe()` function to export dataframe object to EViews as a workfile.

To export the dataframe `EviewsRDataFrame` as a workfile `export_dataframe.wf1` with monthly frequency starting from January 1990:

```
R> export_dataframe(source_description = EviewsRDataFrame, wf = "EviewsR_files/export_dataframe",
+     start_date = "1990", frequency = "m")
```

To export the dataframe as a workfile `export_dataframe1.wf1` with undated frequency. However, if the dataframe contains a column with a regular dated frequency, EViews will automatically detect the date series and create a dated workfile.

```
R> export_dataframe(source_description = EviewsRDataFrame, wf = "EviewsR_files/export_dataframe1")
```

The import_equation() function

The data members of the EViews equation objects are imported in R as a dataframe.

The data members are accessible via:

- `eviews$pageName_equationName$dataMember` in base R or
- `chunkLabel$pageName_equationName$dataMember` in an R Markdown or Quarto document.

To get the value of Akaike Information Criterion (AIC) of an equation object named `OLS` on page `EviewsR` into base R:

```
R> eviews$eviewsr_ols$aic
```

To obtain the same value from the same equation object in an R Markdown or Quarto document, if the `import_equation()` function is called from a chunk label `importEquation`:

```
R> importEquation$eviewsr_ols$aic
```

Note that the `equation`, `graph`, `series` and `table` objects do not need the chunk label in base R. However, where an R environment is required, we use `eviews` instead of the chunk label.

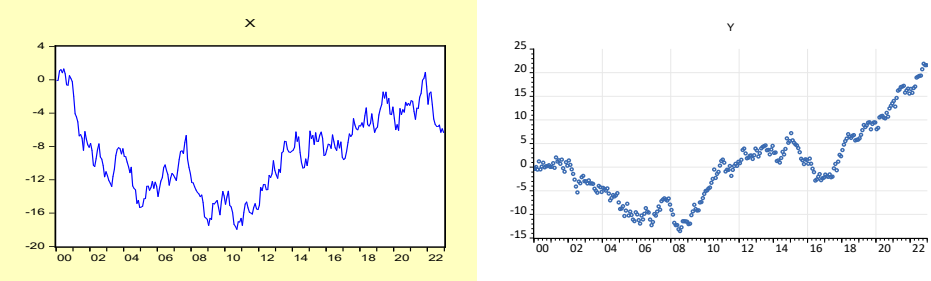

Figure 12: Existing EViews graph imported (chunk: fig-importGraph)

Table 2: EViews table imported as kable

Header 1	Header 2	Header 3	Header 4	Header 5	Header 6	Header 7
1,1	1,2	1,3	1,4	1,5	1,6	1,7
2,1	2,2	2,3	2,4	2,5	2,6	2,7
3,1	3,2	3,3	3,4	3,5	3,6	3,7
4,1	4,2	4,3	4,4	4,5	4,6	4,7
5,1	5,2	5,3	5,4	5,5	5,6	5,7
6,1	6,2	6,3	6,4	6,5	6,6	6,7
7,1	7,2	7,3	7,4	7,5	7,6	7,7
8,1	8,2	8,3	8,4	8,5	8,6	8,7
9,1	9,2	9,3	9,4	9,5	9,6	9,7
10,1	10,2	10,3	10,4	10,5	10,6	10,7

The import_graph() function

Importing existing graph objects from EViews workfile is easy with `import_graph()` function.

The R chunk below imports EViews graph objects `x_graph` and `y_graph` into an R Markdown or Quarto document as Figure 12. Use `graph="*"` and `page="*"` to import all EViews graph objects across all the workfile pages. The `graph` argument accepts any valid EViews pattern or wildcard expressions, such as `graph="x_*"`, `graph="?_graph"`}. Both the `graph` and `page` arguments can be used to choose specific graph object(s) and workfile page(s) respectively.

To import all graph objects from `EviewsR2` of `EviewsR.wf1` workfile that lives in `EviewsR_files/` folder:

```
```{r}
#| label: fig-importGraph
#| fig.cap: "Existing EViews graph imported (chunk: fig-importGraph)"
#| out.width: 45%
#| out.height: 15%

import_graph(wf="EviewsR_files/EviewsR",page="eviewsr2")
```
```

The import_kable() function

EViews table objects can be imported as `kable` object by `import_kable()` function.

To import the entire table object `EviewsRTable` from `EviewsR` page of `EviewsR.wf1` workfile as Table 2:

```
R> import_kable(wf = "EviewsR_files/EviewsR", page = "EviewsR",
+     table = "EviewsRTable", caption = "EViews table imported as kable",
+     format = ifelse(is_html_output(), "html", "latex"), linesep = "")
```

To import certain range of the table object `EviewsRTable` as Table 3:

```
R> import_kable(wf = "EviewsR_files/EviewsR", page = "EviewsR",
```

Table 3: Selected cells of EViews table imported as kable

Header 2	Header 3	Header 4	Header 5	Header 6
1,2	1,3	1,4	1,5	1,6
2,2	2,3	2,4	2,5	2,6
3,2	3,3	3,4	3,5	3,6
4,2	4,3	4,4	4,5	4,6
5,2	5,3	5,4	5,5	5,6

```
+     table = "EviewsRTable", range = "r1c2:r5c6", digits = 3,
+     caption = "Selected cells of EViews table imported as kable",
+     format = ifelse(is_html_output(), "html", "latex"))
```

The import_series() function

Use `import_series()` function to import EViews series objects into R as a dataframe or `xts` object. The function creates a new environment `eviews`, whose objects can be accessed via `eviews$pageName` in base R or `chunkLabel$pageName` in R Markdown or Quarto.

To import EViews series objects from `EviewsR` page of `EviewsR.wf1` workfile into R as a dataframe:

```
R> import_series(wf = "EviewsR_files/EviewsR", page = "EviewsR")
```

To access the imported series of `EviewsR` page in base R:

```
R> eviews$eviewsr
```

To access the imported series of `EviewsR` page in an R Markdown or Quarto document, if the chunk label is `importSeries`:

```
R> importSeries$eviewsr
```

The `series` argument can take any valid EViews wildcard expressions. The series objects are imported to EViews as a dataframe with date column as `POSIXct` by default. We can import the series as `xts` object by setting `class="xts"`.

```
R> import_series(wf = "EviewsR_files/EviewsR", page = "EviewsR",
+     class = "xts")
```

The import_table() function

Use `import_table()` function to import EViews table objects into R as a dataframe. The imported table objects can be accessed via `eviews$pageName_tableName`.

```
R> import_table(wf = "", page = "*", table = "*")
```

The `table` argument accepts valid EViews wildcard expressions.

To import all the table objects in `EviewsR.wf1`:

```
R> import_table(wf = "EviewsR_files/EviewsR")
```

To import `eviewsrtable` from the same workfile but from a page named `EviewsR` only:

```
R> import_table(wf = "EviewsR_files/EviewsR", page = "EviewsR",
+     table = "eviewsrtable")
```

To get the table imported from `EviewsR` in an R Markdown or Quarto document, if the chunk label is `importTable1`:

```
R> importTable1$eviewsr_eviewsrtable
```

The import_workfile() function

To import EViews equation, graph, series and table objects at once from an existing workfile, use the `import_workfile()` function. This function is a combination of `import_equation()`, `import_graph()`, `import_series()` and `import_table()` functions.

To import all the EViews equation, graph, series and table objects from the `EviewsR.wf1` workfile.

````
```{r}
#| label: fig-importWorkfile
#| fig.cap: Graphs imported by EViews chunk (fig-importWorkfile)
#| out.width: 45%
#| out.height: 15%

import_workfile("EviewsR_files/EviewsR")
```
````

We can also obtain the imported EViews equation, series and table objects in the same way as `import_equation()`, `import_series()` and `import_table()`.

The rwalk() function

A set of random walk series can be simulated in R using EViews engine, thanks to `rwalk()` function. The random walk series objects are returned as R dataframe, which can be accessed as `eviews$seriesNames` in base R and as `chunkLabel$seriesNames` in an R Markdown or Quarto document. Removing space from the value(s) of the `series` argument provides the `seriesNames`. Setting argument `class="xts"` returns `xts` object instead of dataframe. This function can be used to replicate examples provided in Econometrics textbooks that use EViews, as random numbers generated in R and EViews differ.

To generate random walk series X Y and Z using EViews and imports them into R as a dataframe `eviews$XYZ}`:

```
R> rwalk(series = "X Y Z", rndseed = 12345, start_date = "1990",
+     frequency = "M", num_observations = 276)
```

To generate random series rw1, rw2 and rw3 each with a drift of 10, on the existing workfile `EviewsR.wf1`:

```
R> rwalk(wf = "EviewsR_files/EviewsR", series = "rw1 rw2 rw3", rndseed = 12345,
+     drift = 10)
```

To obtain the head in base R:

```
R> head(eviews$rw1rw2rw3)
```

To get the head in an R Markdown or Quarto document, if the chunk label is `rwalk1`:

```
R> head(rwalk1$rw1rw2rw3)
```

To plot the random walk dataframe with both **ggplot2** package (Wickham 2016) and `eviews_graph()` function as shown in Figure 13:

````
```{r}
#| label: fig-rwalk
#| out.width: 45%
#| out.height: 15%
#| dim: [7,4]
#| fig.cap: "Random walk generated by EViews"
#| fig.subcap: ["Graph from ggplot package","Graph from EviewsR package"]

ggplot(rwalk$XYZ,aes(x=date)) +
    geom_line(aes(y=x,color="x"))+ geom_line(aes(y=y,color="y"))+
    geom_line(aes(y=z,color="z"))+
````
````

(a) Graph from ggplot2 package **(b)** Graph from EviewsR package

Figure 13: Random walk generated by EViews

```
  xlab('')+ylab('')+
  labs(colour = "")+
  scale_color_manual(values=c(x="blue",y="red",z="#008753"))

eviews_graph(series=rwalk$XYZ,group = TRUE,graph_procs=c('template reverse','legend position(right)'))

```
```

The Set_eviews_path() function

In case of non-standard EViews installation or presence of more than one EViews executable and we do not want to use the latest, we can use this function to set the path to the EViews executable:

To use EViews executable `EViews10`:

```
R> set_eviews_path("C:/Program Files (x86)/EViews 10/EViews10.exe")
```

or

```
R> set_eviews_path("EViews10")
```

Package implementation

The purpose of this section to explain how the package is implemented, so that contributions for further development of the package can be easy and straightforward. We show how the `eviews` knit-engine is added and how the base R functions are created.

Adding eviews as knit-engine

As mentioned earlier, **EviewsR** (Mati 2019b) package adds a knit-engine for Econometric Views (EViews), which is a proprietary econometric software package. The behaviour of EViews is listed below:

1. The file extension of `eviews` code file is `.prg`.
2. The command line execution of the `eviews` code file requires running `"eviewsExecutable exec EviewsFileName.prg"` or opening the `.prg` file.
3. The `eviews` code file can be opened by quoting the path to the EViews program file. If the EViews code file lives in the current working directory (CWD), writing its full file name (together with the `.prg` extension), with or without the quotes, will run the `eviews` code file.

The first step is to create a character string with the EViews's `.prg` file extension. For example, a character string `"MyEviewsFile.prg"` can be assigned to an object `fileName` using the code `fileName <-"MyEviewsFile.prg"`. The second step is to create another R object (`eviewsCode`) and use `writeLines()` function to write the EViews code to the file name object created in step 1. The code `writeLines(eviewsCode,fileName)` creates a file in the EViews's CWD and names it
```

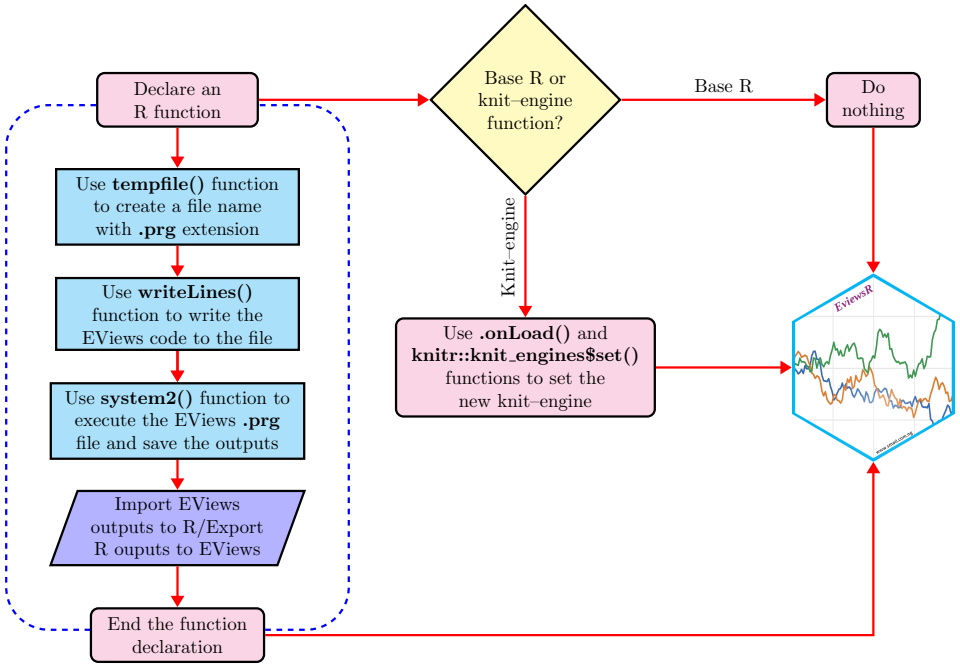

Figure 14: Architecture of EviewsR

`MyEviewsFile.prg`. The third step is to execute the file `MyEviewsFile.prg` using `system2()` function. The code `system2('EViewsExecutable','exec path/to/EViewsFileName.prg')` opens and executes the EViews code file `MyEviewsFile.prg`. The last step is to use `.onLoad()` function along with a function from **knitr** package `knit_engines$set(eviews=eng_eviews)` to set `eviews` as the knit-engine for EViews. Figure 14 presents four easy steps for adding `eviews` to the existing **knitr** engines and creating the R functions.

Creating base R functions

The base R functions are created in a way similar to adding the knit-engine, but they do not depend on **knitr** package. For the sake of demonstration, we use `eviews_wfsave()` function, which saves EViews workfile in various formats. The EViews syntax for `wfsave` command is as follows:

```
EViews> wfsave(options) [path\]filename

EViews> wfsave(options) source_description [@keep keep_list] [@drop drop_list]
+ [@keepmap keepmap_list] [@dropmap dropmap_list] [@smpl smpl_spec]

EViews> wfsave(options) source_description table_description [@keep keep_list]
+[@drop drop_list] [@keepmap keepmap_list] [@dropmap dropmap_list] [@smpl smpl_spec]
```

The last line of the code provides a more general syntax than the first two. Therefore, we use it to create the equivalent base R function. The syntax contains optional arguments such as `@keep keep_list`, `@drop drop_list`, `@drop drop_list`, `@dropmap dropmap_list`, `@smpl smpl_spec`, which are enclosed in square brackets. The arguments `options`, `source_description` and `table_description` are essential. In addition to this, we need to distinguish constant arguments from variable arguments. The constants include `wfsave`, `@keep`, `@drop`, `@keepmap` and `@dropmap`. On the other hand, the variables, which users can change, include `options`, `source_description`, `table_description`, `keep_list`, `drop_list`, `drop_list`, `dropmap_list`, `smpl_spec`. So, we include these variables as the function arguments for the base R function `eviews_wfsave()`. The suffix `eviews_` indicates that the function is based on the EViews command. For details on `wfsave` command, please visit `https://eviews.com/help/helpintro.html#page/content%2Fcommandcmd-pagesave.html`.

Since we need a workfile and a workfile page to work with `wfsave` command, the base R function arguments also include `wf` and `page` for workfile and workfile page respectively. The `wf` argument is necessary, but the `page` is not, due to the fact that active EViews page can be used to execute the function.

Summary and Conclusion

We have provided an overview of the **EviewsR** package, which allows users to run EViews code in R, R Markdown, and Quarto via R functions or knit-engine. The package provides a convenient and efficient way to integrate EViews and R, two of the most popular software packages for econometrics and statistics. The package offers the ability to run EViews code directly from within R, R Markdown, and Quarto, without having to switch between software packages. It also allows users to access and manipulate EViews objects from within R, making it easy to combine EViews analysis with other R-based workflows. In addition, it can be used to generate high-quality reports and presentations using R Markdown and Quarto, with EViews results seamlessly embedded. The package is a valuable tool for economists, statisticians, and other researchers who need to use both software packages.

The **EviewsR** package is still under development, and there are a number of potential future directions. One possibility is to incorporate graph templates similar to **ggplot2**'s themes.

We encourage users to provide feedback and suggestions for the **EviewsR** package, so that it can continue to be improved and meet the needs of the community.

Similar packages include **URooTab** (Mati 2023) , **DynareR** (Mati 2019a) and **gretlR** (Mati 2019c).

Acknowledgements

We are grateful to Yihui Xie for creating the **knitr** package and making his book freely available on-line. We are also grateful to Bob Rudis for explaining how to easily add `Go` knit-engine to **knitr**.

APPENDIX

EViews examples using base R functions

We provide the base R equivalent operations of examples provided by EViews user manual (https://www.eviews.com/help/helpintro.html).

EViews graph command

https://eviews.com/help/helpintro.html#page/content%2Fcgraphs-Creating_a_Graph.html

Example 1

```
EViews> line income

R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "income")
```

Example 2

```
EViews> bar cons

R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "income",
+     graph_command = "bar")
```

Example 3

```
EViews> scat x y z

R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "x y z",
+     group = TRUE, graph_command = "scat")
```

Example 4

```
EViews> bar(rotate) cons

R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "cons", graph_command = "bar",
+     graph_options = "rotate")
```

Or

```
R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "cons", graph_command = "bar(rotate)")
```

Example 5

```
EViews> scat(ab=boxplot) x y z

R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "x y z",
+     group = TRUE, graph_command = "scat", graph_options = "ab=boxplot")
```

Or

```
R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "x y z",
+     group = TRUE, graph_command = "scat(ab=boxplot)")
```

Example 6

```
EViews> ser2.area(n)

R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "ser2", graph_command = "area",
+     graph_options = "n")
```

Or

```
R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "ser2", graph_command = "area(n)")
```

Example 7

```
EViews> grp6.xypair

R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "series1 series2",
+     group = TRUE, graph_command = "xypair")
```

Example 8

```
EViews> group g1 x y z
EViews> g1.scat

R> eviews_graph(wf = "EviewsR_files/EviewsR", series = "x y z",
+     group = TRUE, graph_command = "scat")
```

Example 9

Since EViews `graph` and `freeze` commands create a graph object on the current EViews workfile, we have use `exec_commands()` function instead of `eviews_graph()` function. However, if we want to include the graph in an R Markdown or Quarto document, we have to use `import_graph()` function.

```
EViews> freeze grp6.xypair(m)

R> exec_commands(commands = "freeze grp6.xypair(m)", wf = "EviewsR_files/EviewsR")
```

Example 10

```
EViews> freeze(graph1) grp6.line

R> exec_commands(commands = "freeze(graph1) grp6.line", wf = "EviewsR_files/EviewsR")
```

Example 11

```
EViews> graph gr1 ser1 ser2

R> exec_commands(commands = "graph gr1 ser1 ser2", wf = "EviewsR_files/EviewsR")
```

Example 12

```
EViews> graph gr2.line ser1 ser2

R> exec_commands(commands = "graph gr2.line ser1 ser2", wf = "EviewsR_files/EviewsR")
```

Example 13

```
EViews> graph gr3.xyline group3

R> exec_commands(commands = "graph gr3.xyline group3", wf = "EviewsR_files/EviewsR")
```

Example 14

```
EViews> graph g1.xyline(d) unemp gdp inv

R> exec_commands(commands = "graph g1.xyline(d) unemp gdp inv",
+     wf = "EviewsR_files/EviewsR")
```

Example 15

```
EViews> group grp1 sales1 sales2
EViews> graph grsales.bar(s) grp1
EViews> show grsales

R> commands = "
+ group grp1 sales1 sales2
+ graph grsales.bar(s) grp1
+ show grsales
+ "
R>
R> exec_commands(commands = commands, wf = "EviewsR_files/EviewsR")
```

Example 16

```
EViews> graph gr2.merge gr1 grsales

R> exec_commands(commands = "graph gr2.merge gr1 grsales", wf = "EviewsR_files/EviewsR")
```

EViews import command

https://www.eviews.com/help/helpintro.html#page/content%2Fcommandcmd-import.html

Example 1

```
EViews> import c:\temp\quarterly.xls @freq q 1990

R> eviews_import(source_description = "c:\\temp\\quarterly.xls",
+     frequency = "q", start_date = 1990)
```

Example 2

```
EViews> import(c=s) c:\temp\quarterly.xls range="GDP_SHEET" @freq q 1990 @rename gdp_per_capita gdp

R> eviews_import(source_description = "c:\\temp\\quarterly.xls range=\"GDP_SHEET\"",
+    options = "c=s", frequency = "q", start_date = 1990, rename_string = "gdp_per_capita gdp")
```

Example 3

```
EViews> import(mode=p) c:\temp\annual.txt @freq a 1990 @smpl 1994 1996

R> eviews_import(source_description = "c:\\temp\\annual.txt", options = "mode=p",
+     frequency = "a", start_date = 1990, smpl_string = "1994 1996")
```

Example 4

```
EViews> import(c=max, type=excel) c:\data\stateunemp.xls @id states @destid states

R> eviews_import(source_description = "c:\\data\\stateunemp.xls",
+     type = "excel", options = "c=max", id = "states", destid = "states")
```

Example 5

```
EViews> import c:\data\stategdp.txt colhead=3 delim=comma @id states @date(year) @destid states @date

R> eviews_import(source_description = "c:\\data\\stategdp.txt colhead=3 delim=comma",
+     id = "states @date(year)", destid = "states @date")
```

Example 6

```
EViews> import c:\data\cagdp.xls @id states @date(year) @destid states @date @genr states="CA"

R> eviews_import(source_description = " c:\\data\\cagdp.xls", id = "states @date(year)",
+     destid = "states @date", genr_string = "states=\"CA\"")
```

Example 7

```
EViews> import(resize) sales.dta @smpl @all

R> eviews_import(wf = "EviewsR_files/EviewsR", source_description = "sale.dta")
```

Example 8

```
EViews> import(page=demand) demand.txt @append

R> eviews_import(wf = "EviewsR_files/EviewsR", source_description = "demand.txt",
+     options = "page=demand", append = TRUE)
```

EViews wfcreate command

https://www.eviews.com/help/helpintro.html#page/content%2Fcommandcmd-wfcreate.html

Example 1

```
EViews> wfcreate(wf=storehours) 30MIN(1-6, 8:00-17:00) 1/3/2000 12/30/2000

R> eviews_wfcreate(wf = "storehours", frequency = "30MIN", subperiod_opts = "1-6, 8:00-17:00",
+     start_date = "1/3/2000", end_date = "12/30/2000")
```

Or

```
R> eviews_wfcreate(wf = "storehours", frequency = "30MIN(1-6, 8:00-17:00)",
+     start_date = "1/3/2000", end_date = "12/30/2000")
```

Example 2

```
EViews> wfcreate(wf=storehours) 30MIN(1-6, 8AM-5PM) 1/3/2000 12/30/2000

R> eviews_wfcreate(wf = "storehours", frequency = "30MIN", subperiod_opts = "1-6, 8AM-5PM",
+     start_date = "1/3/2000", end_date = "12/30/2000")
```

Or

```
R> eviews_wfcreate(wf = "storehours", frequency = "30MIN(1-6, 8AM-5PM)",
+     start_date = "1/3/2000", end_date = "12/30/2000")
```

Example 3

```
EViews> wfcreate(wf=storehours) 30MIN(1-7, 10AM-3PM) 1/3/2000 12/30/2000

R> eviews_wfcreate(wf = "storehours", frequency = "30MIN", subperiod_opts = "1-7, 10AM-3PM",
+     start_date = "1/3/2000", end_date = "12/30/2000")
```

Or

```
R> eviews_wfcreate(wf = "storehours", frequency = "30MIN(1-7, 10AM-3PM)",
+     start_date = "1/3/2000", end_date = "12/30/2000")
```

Example 4

```
EViews> wfcreate(wf=storehours) 30MIN(1-6, 8AM-5PM) 1/3/2000 10AM 12/30/2000 2PM

R> eviews_wfcreate(wf = "storehours", frequency = "30MIN", subperiod_opts = "1-6, 8AM-5PM",
+     start_date = "1/3/2000 10AM", end_date = "12/30/2000 2PM")
```

Or

```
R> eviews_wfcreate(wf = "storehours", frequency = "30MIN(1-6, 8AM-5PM)",
+     start_date = "1/3/2000 10AM", end_date = "12/30/2000 2PM")
```

Example 5

```
EViews> wfcreate w(monday) 2000 2010

R> eviews_wfcreate(frequency = "W", subperiod_opts = "monday", start_date = "2000",
+     end_date = "2010")
```

Or

```
R> eviews_wfcreate(frequency = "W(monday)", start_date = "2000",
+     end_date = "2010")
```

Example 6

```
EViews> wfcreate a(july) 2001 2007

R> eviews_wfcreate(frequency = "a", subperiod_opts = "july", start_date = "2001",
+     end_date = "2007")
```

Or

```
R> eviews_wfcreate(frequency = "a(july)", start_date = "2001", end_date = "2007")
```

Example 7

```
EViews> wfcreate w 2000 2010

R> eviews_wfcreate(frequency = "W", start_date = "2000", end_date = "2010")
```

Example 8

```
EViews> wfcreate(wf=annual, page=myproject) a 1950 2005

R> eviews_wfcreate(wf = "annual", page = "myproject", frequency = "a",
+     start_date = "1950", end_date = "2005")
```

Example 9

```
EViews> wfcreate(wf=unstruct, page=undated) u 1000

R> eviews_wfcreate(wf = "unstruct", page = "undated", frequency = "u",
+     num_observations = 1000)
```

Example 10

```
EViews> wfcreate(wf=griliches_grunfeld, page=annual) a 1935 1954 10

R> eviews_wfcreate(wf = "griliches_grunfeld", page = "annual", frequency = "a",
+     start_date = 1935, end_date = 1954, num_cross_sections = 10)
```

Example 11

```
EViews> wfcreate(wf=fourday) D(1,4) 1/3/2000 12/31/2000

R> eviews_wfcreate(wf = "fourday", frequency = "D", subperiod_opts = "1,4",
+     start_date = "1/3/2000", end_date = "12/31/2000")
```

Or

```
R> eviews_wfcreate(wf = "fourday", frequency = "D(1,4)", start_date = "1/3/2000",
+     end_date = "12/31/2000")
```

Example 12

```
EViews> wfcreate(wf=fourday) D(1-4) 1/3/2000 12/31/2000

R> eviews_wfcreate(wf = "fourday", frequency = "D", subperiod_opts = "1-4",
+     start_date = "1/3/2000", end_date = "12/31/2000")
```

Or

```
R> eviews_wfcreate(wf = "fourday", frequency = "D(1-4)", start_date = "1/3/2000",
+     end_date = "12/31/2000")
```

Example 13

```
EViews> wfcreate(wf=captimes) 15SEC(2-4) 1/3/2000 12/30/2000

R> eviews_wfcreate(wf = "captimes", frequency = "15SEC", subperiod_opts = "2-4",
+     start_date = "1/3/2000", end_date = "12/31/2000")
```

Or

```
R> eviews_wfcreate(wf = "captimes", frequency = "15SEC(2-4)", start_date = "1/3/2000",
+     end_date = "12/31/2000")
```

Example 14

```
EViews> wfcreate m 1995 +30

R> eviews_wfcreate(frequency = "m", start_date = "1995", end_date = "+30")
```

EViews pagesave command

https://eviews.com/help/helpintro.html#page/content%2Fcommandcmd-pagesave.html

Example 1

```
EViews> pagesave new_wf

R> eviews_pagesave(wf = "EviewsR_files/EviewsR", source_description = "new_wf")
```

Example 2

```
EViews> pagesave "c:\documents and settings\my data\consump"

R> eviews_pagesave(wf="EviewsR_files/EviewsR",
+                  source_description = "c:\\documents and settings\\my data\\consump")
```

Example 3

```
EViews> pagesave macro @keep gdp unemp

R> eviews_pagesave(wf = "EviewsR_files/EviewsR", source_description = "macro",
+     keep_list = "gdp unemp")
```

Example 4

```
EViews> pagesave macro @dropmap gdp*

R> eviews_pagesave(wf = "EviewsR_files/EviewsR", source_description = "macro",
+     dropmap_list = "gdp*")
```

Example 5

```
EViews> pagesave(type=excelxml, mode=update) macro.xlsx

R> eviews_pagesave(wf = "EviewsR_files/EviewsR", options = "type=excelxml,mode=update",
+     source_description = "macro.xlsx")
```

Example 6

```
EViews> pagesave(type=excelxml, mode=update) macro.xlsx range="Sheet2!a1" byrow @keep gdp unemp

R> eviews_pagesave(wf = "EviewsR_files/EviewsR", options = "type=excelxml,mode=update",
+    source_description = "macro.xlsx", table_description = "range=\"sheet2!a1\" byrow",
+     keep_list = "gdp unemp")
```

Example 7

```
EViews> pagesave(type=excelxml, mode=update) macro.xlsm range="Sheet2!a1" byrow @keep gdp unemp

R> eviews_pagesave(wf = "EviewsR_files/EviewsR", options = "type=excelxml,mode=update",
+     source_description = "macro.xlsm", table_description = "range=\"sheet2!a1\"",
+     keep_list = "gdp unemp")
```

Example 8

```
EViews> pagesave(type=excelxml, noid) macro.xlsx range="Sheet2!a1"

R> eviews_pagesave(wf = "EviewsR_files/EviewsR", options = "type=excelxml,noid",
+     source_description = "macro.xlsx", table_description = "range=\"sheet2!a1\"")
```

EViews wfsave command

https://eviews.com/help/helpintro.html#page/content%2Fcommandcmd-wfsave.html

The EViews commands pagesave and wfsave are similar. Therefore, the above examples about pagesave can be adopted by replacing pagesave with wfsave.

Notes for EViews users

This section explains some peculiarities of R's syntax. It is important for working with **EviewsR**'s functions.

Each of the following is valid way to write a character string containing double quote ("). All of them assign the same string (range="sheet2!a1" byrow) to table_description.

```
R> table_description='range="sheet2!a1" byrow'
R> table_descriptione="range=\"sheet2!a1\" byrow"
R> table_description=r'(range="sheet2!a1" byrow)'
R> table_description=r"(range="sheet2!a1" byrow)"
```

Similarly, we can write string with single quote (') as follows:

```
R> table_description='range=\'sheet2!a1\' byrow'
R> table_descriptione="range='sheet2!a1' byrow"
R> table_description=r'(range='sheet2!a1' byrow)'
R> table_description=r"(range='sheet2!a1' byrow)"

R> set_eviews_path("C:/Program Files (x86)/EViews 10/EViews10.exe")
R> set_eviews_path("C:\\Program Files (x86)\\EViews 10\\EViews10.exe")
R> set_eviews_path(r'(C:\Program Files (x86)\EViews 10\EViews10.exe)')
R> set_eviews_path(r"(C:\Program Files (x86)\EViews 10\EViews10.exe)")
```

Therefore, we recommend using r'()' or r"()" to write complex strings, as both return the strings written between the braces exactly as they are.

Current knitr knit-engines

This section provides the list of knit-engines that are currently available as of November 01, 2023. Please note that eviews is included among the list because **EviewsR** has added it as a knit-engine.

```
#>  [1] "awk"         "bash"        "coffee"      "gawk"        "groovy"
#>  [6] "haskell"     "lein"        "mysql"       "node"        "octave"
#> [11] "perl"        "php"         "psql"        "Rscript"     "ruby"
#> [16] "sas"         "scala"       "sed"         "sh"          "stata"
#> [21] "zsh"         "asis"        "asy"         "block"       "block2"
#> [26] "bslib"       "c"           "cat"         "cc"          "comment"
#> [31] "css"         "ditaa"       "dot"         "embed"       "eviews"
#> [36] "exec"        "fortran"     "fortran95"   "go"          "highlight"
#> [41] "js"          "julia"       "python"      "R"           "Rcpp"
#> [46] "sass"        "scss"        "sql"         "stan"        "targets"
#> [51] "tikz"        "verbatim"    "glue"        "glue_sql"    "gluesql"
#> [56] "theorem"     "lemma"       "corollary"   "proposition" "conjecture"
#> [61] "definition"  "example"     "exercise"    "hypothesis"  "proof"
#> [66] "remark"      "solution"
```

Session information

This section provides detailed information on the Operating System, R packages and their versions used in this document.

```
#> R version 4.2.2 (2022-10-31)
#> Platform: aarch64-apple-darwin20 (64-bit)
#> Running under: macOS 14.0
#>
#> Matrix products: default
#> BLAS:  /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0.dylib
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib
#>
#> locale:
#> [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
#>
#> attached base packages:
#> [1] grid      stats     graphics  grDevices utils     datasets  methods
#> [8] base
#>
#> other attached packages:
#> [1] magrittr_2.0.3   ggplot2_3.4.3    Gmisc_3.0.3      htmlTable_2.4.1
#> [5] Rcpp_1.0.11      kableExtra_1.3.4 knitr_1.45       EviewsR_0.1.5
#>
#> loaded via a namespace (and not attached):
#>  [1] lubridate_1.9.3     svglite_2.1.1       lattice_0.21-9
#>  [4] zoo_1.8-12          digest_0.6.33       utf8_1.2.3
#>  [7] R6_2.5.1            backports_1.4.1     evaluate_0.22
#> [10] httr_1.4.7          pillar_1.9.0        rlang_1.1.1
#> [13] data.table_1.14.8   rstudioapi_0.15.0   rpart_4.1.19
#> [16] checkmate_2.2.0     rmarkdown_2.25      webshot_0.5.5
#> [19] stringr_1.5.0       foreign_0.8-85      htmlwidgets_1.6.2
#> [22] tinytex_0.47        munsell_0.5.0       hunspell_3.0.3
#> [25] compiler_4.2.2      xfun_0.40           pkgconfig_2.0.3
#> [28] systemfonts_1.0.4   base64enc_0.1-3     htmltools_0.5.6.1
#> [31] nnet_7.3-19         forestplot_3.1.3    tidyselect_1.2.0
#> [34] gridExtra_2.3       tibble_3.2.1        bookdown_0.35
#> [37] rjtools_1.0.12      Hmisc_5.1-1         XML_3.99-0.14
#> [40] fansi_1.0.5         viridisLite_0.4.2   withr_2.5.1
#> [43] dplyr_1.1.3         gtable_0.3.4        lifecycle_1.0.3
#> [46] formatR_1.14        scales_1.2.1        cli_3.6.1
#> [49] stringi_1.7.12      fs_1.6.3            xml2_1.3.5
#> [52] xts_0.13.1          yesno_0.1.2         vctrs_0.6.3
#> [55] generics_0.1.3      Formula_1.2-5       tools_4.2.2
#> [58] glue_1.6.2          purrr_1.0.2         abind_1.4-5
#> [61] fastmap_1.1.1       yaml_2.3.7          timechange_0.2.0
#> [64] colorspace_2.1-0    cluster_2.1.4       BiocManager_1.30.22
#> [67] rvest_1.0.3
```

References

Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2020. *Rmarkdown: Dynamic Documents for r*. https://github.com/rstudio/rmarkdown.

Christensen, Garret, and Edward Miguel. 2018. "Transparency, Reproducibility, and the Credibility of Economics Research." *Journal of Economic Literature* 56 (3): 920–80. https://doi.org/10.1257/jel.20171350.

Franco, Annie, Neil Malhotra, and Gabor Simonovits. 2014. "Publication Bias in the Social Sciences: Unlocking the File Drawer." *Science* 345 (6203): 1502–5. https://doi.org/10.1126/science.1255484.

Gerber, Alan, Kevin Arceneaux, Cheryl Boudreau, Conor Dowling, Sunshine Hillygus, Thomas Palfrey, Daniel R. Biggers, and David J. Hendry. 2014. "Reporting Guidelines for Experimental Research: A Report from the Experimental Research Section Standards Committee." *Journal of Experimental Political Science* 1 (1): 81–98. https://doi.org/10.1017/xps.2014.11.

Harvey, Campbell R., Yan Liu, and Heqing Zhu. 2015. "… and the Cross-Section of Expected Returns." *The Review of Financial Studies* 29 (1): 5–68. https://doi.org/10.1093/rfs/hhv059.

Ioannidis, John P. A. 2005. "Why Most Published Research Findings Are False." *PLOS Medicine* 2 (8). https://doi.org/10.1371/journal.pmed.0020124.

Mati, Sagiru. 2019a. *DynareR: Bringing the Power of Dynare to R, R Markdown, and Quarto*. https://CRAN.R-project.org/package=DynareR.

———. 2019b. *EviewsR: A Seamless Integration of EViews and R*. https://CRAN.R-project.org/package=EviewsR.

———. 2019c. *gretlR: A Seamless Integration of Gretl and R*. https://CRAN.R-project.org/package=gretlR.

———. 2023. *URooTab: Tabular Reporting of EViews Unit Root Tests*. CRAN. https://CRAN.R-project.org/package=EviewsR.

Mati, Sagiru, Irfan Civcir, and Huseyin Ozdeser. 2023. "ECOWAS Common Currency, a Mirage or Possibility?" *Panoeconomicus*, 1–25. https://doi.org/10.2298/PAN191119015M.

Mati, Sagiru, Irfan Civcir, and Hüseyin Ozdeser. 2019. "ECOWAS Common Currency: How Prepared Are Its Members?" *Investigación Económica* 78 (May): 89. https://doi.org/10.22201/fe.01851667p.2019.308.69625.

Murrell, Paul. 2019. **hexView***: Viewing Binary Files*. https://CRAN.R-project.org/package=hexView.

Pretis, Felix, J. James Reade, and Genaro Sucarrat. 2018. "Automated General-to-Specific (**GETS**) Regression Modeling and Indicator Saturation for Outliers and Structural Breaks." *Journal of Statistical Software* 86 (3): 1–44. https://doi.org/10.18637/jss.v086.i03.

R Core Team. 2019. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Sandve, Anton AND Taylor, Geir Kjetil AND Nekrutenko. 2013. "Ten Simple Rules for Reproducible Computational Research." *PLOS Computational Biology* 9 (10): 1–4. https://doi.org/10.1371/journal.pcbi.1003285.

Simmons, Joseph P., Leif D. Nelson, and Uri Simonsohn. 2011. "False-Positive Psychology: Undisclosed Flexibility in Data Collection and Analysis Allows Presenting Anything as Significant." *Psychological Science* 22 (11): 1359–66. https://doi.org/10.1177/0956797611417632.

Startz, Richard. 2019. "EViews Illustrated." *University of California: Santa Barbara, CA, USA*.

Stodden, Victoria, Friedrich Leisch, and Roger D Peng. 2014. *Implementing Reproducible Research*. Crc Press.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Xie, Yihui. 2014. "**knitr**: A Comprehensive Tool for Reproducible Research in r." In *Implementing Reproducible Computational Research*, edited by Victoria Stodden, Friedrich Leisch, and Roger D. Peng. Chapman; Hall/CRC. http://www.crcpress.com/product/isbn/9781466561595.

———. 2015. *Dynamic Documents with r and* **knitr**. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.name/knitr/.

———. 2019. **knitr***: A General-Purpose Package for Dynamic Report Generation in r*. https://yihui.name/knitr/.

Sagiru Mati
Near East University
[1]*Operational Research Center in Healthcare*
Near East University, Nicosia, North Cyprus
[2]*Department of Economics*
Yusuf Maitama Sule University, Nigeria
https://www.smati.com.ng

ORCiD: 0000-0003-1413-3974
sagirumati@gmail.com

Irfan Civcir
Ankara University
Department of Economics
Ankara, Turkey
http://cv.ankara.edu.tr/kisi.php?id=civcir@politics.ankara.edu.tr&deger=2
ORCiD: 0000-0002-2557-2625
civcir@politics.ankara.edu.tr

S. I. Abba
King Fahd University of Petroleum and Minerals
Interdisciplinary Research Center for Membranes and Water Security
Dhahran 31261, Saudi Arabia
https://scholar.google.com/citations?user=4h2JX7YAAAAJ&hl=en
ORCiD: 0000-0001-9356-2798
sani.abba@kfupm.edu.sa