Tutorial

Data Envelopment Analysis with deaR

Version 1.0 (English)

(Noviembre 2018)

Vicente Coll-Serrano[1]

Rafael Benítez[2]

Vicente J. Bolós[3]

(1) Department of Applied Economics. Vicente.Coll@uv.es

Quantitative Methods for Measuring Culture (MC2)

(2) Department of Business Mathematics. Benitez.Suarez@uv.es

(3) Department of Business Mathematics. Vicente.Bolos@uv.es

School of Economics (www.uv.es/economia)

University of Valencia (Spain)

Index

1. INTRODUCTION.

deaR is a new and versatile R package (free software) that allows to run a wide variety of models based on Data Envelopment Analysis.

This tutorial is written so that non-R users can use deaR, but it is not a manual of introduction to R[1].

If you are an R user, you can directly go to section 5 and 7.

We want **deaR** to become the reference software tool for those researchers, practitioners, teachers, students and users of the Data Envelopment Analysis methodology. For this, any comments and suggestions to improve **deaR** will be really appreciated. We also accept suggestions of DEA models to be considered for being programmed in new versions of **deaR**. In this sense, we like to anticipate that a new version of **deaR** will include models or features not currently covered such as: stochastic DEA, network DEA, Malmquist index (other decompositions and bootstrapping), negative values and extension of undesirable inputs/outputs to other models in which they are not available now, etc.

We will release a **deaR shiny version** (an interactive web app) soon. We will inform you when it available.

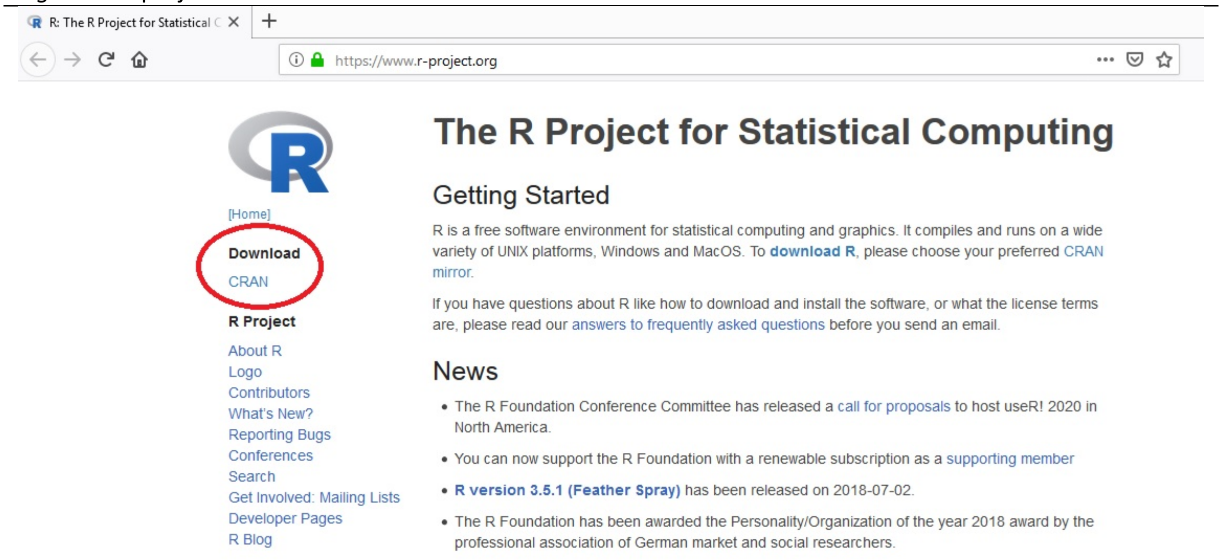
2. Download and Install R and RStudio.

To use **deaR**, firstly we have to download and install R and Rstudio.

2.1. R installation.

To install R, we go to the R-project website: http://www.r-project.org (see Figure 1).

Figure 1. R-project website.



The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.

If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

News

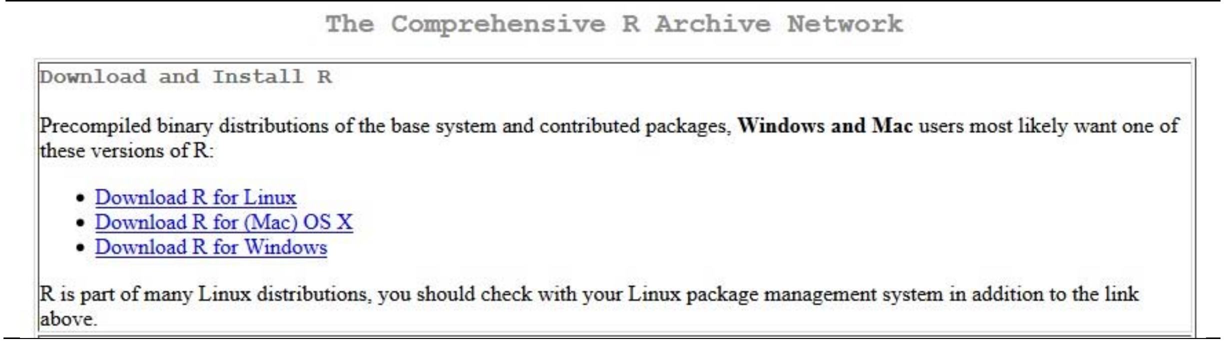
- The R Foundation Conference Committee has released a call for proposals to host useR! 2020 in North America.
- You can now support the R Foundation with a renewable subscription as a supporting member
- **R version 3.5.1 (Feather Spray)** has been released on 2018-07-02.
- The R Foundation has been awarded the Personality/Organization of the year 2018 award by the professional association of German market and social researchers.

To download R, we click the *CRAN* link and select the "mirror site" nearest to our location.

Now, depending on the operating system of our computer, we select the appropriate option (see Figure 2).

[1]Manual of introduction to R: https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf

Figure 2. Versions of R.



The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

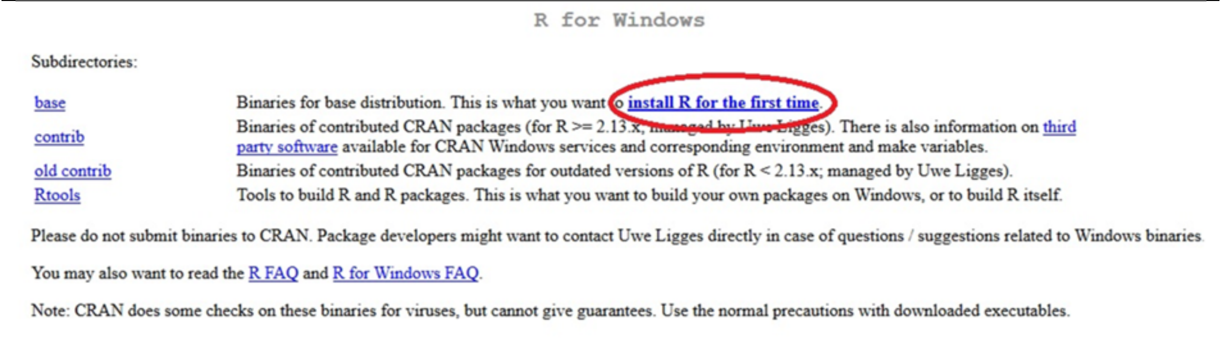
- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

a) Installing R on Windows.

By clicking the *Download R for Windows* link, we will go to the webpage shown in the Figure below. We click on *install R for the first time* (see Figure 3).

Figure 3. R for Windows.



R for Windows

Subdirectories:

base	Binaries for base distribution. This is what you want to install R for the first time.
contrib	Binaries of contributed CRAN packages (for R >= 2.13.x; managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).
Rtools	Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

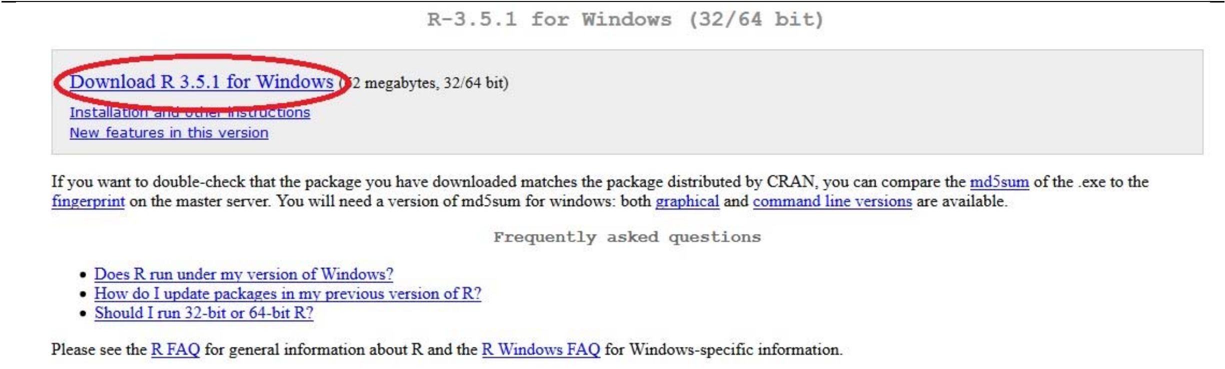
Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the R FAQ and R for Windows FAQ.

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

Next, we click on ***Download R 3.5.1 for Windows*** and save the installation file (see Figure 4)

Figure 4. Download R for Windows.



R-3.5.1 for Windows (32/64 bit)

Download R 3.5.1 for Windows (62 megabytes, 32/64 bit)
Installation and other instructions
New features in this version

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server. You will need a version of md5sum for windows: both graphical and command line versions are available.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

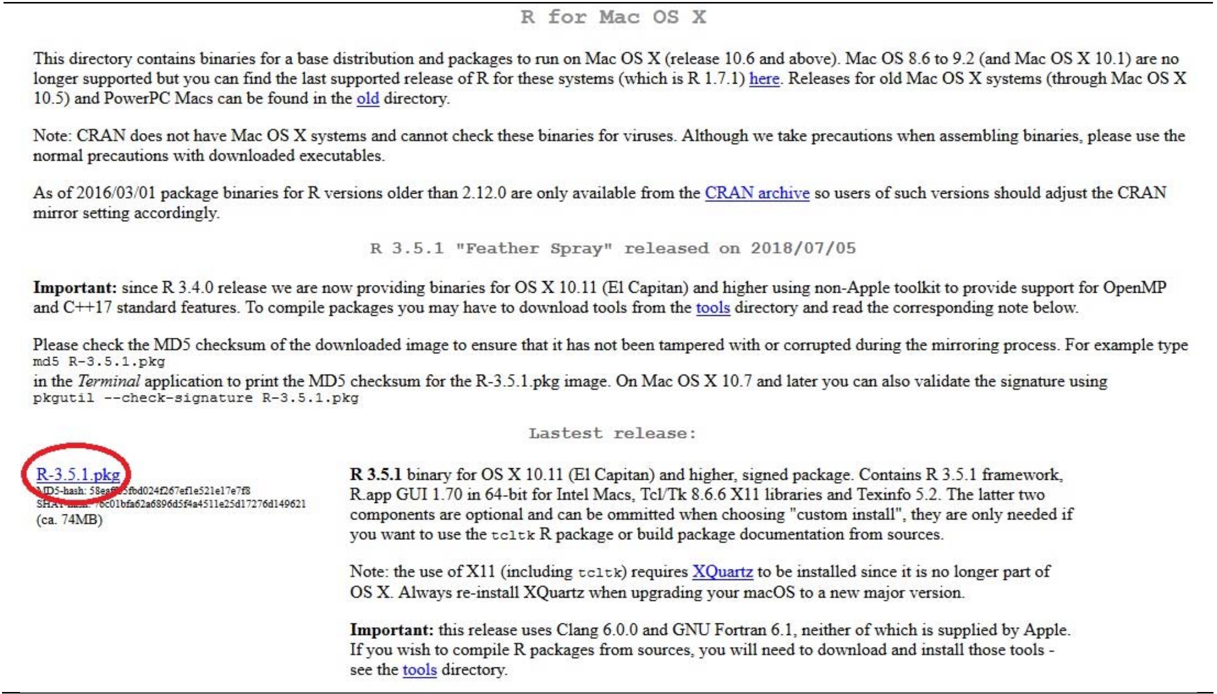
Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

Now, we open the downloaded file (by double-clicking on the file) to install R.

b) Installing R on Mac.

By clicking the *Download R for (Mac) OS X* link, we will go to the webpage shown in the Figure below. We click on ***R-3.5.1.pkg*** to download the installation file (see Figure 5).

Figure 5. R for Mac.



R for Mac OS X

This directory contains binaries for a base distribution and packages to run on Mac OS X (release 10.6 and above). Mac OS 8.6 to 9.2 (and Mac OS X 10.1) are no longer supported but you can find the last supported release of R for these systems (which is R 1.7.1) here. Releases for old Mac OS X systems (through Mac OS X 10.5) and PowerPC Macs can be found in the old directory.

Note: CRAN does not have Mac OS X systems and cannot check these binaries for viruses. Although we take precautions when assembling binaries, please use the normal precautions with downloaded executables.

As of 2016/03/01 package binaries for R versions older than 2.12.0 are only available from the CRAN archive so users of such versions should adjust the CRAN mirror setting accordingly.

R 3.5.1 "Feather Spray" released on 2018/07/05

Important: since R 3.4.0 release we are now providing binaries for OS X 10.11 (El Capitan) and higher using non-Apple toolkit to provide support for OpenMP and C++17 standard features. To compile packages you may have to download tools from the tools directory and read the corresponding note below.

Please check the MD5 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type
`md5 R-3.5.1.pkg`
in the *Terminal* application to print the MD5 checksum for the R-3.5.1.pkg image. On Mac OS X 10.7 and later you can also validate the signature using
`pkgutil --check-signature R-3.5.1.pkg`

Lastest release:

R-3.5.1.pkg
MD5-hash: 58ea[illegible]5fbd024f267ef1e521e17e7f8
SHA[illegible]6c01bfa62a6896d5f4a4511e25d17276d149621
(ca. 74MB)

R 3.5.1 binary for OS X 10.11 (El Capitan) and higher, signed package. Contains R 3.5.1 framework, R.app GUI 1.70 in 64-bit for Intel Macs, Tcl/Tk 8.6.6 X11 libraries and Texinfo 5.2. The latter two components are optional and can be ommitted when choosing "custom install", they are only needed if you want to use the `tcltk` R package or build package documentation from sources.

Note: the use of X11 (including `tcltk`) requires XQuartz to be installed since it is no longer part of OS X. Always re-install XQuartz when upgrading your macOS to a new major version.

Important: this release uses Clang 6.0.0 and GNU Fortran 6.1, neither of which is supplied by Apple. If you wish to compile R packages from sources, you will need to download and install those tools - see the tools directory.

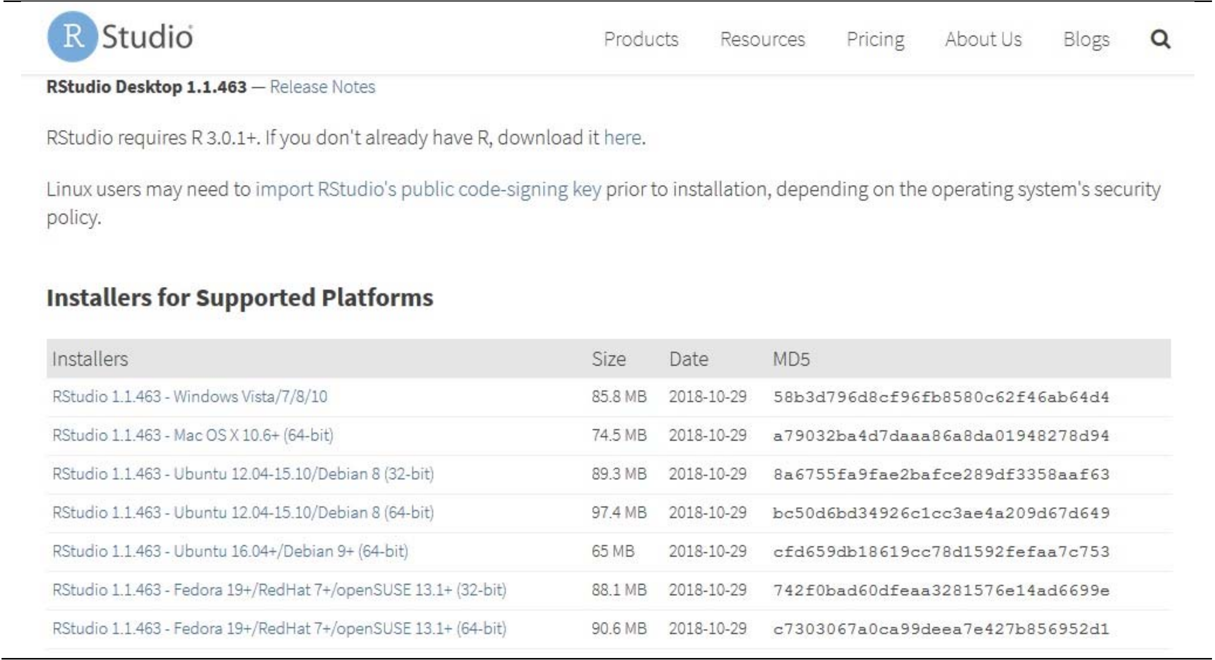
We open *R-3.5.1.pkg* and follow the instructions to install R.

2.2. RStudio installation.

Once we have installed R, we download RStudio from the following link (see Figure 6):

https://www.rstudio.com/products/rstudio/download/#download

Figure 6. Download RStudio.



RStudio Desktop 1.1.463 — Release Notes

RStudio requires R 3.0.1+. If you don't already have R, download it here.

Linux users may need to import RStudio's public code-signing key prior to installation, depending on the operating system's security policy.

Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 1.1.463 - Windows Vista/7/8/10	85.8 MB	2018-10-29	58b3d796d8cf96fb8580c62f46ab64d4
RStudio 1.1.463 - Mac OS X 10.6+ (64-bit)	74.5 MB	2018-10-29	a79032ba4d7daaa86a8da01948278d94
RStudio 1.1.463 - Ubuntu 12.04-15.10/Debian 8 (32-bit)	89.3 MB	2018-10-29	8a6755fa9fae2bafce289df3358aaf63
RStudio 1.1.463 - Ubuntu 12.04-15.10/Debian 8 (64-bit)	97.4 MB	2018-10-29	bc50d6bd34926c1cc3ae4a209d67d649
RStudio 1.1.463 - Ubuntu 16.04+/Debian 9+ (64-bit)	65 MB	2018-10-29	cfd659db18619cc78d1592fefaa7c753
RStudio 1.1.463 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	88.1 MB	2018-10-29	742f0bad60dfeaa3281576e14ad6699e
RStudio 1.1.463 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	90.6 MB	2018-10-29	c7303067a0ca99deea7e427b856952d1

To download the executable file, we select the option according to our operating system:

- RStudio 1.1.463 - Windows Vista/7/8/10
- RStudio 1.1.463 - Mac OS X 10.6+ (64-bit)

First, we save the file. Then, we open it to install RStudio. We follow the installation instructions.

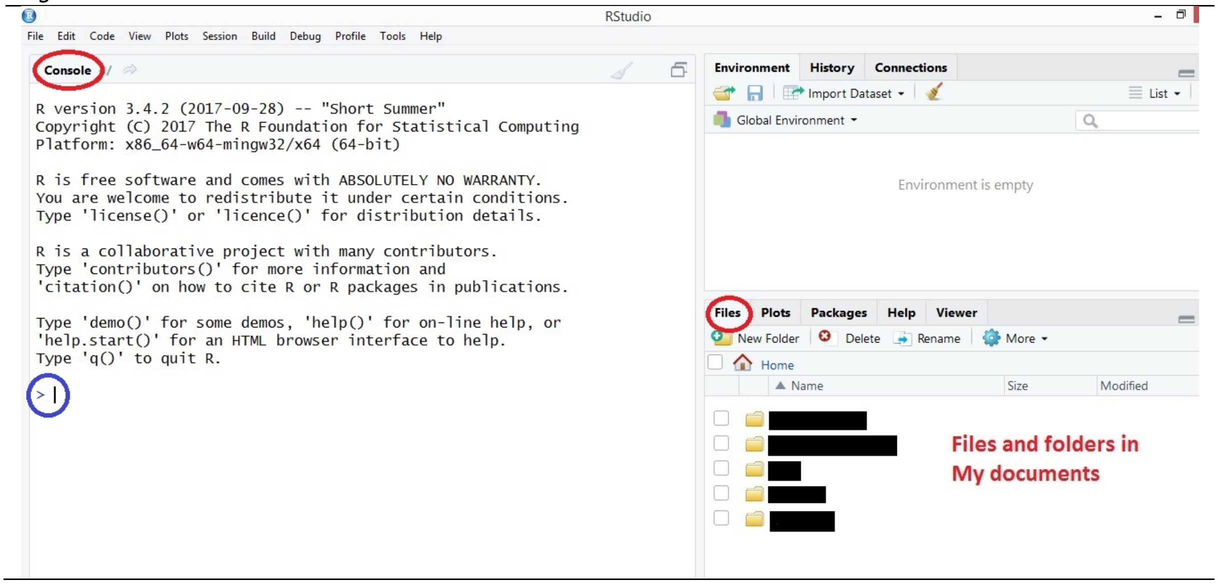
3. Getting started with RStudio.

In general, we use RStudio instead of R because of the friendly interface of the former.

To boot up RStudio, we click on the RStudio icon:

By opening RStudio, we should see something like the Figure 7:

Figure 7. The RStudio interface.



```
R version 3.4.2 (2017-09-28) -- "Short Summer"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Console
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home
Name Size Modified
Files and folders in My documents

[redacted]

By default, the *Console* is in the left side. After a brief informative text, the system prompt appears (">"). Can you see the flashing cursor? Here is where we can write the commands or instructions to be run (by R). To execute an instruction in the *Console* and get the result we press *Enter*.

3.1. How to create a script.

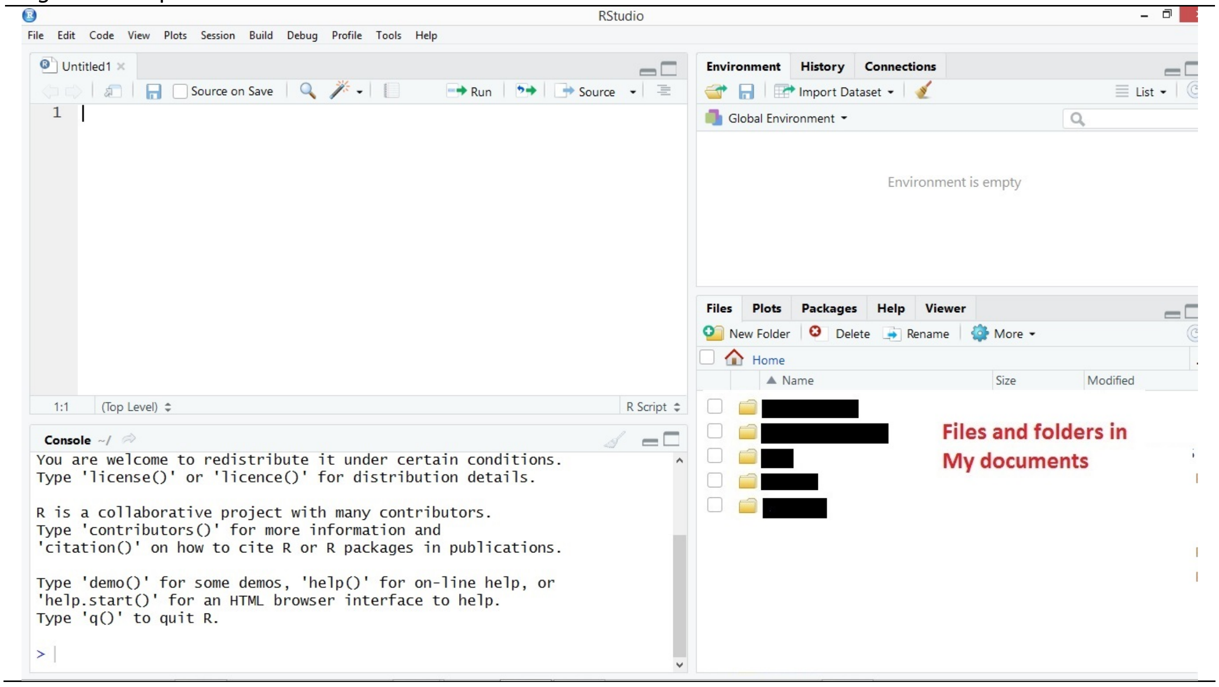
Working in the Console is very limited since in the Console the instructions are generally written and executed one by one. For this reason, it is usual working with **scripts** or instructions files. These files have the extension "*.R*".

To create a script we select *File > New File > R Script*.

Now, the script is the top left panel and the *Console* is the bottom left panel. By default, the name of the script is "*Untitled1*" (see Figure 8).

In a script we can write the instructions line by line. The instructions can be run one by one or we can select all (or some) of them and run the selection. We click on Run to run them.

Figure 8. Script.


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1
Source on Save
Run
Source
1:1
(Top Level)
R Script
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home
Name Size Modified
Files and folders in My documents
Console ~/

```
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

[redacted]

4. How to create and work with projects in RStudio.

Once we boot up RStudio, we should set the working directory to indicate R and RStudio where our data, scripts, etc. are. However, in our opinion, the best option is to create a project in RStudio. Creating a project, all the files (data, scripts, etc.) and directories inside the project are directly linked to the project. That is to say, all the work on a project will be self-contained in the directory of the project. In this way, we can easily share the project with someone else or copy-paste the project in another computer, etc.

Important note:

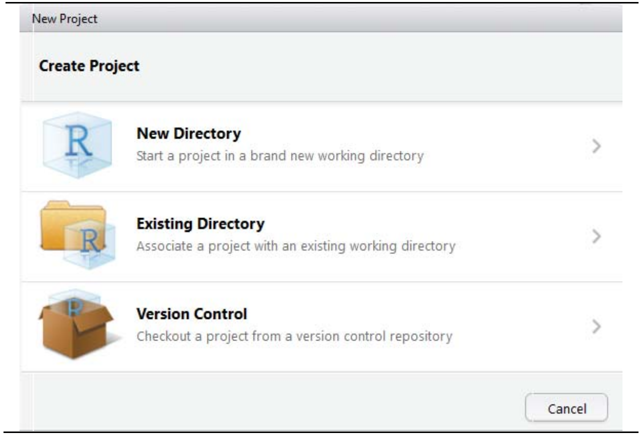
We create a project for each activity/work (for example, for each article/paper).

To organize a project properly, we recommend to create specific directories for raw data, results, text documents, etc.

4.1. Creating a project.

To create a project in RStudio, we select *File* > *New project...* We should see something similar as the Figure below:

Figure 9. Create a project.


New Project
Create Project
New Directory
Start a project in a brand new working directory
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository
Cancel

To create a project in a new directory, we click on the *New Directory* button. Then, we select the type of project. In our case, *New Project*. Now, we name the directory (or folder) that we are creating. It will also be the name of the project. Finally, we click on *Create Project*. Following this process, a folder in Documents will be created, and inside it we will find the file: "*folder_name.Rproj*".

To create a project in an existing directory, we click on the *Existing Directory* button, and then we select the directory or folder by clicking on the *Browse*… button. Finally, we click on *Create Project*.

4.2. Opening a project.

To open a Project we double click on the file with extension "*.Rproj*". We can also open the project from the RStudio menu: *File > Open Project…*

Advantage of projects: any file we save while we are working in a project will be saved inside the project's directory.

4.3. Additional information.

Here you have two readings about projects in RStudio. The second reading includes information about how to create projects.

- https://www.r-bloggers.com/managing-projects-using-rstudio/
- https://www.ssc.wisc.edu/sscc/pubs/RFR/RFR_Projects.html#projects

4.4. How to create a project.

Step 1. Open RStudio

Step 2. Select *File > New Project*

Step 3. Select *New Directory*

Step 4. *Project Type*: select *New Project*

Step 5. *Directory name*: Paper_1

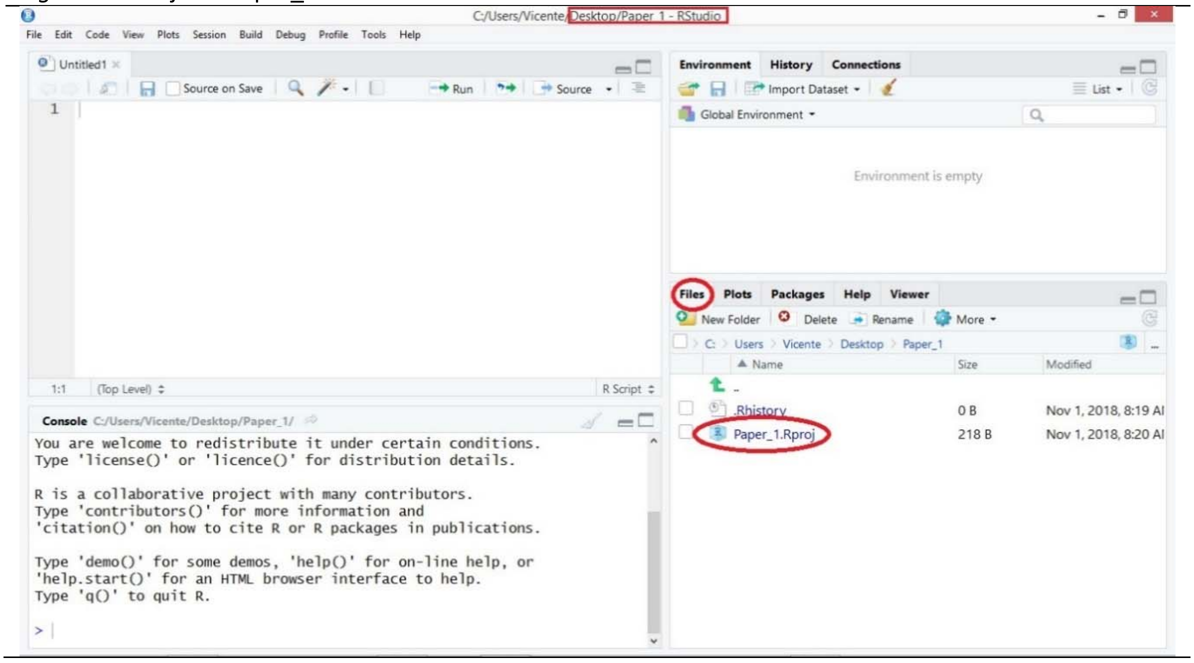
Step 6: *Create project as subdirectory of*: by using the Browse, select the route where the project folder will be created. For example, select the Desktop folder.

Step 7. Click on *Create Project*

Result: The "*Paper_1*" folder will be created in the Desktop folder. This folder will have two files (see Figure 10).

- **Paper_1** (type: R Project)
- .Rhistory (type: RHISTORY file)

Figure 10. Project "Paper_1".


C:/Users/Vicente/Desktop/Paper 1 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1
Source on Save
Run
Source
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
New Folder Delete Rename More
C: > Users > Vicente > Desktop > Paper_1
Name Size Modified
1:1 (Top Level) R Script
Console C:/Users/Vicente/Desktop/Paper_1/

Name	Size	Modified
.Rhistory	0 B	Nov 1, 2018, 8:19 AI
Paper_1.Rproj	218 B	Nov 1, 2018, 8:20 AI

```
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Important note:

The working session with a project will be always opened

double-clicking on the file with extension "*.Rproj*"

or

File > Open Project and selecting the project

Working simultaneously with several projects is not recommended.

5. How to install and load deaR.

Open RStudio or the project: "*Paper_1*".

Select *File > New File > R script*

5.1. Installing deaR.

To install **deaR**, we write into the script:

install.packages("deaR")

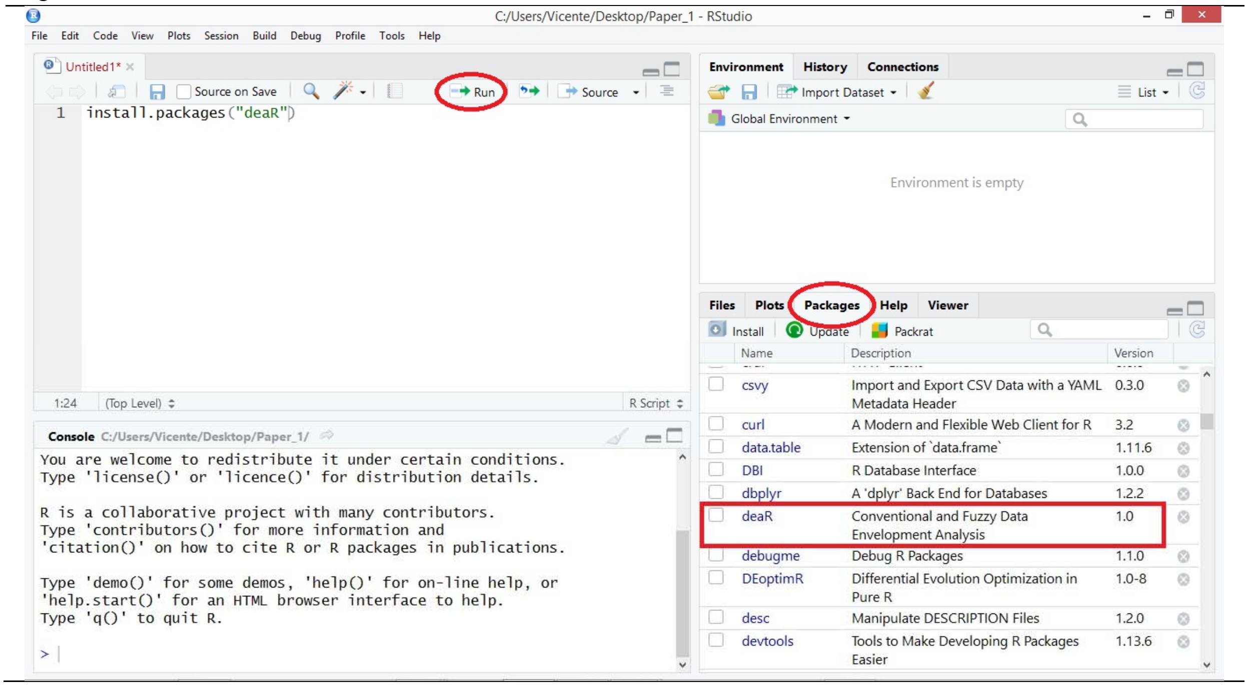
and we run the instruction. For that, we click on the *Run* button: Run (see Figure 11).

By clicking on the *Packages* menu in the bottom right window, a list of all installed packages will be shown. Now, **deaR** package should be listed.

Important note:

We have to install **deaR** only once. Then, we can get the new **deaR** versions updating the package.

Figure 11. Install **deaR**.


C:/Users/Vicente/Desktop/Paper_1 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
Source on Save
Run
Source
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
Install Update Packrat
1:24 (Top Level) R Script
Console C:/Users/Vicente/Desktop/Paper_1/

```
install.packages("deaR")
```

```
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Name	Description	Version
csvy	Import and Export CSV Data with a YAML Metadata Header	0.3.0
curl	A Modern and Flexible Web Client for R	3.2
data.table	Extension of `data.frame`	1.11.6
DBI	R Database Interface	1.0.0
dbplyr	A 'dplyr' Back End for Databases	1.2.2
deaR	Conventional and Fuzzy Data Envelopment Analysis	1.0
debugme	Debug R Packages	1.1.0
DEoptimR	Differential Evolution Optimization in Pure R	1.0-8
desc	Manipulate DESCRIPTION Files	1.2.0
devtools	Tools to Make Developing R Packages Easier	1.13.6

5.2. Loading deaR.

Once we have installed **deaR**, or any other R package, to use it we have to load it first (see Figure 12). For that, we write into the script:

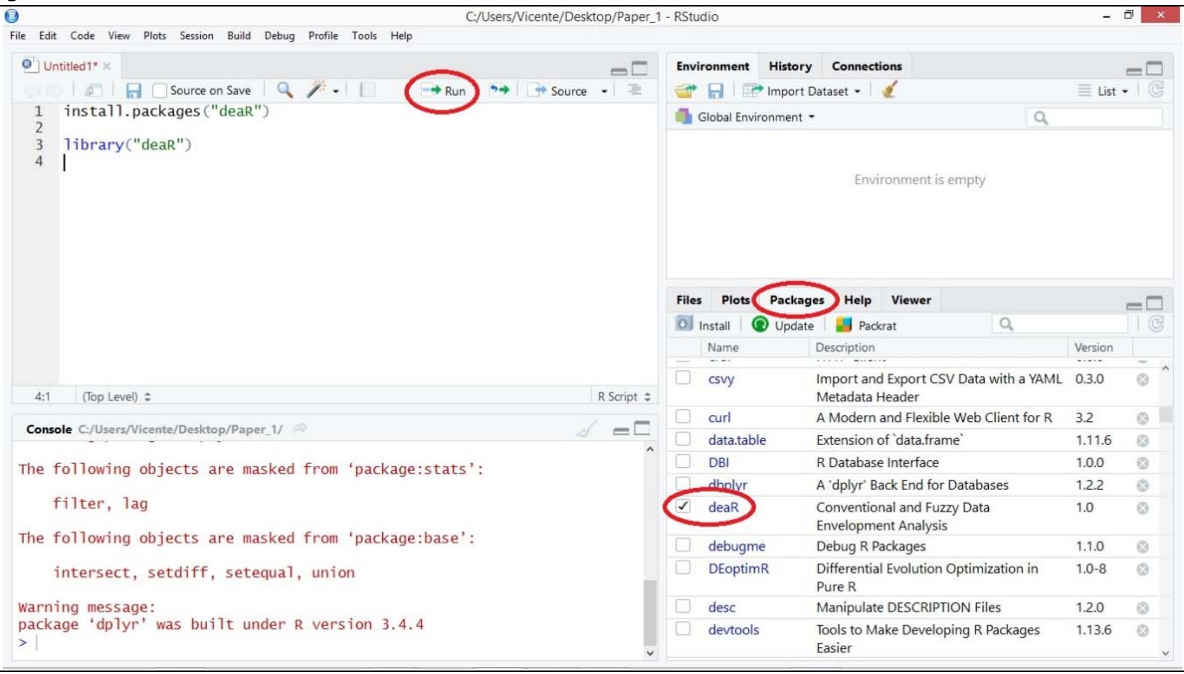
library("deaR")

and we run the instruction clicking on the Run button: Run

Important note:

It is necessary to load **deaR** package in each working session.

Figure 12. Load **deaR**.


C:/Users/Vicente/Desktop/Paper_1 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
Source on Save
Run
Source
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
Install Update Packrat
4:1 (Top Level)
R Script
Console C:/Users/Vicente/Desktop/Paper_1/

```r
install.packages("deaR")

library("deaR")
```

```
The following objects are masked from ‘package:stats’:

    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union

Warning message:
package ‘dplyr’ was built under R version 3.4.4
>
```

	Name	Description	Version
	csvy	Import and Export CSV Data with a YAML Metadata Header	0.3.0
	curl	A Modern and Flexible Web Client for R	3.2
	data.table	Extension of `data.frame`	1.11.6
	DBI	R Database Interface	1.0.0
	dbplyr	A 'dplyr' Back End for Databases	1.2.2
✓	deaR	Conventional and Fuzzy Data Envelopment Analysis	1.0
	debugme	Debug R Packages	1.1.0
	DEoptimR	Differential Evolution Optimization in Pure R	1.0-8
	desc	Manipulate DESCRIPTION Files	1.2.0
	devtools	Tools to Make Developing R Packages Easier	1.13.6

6. Save the script and close the working session.

To save the script, we select *File > Save as...*

Now, we name the file (for example: session_1) and click on the *Save* button. By default, the script will be saved in the project folder: “*Paper_1*”. Remember, this is an advantage of working with projects.

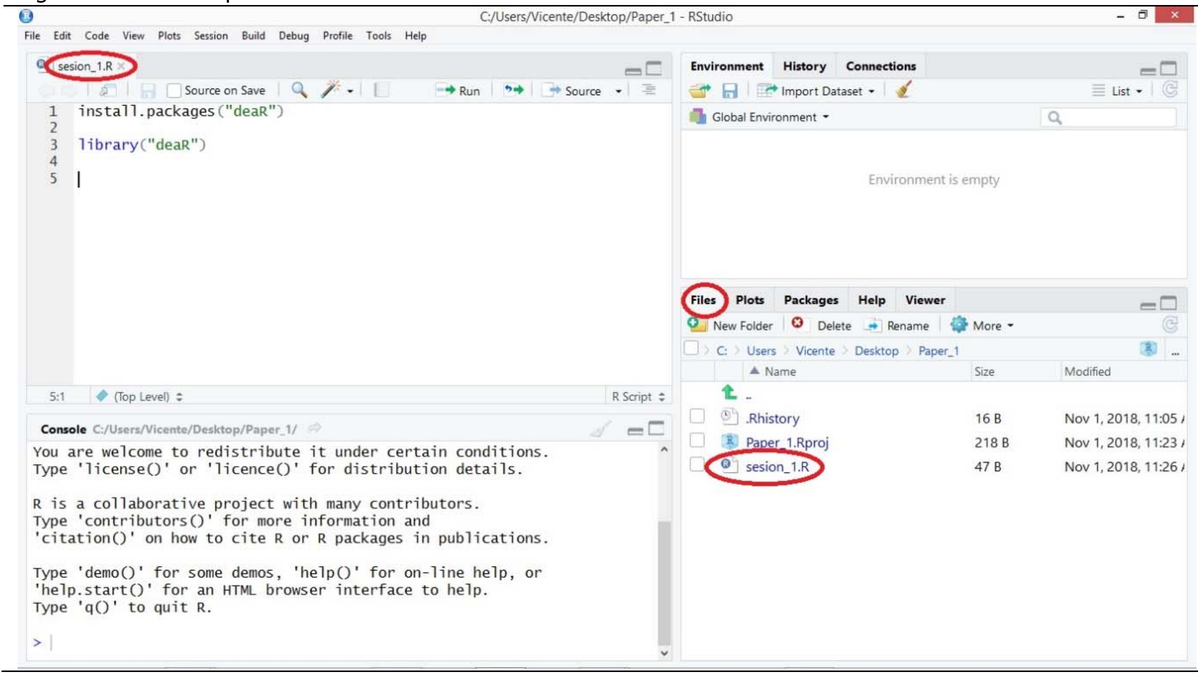
Notice that the name of the R file (script) is *“session_1.R*” instead of “*Untitled1”* (see Figure 13). This file is shown in the list of files that are part of the project Paper_1.

Note: The content of this first script is completely irrelevant up to now. What is important is to remember that we can save a set of instructions in a script.

Important note:

Give names to scripts that reflect their content.

Figure 13. Save script.


C:/Users/Vicente/Desktop/Paper_1 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
sesion_1.R
Source on Save
Run
Source
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
New Folder Delete Rename More
C: > Users > Vicente > Desktop > Paper_1
Name Size Modified
.Rhistory 16 B Nov 1, 2018, 11:05
Paper_1.Rproj 218 B Nov 1, 2018, 11:23
sesion_1.R 47 B Nov 1, 2018, 11:26
5:1 (Top Level) R Script
Console C:/Users/Vicente/Desktop/Paper_1/

```
install.packages("deaR")

library("deaR")

```

```
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

If we want to close the project but not Rstudio: *File > Close project*

If we want to close the project and Rstudio: *File > Quit Session*

At this point, **we save "*session_1.R*", close the Project "*Paper_1*" and quit RStudio**.

7. Data Envelopment Analysis with deaR.

We start by opening the project "*Paper_1*". For that, we double click on the file "*Paper_1.Rproj*". RStudio and the project will open. Remember, you can also open RStudio first and then the project (*File > Open Project*).

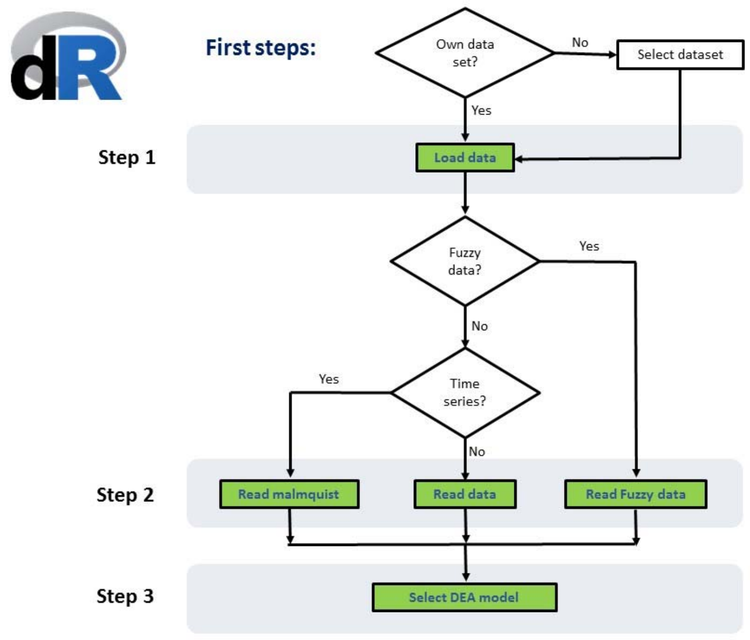
Next, we create a new script (*File > New File > R Script*). In this new script we write the instruction to load **deaR**:

library("deaR")

(Note: we execute the instruction by clicking on the *Run* button Run).

In the flow chart below (see Figure 14), we can see the steps to perform any DEA (Data Envelopment Analysis) with **deaR**.

Figure 14. Steps to use **deaR**.



7.1. Importing data into R.

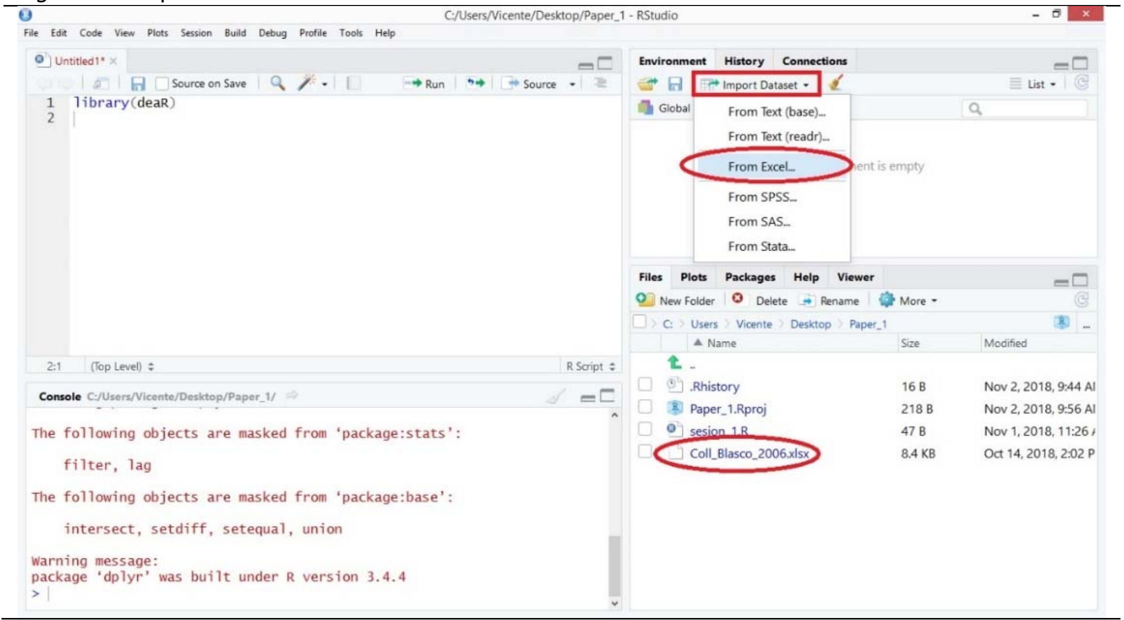
The first step is to import the data we are going to use to perform a DEA analysis. The non-R user can use the *Import dataset* option. Let's see how to import data with Example 1.

Example 1. Import data into R data from an Excel file:

1. Download the data of the example from www.uv.es/vcoll/Coll_Blasco_2006.xlsx and save the file into the folder of the project "*Paper_1*".

2. From RStudio *Environment* menu in the top left window, we select the option: *Import Dataset < From Excel* (see Figure 15).

Figure 15. Import data from Excel.

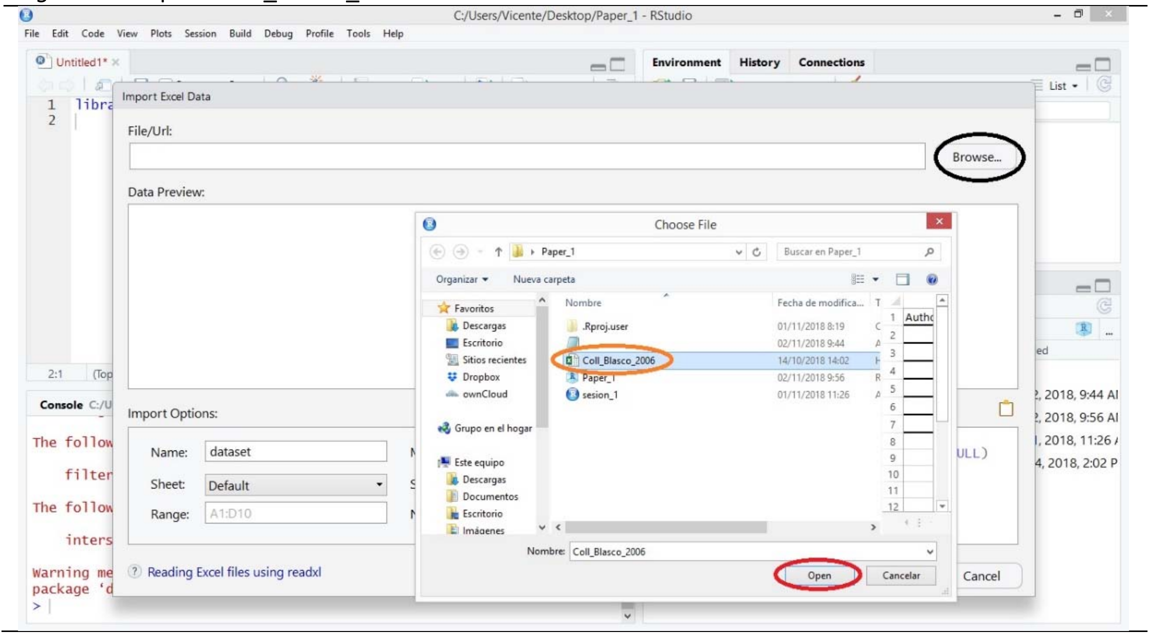

C:/Users/Vicente/Desktop/Paper_1 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
Source on Save
Run
Source
2:1
(Top Level)
R Script
Console C:/Users/Vicente/Desktop/Paper_1/
Environment History Connections
Import Dataset
List
Global
From Text (base)...
From Text (readr)...
From Excel...
From SPSS...
From SAS...
From Stata...
ent is empty
Files Plots Packages Help Viewer
New Folder Delete Rename More
C: > Users > Vicente > Desktop > Paper_1
Name Size Modified
.Rhistory 16 B Nov 2, 2018, 9:44 AM
Paper_1.Rproj 218 B Nov 2, 2018, 9:56 AM
sesion_1.R 47 B Nov 1, 2018, 11:26
Coll_Blasco_2006.xlsx 8.4 KB Oct 14, 2018, 2:02 P

```
library(deaR)
```

```
The following objects are masked from ‘package:stats’:

    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union

Warning message:
package ‘dplyr’ was built under R version 3.4.4
>
```

3. The window *Import Excel Data* opens. We click on the *Browse* button to choose the Excel file ("*Coll_Blasco_2006.xlsx*")[2] and then on the *Open* button (see Figure 16).

Figure 16. Import "Coll_Blasco_2006.xlsx".

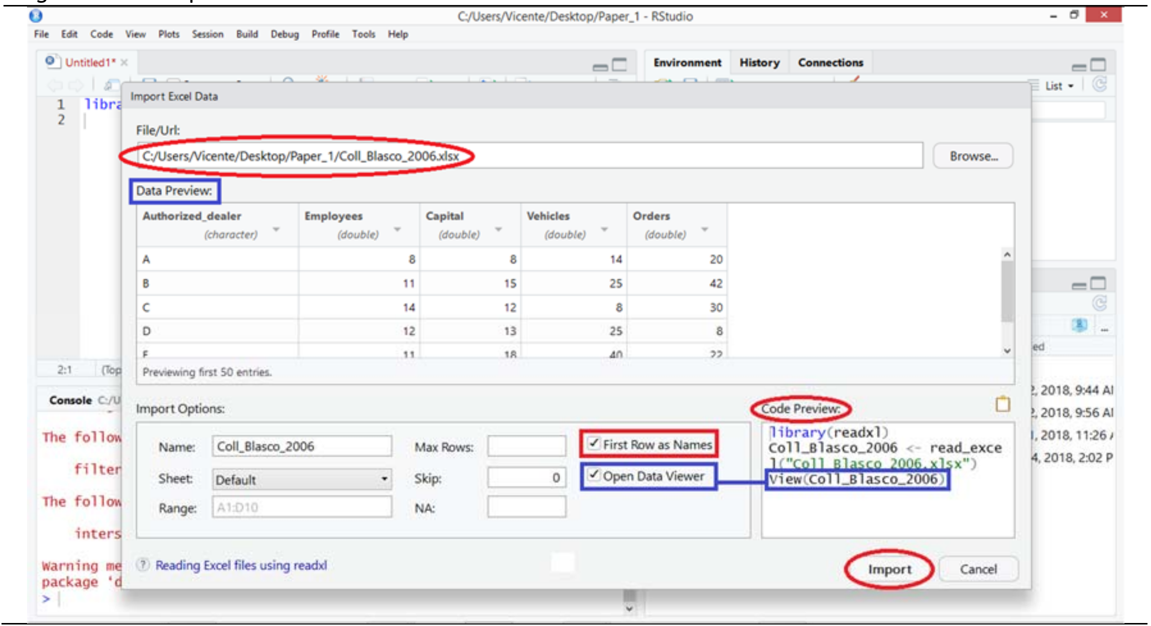


Import Options:

Name: dataset
Sheet: Default
Range: A1:D10

4. Now, we can previsualize "*Coll_Blasco_2006.xlsx*" and the import data options used (see left side in Figure 17). As you can see in the Figure below, the R code used to import the data is shown in the bottom right side of this window (see Figure 17). Since the *Open Data Viewer* is selected[3], the dataset will be opened when the import process concludes. For that, we click on the *Import* button.

Figure 17. Data preview.


C:/Users/Vicente/Desktop/Paper_1 - RStudio
Import Excel Data
Browse...
Code Preview:
Import
Cancel
Reading Excel files using readxl

File/Url: C:/Users/Vicente/Desktop/Paper_1/Coll_Blasco_2006.xlsx

Data Preview:

Authorized_dealer (character)	Employees (double)	Capital (double)	Vehicles (double)	Orders (double)
A	8	8	14	20
B	11	15	25	42
C	14	12	8	30
D	12	13	25	8
E	11	18	40	22

Previewing first 50 entries.

Import Options:

Name: Coll_Blasco_2006
Sheet: Default
Range: A1:D10
Max Rows:
Skip: 0
NA:
✓ First Row as Names
✓ Open Data Viewer

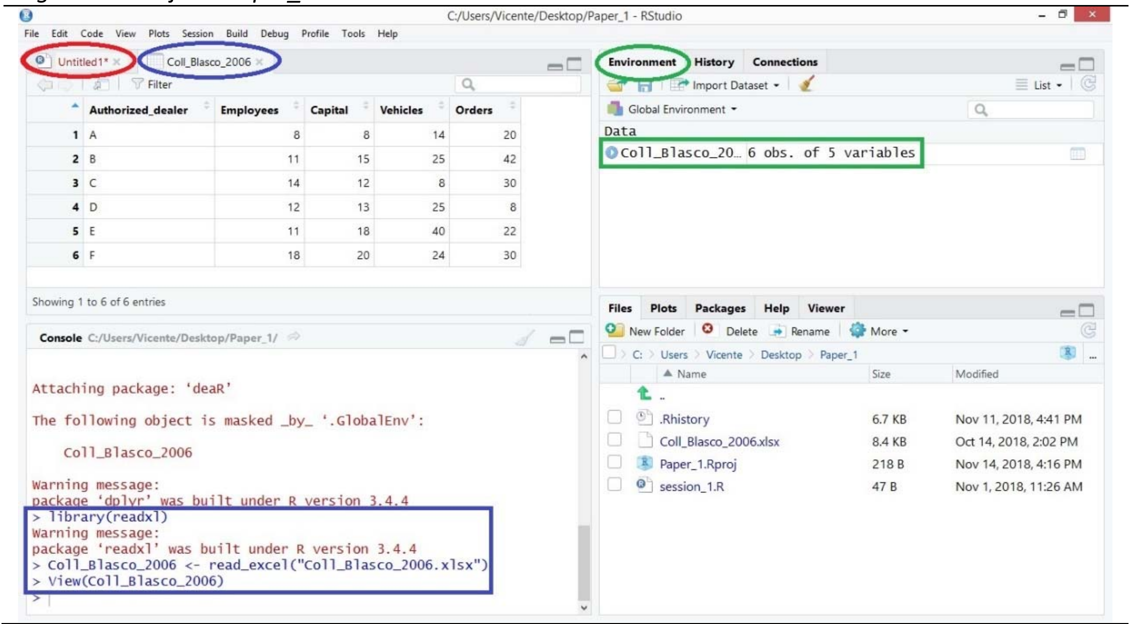
```
library(readxl)
Coll_Blasco_2006 <- read_exce
l("Coll_Blasco_2006.xlsx")
View(Coll_Blasco_2006)
```

[2] Coll-Serrano, V.; Blasco-Blasco, O. (2006). *Evaluación de la Eficiencia mediante el Análisis Envolvente de Datos. Introducción a los Modelos Básicos*. www.eumed.net/libros/2006c/197/

[3] It corresponds to the R code: **View(Coll_Blasco_2006)**

5. After clicking on the *Import* button, we come back to the main window of the project "*Paper_1*". The imported data are shown in a new sheet (it has the same name as the dataset) (see Figure 18). Besides, in the *Environment* menu (top right panel) appears a list of all objects[4] that we are creating in R. Right now, we only have one object: "*Coll_Blasco_2006*". Actually, this object is a dataframe with 6 observations and 5 variables.

Figure 18. Project "*Paper_1*".


C:/Users/Vicente/Desktop/Paper_1 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
Coll_Blasco_2006
Filter
Environment History Connections
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
New Folder Delete Rename More
C: > Users > Vicente > Desktop > Paper_1
Console C:/Users/Vicente/Desktop/Paper_1/

	Authorized_dealer	Employees	Capital	Vehicles	Orders
1	A	8	8	14	20
2	B	11	15	25	42
3	C	14	12	8	30
4	D	12	13	25	8
5	E	11	18	40	22
6	F	18	20	24	30

Showing 1 to 6 of 6 entries

```
Data
Coll_Blasco_20… 6 obs. of 5 variables
```

```
Attaching package: 'deaR'

The following object is masked _by_ '.GlobalEnv':

    Coll_Blasco_2006

Warning message:
package 'dplyr' was built under R version 3.4.4
> library(readxl)
Warning message:
package 'readxl' was built under R version 3.4.4
> Coll_Blasco_2006 <- read_excel("Coll_Blasco_2006.xlsx")
> View(Coll_Blasco_2006)
>
```

Name	Size	Modified
..		
.Rhistory	6.7 KB	Nov 11, 2018, 4:41 PM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
Paper_1.Rproj	218 B	Nov 14, 2018, 4:16 PM
session_1.R	47 B	Nov 1, 2018, 11:26 AM

We can see the R code used to import the data in the *Console* (see Figure 18):

- **library(readxl)** → load the package: "*readxl*"
- **Coll_Blasco_2006 <- read_excel("Coll_Blasco_2006.xlsx")** → the "*read_excel*" function (from the *readlx* package) reads the data file and assign ("<- ") the data to the object "*Coll_Blasco_2006*".

  Very important:

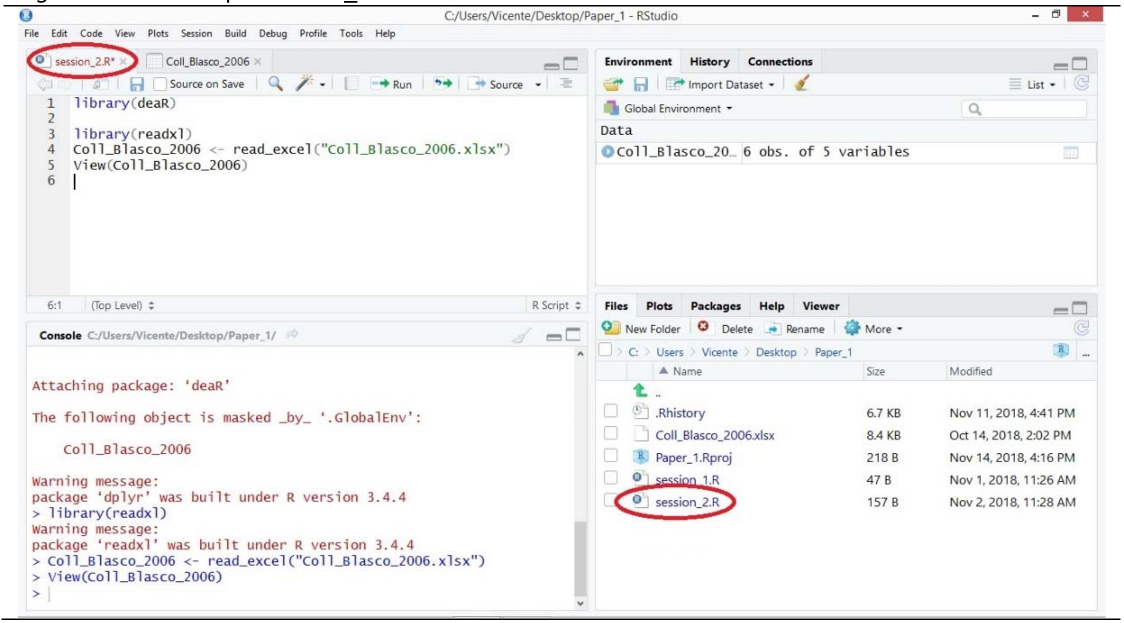
  R terminology: **A <- B** This means that B (whatever B: a dataset, a function, a matrix, the result of a mathematical operation, etc.) is assigned (<-) to the object A.
- **View(Coll_Blasco_2006)** → visualize (View) the object "*Coll_Blasco_2006*".

6. We copy the above code into the script "*Untitled1*" and save it with the name "*session_2*".

Our project should show something like that (see Figure 19).

[4] Everything in R is an object. An object can be a dataset, a function, an instruction, a matrix, etc.

Figure 19. Save script "session_2".



C:/Users/Vicente/Desktop/Paper_1 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
session_2.R*
Coll_Blasco_2006
Source on Save
Run
Source
Environment History Connections
Import Dataset
List
Global Environment
6:1 (Top Level)
R Script
Console C:/Users/Vicente/Desktop/Paper_1/
Files Plots Packages Help Viewer
New Folder Delete Rename More
C: Users Vicente Desktop Paper_1

```
library(deaR)

library(readxl)
Coll_Blasco_2006 <- read_excel("Coll_Blasco_2006.xlsx")
View(Coll_Blasco_2006)
```

```
Data
Coll_Blasco_20… 6 obs. of 5 variables
```

```
Attaching package: 'deaR'

The following object is masked _by_ '.GlobalEnv':

    Coll_Blasco_2006

Warning message:
package 'dplyr' was built under R version 3.4.4
> library(readxl)
Warning message:
package 'readxl' was built under R version 3.4.4
> Coll_Blasco_2006 <- read_excel("Coll_Blasco_2006.xlsx")
> View(Coll_Blasco_2006)
>
```

Name	Size	Modified
.Rhistory	6.7 KB	Nov 11, 2018, 4:41 PM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
Paper_1.Rproj	218 B	Nov 14, 2018, 4:16 PM
session_1.R	47 B	Nov 1, 2018, 11:26 AM
session_2.R	157 B	Nov 2, 2018, 11:28 AM

It will not be necessary to repeat steps 1-5 on all subsequent sessions because we have the whole code in the file "*session_2.R*". To import the data from "*Coll_Blasco_2006.xlsx*" again, we only have to execute the code in "*session_2.R*".

deaR also provides a large collection of datasets. These datasets come from articles already published and they are used to replicate the results of the articles. We think this is a great added value that **deaR** offers both researchers and practitioners, since it can be an important help and support for teaching and learning DEA methodology. We can see the structure and source of the data by getting help from **deaR**[5].

Let's see how to load a dataset in Example 2.

Example 2. Loading datasets available in **deaR**.

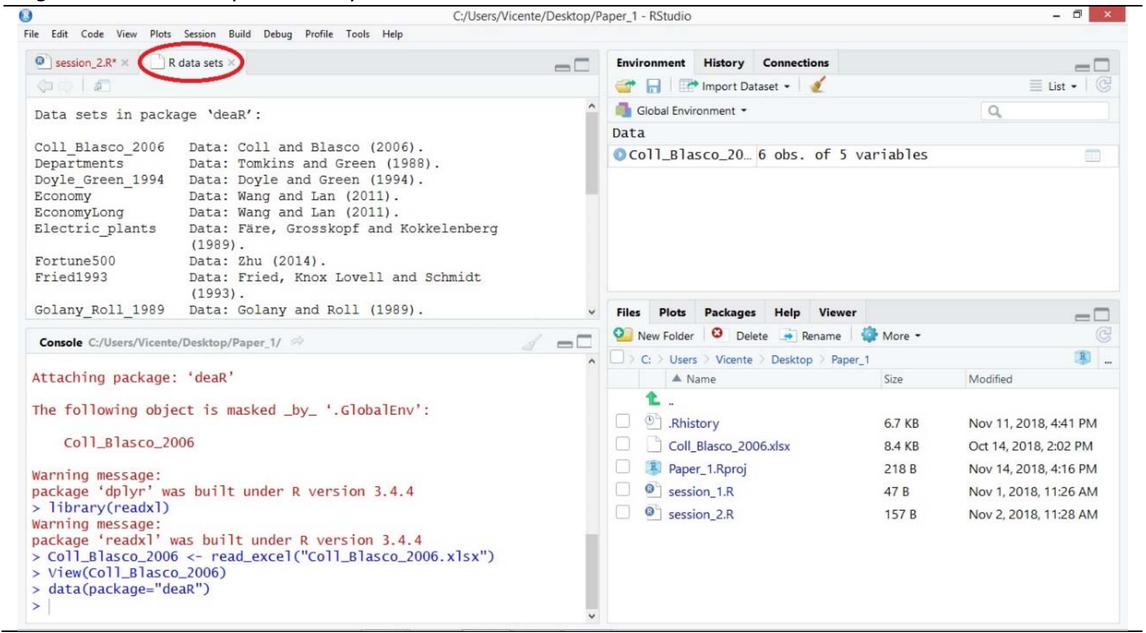
To see what datasets are available in **deaR**, we write

```
data(package="deaR")
```

and run the instruction.

[5] Getting help with R: https://www.r-project.org/help.html

Figure 20. Datasets proviede by **deaR**.


C:/Users/Vicente/Desktop/Paper_1 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
session_2.R*
R data sets
Environment History Connections
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
New Folder Delete Rename More
C: > Users > Vicente > Desktop > Paper_1
Console C:/Users/Vicente/Desktop/Paper_1/

```
Data sets in package 'deaR':

Coll_Blasco_2006     Data: Coll and Blasco (2006).
Departments          Data: Tomkins and Green (1988).
Doyle_Green_1994     Data: Doyle and Green (1994).
Economy              Data: Wang and Lan (2011).
EconomyLong          Data: Wang and Lan (2011).
Electric_plants      Data: Färe, Grosskopf and Kokkelenberg
                     (1989).
Fortune500           Data: Zhu (2014).
Fried1993            Data: Fried, Knox Lovell and Schmidt
                     (1993).
Golany_Roll_1989     Data: Golany and Roll (1989).
```

```
Data
Coll_Blasco_20… 6 obs. of 5 variables
```

Name	Size	Modified
..		
.Rhistory	6.7 KB	Nov 11, 2018, 4:41 PM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
Paper_1.Rproj	218 B	Nov 14, 2018, 4:16 PM
session_1.R	47 B	Nov 1, 2018, 11:26 AM
session_2.R	157 B	Nov 2, 2018, 11:28 AM

```
Attaching package: 'deaR'

The following object is masked _by_ '.GlobalEnv':

    Coll_Blasco_2006

Warning message:
package 'dplyr' was built under R version 3.4.4
> library(readxl)
Warning message:
package 'readxl' was built under R version 3.4.4
> Coll_Blasco_2006 <- read_excel("Coll_Blasco_2006.xlsx")
> View(Coll_Blasco_2006)
> data(package="deaR")
> 
```

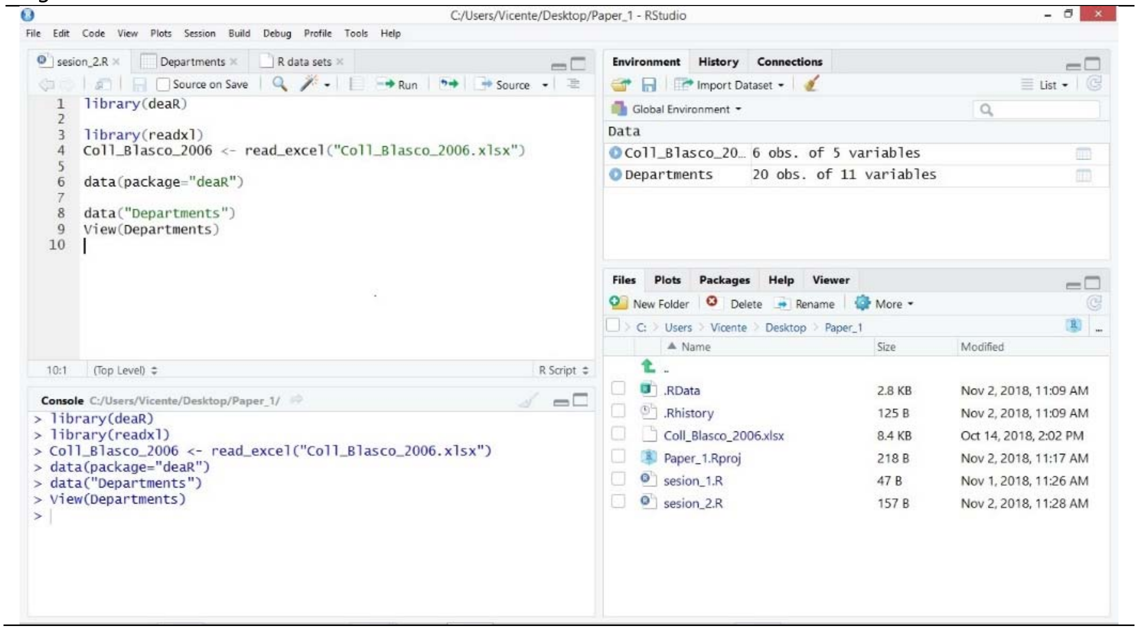
To load a dataset we use the **data()** function. For example, we are going to load the data from Tomkins and Green (1988)[6]. The name of this dataset is "*Departments*". Therefore, we write into the script "*session2.R*" (see Figure 21):

data("Departments")

If we want to visualize the dataset, we write (see Figure 21):

View(Departments)

Figure 21. Load and view data.


C:/Users/Vicente/Desktop/Paper_1 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
sesion_2.R
Departments
R data sets
Source on Save
Run
Source
10:1
(Top Level)
R Script
Environment History Connections
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
New Folder Delete Rename More
C: > Users > Vicente > Desktop > Paper_1
Console C:/Users/Vicente/Desktop/Paper_1/

```
library(deaR)

library(readxl)
Coll_Blasco_2006 <- read_excel("Coll_Blasco_2006.xlsx")

data(package="deaR")

data("Departments")
View(Departments)
```

```
Data
Coll_Blasco_20… 6 obs. of 5 variables
Departments     20 obs. of 11 variables
```

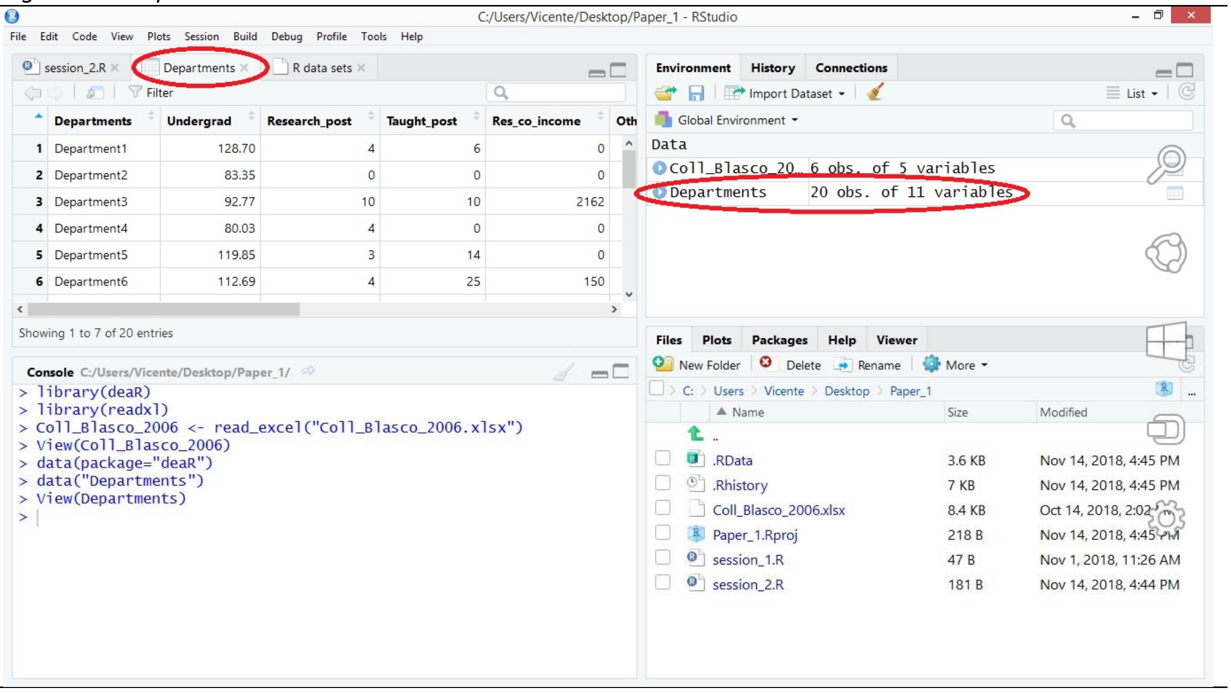
Name	Size	Modified
..		
.RData	2.8 KB	Nov 2, 2018, 11:09 AM
.Rhistory	125 B	Nov 2, 2018, 11:09 AM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
Paper_1.Rproj	218 B	Nov 2, 2018, 11:17 AM
sesion_1.R	47 B	Nov 1, 2018, 11:26 AM
sesion_2.R	157 B	Nov 2, 2018, 11:28 AM

```
> library(deaR)
> library(readxl)
> Coll_Blasco_2006 <- read_excel("Coll_Blasco_2006.xlsx")
> data(package="deaR")
> data("Departments")
> View(Departments)
> 
```

We execute the instructions by clicking on Run (see Figure 22).

[6] Tomkins, C.; Green, R. (1988). "An Experiment in the Use of Data Envelopment Analysis for Evaluating the Efficiency of UK University Departments of Accounting", Financial Accountability and Management, 4(2), 147-164. https://doi.org/10.1111/j.1468-0408.1988.tb00296.x

Figure 22. "Departments".


C:/Users/Vicente/Desktop/Paper_1 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
session_2.R
Departments
R data sets
Filter
Environment History Connections
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
New Folder Delete Rename More

	Departments	Undergrad	Research_post	Taught_post	Res_co_income	Oth
1	Department1	128.70	4	6	0	
2	Department2	83.35	0	0	0	
3	Department3	92.77	10	10	2162	
4	Department4	80.03	4	0	0	
5	Department5	119.85	3	14	0	
6	Department6	112.69	4	25	150	

Showing 1 to 7 of 20 entries

Console C:/Users/Vicente/Desktop/Paper_1/

```
> library(deaR)
> library(readxl)
> Coll_Blasco_2006 <- read_excel("Coll_Blasco_2006.xlsx")
> View(Coll_Blasco_2006)
> data(package="deaR")
> data("Departments")
> View(Departments)
> |
```

```
Data
Coll_Blasco_20… 6 obs. of 5 variables
Departments     20 obs. of 11 variables
```

C: > Users > Vicente > Desktop > Paper_1

Name	Size	Modified
..		
.RData	3.6 KB	Nov 14, 2018, 4:45 PM
.Rhistory	7 KB	Nov 14, 2018, 4:45 PM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02
Paper_1.Rproj	218 B	Nov 14, 2018, 4:45 PM
session_1.R	47 B	Nov 1, 2018, 11:26 AM
session_2.R	181 B	Nov 14, 2018, 4:44 PM

Notice that we now have two objects: "*Coll_Blasco_2006*" and "*Departments*". "*Departments*" is a dataframe consisting of 20 observations (DMUs) and 11 variables.

Now, **we save "*session_2.R*" and close the project "*Paper_1*". Quit RStudio.**

7.2. Adapting the data to deaR

Once the data have been loaded, the next step is to adapt them to the format that **deaR** uses to read them. **deaR** has three different data reading functions. Each reading function responds to a typology of DEA model. These reading functions are:

Read_data(): if we are going to run a conventional DEA model.

Read_malmquist(): if we want to perform the Malmquist productivity index.

Read_data_fuzzy(): if we want to run a DEA model with uncertain data (fuzzy DEA).

7.2.1. Getting help from deaR[7].

We can learn how to use a specific function by using the help from **deaR**. In the help for **deaR** we can read about the arguments of a function or examples about how to use it. Let's see how to use the help function with Example 3.

Example 3. Getting help for **deaR**.

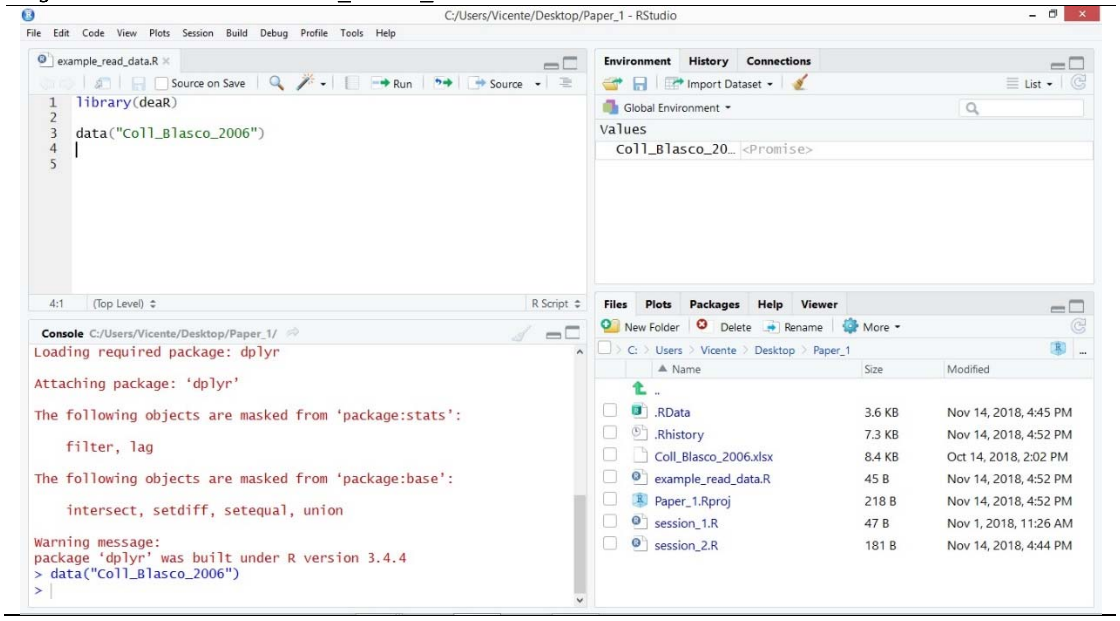
1. We open the project "*Paper_1*" and create a new script. We name the script: "*example_read_data*"

2. We load **deaR**

3. We load the dataset provided by **deaR**: "*Coll_Blasco_2006*" (see Figure 23).

[7] Getting help with R: https://www.r-project.org/help.html

Figure 23. Load dataset: “Coll_Blasco_2006”



```
library(deaR)

data("Coll_Blasco_2006")
```

```
Loading required package: dplyr

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union

Warning message:
package 'dplyr' was built under R version 3.4.4
> data("Coll_Blasco_2006")
>
```

Values
Coll_Blasco_20... <Promise>

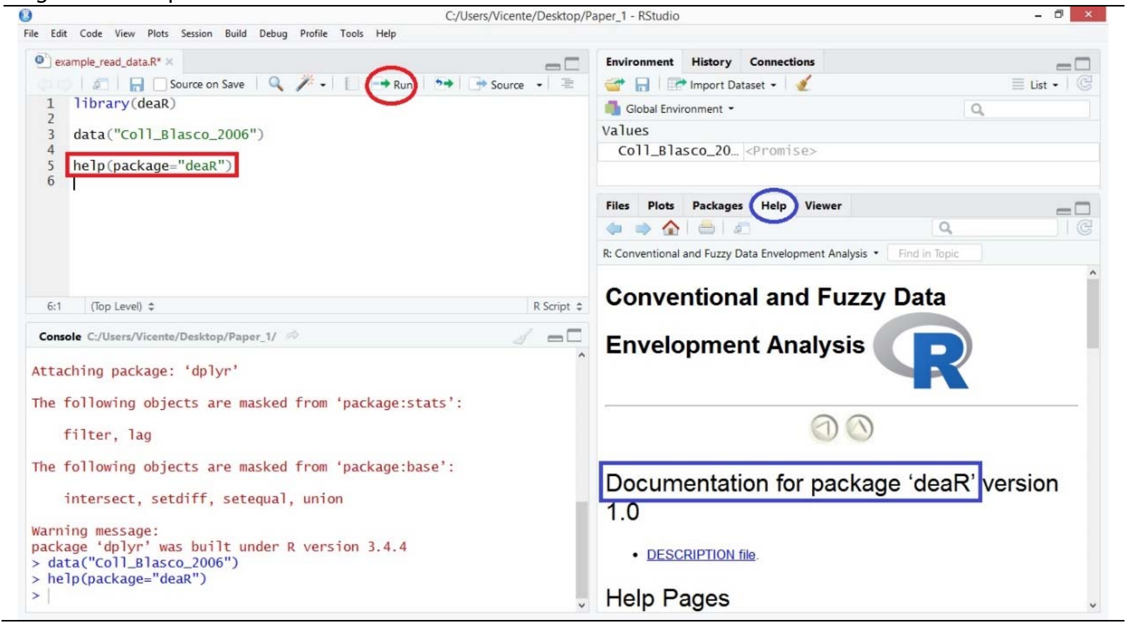
Name	Size	Modified
.RData	3.6 KB	Nov 14, 2018, 4:45 PM
.Rhistory	7.3 KB	Nov 14, 2018, 4:52 PM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
example_read_data.R	45 B	Nov 14, 2018, 4:52 PM
Paper_1.Rproj	218 B	Nov 14, 2018, 4:52 PM
session_1.R	47 B	Nov 1, 2018, 11:26 AM
session_2.R	181 B	Nov 14, 2018, 4:44 PM

To access the documentation for a function of the package **deaR** we use the **help()** function. We write:

help(package="deaR")

In the *Help* menu (see bottom right window) a list of all functions and datasets in **deaR** will appear. It is the documentation for the package (see Figure 24).

Figure 24. Help for **deaR**.



```
library(deaR)

data("Coll_Blasco_2006")

help(package="deaR")
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union

Warning message:
package 'dplyr' was built under R version 3.4.4
> data("Coll_Blasco_2006")
> help(package="deaR")
>
```

R: Conventional and Fuzzy Data Envelopment Analysis

Conventional and Fuzzy Data Envelopment Analysis

Documentation for package 'deaR' version 1.0

- DESCRIPTION file.

Help Pages

7.2.2. read_data() function:

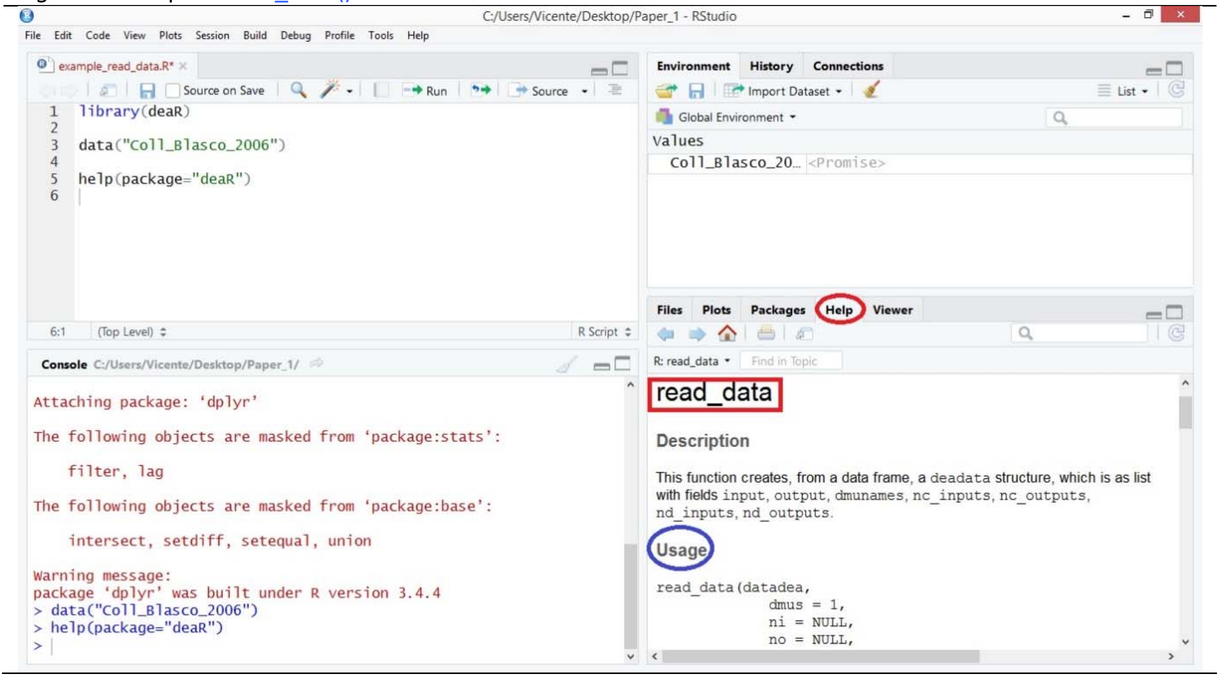
Once we have accessed the documentation (help) for **deaR**, if we clic on the "*read_data*" link we will get the specific help about this function. We can also get the same result if we write:

help(read_data) (or **?read_data**)

into the script and run the instruction.

All the arguments of the read_data() function are listed in the *Usage* section, and in the *Arguments* section we can read a brief explanation of them (see Figure 25).

Figure 25. Help for read_data() function.


C:/Users/Vicente/Desktop/Paper_1 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
example_read_data.R*
Source on Save
Run
Source
Environment History Connections
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
R: read_data
Find in Topic
6:1 (Top Level)
R Script
Console C:/Users/Vicente/Desktop/Paper_1/

```
library(deaR)

data("Coll_Blasco_2006")

help(package="deaR")
```

```
Values
  Coll_Blasco_20... <Promise>
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union

Warning message:
package 'dplyr' was built under R version 3.4.4
> data("Coll_Blasco_2006")
> help(package="deaR")
>
```

read_data

Description

This function creates, from a data frame, a `deadata` structure, which is as list with fields `input`, `output`, `dmunames`, `nc_inputs`, `nc_outputs`, `nd_inputs`, `nd_outputs`.

Usage

```
read_data(datadea,
          dmus = 1,
          ni = NULL,
          no = NULL,
```

The read_data() function takes the following arguments:

- *datadea*: It is the dataset (It must be a dataframe).
- *dmus*: Number of the column where the DMUs are. By default, **deaR** considers DMUs are in the first column of the dataset.
- *ni*: It is the number of inputs.
- *no*: It is the number of outputs.
- *inputs*: We can indicate the numbers of the columns where the inputs are instead of indicating the number of inputs.
- *Outputs*: We can indicate the numbers of the columns where the outputs are instead of indicating the number of outputs.
- *nc_inputs*: If we have non-controllable inputs, we can indicate which are them.
- *nc_outputs*: If we have non-controllable outputs, we can indicate which are them.
- *nd_inputs*: If we have non-discretionary inputs, we can indicate which are them.
- *nd_outputs*: If we have non-discretionary outputs, we can indicate which are them.
- *ud_inputs*: If we have undesirable (bad) inputs, we can indicate which are them.

- *ud_outputs*: If we have undesirable (bad) outputs, we can indicate which are them.

Example 4 shows how to use the read_data() function. The documentation for **deaR** provides examples for the all functions in the package.

Example 4. Using the read_data() function.

Currently, we have loaded the dataset "*Coll_Blasco_2006*". This dataset has 6 DMUs (column 1), 2 inputs (columns 2 and 3) and 2 outputs (columns 4 and 5).

Suppose that we want to measure the efficiency of these DMUs using the BCC DEA model.

As BCC model is a conventional DEA model (it is not a fuzzy DEA model), we must use the read_data() function to adapt the original dataset to the format used by **deaR**.

We write the following text into the script "*example_read_data*" (see Figure 26):

data_example <- read_data(Coll_Blasco_2006, ni=2, no=2)

Very important: Reading and understanding the instruction.

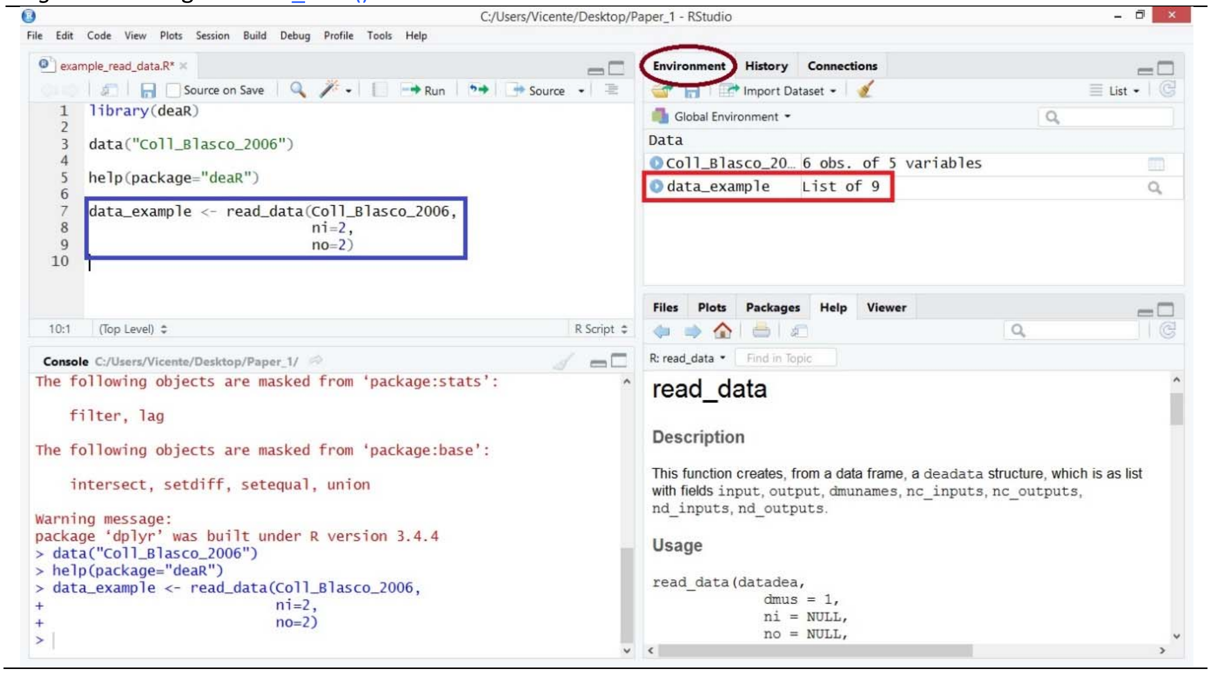
The expression on the right side of the assignment symbol (<-) means: read the data (read_data) in Coll_Blasco_2006; it has 2 inputs (ni=2) and 2 outputs(no=2). DMUs are in column 1 (dmus=1). This argument doesn't appear since it is the default value.

Then, the result of this function is assigned (<-) to the object *data_example*.

We execute the instruction (Run).

Notice that now the objects "*Coll_Blasco_2006*" and "*data_example*" appear listed in the *Environment* menu. Notice also that "*data_example*" is a list of 9 components. (see Figure 26).

Figure 26. Using the read_data() function.



```
library(deaR)

data("Coll_Blasco_2006")

help(package="deaR")

data_example <- read_data(Coll_Blasco_2006,
                          ni=2,
                          no=2)
```

```
Data
Coll_Blasco_20…  6 obs. of 5 variables
data_example     List of 9
```

```
The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union

Warning message:
package 'dplyr' was built under R version 3.4.4
> data("Coll_Blasco_2006")
> help(package="deaR")
> data_example <- read_data(Coll_Blasco_2006,
+                           ni=2,
+                           no=2)
> 
```

read_data

Description

This function creates, from a data frame, a deadata structure, which is as list with fields input, output, dmunames, nc_inputs, nc_outputs, nd_inputs, nd_outputs.

Usage

```
read_data(datadea,
          dmus = 1,
          ni = NULL,
          no = NULL,
```

Now, the data are ready to be used with a conventional DEA model (see section 7.3.). For that, we should use the object we have created: "*data_example*".

Recommendation: Practice the function read_data() with the examples in the documentation.

Save "*example_read_data.R*".

7.2.3. read_malmquist() function.

If we have time series data and we want to measure the Malmquist productivity index, we must use the read_malmquist() function first to adapt them to the format that **deaR** reads.

deaR admits the time series data in two different formats:

- **Wide format**: DMUs by column. Inputs and outputs of different periods of time by column. For example, see the dataset "*Economy*"[8] in **deaR** (Figure 27).
- **Long format**: Time by column. DMUs, inputs and outputs by column, but grouped by time. For example, see the dataset "*EconomyLong*" in **deaR** (Figure 28).

In the following example we try to show the above explanation.

Example 5. Wide and Long data formats.

Now, we are going to show these different data formats. For that:

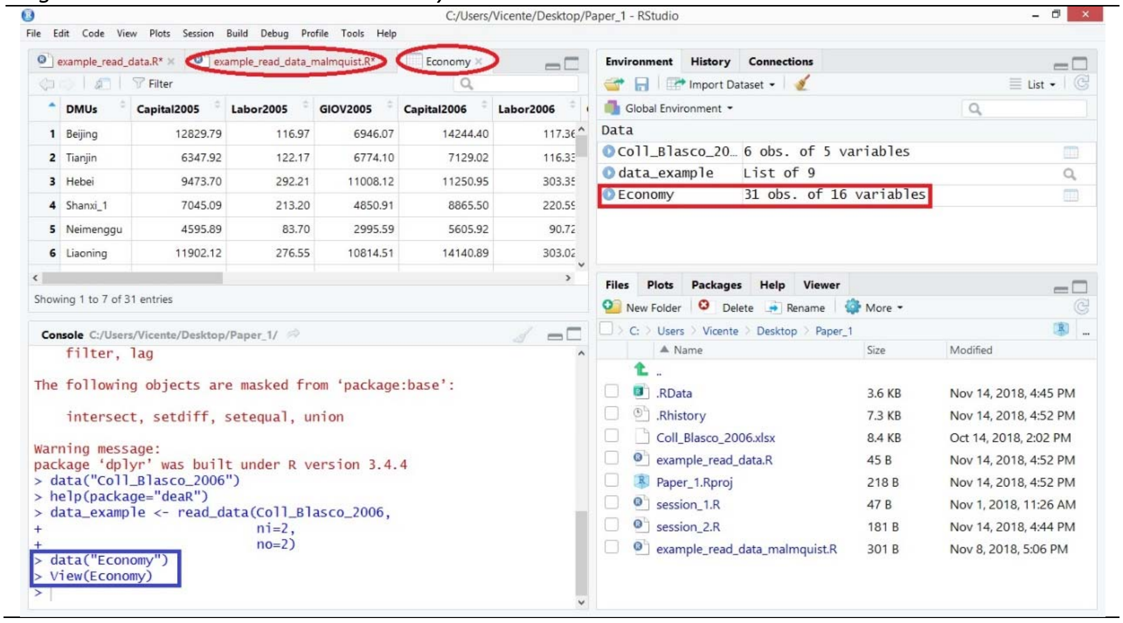
1. We create a new script. Name it: "*example_read_malmquist*".

   Note: If we have closed the working session, we open the project "*Paper_1*" and then create the new script. We also have to load **deaR**.

2. We load the dataset in **deaR**: "*Economy*" (see Figure 27).
3. Visualize "*Economy*". This object (dataset) has 31 DMUs with 2 inputs (Capital and Labor) and 1 output (GIOV) for 5 years (from 2005 to 2009). "*Economy*" is a dataset in wide format.

[8] Wang, Y.; Lan, Y. (2011). "Measuring Malmquist Productiviy Index: A New Approach Based on Double Frontiers Data Envelopment Analysis". Mathematical and Computer Modelling, 54, 2760-2771. https://doi.org/10.1016/j.mcm.2011.06.064

Figure 27. Wide data format: "*Economy*".


C:/Users/Vicente/Desktop/Paper_1 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
example_read_data.R*
example_read_data_malmquist.R*
Economy
Filter
Environment History Connections
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
New Folder Delete Rename More
C: > Users > Vicente > Desktop > Paper_1

	DMUs	Capital2005	Labor2005	GIOV2005	Capital2006	Labor2006
1	Beijing	12829.79	116.97	6946.07	14244.40	117.3
2	Tianjin	6347.92	122.17	6774.10	7129.02	116.3
3	Hebei	9473.70	292.21	11008.12	11250.95	303.3
4	Shanxi_1	7045.09	213.20	4850.91	8865.50	220.5
5	Neimenggu	4595.89	83.70	2995.59	5605.92	90.72
6	Liaoning	11902.12	276.55	10814.51	14140.89	303.02

Showing 1 to 7 of 31 entries

Console C:/Users/Vicente/Desktop/Paper_1/

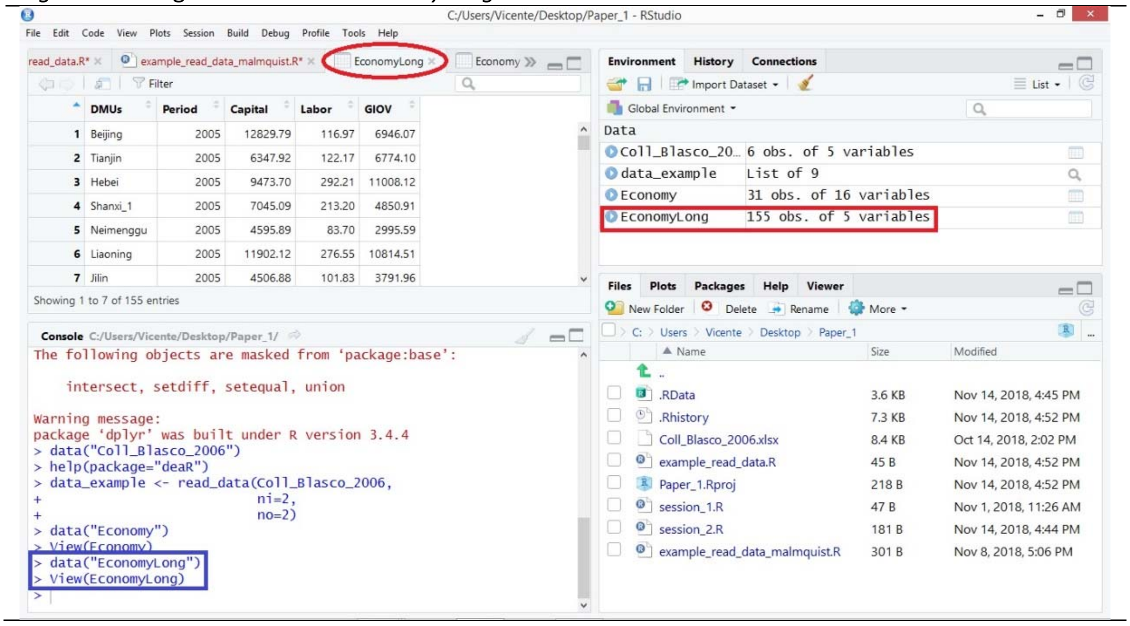
```
    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union

Warning message:
package ‘dplyr’ was built under R version 3.4.4
> data("Coll_Blasco_2006")
> help(package="deaR")
> data_example <- read_data(Coll_Blasco_2006,
+                          ni=2,
+                          no=2)
> data("Economy")
> View(Economy)
>
```

```
Data
Coll_Blasco_20…  6 obs. of 5 variables
data_example     List of 9
Economy          31 obs. of 16 variables
```

Name	Size	Modified
..		
.RData	3.6 KB	Nov 14, 2018, 4:45 PM
.Rhistory	7.3 KB	Nov 14, 2018, 4:52 PM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
example_read_data.R	45 B	Nov 14, 2018, 4:52 PM
Paper_1.Rproj	218 B	Nov 14, 2018, 4:52 PM
session_1.R	47 B	Nov 1, 2018, 11:26 AM
session_2.R	181 B	Nov 14, 2018, 4:44 PM
example_read_data_malmquist.R	301 B	Nov 8, 2018, 5:06 PM

4. Now, load the dataset "EconomyLong".
5. We visualize "*EconomyLong*" (see Figure 28). This new R object (actually, it is a dataset) has 155 observations (31 DMUs x 5 years), with 2 inputs (Capital and Labor) and 1 output (GIOV). The *Period* column refers to the time period 2005-2009.

Figure 28. Long data format: "*EconomyLong*".


C:/Users/Vicente/Desktop/Paper_1 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
read_data.R*
example_read_data_malmquist.R*
EconomyLong
Economy
Filter
Environment History Connections
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
New Folder Delete Rename More
C: > Users > Vicente > Desktop > Paper_1

	DMUs	Period	Capital	Labor	GIOV
1	Beijing	2005	12829.79	116.97	6946.07
2	Tianjin	2005	6347.92	122.17	6774.10
3	Hebei	2005	9473.70	292.21	11008.12
4	Shanxi_1	2005	7045.09	213.20	4850.91
5	Neimenggu	2005	4595.89	83.70	2995.59
6	Liaoning	2005	11902.12	276.55	10814.51
7	Jilin	2005	4506.88	101.83	3791.96

Showing 1 to 7 of 155 entries

Console C:/Users/Vicente/Desktop/Paper_1/

```
The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union

Warning message:
package ‘dplyr’ was built under R version 3.4.4
> data("Coll_Blasco_2006")
> help(package="deaR")
> data_example <- read_data(Coll_Blasco_2006,
+                          ni=2,
+                          no=2)
> data("Economy")
> View(Economy)
> data("EconomyLong")
> View(EconomyLong)
>
```

```
Data
Coll_Blasco_20…  6 obs. of 5 variables
data_example     List of 9
Economy          31 obs. of 16 variables
EconomyLong      155 obs. of 5 variables
```

Name	Size	Modified
..		
.RData	3.6 KB	Nov 14, 2018, 4:45 PM
.Rhistory	7.3 KB	Nov 14, 2018, 4:52 PM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
example_read_data.R	45 B	Nov 14, 2018, 4:52 PM
Paper_1.Rproj	218 B	Nov 14, 2018, 4:52 PM
session_1.R	47 B	Nov 1, 2018, 11:26 AM
session_2.R	181 B	Nov 14, 2018, 4:44 PM
example_read_data_malmquist.R	301 B	Nov 8, 2018, 5:06 PM

Note: Notice that all objects we are creating during the working session are listed in the *Environment* menu (top right window).

The read_malmquist() function takes the following arguments[9]:

- *datadea*: It is the dataset (It must be a dataframe).
- *nper*: It is the number of years (with data in wide format).
- *percol*: It is the number of the column where the variable time is (with data in long format).
- *arrangement*: "horizontal" with data in wide format or "vertical" with data in long format.
- *dmus*: Number of the column where the DMUs are. By default, **deaR** considers DMUs are in the first column of the dataset.
- *ni*: It is the number of inputs.
- *no*: It is the number of outputs.
- *inputs*: We can indicate the numbers of the columns where the inputs are instead of indicating the number of inputs.
- *Outputs*: We can indicate the numbers of the columns where the outputs are instead of indicating the number of outputs.
- *nc_inputs*: If we have non-controllable inputs, we can indicate which are them.
- *nc_outputs*: If we have non-controllable outputs, we can indicate which are them.
- *nd_inputs*: If we have non-discretionary inputs, we can indicate which are them.
- *nd_outputs*: If we have non-discretionary outputs, we can indicate which are them.
- *ud_inputs*: If we have undesirable (bad) inputs, we can indicate which are them.
- *ud_outputs*: If we have undesirable (bad) outputs, we can indicate which are them.

The current version of **deaR** cannot manage undesirable inputs/outputs to calculate the Malmquist productivity index. This feature will be incorporated in a future version.

In Examples 6 and 7 we can see how to use the read_malmquist() function with wide and long data respectively.

Example 6. read_malmquist() function with wide data format.

For this example, we are going to use the dataset "*Economy*". This dataset is already loaded in the current working session.

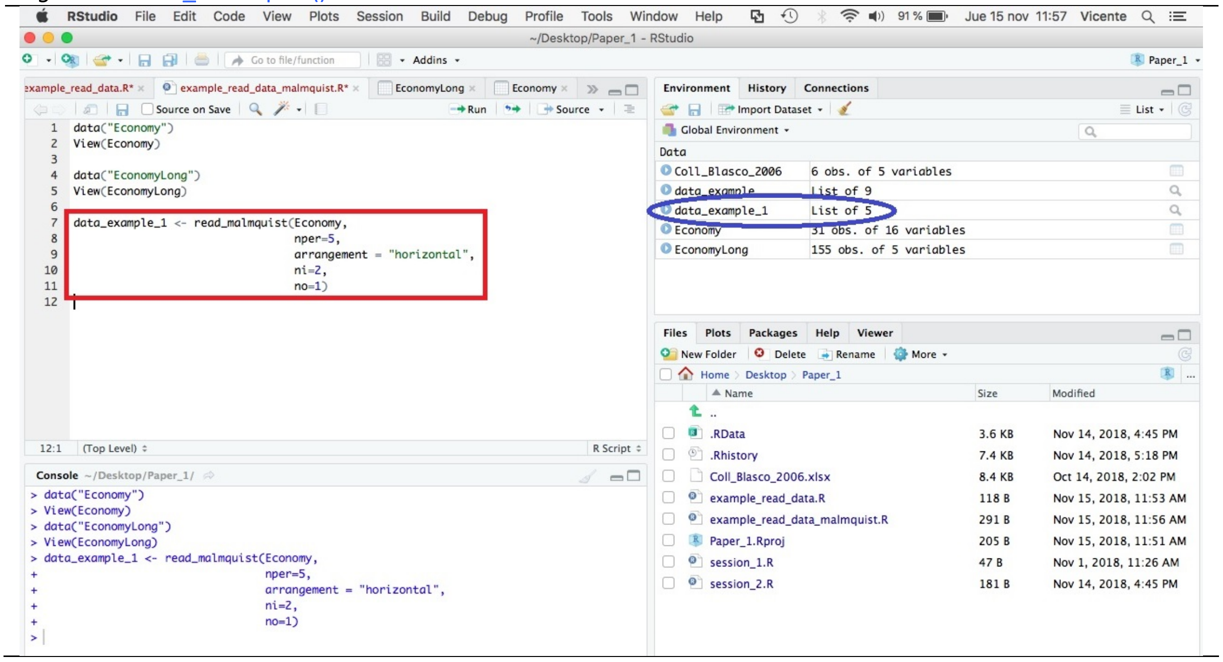
To adapt "*Economy*" to the reading format used by **deaR**, we write into the script ("*example_read_malmquist*") the following text:

```
data_example_1 <- read_malmquist(Economy,
                                 nper=5,
                                 arrangement=”horizontal”,
                                 ni=2,
                                 no=1)
```

[9] See **help(read_malmquist)**.

By running the instruction, a new object is created ("*data_example_1*") and listed in the *Environment* (see Figure 29).

Figure 29. read_malmquist() funtion with wide data format.



```
data("Economy")
View(Economy)

data("EconomyLong")
View(EconomyLong)

data_example_1 <- read_malmquist(Economy,
                                 nper=5,
                                 arrangement = "horizontal",
                                 ni=2,
                                 no=1)
```

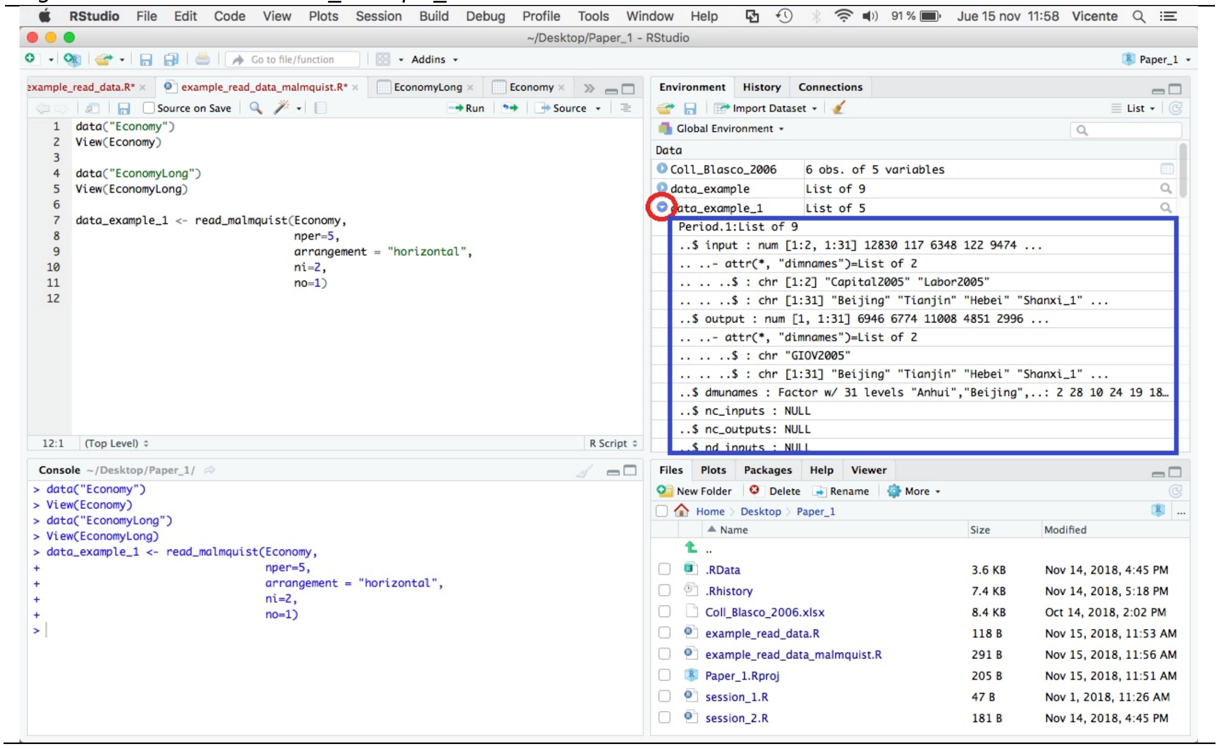
```
Data
Coll_Blasco_2006    6 obs. of 5 variables
data_example        List of 9
data_example_1      List of 5
Economy             31 obs. of 16 variables
EconomyLong         155 obs. of 5 variables
```

```
> data("Economy")
> View(Economy)
> data("EconomyLong")
> View(EconomyLong)
> data_example_1 <- read_malmquist(Economy,
+                                  nper=5,
+                                  arrangement = "horizontal",
+                                  ni=2,
+                                  no=1)
>
```

Name	Size	Modified
.RData	3.6 KB	Nov 14, 2018, 4:45 PM
.Rhistory	7.4 KB	Nov 14, 2018, 5:18 PM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
example_read_data.R	118 B	Nov 15, 2018, 11:53 AM
example_read_data_malmquist.R	291 B	Nov 15, 2018, 11:56 AM
Paper_1.Rproj	205 B	Nov 15, 2018, 11:51 AM
session_1.R	47 B	Nov 1, 2018, 11:26 AM
session_2.R	181 B	Nov 14, 2018, 4:45 PM

As we can see, "*data_example_1*" is a list of 5 components. If we click on the icon next to the name of the object, we can see the structure of this object (see Figure 30).

Figure 30. Structure of "*data_example_1*".

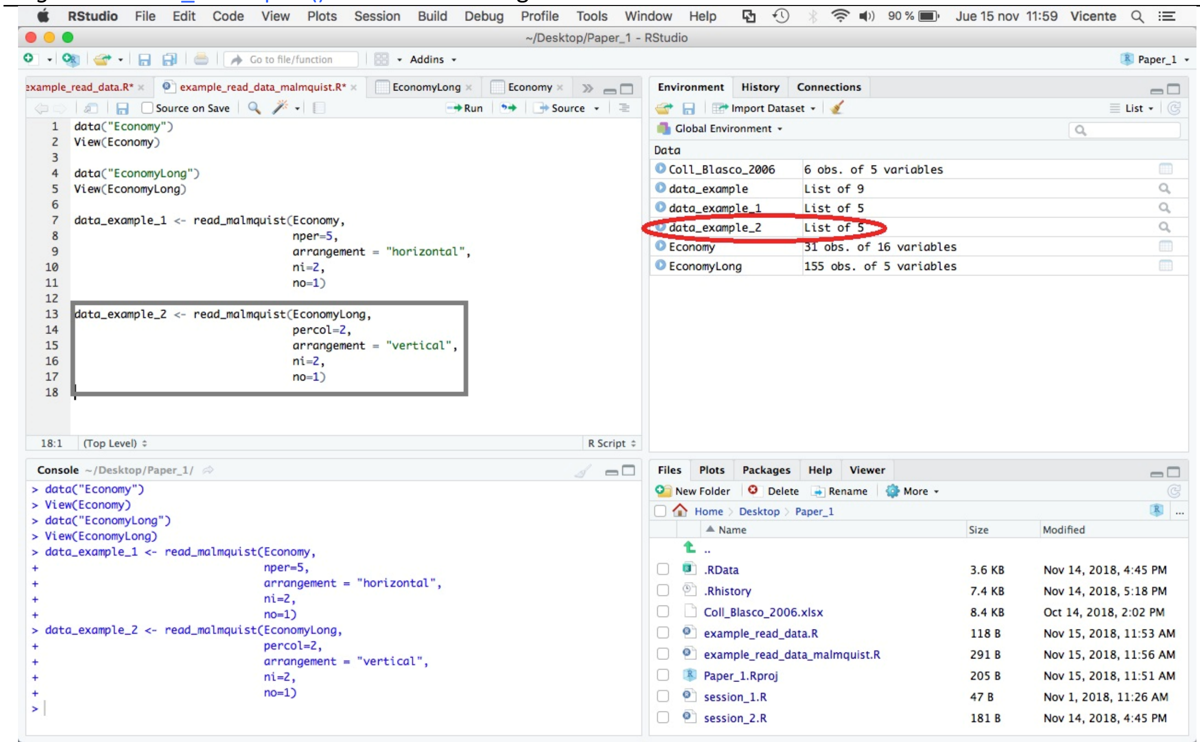


```
Data
Coll_Blasco_2006    6 obs. of 5 variables
data_example        List of 9
data_example_1      List of 5
 Period.1:List of 9
 ..$ input : num [1:2, 1:31] 12830 117 6348 122 9474 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : chr [1:2] "Capital2005" "Labor2005"
 .. .. ..$ : chr [1:31] "Beijing" "Tianjin" "Hebei" "Shanxi_1" ...
 ..$ output : num [1, 1:31] 6946 6774 11008 4851 2996 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : chr "GIOV2005"
 .. .. ..$ : chr [1:31] "Beijing" "Tianjin" "Hebei" "Shanxi_1" ...
 ..$ dmunames : Factor w/ 31 levels "Anhui","Beijing",..: 2 28 10 24 19 18…
 ..$ nc_inputs : NULL
 ..$ nc_outputs: NULL
 ..$ nd_inputs : NULL
```

Example 7. read_malmquist() function with long data format.

Now, we are going to use the dataset "*EconomyLong*". The Figure 31 shows the instruction to adapt the dataset to the reading format in **deaR**.

Figure 31. read_malmquist() function with long data format.



```
data("Economy")
View(Economy)

data("EconomyLong")
View(EconomyLong)

data_example_1 <- read_malmquist(Economy,
                                 nper=5,
                                 arrangement = "horizontal",
                                 ni=2,
                                 no=1)

data_example_2 <- read_malmquist(EconomyLong,
                                 percol=2,
                                 arrangement = "vertical",
                                 ni=2,
                                 no=1)
```

```
> data("Economy")
> View(Economy)
> data("EconomyLong")
> View(EconomyLong)
> data_example_1 <- read_malmquist(Economy,
+                                  nper=5,
+                                  arrangement = "horizontal",
+                                  ni=2,
+                                  no=1)
> data_example_2 <- read_malmquist(EconomyLong,
+                                  percol=2,
+                                  arrangement = "vertical",
+                                  ni=2,
+                                  no=1)
>
```

Data	
Coll_Blasco_2006	6 obs. of 5 variables
data_example	List of 9
data_example_1	List of 5
data_example_2	List of 5
Economy	31 obs. of 16 variables
EconomyLong	155 obs. of 5 variables

Name	Size	Modified
.RData	3.6 KB	Nov 14, 2018, 4:45 PM
.Rhistory	7.4 KB	Nov 14, 2018, 5:18 PM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
example_read_data.R	118 B	Nov 15, 2018, 11:53 AM
example_read_data_malmquist.R	291 B	Nov 15, 2018, 11:56 AM
Paper_1.Rproj	205 B	Nov 15, 2018, 11:51 AM
session_1.R	47 B	Nov 1, 2018, 11:26 AM
session_2.R	181 B	Nov 14, 2018, 4:45 PM

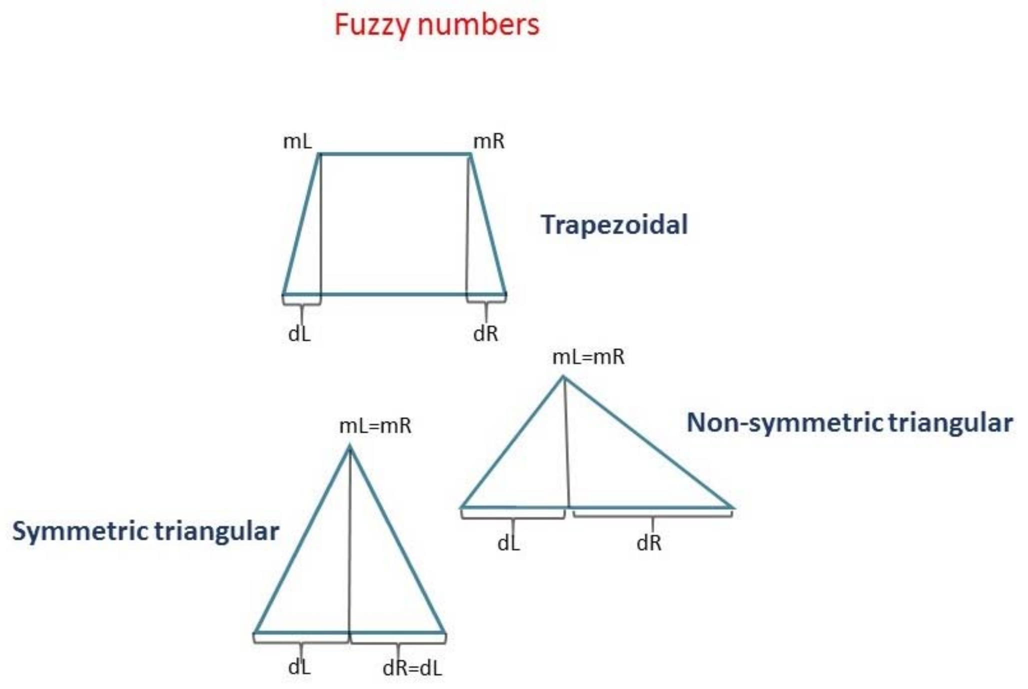
Save "example_read_malmquist.R".

7.2.4. read_data_fuzzy() function.

If we are working with uncertain data and we want to measure the efficiency with a fuzzy DEA model so that **deaR** can read the data, we have to use the read_data_fuzzy() function to adapt them first.

deaR can manage trapezoidal, symmetric triangular, and non-symmetric triangular fuzzy numbers (see Figure 32).

Figure 32. Fuzzy numbers.



The read_data_fuzzy() function takes the following arguments[10]:

- *datadea*: It is the dataset (It must be a dataframe).
- *dmus*: Number of the column where the DMUs are. By default, **deaR** considers DMUs are in the first column of the dataset.
- *Inputs.mL*: Number of the columns where the mL values of the inputs are.
- *Inputs.mR*: Number of the columns where the mR values of the inputs are.
- *Inputs.dL*: Number of the columns where the dL values of the inputs are.
- *Inputs.dR*: Number of the columns where the dR values of the inputs are
- *Outputs.mL*: Number of the columns where the mL. values of the outputs are.
- *Outputs.mR*: Number of the columns where the mR values of the outputs are.
- *Outputs.dL*: Number of the columns where the dL values of the outputs are.
- *Outputs.dR*: Number of the columns where the dR values of the outputs are.
- *nc_inputs*: If we have non-controllable inputs, we can indicate which are them.
- *nc_outputs*: If we have non-controllable outputs, we can indicate which are them.
- *nd_inputs*: If we have non-discretionary inputs, we can indicate which are them.
- *nd_outputs*: If we have non-discretionary outputs, we can indicate which are them.
- *ud_inputs*: If we have undesirable (bad) inputs, we can indicate which are them.

[10] See **help(read_data_fuzzy)**.

- *ud_outputs*: If we have undesirable (bad) outputs, we can indicate which are them.

Currently, **deaR** only considers the undesirable inputs/outputs in the fuzzy DEA models based on the BCC DEA model.

Let's see it in Example 8.

Example 8. read_data_fuzzy() function.

1. We create a new script. Name it: "*example_read_data_fuzzy*".

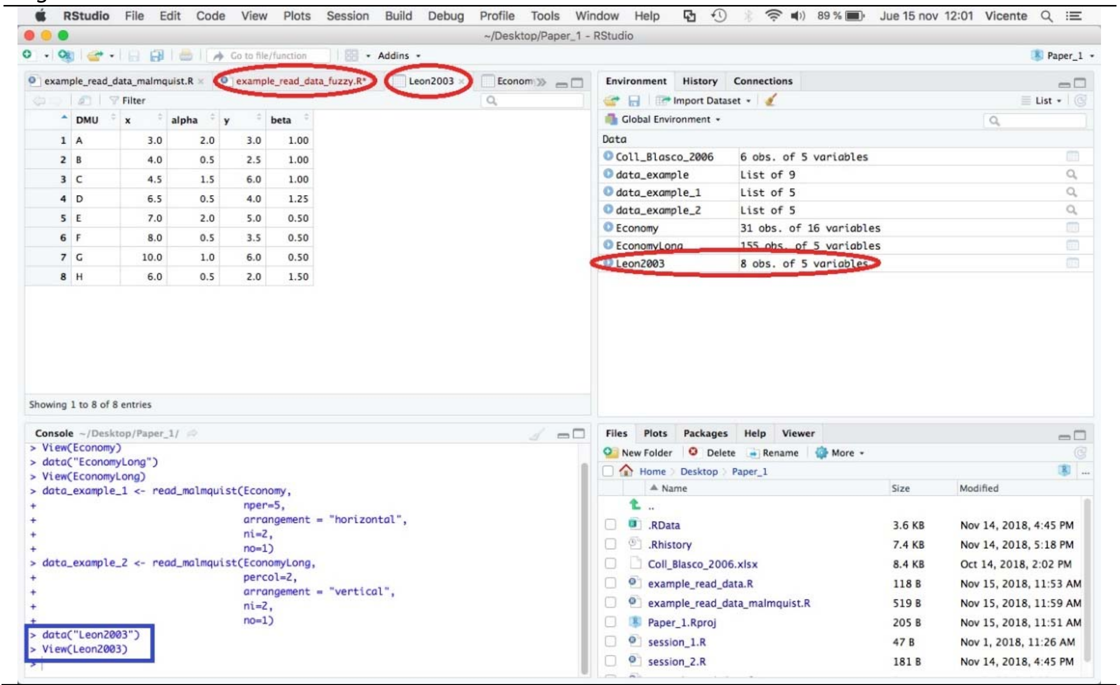
   Note: If we have closed the working session, we open the project "*Paper_1*" and then create the new script. We also have to load **deaR**.

2. We load the dataset "*Leon2003*"[11]:

   data("Leon2003").

   This dataset (see Figure 33) has 8 DMUs with 1 symmetric triangular fuzzy input (alpha is the spread of the input) and 1 symmetric triangular fuzzy output (beta is the spread of the output).

Figure 33. Dataset: Leon2003.


RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
89 % Jue 15 nov 12:01 Vicente
~/Desktop/Paper_1 - RStudio
Go to file/function
Addins
Paper_1
example_read_data_malmquist.R
example_read_data_fuzzy.R*
Leon2003
Econom
Filter
Environment
History
Connections
Import Dataset
List
Global Environment
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
Desktop
Paper_1

	DMU	x	alpha	y	beta
1	A	3.0	2.0	3.0	1.00
2	B	4.0	0.5	2.5	1.00
3	C	4.5	1.5	6.0	1.00
4	D	6.5	0.5	4.0	1.25
5	E	7.0	2.0	5.0	0.50
6	F	8.0	0.5	3.5	0.50
7	G	10.0	1.0	6.0	0.50
8	H	6.0	0.5	2.0	1.50

Showing 1 to 8 of 8 entries

Data	
Coll_Blasco_2006	6 obs. of 5 variables
data_example	List of 9
data_example_1	List of 5
data_example_2	List of 5
Economy	31 obs. of 16 variables
EconomyLong	155 obs. of 5 variables
Leon2003	8 obs. of 5 variables

Console ~/Desktop/Paper_1/

```
> View(Economy)
> data("EconomyLong")
> View(EconomyLong)
> data_example_1 <- read_malmquist(Economy,
+                                  nper=5,
+                                  arrangement = "horizontal",
+                                  ni=2,
+                                  no=1)
> data_example_2 <- read_malmquist(EconomyLong,
+                                  percol=2,
+                                  arrangement = "vertical",
+                                  ni=2,
+                                  no=1)
> data("Leon2003")
> View(Leon2003)
>
```

Name	Size	Modified
..		
.RData	3.6 KB	Nov 14, 2018, 4:45 PM
.Rhistory	7.4 KB	Nov 14, 2018, 5:18 PM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
example_read_data.R	118 B	Nov 15, 2018, 11:53 AM
example_read_data_malmquist.R	519 B	Nov 15, 2018, 11:59 AM
Paper_1.Rproj	205 B	Nov 15, 2018, 11:51 AM
session_1.R	47 B	Nov 1, 2018, 11:26 AM
session_2.R	181 B	Nov 14, 2018, 4:45 PM

3. Now, we read_data_fuzzy() to adapt the data. For that, we write into the script:

```
data_example_3 <- read_data_fuzzy(Leon2003,
                                  inputs.mL=2,
                                  inputs.dL=3,
                                  outputs.mL=4,
                                  outputs.dL=5)
```

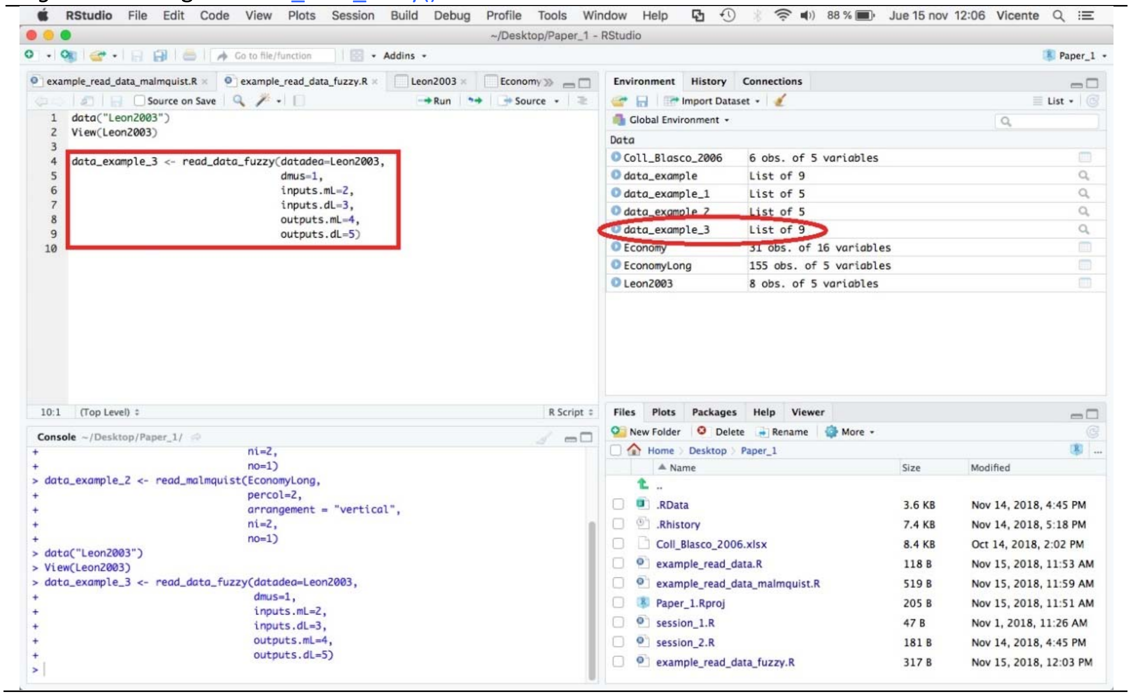
   and run it.

---

[11] León, T.; Liern, V. Ruiz, J.; Sirvent, I. (2003). "A Possibilistic Programming Approach to the Assessment of Efficiency with DEA Models", Fuzzy Sets and Systems, 139, 407–419.https://doi.org/10.1016/S0165-0114(02)00608-5

Our working session should look like the screenshot below (Figure 34):

Figure 34. Using the read_data_fuzzy() function.



```
data("Leon2003")
View(Leon2003)

data_example_3 <- read_data_fuzzy(datadea=Leon2003,
                                  dmus=1,
                                  inputs.mL=2,
                                  inputs.dL=3,
                                  outputs.mL=4,
                                  outputs.dL=5)
```

```
+                         ni=2,
+                         no=1)
> data_example_2 <- read_malmquist(EconomyLong,
+                         percol=2,
+                         arrangement = "vertical",
+                         ni=2,
+                         no=1)
> data("Leon2003")
> View(Leon2003)
> data_example_3 <- read_data_fuzzy(datadea=Leon2003,
+                          dmus=1,
+                          inputs.mL=2,
+                          inputs.dL=3,
+                          outputs.mL=4,
+                          outputs.dL=5)
> |
```

Data	
Coll_Blasco_2006	6 obs. of 5 variables
data_example	List of 9
data_example_1	List of 5
data_example_2	List of 5
data_example_3	List of 9
Economy	31 obs. of 16 variables
EconomyLong	155 obs. of 5 variables
Leon2003	8 obs. of 5 variables

Name	Size	Modified
..		
.RData	3.6 KB	Nov 14, 2018, 4:45 PM
.Rhistory	7.4 KB	Nov 14, 2018, 5:18 PM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
example_read_data.R	118 B	Nov 15, 2018, 11:53 AM
example_read_data_malmquist.R	519 B	Nov 15, 2018, 11:59 AM
Paper_1.Rproj	205 B	Nov 15, 2018, 11:51 AM
session_1.R	47 B	Nov 1, 2018, 11:26 AM
session_2.R	181 B	Nov 14, 2018, 4:45 PM
example_read_data_fuzzy.R	317 B	Nov 15, 2018, 12:03 PM

In this example, inputs.dL=alpha and ouputs.dL=beta. Since we have symmetric triangular fuzzy numbers: inputs.dL=inputs.dR and outputs.dL=outputs.dR

We have created a new R object: "*data_example_3*". This object will be the one we will use to run a given fuzzy DEA model.

We save "*example_read_data_fuzzy.R*", close the project and quit RStudio.

7.3. Select and run a DEA model.

Once we have adapted the data to the reading format in **deaR**, the next step is to select the DEA model and run it.

In the current version of **deaR** (version 1.0), the following DEA models are available:

Conventional DEA models
Basic (radial) models (envelopment and multiplier forms)
Directional distance function model
(Weighted) Additive model
Super-efficiency additive model
Radial Super-efficiency model
(Weighted) Non-radial model
Preference Structure model
(Weighted) Slack-based model
(Weighted) Super-efficiency slack-based model

Conventional DEA models
Cross-efficiency (crs[12] and vrs[13])
Bootstrapping (Simar and Wilson algorithm)
FDH model
Productivity
Malmquist index
Fuzzy DEA models
Kao and Liu model[14]
Possibilistic model
Guo and Tanaka model
Fuzzy cross-efficiency[15] (only crs)

In the documentation for **deaR** we can find all the details about how to use the different functions and examples putting them into practice.

Although the DEA models available in **deaR** are the listed above, given the flexibility in programming **deaR** (here is an important strength of the package), the user himself can experiment, try and implement variants of these models. For example, if the user defines the weights appropriately in the "*Additive model*", MPI[16] or RAM[17] models can be obtained. Another example where the user can define weights and obtain different models is from "*Preference Structure Model*" (weighted no n radial model). In this case, the user can calculate the "*Cost Efficiency*", "*Revenue Efficiency*" or "*Profit Efficiency*" models.

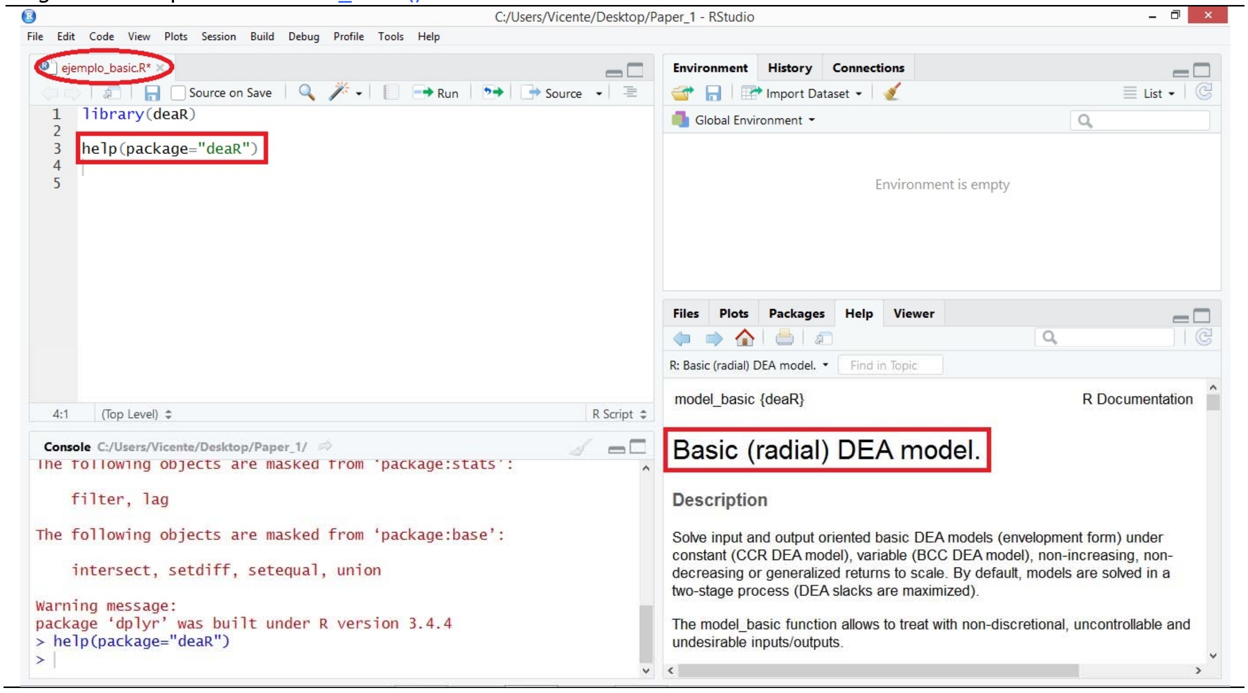
Now, we are going how to use the model_basic() function to perform basic DEA models. For that:

- Open the project "*Paper_1*" and create a new script. Give it the name: "*example_basic*".
- Load **deaR**.
- Go to the help for **deaR**.
- Clic on the *model_basic* link (see Figure 35). We can also write into the script:

help("model_basic")

[12] crs = constant returns-to-scale.
[13] vrs = variable returns-to-scale.
[14] This Fuzzy DEA model has been extended to a several DEA models. See the help for the package: **help("modelfuzzy_kaoliu").**
[15] Based on Guo and Tanaka's model.
[16] Measure of Inefficiency Proportions (MPI).
[17] Range Adjusted Measure (RAM).

Figure 35. Help for the model_basic() function.


C:/Users/Vicente/Desktop/Paper_1 - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
ejemplo_basic.R*
Source on Save
Run
Source
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
R: Basic (radial) DEA model.
Find in Topic
4:1
(Top Level)
R Script
Console C:/Users/Vicente/Desktop/Paper_1/

```
library(deaR)

help(package="deaR")
```

```
The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union

Warning message:
package 'dplyr' was built under R version 3.4.4
> help(package="deaR")
>
```

model_basic {deaR} R Documentation

Basic (radial) DEA model.

Description

Solve input and output oriented basic DEA models (envelopment form) under constant (CCR DEA model), variable (BCC DEA model), non-increasing, non-decreasing or generalized returns to scale. By default, models are solved in a two-stage process (DEA slacks are maximized).

The model_basic function allows to treat with non-discretional, uncontrollable and undesirable inputs/outputs.

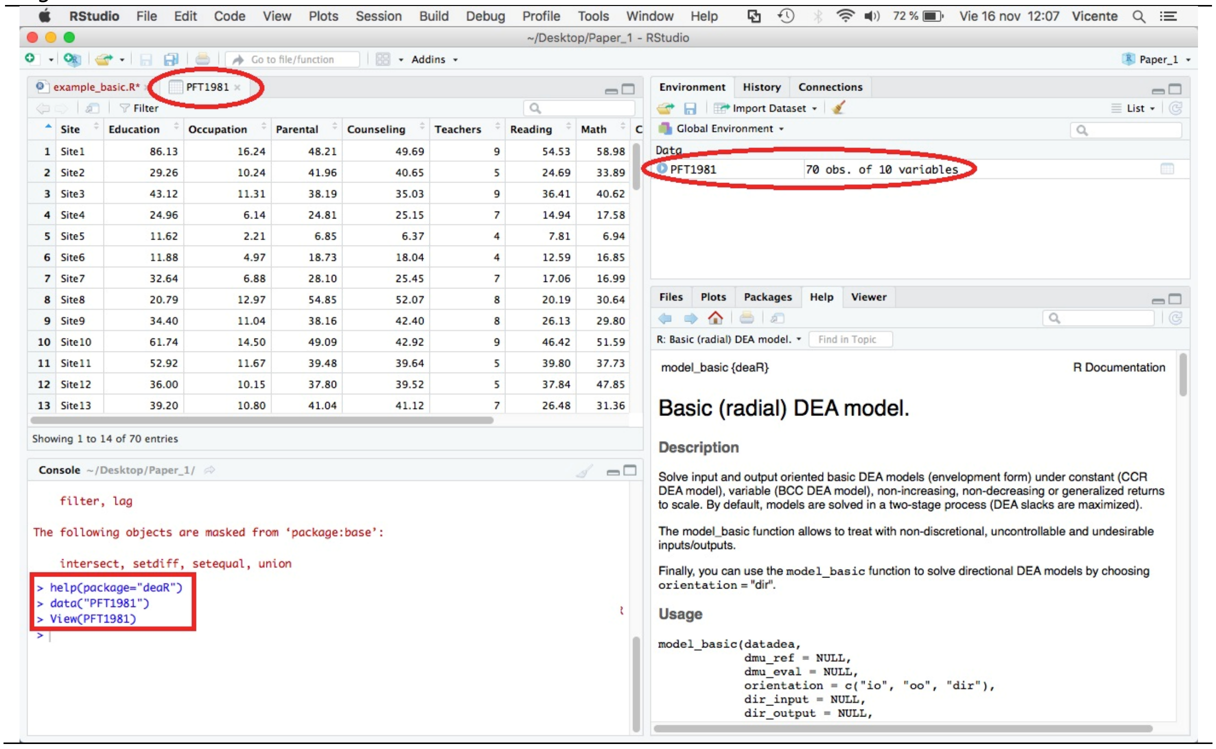
The model_basic() function takes the following arguments:

- datadea: It is the dataset (It must be a dataframe).
- *dmu_ref*: Selection of a subset of DMUs.
- *dmu_eval*: DMUs to be evaluated from the selected subset.
- *orientation*: Orientation of the DEA model: input-oriented, output-oriented o direccional.
- *dir_input*: Input direction vector in directional models.
- *dir_output*: Output direction vector in directional models.
- rts: Returns to scale (constant, variable, non-increasing, non-decreasing, generalized).
- *L*: If we select generalized rts, lower bound.
- *U*: If we select generalized rts, upper bound.
- *Maxslack*: By default, slacks are maximized in a second stage.
- *weight_slack_i*: The user can set weights for the input slacks to be maximized in the second stage.
- *weight_slack_o*: The user can set weights for the output slacks to be maximized in the second stage.
- *vtrans_i*: With undesirable inputs and variable returns-to-scale, the user can set a translation vector. By default, the translation vector is (max + 1).
- *vtrans_o*: With undesirable outputs and variable returns-to-scale, the user can set a translation vector. By default, the translation vector is (max + 1).
- *compute_target*: To calculate the targets for the max slack solution.
- *compute_multiplier*: It returns the multipliers (or weights) of the multiplier form.

- *returnlp*: For each DMU, this argument returns the linear problem of the first stage.

Now, we are going to learn how to use the model_basic() function by doing Examples 9 and 10. To follow these examples, we have to load the dataset in **deaR**: "*PFT1981*"[18]. The dataset has 70 DMUs (school sites) with 5 inputs (*Education*, *Occupation*, *Parental*, *Counseling*, *Teachers*) and 3 outputs (*Reading*, *Math*, *Coopersmith*). The Figure below (Figure 36) shows the instructions that we should write into the script "*example_basic*" to load the data.

Figure 36. Dataset: PTF1981.



	Site	Education	Occupation	Parental	Counseling	Teachers	Reading	Math	C
1	Site1	86.13	16.24	48.21	49.69	9	54.53	58.98	
2	Site2	29.26	10.24	41.96	40.65	5	24.69	33.89	
3	Site3	43.12	11.31	38.19	35.03	9	36.41	40.62	
4	Site4	24.96	6.14	24.81	25.15	7	14.94	17.58	
5	Site5	11.62	2.21	6.85	6.37	4	7.81	6.94	
6	Site6	11.88	4.97	18.73	18.04	4	12.59	16.85	
7	Site7	32.64	6.88	28.10	25.45	7	17.06	16.99	
8	Site8	20.79	12.97	54.85	52.07	8	20.19	30.64	
9	Site9	34.40	11.04	38.16	42.40	8	26.13	29.80	
10	Site10	61.74	14.50	49.09	42.92	9	46.42	51.59	
11	Site11	52.92	11.67	39.48	39.64	5	39.80	37.73	
12	Site12	36.00	10.15	37.80	39.52	5	37.84	47.85	
13	Site13	39.20	10.80	41.04	41.12	7	26.48	31.36	

Showing 1 to 14 of 70 entries

```
    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union

> help(package="deaR")
> data("PFT1981")
> View(PFT1981)
>
```

Data
PFT1981   70 obs. of 10 variables

R: Basic (radial) DEA model.

model_basic {deaR}   R Documentation

Basic (radial) DEA model.

Description

Solve input and output oriented basic DEA models (envelopment form) under constant (CCR DEA model), variable (BCC DEA model), non-increasing, non-decreasing or generalized returns to scale. By default, models are solved in a two-stage process (DEA slacks are maximized).

The model_basic function allows to treat with non-discretional, uncontrollable and undesirable inputs/outputs.

Finally, you can use the `model_basic` function to solve directional DEA models by choosing `orientation` = "dir".

Usage

```
model_basic(datadea,
            dmu_ref = NULL,
            dmu_eval = NULL,
            orientation = c("io", "oo", "dir"),
            dir_input = NULL,
            dir_output = NULL,
```

Example 9. model_basic() function.

Of 70 DMUs (school sites) in "*PTF1981*", there are 49 DMUs in Project Follow Through (PFT) and 21 DMUs in Non-Follow Through (NFT).

In this example, we are going to calculate the efficiency of the 49 DMUs in PFT by using the input-oriented CCR DEA moldel.

First, we have to use the read_data() function to adapt the data to the reading format in **deaR** (see section 7.2.2.). Run the instruction by (Run on

Now, we have to use the model_basic() function because we want to measure the efficiency by using the CCR DEA model (a conventional DEA model). But we want to calculate the efficiency of the first 49 DMUS; that is to say, the DMUs participating in PFT. To make this selection, we use the argument: *dmu_ref=1:49*. To evaluate all these units, we use the argument: *dmu_eval=1:49*. Therefore, we have to write into the script "*example_basic*" the following text:

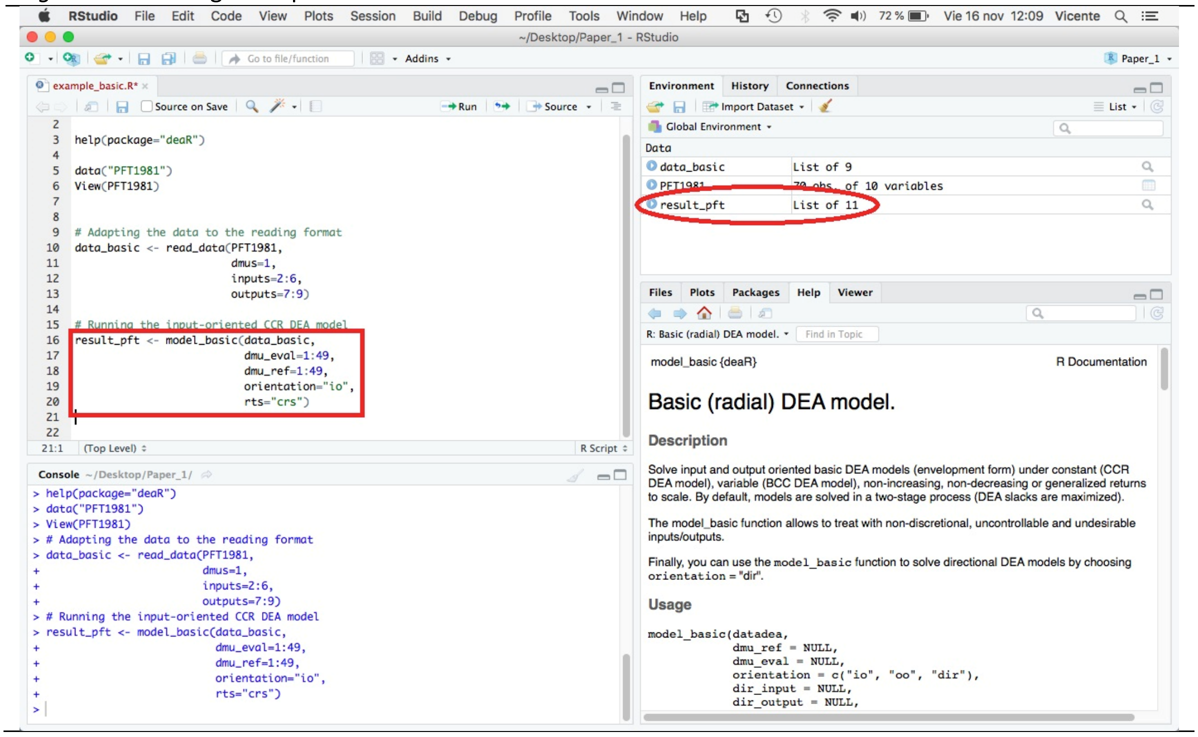
[18] Charnes, A.; Cooper, W.W.; Rhodes, E. (1981). "Evaluating Program and Managerial Efficiency: An Application of Data Envelopment Analysis to Program Follow Through", Management Science, 27(6), 668-697. https://pubsonline.informs.org/doi/abs/10.1287/mnsc.27.6.668

```
result_pft <- model_basci(PFT1981,
                          dmu_ref=1:49,
                          dmu_eval=1:49,
                          orientation=”io”,
                          rts=”crs”)
```

and run it (see Figure 37).

> Note: We can also select all the instructions and run them instead of doing it instruction by instruction.

Figure 37. Running the input-oriented CCR DEA model for DMUS in PFT.



```

help(package="deaR")

data("PFT1981")
View(PFT1981)


# Adapting the data to the reading format
data_basic <- read_data(PFT1981,
                        dmus=1,
                        inputs=2:6,
                        outputs=7:9)

# Running the input-oriented CCR DEA model
result_pft <- model_basic(data_basic,
                          dmu_eval=1:49,
                          dmu_ref=1:49,
                          orientation="io",
                          rts="crs")

```

```
> help(package="deaR")
> data("PFT1981")
> View(PFT1981)
> # Adapting the data to the reading format
> data_basic <- read_data(PFT1981,
+                         dmus=1,
+                         inputs=2:6,
+                         outputs=7:9)
> # Running the input-oriented CCR DEA model
> result_pft <- model_basic(data_basic,
+                           dmu_eval=1:49,
+                           dmu_ref=1:49,
+                           orientation="io",
+                           rts="crs")
> |
```

```
Data
data_basic    List of 9
PFT1981       70 obs. of 10 variables
result_pft    List of 11
```

model_basic {deaR}   R Documentation

Basic (radial) DEA model.

Description

Solve input and output oriented basic DEA models (envelopment form) under constant (CCR DEA model), variable (BCC DEA model), non-increasing, non-decreasing or generalized returns to scale. By default, models are solved in a two-stage process (DEA slacks are maximized).

The model_basic function allows to treat with non-discretional, uncontrollable and undesirable inputs/outputs.

Finally, you can use the `model_basic` function to solve directional DEA models by choosing `orientation` = "dir".

Usage

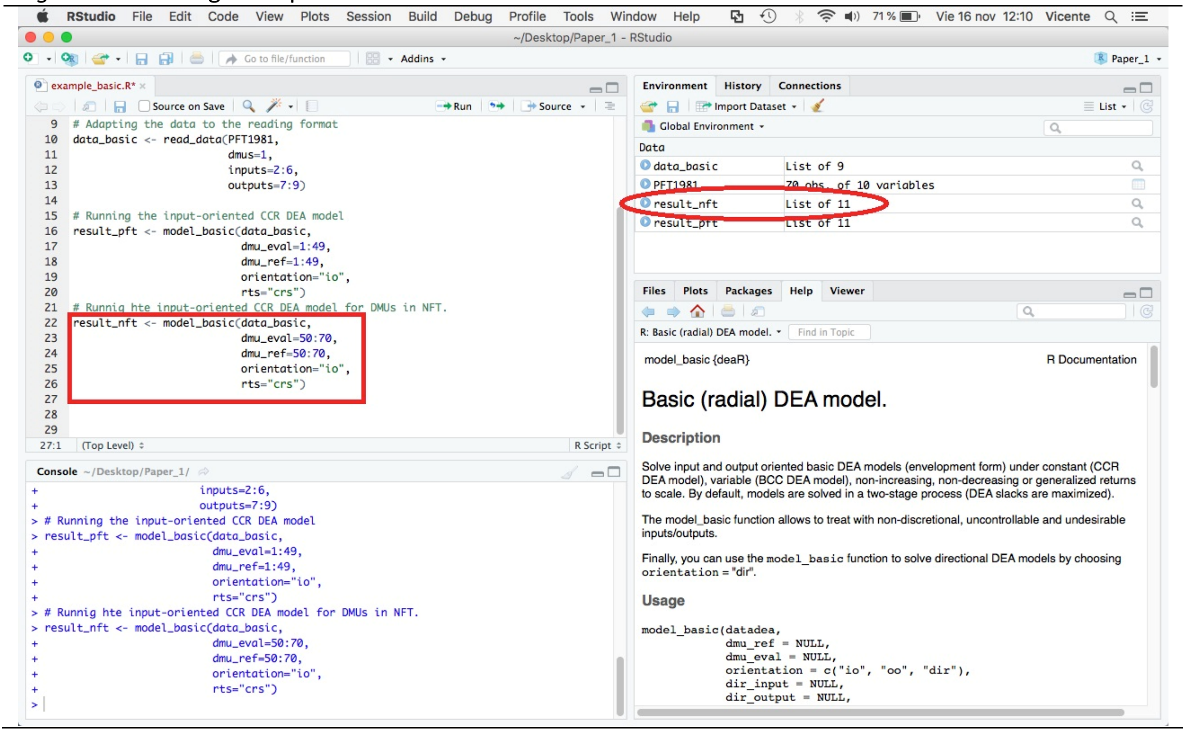
```
model_basic(datadea,
            dmu_ref = NULL,
            dmu_eval = NULL,
            orientation = c("io", "oo", "dir"),
            dir_input = NULL,
            dir_output = NULL,
```

Example 10. model_basic() function.

At this point, **could you run the CCR input-oriented DEA model for the DMUs in NFT?** (DMUs in NFT are the DMUs from 50 to 70).

The answer to this question is in next page (Figure 38)

Figure 38. Running the input-oriented CCR DEA model for DMUs in NFT.



```
# Adapting the data to the reading format
data_basic <- read_data(PFT1981,
                        dmus=1,
                        inputs=2:6,
                        outputs=7:9)

# Running the input-oriented CCR DEA model
result_pft <- model_basic(data_basic,
                          dmu_eval=1:49,
                          dmu_ref=1:49,
                          orientation="io",
                          rts="crs")
# Runnig hte input-oriented CCR DEA model for DMUs in NFT.
result_nft <- model_basic(data_basic,
                          dmu_eval=50:70,
                          dmu_ref=50:70,
                          orientation="io",
                          rts="crs")
```

```
+                         inputs=2:6,
+                         outputs=7:9)
> # Running the input-oriented CCR DEA model
> result_pft <- model_basic(data_basic,
+                           dmu_eval=1:49,
+                           dmu_ref=1:49,
+                           orientation="io",
+                           rts="crs")
> # Runnig hte input-oriented CCR DEA model for DMUs in NFT.
> result_nft <- model_basic(data_basic,
+                           dmu_eval=50:70,
+                           dmu_ref=50:70,
+                           orientation="io",
+                           rts="crs")
>
```

```
Data
data_basic     List of 9
PFT1981        70 obs. of 10 variables
result_nft     List of 11
result_pft     List of 11
```

model_basic {deaR}   R Documentation

Basic (radial) DEA model.

Description

Solve input and output oriented basic DEA models (envelopment form) under constant (CCR DEA model), variable (BCC DEA model), non-increasing, non-decreasing or generalized returns to scale. By default, models are solved in a two-stage process (DEA slacks are maximized).

The model_basic function allows to treat with non-discretional, uncontrollable and undesirable inputs/outputs.

Finally, you can use the model_basic function to solve directional DEA models by choosing orientation = "dir".

Usage

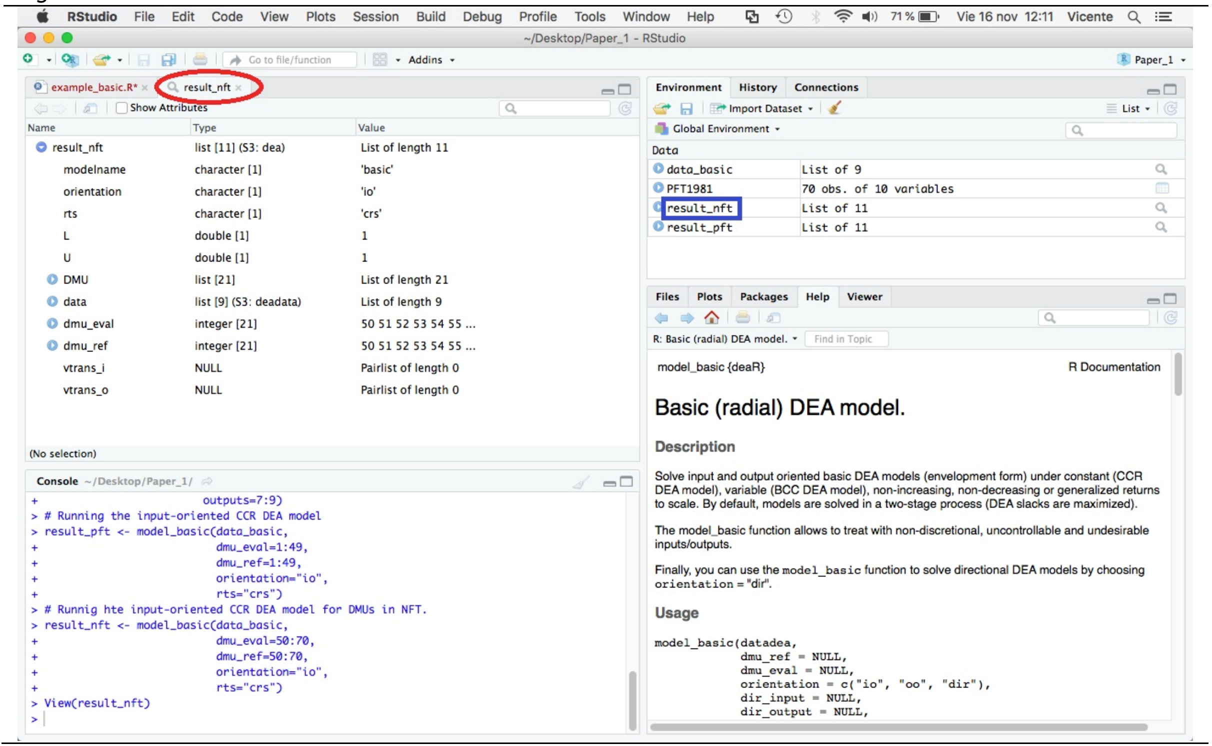
```
model_basic(datadea,
            dmu_ref = NULL,
            dmu_eval = NULL,
            orientation = c("io", "oo", "dir"),
            dir_input = NULL,
            dir_output = NULL,
```

Save the script "*example_basic*".

7.4. Extracting the main results.

In Examples 9 and 10, after running the model_basic() function, all the results are saved in the objects "*result_pft*" and "*result_nft*" respectively. We can see that these objects are a list of 11 components. We can show the content of each list by clicking on the name of the object (see Figure 39) or its structure clicking on the blue arrow (▶).

Figure 39. Structure of the results for a basic DEA model.



Name	Type	Value
result_nft	list [11] (S3: dea)	List of length 11
modelname	character [1]	'basic'
orientation	character [1]	'io'
rts	character [1]	'crs'
L	double [1]	1
U	double [1]	1
DMU	list [21]	List of length 21
data	list [9] (S3: deadata)	List of length 9
dmu_eval	integer [21]	50 51 52 53 54 55 ...
dmu_ref	integer [21]	50 51 52 53 54 55 ...
vtrans_i	NULL	Pairlist of length 0
vtrans_o	NULL	Pairlist of length 0

```
+                         outputs=7:9)
> # Running the input-oriented CCR DEA model
> result_pft <- model_basic(data_basic,
+                           dmu_eval=1:49,
+                           dmu_ref=1:49,
+                           orientation="io",
+                           rts="crs")
> # Runnig hte input-oriented CCR DEA model for DMUs in NFT.
> result_nft <- model_basic(data_basic,
+                           dmu_eval=50:70,
+                           dmu_ref=50:70,
+                           orientation="io",
+                           rts="crs")
> View(result_nft)
>
```

```
Data
data_basic     List of 9
PFT1981        70 obs. of 10 variables
result_nft     List of 11
result_pft     List of 11
```

To extract the main results of the DEA analysis, **deaR** has a several specific functions. These functions are:

- ✓ **efficiencies()**: To extract the efficiency scores.
- ✓ **slacks()**: To extract input and output slacks.
- ✓ **targets()**: To extract the input and output target values.
- ✓ **lambdas()**: To extract the lambdas (or intensities).
- ✓ **references()**: To extract the reference set for inefficient DMUs.
- ✓ **rts()**: To extract the returns-to-scale.
- ✓ **multipliers()**: To extract the multipliers (or weights) of the multiplier DEA form.

In the following Example 11 we are going to practice with these functions.

Example 11. Extracting the results.

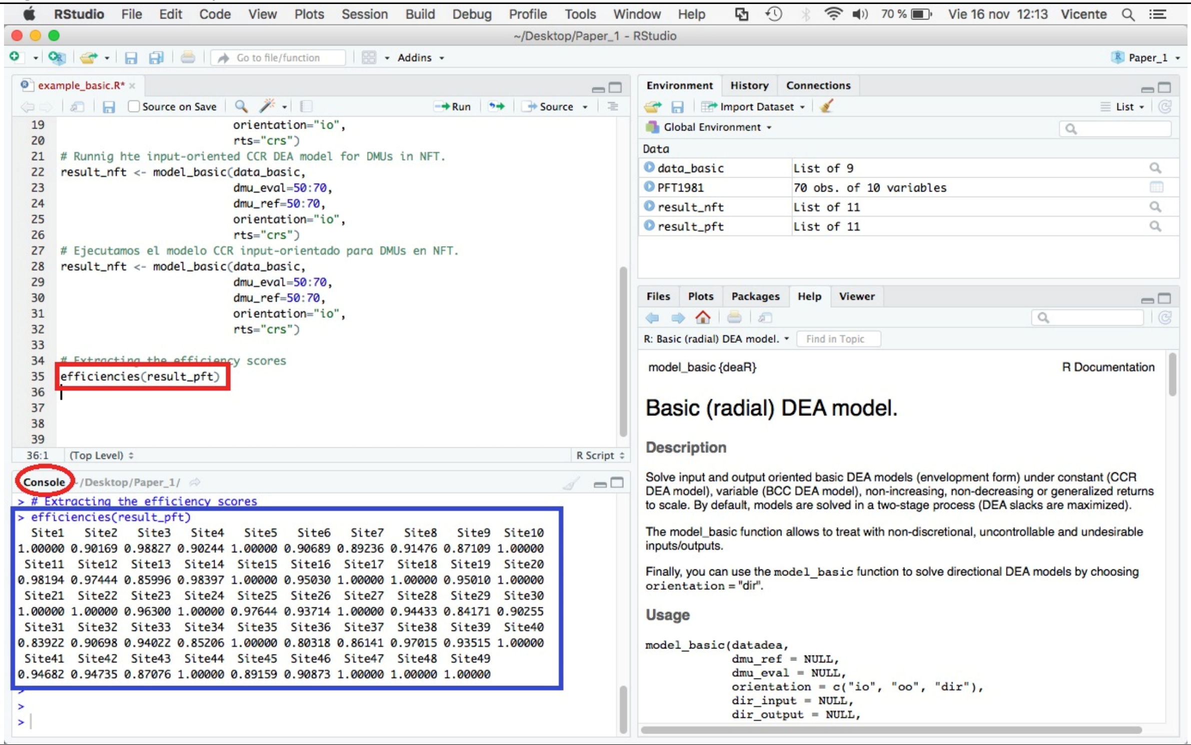
To extract the efficiency scores from "*result_pft*", we only have to write into the script the instruction:

efficiencies(result_pft)

and run it.

The efficiency scores will be shown in the *Console*, as the following Figure shows.

Figure 40. Efficiency scores of DMUs in PFT.



```
                          orientation="io",
                          rts="crs")
# Runnig hte input-oriented CCR DEA model for DMUs in NFT.
result_nft <- model_basic(data_basic,
                          dmu_eval=50:70,
                          dmu_ref=50:70,
                          orientation="io",
                          rts="crs")
# Ejecutamos el modelo CCR input-orientado para DMUs en NFT.
result_nft <- model_basic(data_basic,
                          dmu_eval=50:70,
                          dmu_ref=50:70,
                          orientation="io",
                          rts="crs")

# Extracting the efficiency scores
efficiencies(result_pft)
```

```
> # Extracting the efficiency scores
> efficiencies(result_pft)
  Site1   Site2   Site3   Site4   Site5   Site6   Site7   Site8   Site9  Site10 
1.00000 0.90169 0.98827 0.90244 1.00000 0.90689 0.89236 0.91476 0.87109 1.00000 
 Site11  Site12  Site13  Site14  Site15  Site16  Site17  Site18  Site19  Site20 
0.98194 0.97444 0.85996 0.98397 1.00000 0.95030 1.00000 1.00000 0.95010 1.00000 
 Site21  Site22  Site23  Site24  Site25  Site26  Site27  Site28  Site29  Site30 
1.00000 1.00000 0.96300 1.00000 0.97644 0.93714 1.00000 0.94433 0.84171 0.90255 
 Site31  Site32  Site33  Site34  Site35  Site36  Site37  Site38  Site39  Site40 
0.83922 0.90698 0.94022 0.85206 1.00000 0.80318 0.86141 0.97015 0.93515 1.00000 
 Site41  Site42  Site43  Site44  Site45  Site46  Site47  Site48  Site49 
0.94682 0.94735 0.87076 1.00000 0.89159 0.90873 1.00000 1.00000 1.00000 
>
>
>
```

Data		
data_basic	List of 9	
PFT1981	70 obs. of 10 variables	
result_nft	List of 11	
result_pft	List of 11	

model_basic {deaR} R Documentation

Basic (radial) DEA model.

Description

Solve input and output oriented basic DEA models (envelopment form) under constant (CCR DEA model), variable (BCC DEA model), non-increasing, non-decreasing or generalized returns to scale. By default, models are solved in a two-stage process (DEA slacks are maximized).

The model_basic function allows to treat with non-discretional, uncontrollable and undesirable inputs/outputs.

Finally, you can use the `model_basic` function to solve directional DEA models by choosing `orientation` = "dir".

Usage

```
model_basic(datadea,
            dmu_ref = NULL,
            dmu_eval = NULL,
            orientation = c("io", "oo", "dir"),
            dir_input = NULL,
            dir_output = NULL,
```

Similarly, to extract the reference set of the inefficient DMUs we write and execute the instruction:

references(result_pft)

The remaining functions (slacks(), targets(), lambdas(), rts(), and multipliers()) work in a similar way. Try and practice with these functions and pay attention to the results obtained.

Save the script “*example_basic..R*”, close the project and quit RStudio.

7.5. summary() function.

In addition to the previous functions (efficiencies(), lambdas(), etc.), **deaR** has the summary() function, which summarizes the results of the DEA analysis. The summary() function serves to summarize the results of both conventional DEA models and fuzzy DEA models. To use this function we have to know that it has the following structure:[19]:

summary(objet, exportExcel = TRUE, filename = NULL)

The arguments of this function refer to:

- *object*: It is the object to which we have assigned the results of a conventional DEA or fuzzy DEA model.
- *exportExcel*: The default value for this argument is TRUE. Therefore, the summary() function will automatically create an Excel file with the main results in the working directory. The user can choose not to create the Excel file (exportExcel = FALSE).
- *filename*: By default, the value of the argument is NULL. Therefore, the default name of the Excel file will be: "*ResutsDEAyear-month-dat_hour:minute*:second.xlsx". Of course, the user can give other name to the Excel file.

As we have said, the way to use the summary() function to summarize the results of a DEA analysis is identical in both conventional DEA models and fuzzy DEA models. The difference is found in the summary of the Excel file generated. That is, the sheets of the Excel workbook created are different depending on the model. Let’s see this in Examples 12 and 13.

Example 12. Summary of results: conventional DEA model.

Follow the following steps:

1. Open the project “*Paper_1*”.
2. Create a new script: “*Summary_DEA*”
3. Load **deaR**.

Assumption: Load the dataset provided by **deaR**: “*Hua_Bian_2007*”[20]. This dataset has 30 DMUs with 2 inputs (*D-Input1*, *D-Input2*), 2 outputs (*D-Output1*, *D-Output2*) and 1 undesirable output (*UD-Output1*) (in this same order).

We want to get the summary of results from the output-oriented BBC DEA model. As there is an undesirable output, to take this into account in the analysis we will use as the translation parameter the one used by Hua and Bian (2007): vtrans_o=1500.

[19] For more details: **help(summary.dea)** o **help(summary.dea_fuzzy)**.

[20] Hua Z.; Bian Y. (2007). DEA with Undesirable Factors. In: Zhu J., Cook W.D. (eds) Modeling Data Irregularities and Structural Complexities in Data Envelopment Analysis. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-71607-7_6

Important: Remember the steps to use **deaR**.

Step 1. Load the data.

Step 2. Adapt the data to the format used by **deaR**.

Step 3. Run the DEA model.

Step 4. Extract the results.

Try it!!!

(the solution is shown in the next page)

Solution: As we can see in Figure 41, to solve Example 12 we have to write into the script "Summary_*DEA*" the following instructions:

Step 1. Load the data:

data("Hua_Bian_2007")

Step 2. Adapt the data:

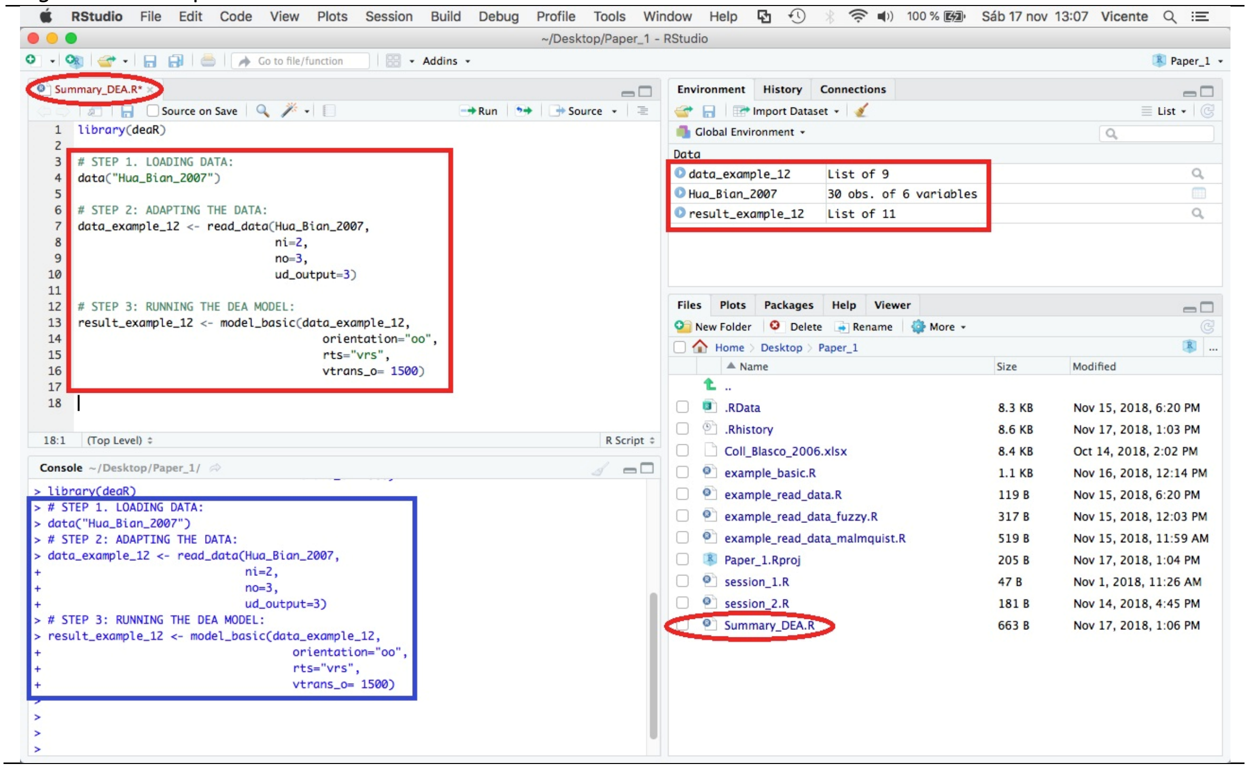
```
data_example_12 <- read_data(Hua_Bian_2007,
                             ni=2,
                             no=3,
                             ud_output=3)
```

Note: We have 3 outputs (no=3) and the third output is the undesirable output (ud_output=3)

Step 3. Run the DEA model:

```
result_example_12 <- model_basic(data_example_12,
                                 orientation="oo",
                                 rts="vrs",
                                 vtrans_o= 1500)
```

Figure 41. Example 12 solution.


RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
100 % Sáb 17 nov 13:07 Vicente
~/Desktop/Paper_1 - RStudio
Go to file/function
Addins
Paper_1
Summary_DEA.R*
Source on Save
Run
Source
18:1
(Top Level)
R Script
Console ~/Desktop/Paper_1/
Environment History Connections
Import Dataset
List
Global Environment
Data
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home > Desktop > Paper_1

```
library(deaR)

# STEP 1. LOADING DATA:
data("Hua_Bian_2007")

# STEP 2: ADAPTING THE DATA:
data_example_12 <- read_data(Hua_Bian_2007,
                             ni=2,
                             no=3,
                             ud_output=3)

# STEP 3: RUNNING THE DEA MODEL:
result_example_12 <- model_basic(data_example_12,
                                 orientation="oo",
                                 rts="vrs",
                                 vtrans_o= 1500)
```

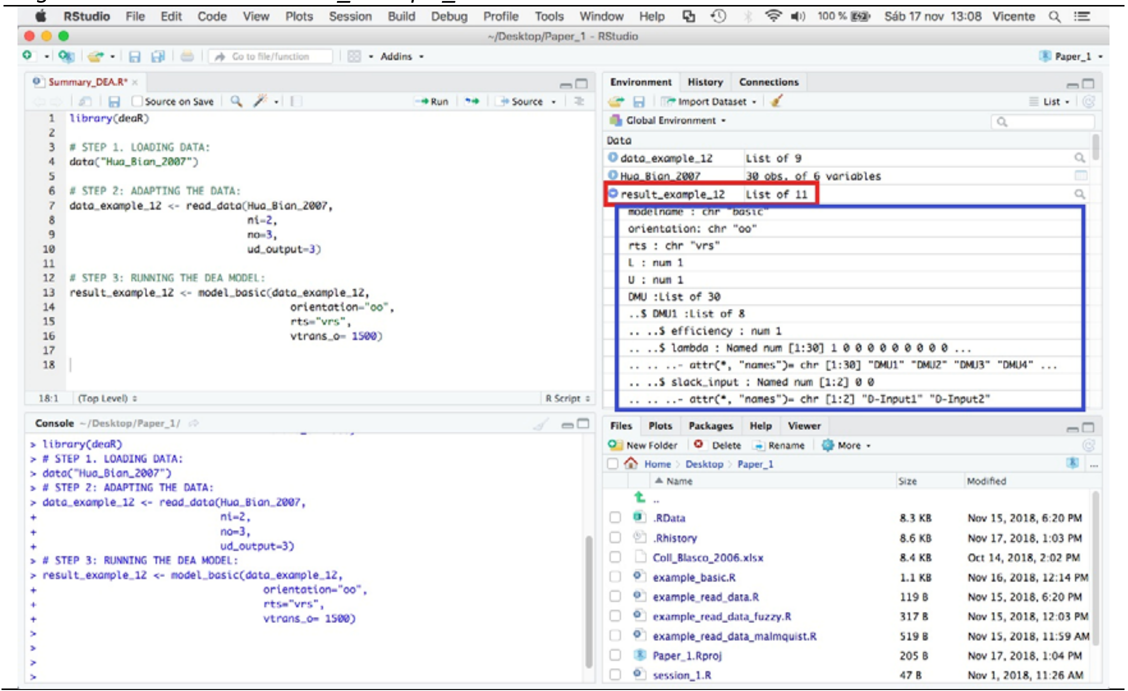
```
> library(deaR)
> # STEP 1. LOADING DATA:
> data("Hua_Bian_2007")
> # STEP 2: ADAPTING THE DATA:
> data_example_12 <- read_data(Hua_Bian_2007,
+                              ni=2,
+                              no=3,
+                              ud_output=3)
> # STEP 3: RUNNING THE DEA MODEL:
> result_example_12 <- model_basic(data_example_12,
+                                  orientation="oo",
+                                  rts="vrs",
+                                  vtrans_o= 1500)
>
>
>
```

data_example_12	List of 9
Hua_Bian_2007	30 obs. of 6 variables
result_example_12	List of 11

Name	Size	Modified
..		
.RData	8.3 KB	Nov 15, 2018, 6:20 PM
.Rhistory	8.6 KB	Nov 17, 2018, 1:03 PM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
example_basic.R	1.1 KB	Nov 16, 2018, 12:14 PM
example_read_data.R	119 B	Nov 15, 2018, 6:20 PM
example_read_data_fuzzy.R	317 B	Nov 15, 2018, 12:03 PM
example_read_data_malmquist.R	519 B	Nov 15, 2018, 11:59 AM
Paper_1.Rproj	205 B	Nov 17, 2018, 1:04 PM
session_1.R	47 B	Nov 1, 2018, 11:26 AM
session_2.R	181 B	Nov 14, 2018, 4:45 PM
Summary_DEA.R	663 B	Nov 17, 2018, 1:06 PM

All the results of the output-oriented BCC DEA model that we have executed can be found in the object "*result_example_12*", which is a list of 11 components (see Figure 42).

Figure 42. Structure of "*result_example_12*".



```
library(deaR)

# STEP 1. LOADING DATA:
data("Hua_Bian_2007")

# STEP 2: ADAPTING THE DATA:
data_example_12 <- read_data(Hua_Bian_2007,
                             ni=2,
                             no=3,
                             ud_output=3)

# STEP 3: RUNNING THE DEA MODEL:
result_example_12 <- model_basic(data_example_12,
                                 orientation="oo",
                                 rts="vrs",
                                 vtrans_o= 1500)
```

```
Data
data_example_12     List of 9
Hua_Bian_2007       30 obs. of 6 variables
result_example_12   List of 11
   modelname : chr "basic"
   orientation: chr "oo"
   rts : chr "vrs"
   L : num 1
   U : num 1
   DMU :List of 30
   ..$ DMU1 :List of 8
   .. ..$ efficiency : num 1
   .. ..$ lambda : Named num [1:30] 1 0 0 0 0 0 0 0 0 0 ...
   .. .. ..- attr(*, "names")= chr [1:30] "DMU1" "DMU2" "DMU3" "DMU4" ...
   .. ..$ slack_input : Named num [1:2] 0 0
   .. .. ..- attr(*, "names")= chr [1:2] "D-Input1" "D-Input2"
```

```
> library(deaR)
> # STEP 1. LOADING DATA:
> data("Hua_Bian_2007")
> # STEP 2: ADAPTING THE DATA:
> data_example_12 <- read_data(Hua_Bian_2007,
+                              ni=2,
+                              no=3,
+                              ud_output=3)
> # STEP 3: RUNNING THE DEA MODEL:
> result_example_12 <- model_basic(data_example_12,
+                                  orientation="oo",
+                                  rts="vrs",
+                                  vtrans_o= 1500)
>
>
>
>
```

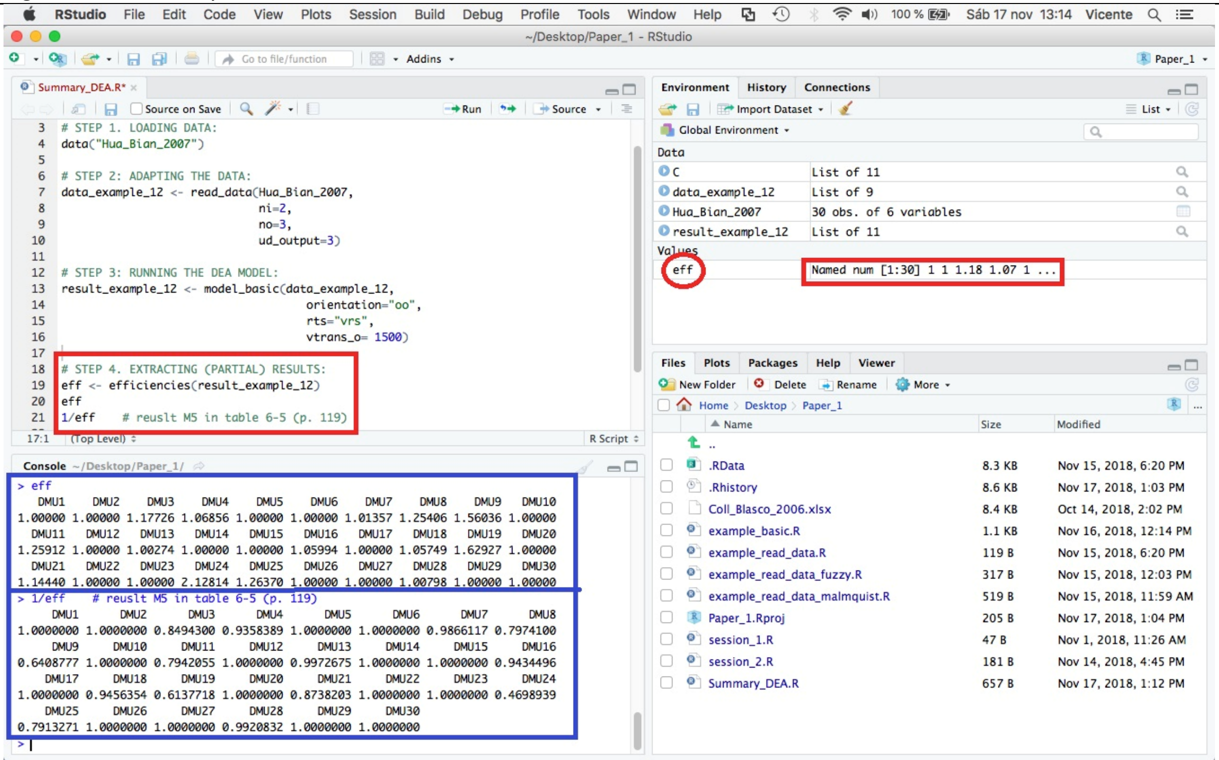
At this point, we can extract the partial results with the functions: efficiencies(), lambdas(), multipliers(), rts(), references(), slacks(), targets(); and the summary of results with the summary() function.

As the Figure 43 shows, we extract the efficiency scores of the DMUs with the efficiencies() function. The efficiencies are listed in the *Console* and assigned to the object "*eff*". To obtain the same results as those obtained by Hua and Bian (2007), we write into the script:

1/eff

and run the instruction.

Figure 43. Efficiency scores.



```
# STEP 4. EXTRACTING (PARTIAL) RESULTS:
eff <- efficiencies(result_example_12)
eff
1/eff    # reuslt M5 in table 6-5 (p. 119)
```

```
Data
C                   List of 11
data_example_12     List of 9
Hua_Bian_2007       30 obs. of 6 variables
result_example_12   List of 11
Values
eff                 Named num [1:30] 1 1 1.18 1.07 1 ...
```

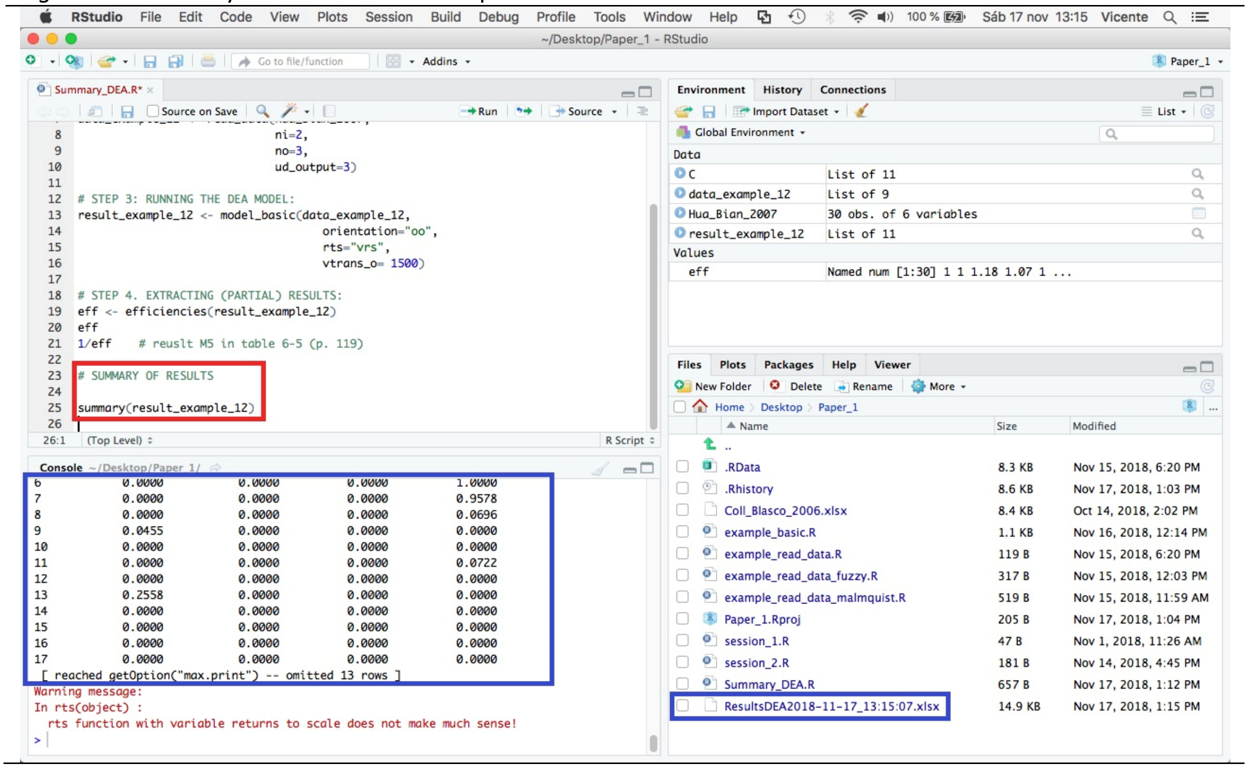
```
> eff
   DMU1    DMU2    DMU3    DMU4    DMU5    DMU6    DMU7    DMU8    DMU9   DMU10
1.00000 1.00000 1.17726 1.06856 1.00000 1.00000 1.01357 1.25406 1.56036 1.00000
  DMU11   DMU12   DMU13   DMU14   DMU15   DMU16   DMU17   DMU18   DMU19   DMU20
1.25912 1.00000 1.00274 1.00000 1.00000 1.05994 1.00000 1.05749 1.62927 1.00000
  DMU21   DMU22   DMU23   DMU24   DMU25   DMU26   DMU27   DMU28   DMU29   DMU30
1.14440 1.00000 1.00000 2.12814 1.26370 1.00000 1.00000 1.00798 1.00000 1.00000
> 1/eff    # reuslt M5 in table 6-5 (p. 119)
     DMU1      DMU2      DMU3      DMU4      DMU5      DMU6      DMU7      DMU8
1.0000000 1.0000000 0.8494300 0.9358389 1.0000000 1.0000000 0.9866117 0.7974100
     DMU9     DMU10     DMU11     DMU12     DMU13     DMU14     DMU15     DMU16
0.6408777 1.0000000 0.7942055 1.0000000 0.9972675 1.0000000 1.0000000 0.9434496
    DMU17     DMU18     DMU19     DMU20     DMU21     DMU22     DMU23     DMU24
1.0000000 0.9456354 0.6137718 1.0000000 0.8738203 1.0000000 1.0000000 0.4698939
    DMU25     DMU26     DMU27     DMU28     DMU29     DMU30
0.7913271 1.0000000 1.0000000 0.9920832 1.0000000 1.0000000
>
```

To get a summary of all the results obtained by running the output-oriented BCC DEA model, we use the summary() function. We write into the script "*Summary_DEA*":

summary(result_example_12)

All results are dumped on the *Console*. Although there are not many DMUs (only 30), there are many results. It is not practical to use the summary() function to see the results on the screen. However, as we can see in Figure 44, **deaR** has also created an Excel file that has all these results. Notice the name that has been given to the file by default.

Figure 44. Summary of results of the Example 12.



```
                                   ni=2,
                                   no=3,
                                   ud_output=3)

# STEP 3: RUNNING THE DEA MODEL:
result_example_12 <- model_basic(data_example_12,
                                 orientation="oo",
                                 rts="vrs",
                                 vtrans_o= 1500)

# STEP 4. EXTRACTING (PARTIAL) RESULTS:
eff <- efficiencies(result_example_12)
eff
1/eff    # reuslt M5 in table 6-5 (p. 119)

# SUMMARY OF RESULTS

summary(result_example_12)
```

```
6           0.0000        0.0000        0.0000        1.0000
7           0.0000        0.0000        0.0000        0.9578
8           0.0000        0.0000        0.0000        0.0696
9           0.0455        0.0000        0.0000        0.0000
10          0.0000        0.0000        0.0000        0.0000
11          0.0000        0.0000        0.0000        0.0722
12          0.0000        0.0000        0.0000        0.0000
13          0.2558        0.0000        0.0000        0.0000
14          0.0000        0.0000        0.0000        0.0000
15          0.0000        0.0000        0.0000        0.0000
16          0.0000        0.0000        0.0000        0.0000
17          0.0000        0.0000        0.0000        0.0000
 [ reached getOption("max.print") -- omitted 13 rows ]
Warning message:
In rts(object) :
  rts function with variable returns to scale does not make much sense!
>
```

Global Environment

Data

C	List of 11
data_example_12	List of 9
Hua_Bian_2007	30 obs. of 6 variables
result_example_12	List of 11

Values

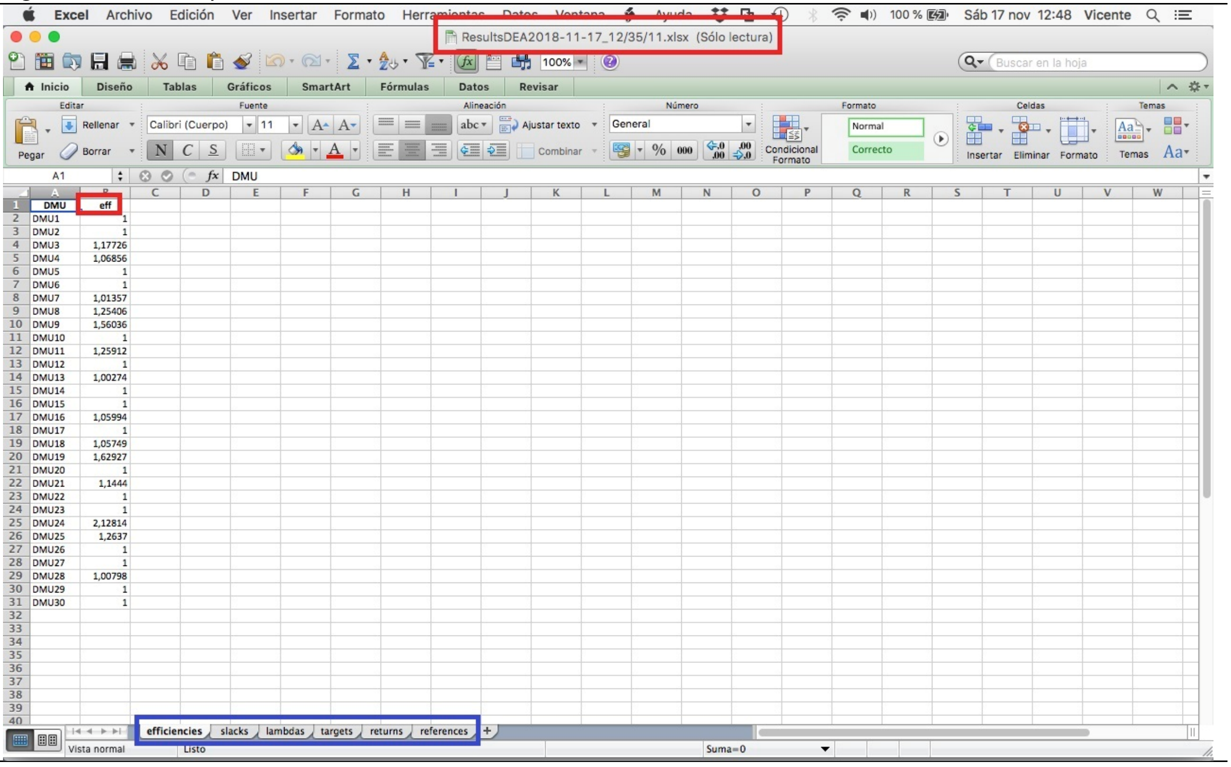
eff	Named num [1:30] 1 1 1.18 1.07 1 ...

Home > Desktop > Paper_1

Name	Size	Modified
..		
.RData	8.3 KB	Nov 15, 2018, 6:20 PM
.Rhistory	8.6 KB	Nov 17, 2018, 1:03 PM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
example_basic.R	1.1 KB	Nov 16, 2018, 12:14 PM
example_read_data.R	119 B	Nov 15, 2018, 6:20 PM
example_read_data_fuzzy.R	317 B	Nov 15, 2018, 12:03 PM
example_read_data_malmquist.R	519 B	Nov 15, 2018, 11:59 AM
Paper_1.Rproj	205 B	Nov 17, 2018, 1:04 PM
session_1.R	47 B	Nov 1, 2018, 11:26 AM
session_2.R	181 B	Nov 14, 2018, 4:45 PM
Summary_DEA.R	657 B	Nov 17, 2018, 1:12 PM
ResultsDEA2018-11-17_13:15:07.xlsx	14.9 KB	Nov 17, 2018, 1:15 PM

Open the Excel file to see how the results are shown (see Figure 45). To do this, we click on the file "*ResutsDEAyear-month-dat_hour:minute*:second.xlsx " and select the option "*View file*".

Figure 45. Summary results in Excel.



DMU	eff
DMU1	1
DMU2	1
DMU3	1,17726
DMU4	1,06856
DMU5	1
DMU6	1
DMU7	1,01357
DMU8	1,25406
DMU9	1,56036
DMU10	1
DMU11	1,25912
DMU12	1
DMU13	1,00274
DMU14	1
DMU15	1
DMU16	1,05994
DMU17	1
DMU18	1,05749
DMU19	1,62927
DMU20	1
DMU21	1,1444
DMU22	1
DMU23	1
DMU24	2,12814
DMU25	1,2637
DMU26	1
DMU27	1
DMU28	1,00798
DMU29	1
DMU30	1

We save the script "*summary_DEA*".

Now, in Example 13 we are going to replicate the results in León, et al. (2003)[21].

Example 13. Summary of results: Posibilistic fuzzy DEA model.

León et al. (2013) use possibilistic programming techniques to measure the efficiency based on the envelopment form of the input-oriented BCC model. Data used by the authors in their article can be found in the dataset "Leon2003", which is provided by **deaR**.

We create a new script: "*Summary_DEA_fuzzy*". To replicate the results that the authors show in Table 2 (page 419), we write into the script:

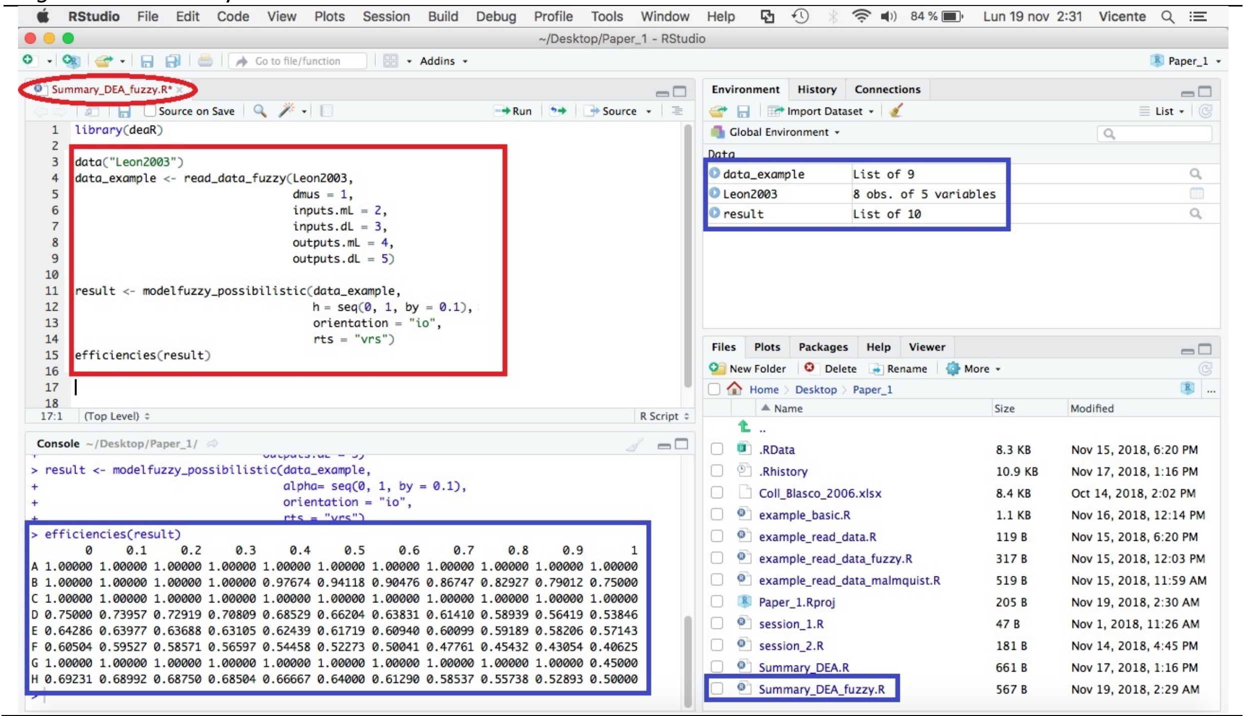
```
data("Leon2003")
data_example <- read_data_fuzzy(Leon2003,
                                dmus = 1,
                                inputs.mL = 2,
                                inputs.dL = 3,
                                outputs.mL = 4,
                                outputs.dL = 5)

result <- modelfuzzy_possibilistic(data_example,
                                   h= seq(0, 1, by = 0.1),
                                   orientation = "io",
                                   rts = "vrs")
efficiencies(result)
```

[21] Léon, T.; Liern, V. Ruiz, J.; Sirvent, I. (2003). "A Possibilistic Programming Approach to the Assessment of Efficiency with DEA Models", Fuzzy Sets and Systems, 139, 407–419. https://doi.org/10.1016/S0165-0114(02)00608-5

Note: h = seq (0, 1, by = 0.1) generates a sequence of values for the different levels of possibility: h = (0, 0.1, 0.2, ..., 1).

We run the instructions. The efficiency scores of the BCC input-oriented model will be shown for the different levels of possibility h in the *Console* (see Figure 46)

Figure 46. Efficiency scores for different h-levels.



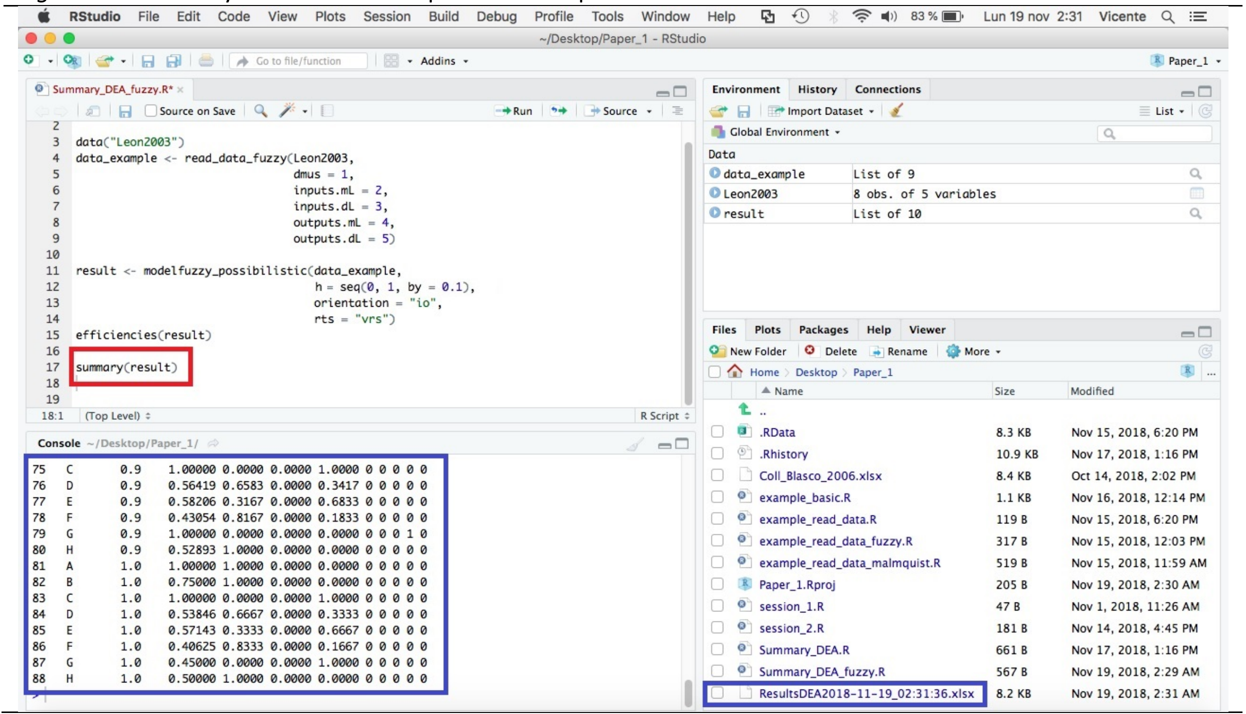
```
library(deaR)

data("Leon2003")
data_example <- read_data_fuzzy(Leon2003,
                                dmus = 1,
                                inputs.mL = 2,
                                inputs.dL = 3,
                                outputs.mL = 4,
                                outputs.dL = 5)

result <- modelfuzzy_possibilistic(data_example,
                                   h = seq(0, 1, by = 0.1),
                                   orientation = "io",
                                   rts = "vrs")

efficiencies(result)
```

```
> result <- modelfuzzy_possibilistic(data_example,
+                                    alpha= seq(0, 1, by = 0.1),
+                                    orientation = "io",
+                                    rts = "vrs")
> efficiencies(result)
        0     0.1     0.2     0.3     0.4     0.5     0.6     0.7     0.8     0.9       1
A 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000
B 1.00000 1.00000 1.00000 1.00000 0.97674 0.94118 0.90476 0.86747 0.82927 0.79012 0.75000
C 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000
D 0.75000 0.73957 0.72919 0.70809 0.68529 0.66204 0.63831 0.61410 0.58939 0.56419 0.53846
E 0.64286 0.63977 0.63688 0.63105 0.62439 0.61719 0.60940 0.60099 0.59189 0.58206 0.57143
F 0.60504 0.59527 0.58571 0.56597 0.54458 0.52273 0.50041 0.47761 0.45432 0.43054 0.40625
G 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 0.45000
H 0.69231 0.68992 0.68750 0.68504 0.66667 0.64000 0.61290 0.58537 0.55738 0.52893 0.50000
```

To get a summary of the results on the screen (see Figure 47) and export them to an Excel file, we write the following instruction into the script and run it:

summary(result)

Figure 47. Summary of results for the possibilistic input-oriented BCC DEA model.

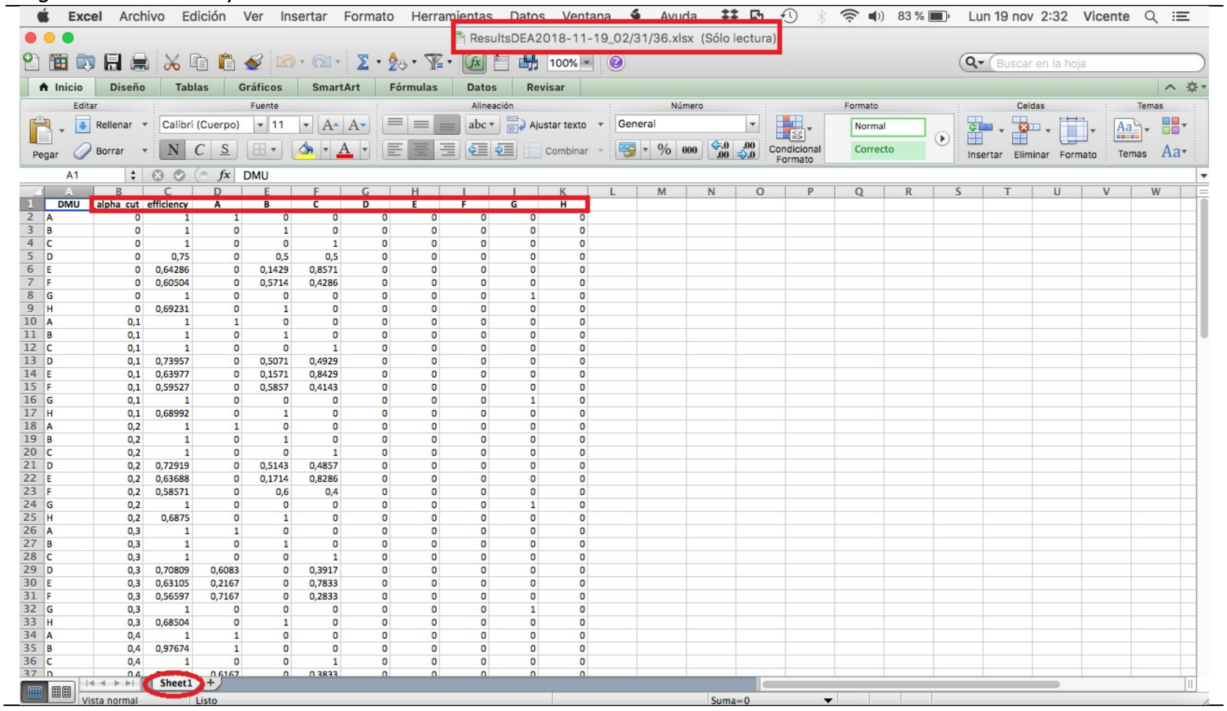


```
data("Leon2003")
data_example <- read_data_fuzzy(Leon2003,
                                dmus = 1,
                                inputs.mL = 2,
                                inputs.dL = 3,
                                outputs.mL = 4,
                                outputs.dL = 5)

result <- modelfuzzy_possibilistic(data_example,
                                   h = seq(0, 1, by = 0.1),
                                   orientation = "io",
                                   rts = "vrs")

efficiencies(result)

summary(result)
```

```
75   C      0.9    1.00000 0.0000 0.0000 1.0000 0 0 0 0 0
76   D      0.9    0.56419 0.6583 0.0000 0.3417 0 0 0 0 0
77   E      0.9    0.58206 0.3167 0.0000 0.6833 0 0 0 0 0
78   F      0.9    0.43054 0.8167 0.0000 0.1833 0 0 0 0 0
79   G      0.9    1.00000 0.0000 0.0000 0.0000 0 0 0 1 0
80   H      0.9    0.52893 1.0000 0.0000 0.0000 0 0 0 0 0
81   A      1.0    1.00000 1.0000 0.0000 0.0000 0 0 0 0 0
82   B      1.0    0.75000 1.0000 0.0000 0.0000 0 0 0 0 0
83   C      1.0    1.00000 0.0000 0.0000 1.0000 0 0 0 0 0
84   D      1.0    0.53846 0.6667 0.0000 0.3333 0 0 0 0 0
85   E      1.0    0.57143 0.3333 0.0000 0.6667 0 0 0 0 0
86   F      1.0    0.40625 0.8333 0.0000 0.1667 0 0 0 0 0
87   G      1.0    0.45000 0.0000 0.0000 1.0000 0 0 0 0 0
88   H      1.0    0.50000 1.0000 0.0000 0.0000 0 0 0 0 0
```

For the possibilistic model, the summary of results consists of: (1) the efficiency scores and (2) the reference sets. These are the results that are exported to the Excel workbook (see Figure 48).

Figure 48. Summary results in Excel.


ResultsDEA2018-11-19_02/31/36.xlsx (Sólo lectura)

	A	B	C	D	E	F	G	H	I	J	K
1	DMU	alpha_cut	efficiency	A	B	C	D	E	F	G	H
2	A	0	1	1	0	0	0	0	0	0	0
3	B	0	1	0	1	0	0	0	0	0	0
4	C	0	1	0	0	1	0	0	0	0	0
5	D	0	0,75	0	0,5	0,5	0	0	0	0	0
6	E	0	0,64286	0	0,1429	0,8571	0	0	0	0	0
7	F	0	0,60504	0	0,5714	0,4286	0	0	0	0	0
8	G	0	1	0	0	0	0	0	0	1	0
9	H	0	0,69231	0	1	0	0	0	0	0	0
10	A	0,1	1	1	0	0	0	0	0	0	0
11	B	0,1	1	0	1	0	0	0	0	0	0
12	C	0,1	1	0	0	1	0	0	0	0	0
13	D	0,1	0,73957	0	0,5071	0,4929	0	0	0	0	0
14	E	0,1	0,63977	0	0,1571	0,8429	0	0	0	0	0
15	F	0,1	0,59527	0	0,5857	0,4143	0	0	0	0	0
16	G	0,1	1	0	0	0	0	0	0	1	0
17	H	0,1	0,68992	0	1	0	0	0	0	0	0
18	A	0,2	1	1	0	0	0	0	0	0	0
19	B	0,2	1	0	1	0	0	0	0	0	0
20	C	0,2	1	0	0	1	0	0	0	0	0
21	D	0,2	0,72919	0	0,5143	0,4857	0	0	0	0	0
22	E	0,2	0,63688	0	0,1714	0,8286	0	0	0	0	0
23	F	0,2	0,58571	0	0,6	0,4	0	0	0	0	0
24	G	0,2	1	0	0	0	0	0	0	1	0
25	H	0,2	0,6875	0	1	0	0	0	0	0	0
26	A	0,3	1	1	0	0	0	0	0	0	0
27	B	0,3	1	0	1	0	0	0	0	0	0
28	C	0,3	1	0	0	1	0	0	0	0	0
29	D	0,3	0,70809	0,6083	0	0,3917	0	0	0	0	0
30	E	0,3	0,63105	0,2167	0	0,7833	0	0	0	0	0
31	F	0,3	0,56597	0,7167	0	0,2833	0	0	0	0	0
32	G	0,3	1	0	0	0	0	0	0	1	0
33	H	0,3	0,68504	0	1	0	0	0	0	0	0
34	A	0,4	1	1	0	0	0	0	0	0	0
35	B	0,4	0,97674	1	0	0	0	0	0	0	0
36	C	0,4	1	0	0	1	0	0	0	0	0
37	D	0,4	[illegible]	0,6167	0	0,3833	0	0	0	0	0

We save the script "*Summary_DEA_fuzzy*".

7.6. Graphical results. The plot() function.

With the plot() function we can obtain some plots of the results from a conventional DEA model, Malmquist index or cross-efficiency model. This is shown in Examples 14, 15, and 16 respectively.

The plot() function is used in the following way:

plot(*x*)

where **x** is the object in which the result of a DEA model has been stored.

After using the plot() function, we will get the plot in the *Viewer* (right bottom window). We can save a plot by clicking on the *Export* tag of the *Viewer*. It is possible to save the plot as an image in format: "*png*", "*jpeg*" or "*tiff*". We can copy the plot to the system clipboard and then paste it, for example, in a word document.

Graphs for fuzzy DEA models are not available still in this version of **deaR**.

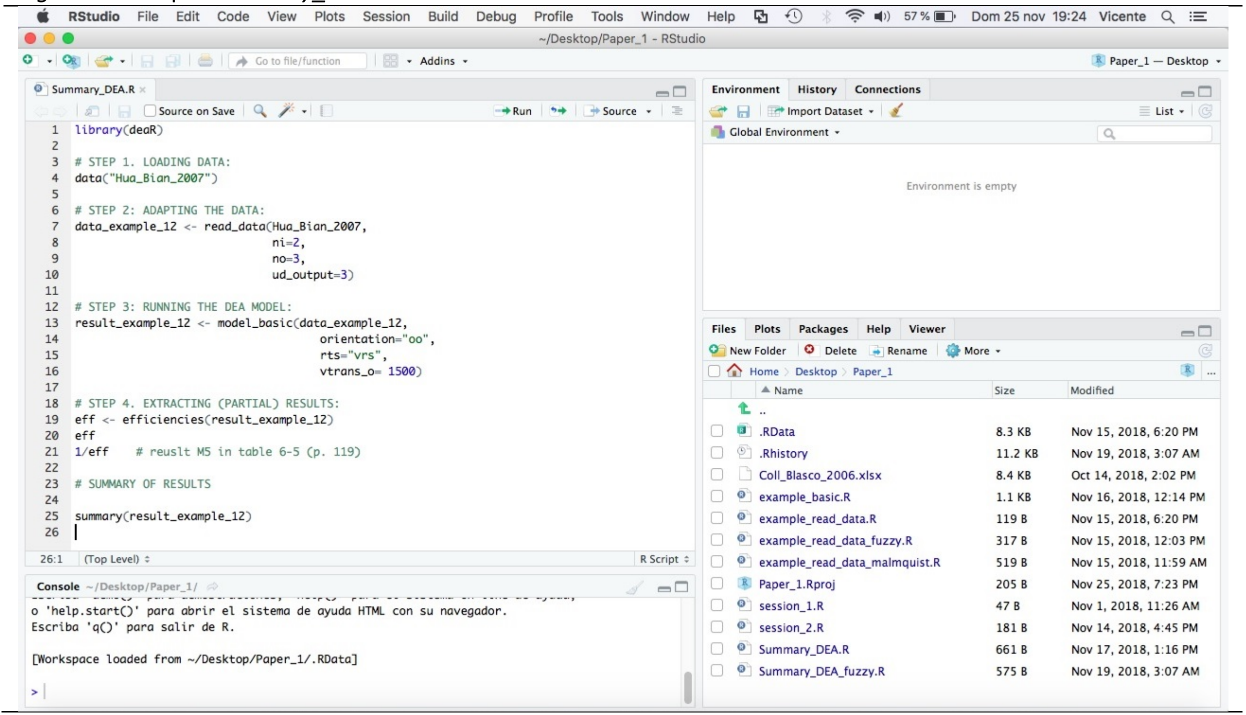
We are currently working to improve the graphical results in **deaR**, which will be included in the next version.

Example 14. Plot: Basic DEA model.

We open the script "*Summary_DEA*". If we have closed the working session, we need to open the project "*Paper_1*", load **deaR**, and open the script.

Our working session should be similar to the one shown in Figure 49.

Figure 49. Script "*Summary_DEA*".



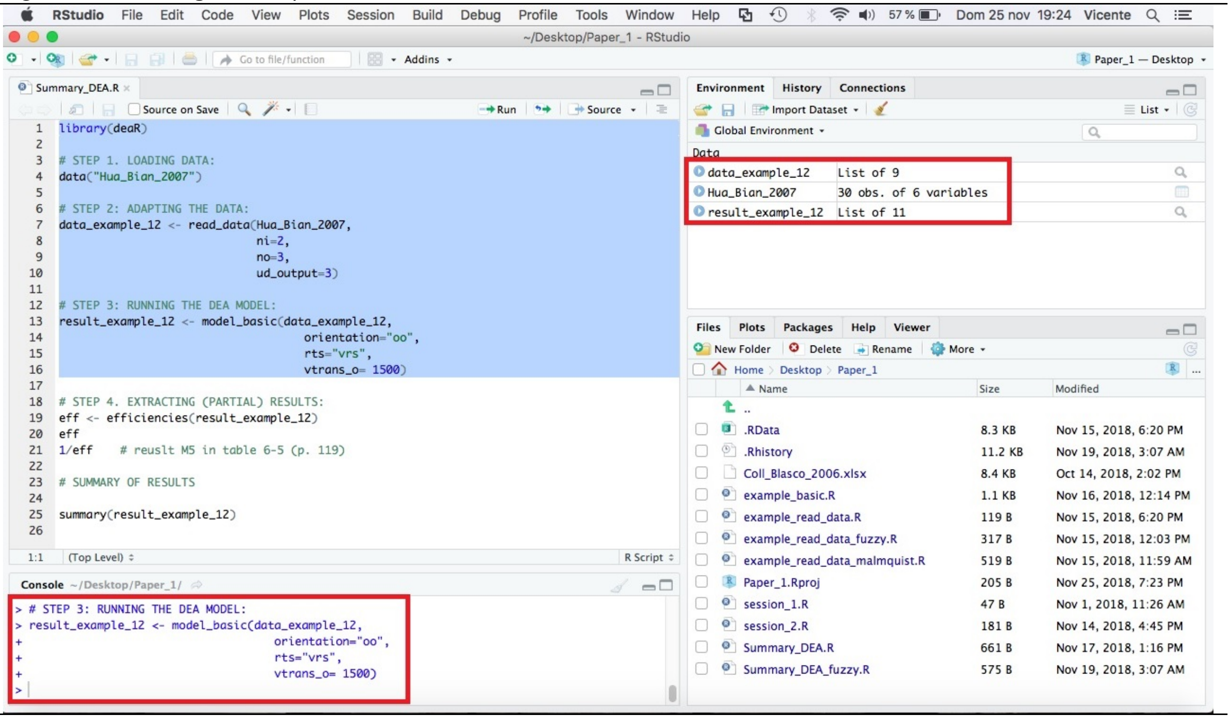
```
1   library(deaR)
2
3   # STEP 1. LOADING DATA:
4   data("Hua_Bian_2007")
5
6   # STEP 2: ADAPTING THE DATA:
7   data_example_12 <- read_data(Hua_Bian_2007,
8                                ni=2,
9                                no=3,
10                               ud_output=3)
11
12  # STEP 3: RUNNING THE DEA MODEL:
13  result_example_12 <- model_basic(data_example_12,
14                                   orientation="oo",
15                                   rts="vrs",
16                                   vtrans_o= 1500)
17
18  # STEP 4. EXTRACTING (PARTIAL) RESULTS:
19  eff <- efficiencies(result_example_12)
20  eff
21  1/eff    # reuslt M5 in table 6-5 (p. 119)
22
23  # SUMMARY OF RESULTS
24
25  summary(result_example_12)
26
```

```
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

[Workspace loaded from ~/Desktop/Paper_1/.RData]

> 
```

Name	Size	Modified
..		
.RData	8.3 KB	Nov 15, 2018, 6:20 PM
.Rhistory	11.2 KB	Nov 19, 2018, 3:07 AM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
example_basic.R	1.1 KB	Nov 16, 2018, 12:14 PM
example_read_data.R	119 B	Nov 15, 2018, 6:20 PM
example_read_data_fuzzy.R	317 B	Nov 15, 2018, 12:03 PM
example_read_data_malmquist.R	519 B	Nov 15, 2018, 11:59 AM
Paper_1.Rproj	205 B	Nov 25, 2018, 7:23 PM
session_1.R	47 B	Nov 1, 2018, 11:26 AM
session_2.R	181 B	Nov 14, 2018, 4:45 PM
Summary_DEA.R	661 B	Nov 17, 2018, 1:16 PM
Summary_DEA_fuzzy.R	575 B	Nov 19, 2018, 3:07 AM

As Figure 50 shows, we select from line 1, **library(deaR)**, to line 16 and run the selection. Immediately, three objects are listed in the *Environment*: "*data_example_12*", "*Hua_Bian_2007*" and "*result_example_12*".

Figure 50. Running the script.



```
1   library(deaR)
2
3   # STEP 1. LOADING DATA:
4   data("Hua_Bian_2007")
5
6   # STEP 2: ADAPTING THE DATA:
7   data_example_12 <- read_data(Hua_Bian_2007,
8                                ni=2,
9                                no=3,
10                               ud_output=3)
11
12  # STEP 3: RUNNING THE DEA MODEL:
13  result_example_12 <- model_basic(data_example_12,
14                                   orientation="oo",
15                                   rts="vrs",
16                                   vtrans_o= 1500)
17
18  # STEP 4. EXTRACTING (PARTIAL) RESULTS:
19  eff <- efficiencies(result_example_12)
20  eff
21  1/eff    # reuslt M5 in table 6-5 (p. 119)
22
23  # SUMMARY OF RESULTS
24
25  summary(result_example_12)
26
```

```
> # STEP 3: RUNNING THE DEA MODEL:
> result_example_12 <- model_basic(data_example_12,
+                                  orientation="oo",
+                                  rts="vrs",
+                                  vtrans_o= 1500)
> 
```

Data

data_example_12	List of 9
Hua_Bian_2007	30 obs. of 6 variables
result_example_12	List of 11

Name	Size	Modified
..		
.RData	8.3 KB	Nov 15, 2018, 6:20 PM
.Rhistory	11.2 KB	Nov 19, 2018, 3:07 AM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
example_basic.R	1.1 KB	Nov 16, 2018, 12:14 PM
example_read_data.R	119 B	Nov 15, 2018, 6:20 PM
example_read_data_fuzzy.R	317 B	Nov 15, 2018, 12:03 PM
example_read_data_malmquist.R	519 B	Nov 15, 2018, 11:59 AM
Paper_1.Rproj	205 B	Nov 25, 2018, 7:23 PM
session_1.R	47 B	Nov 1, 2018, 11:26 AM
session_2.R	181 B	Nov 14, 2018, 4:45 PM
Summary_DEA.R	661 B	Nov 17, 2018, 1:16 PM
Summary_DEA_fuzzy.R	575 B	Nov 19, 2018, 3:07 AM

The results of the DEA model are assigned to (or stored in) the object "*result_example_12*". To plot some of these results, we will write the following instruction on the line 27 of the script:

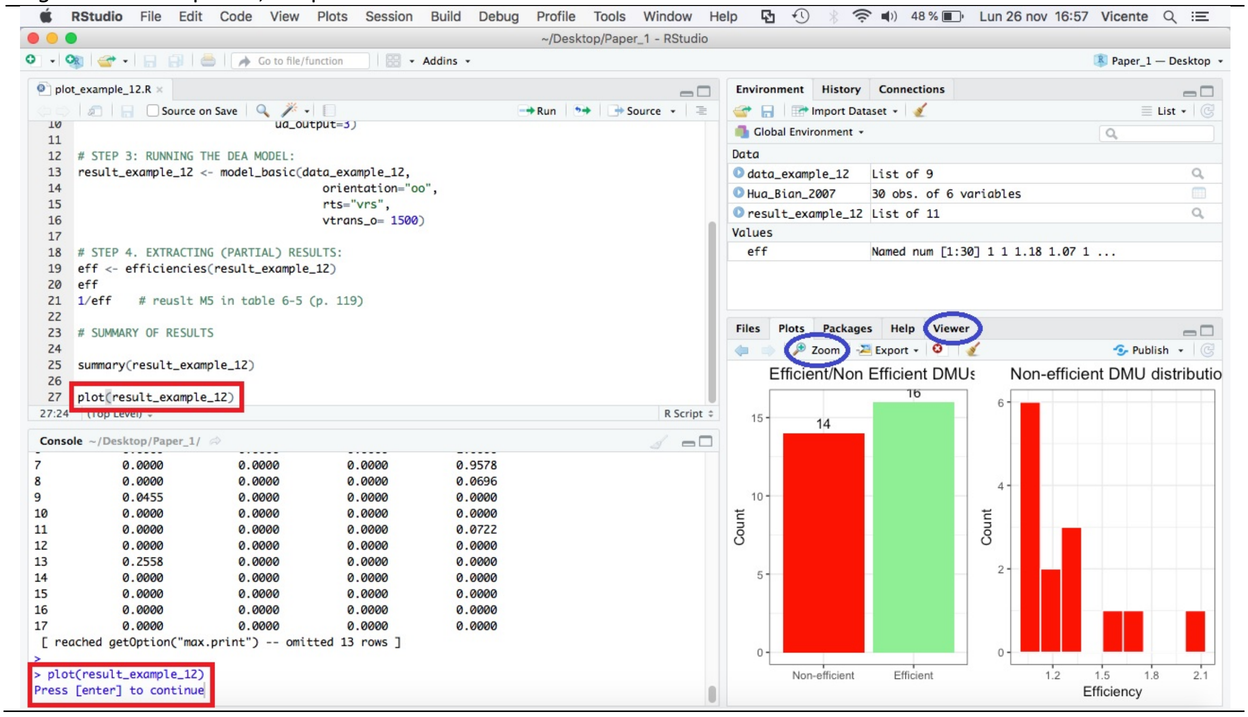
plot(result_example_12)

and run it. The following message will be displayed in the *Console*:

Press [enter] to continue

and a plot will be shown in the *Viewer* tag (right bottom window), as we can see in Figure 51.

Figure 51. Example 12, Graphic 1.



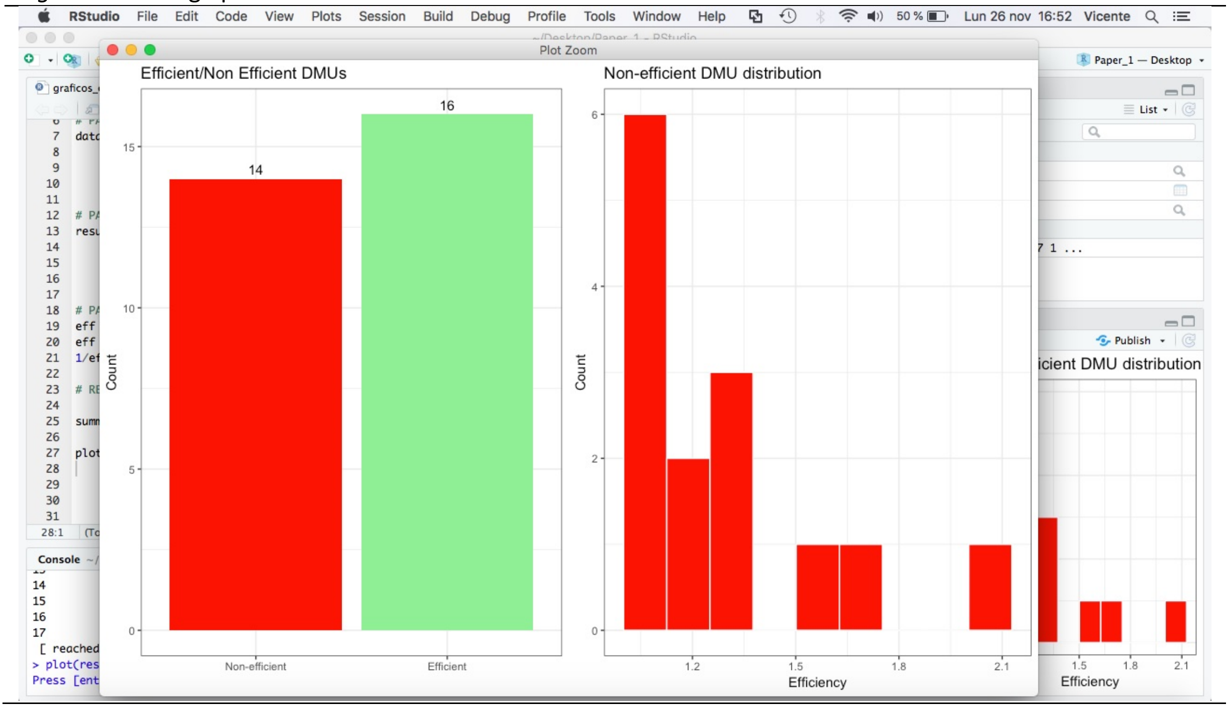
```
                                ud_output=3)

# STEP 3: RUNNING THE DEA MODEL:
result_example_12 <- model_basic(data_example_12,
                                 orientation="oo",
                                 rts="vrs",
                                 vtrans_o= 1500)

# STEP 4. EXTRACTING (PARTIAL) RESULTS:
eff <- efficiencies(result_example_12)
eff
1/eff    # reuslt M5 in table 6-5 (p. 119)

# SUMMARY OF RESULTS

summary(result_example_12)

plot(result_example_12)
```

```
7          0.0000      0.0000      0.0000      0.9578
8          0.0000      0.0000      0.0000      0.0696
9          0.0455      0.0000      0.0000      0.0000
10         0.0000      0.0000      0.0000      0.0000
11         0.0000      0.0000      0.0000      0.0722
12         0.0000      0.0000      0.0000      0.0000
13         0.2558      0.0000      0.0000      0.0000
14         0.0000      0.0000      0.0000      0.0000
15         0.0000      0.0000      0.0000      0.0000
16         0.0000      0.0000      0.0000      0.0000
17         0.0000      0.0000      0.0000      0.0000
 [ reached getOption("max.print") -- omitted 13 rows ]
>
> plot(result_example_12)
Press [enter] to continue
```

```
Data
data_example_12     List of 9
Hua_Bian_2007       30 obs. of 6 variables
result_example_12   List of 11
Values
eff                 Named num [1:30] 1 1 1.18 1.07 1 ...
```

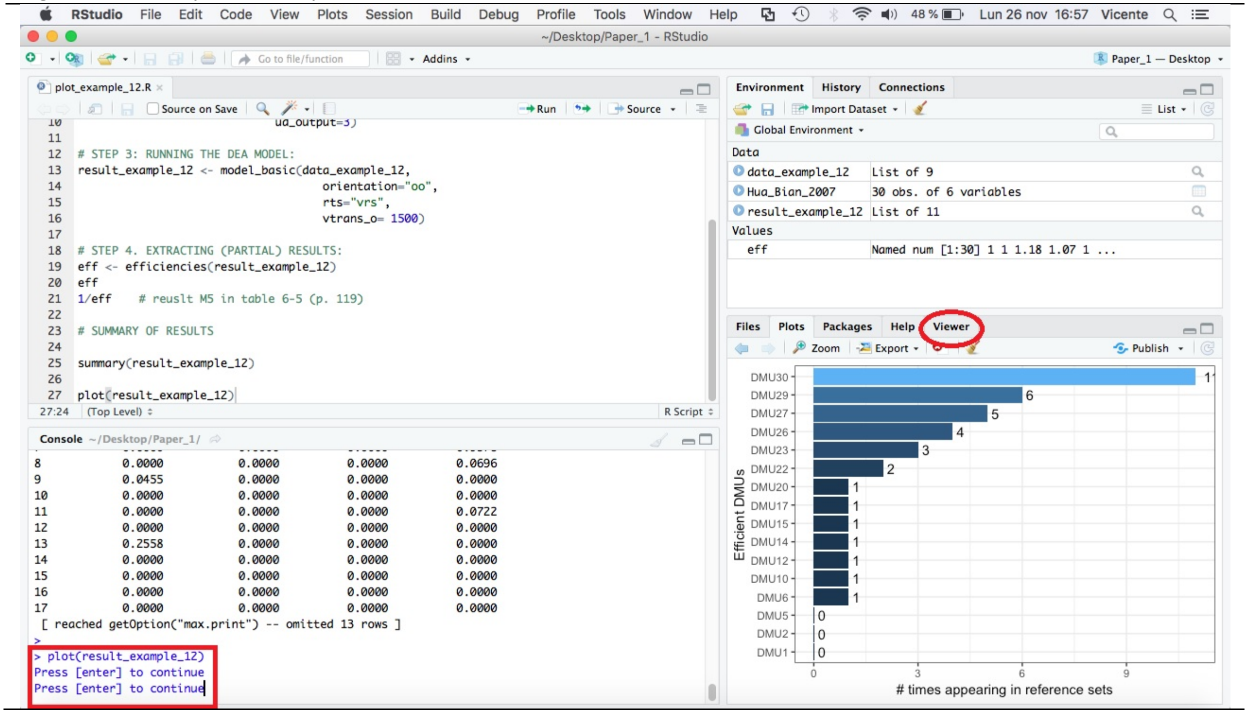
This Graphic 1 shows the number of efficient and inefficient DMUs (right graph), and the distribution of the efficiency score of the latter as well (left graph). If we click on the Zoom bottom, the plot will be enlarged (see Figure 52).

Figura 52. Enlarge plot.



Now, if we press the *Enter* key, a second graphic will be displayed. On this occasion, what is depicted is the number of times an efficient DMU is part of the inefficient DMUs reference set.

Figure 53. Example 12, Graphic 2.



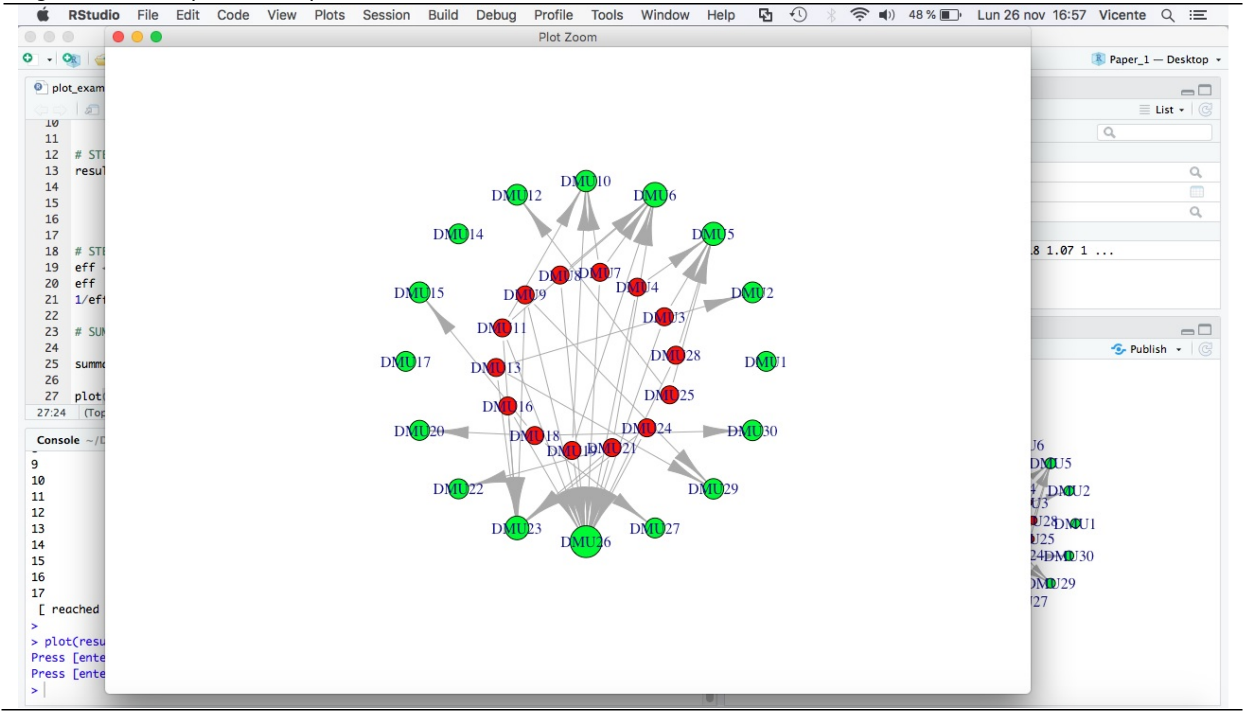
```
                                   ud_output=3)

# STEP 3: RUNNING THE DEA MODEL:
result_example_12 <- model_basic(data_example_12,
                                 orientation="oo",
                                 rts="vrs",
                                 vtrans_o= 1500)

# STEP 4. EXTRACTING (PARTIAL) RESULTS:
eff <- efficiencies(result_example_12)
eff
1/eff    # reuslt M5 in table 6-5 (p. 119)

# SUMMARY OF RESULTS

summary(result_example_12)

plot(result_example_12)
```

```
8         0.0000      0.0000      0.0000      0.0696
9         0.0455      0.0000      0.0000      0.0000
10        0.0000      0.0000      0.0000      0.0000
11        0.0000      0.0000      0.0000      0.0722
12        0.0000      0.0000      0.0000      0.0000
13        0.2558      0.0000      0.0000      0.0000
14        0.0000      0.0000      0.0000      0.0000
15        0.0000      0.0000      0.0000      0.0000
16        0.0000      0.0000      0.0000      0.0000
17        0.0000      0.0000      0.0000      0.0000
 [ reached getOption("max.print") -- omitted 13 rows ]
>
> plot(result_example_12)
Press [enter] to continue
Press [enter] to continue
```

Data	
data_example_12	List of 9
Hua_Bian_2007	30 obs. of 6 variables
result_example_12	List of 11
Values	
eff	Named num [1:30] 1 1 1.18 1.07 1 ...

Finally, by pressing the *Enter* key again we can see the last graphic (see Figure 54).

Figure 54. Example 12, Graphic 3.



In Figure 54 we see a network graph in which the green circles represent the efficient DMUs and the red circles the inefficient ones. This graph shows how inefficient DMUs are related to efficient DMUs, trying to convey the idea that the efficient ones form the efficient frontier. In addition, notice that not all green circles have the same size. In this case, the size of the circle aims to convey the idea of how important is the efficient DMU for the set of inefficient DMUs.

We save the script as: “*plot_example_12*”.

Example 15. Plot: Malmquist index.

We create a new script with the name “*plot_malmquist*”. If we have closed the working session, we need to open the project “*Paper_1*”, load **deaR**, and create the script.

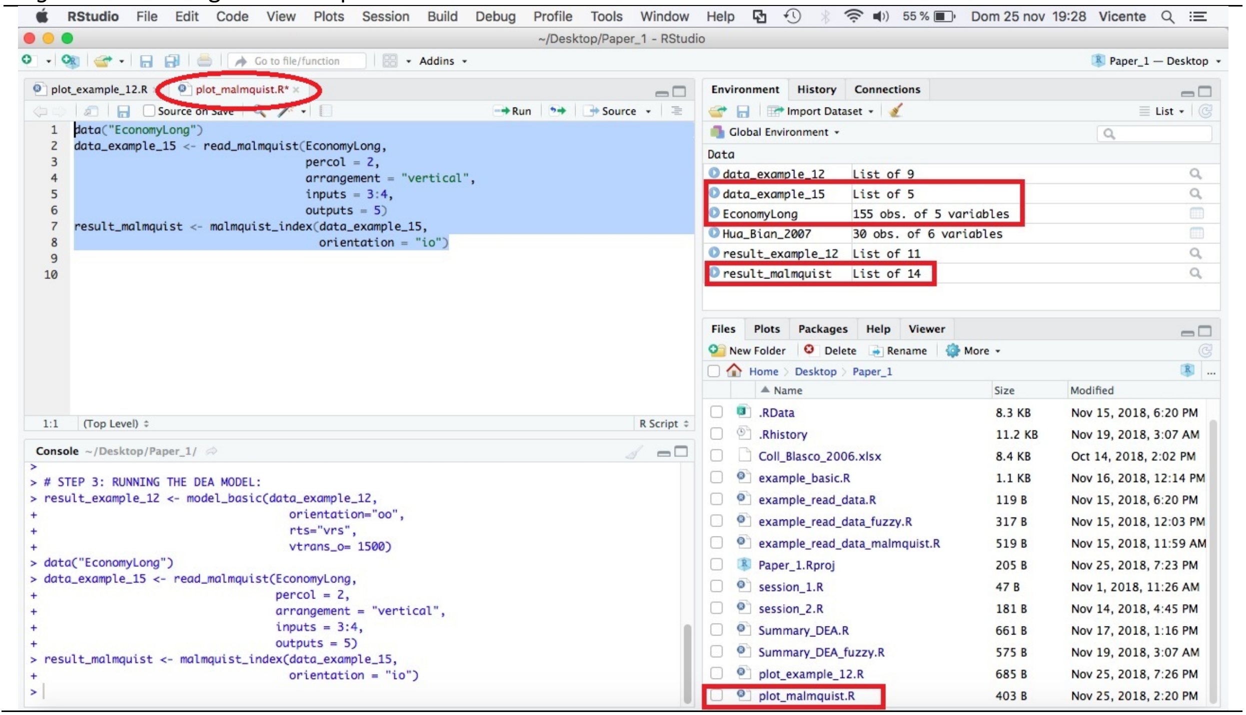
Now, we are going to run the Malmquist index. For that, we write into the script:

```
data("EconomyLong")

data_example_15 <- read_malmquist(EconomyLong,
                                  percol = 2,
                                  arrangement = "vertical",
                                  inputs = 3:4,
                                  outputs = 5)

result_malmquist <- malmquist_index(data_example_15,
                                    orientation = "io")
```

Note: The data are in long format.

We run the instructions of the script (see Figure 55).

Figure 55. Running the Malmquist index.



```
data("EconomyLong")
data_example_15 <- read_malmquist(EconomyLong,
                                  percol = 2,
                                  arrangement = "vertical",
                                  inputs = 3:4,
                                  outputs = 5)
result_malmquist <- malmquist_index(data_example_15,
                                    orientation = "io")
```

Console ~/Desktop/Paper_1/

```
>
> # STEP 3: RUNNING THE DEA MODEL:
> result_example_12 <- model_basic(data_example_12,
+                                  orientation="oo",
+                                  rts="vrs",
+                                  vtrans_o= 1500)
> data("EconomyLong")
> data_example_15 <- read_malmquist(EconomyLong,
+                                   percol = 2,
+                                   arrangement = "vertical",
+                                   inputs = 3:4,
+                                   outputs = 5)
> result_malmquist <- malmquist_index(data_example_15,
+                                     orientation = "io")
> |
```

Environment — Global Environment — Data

data_example_12	List of 9
data_example_15	List of 5
EconomyLong	155 obs. of 5 variables
Hua_Bian_2007	30 obs. of 6 variables
result_example_12	List of 11
result_malmquist	List of 14

Files — Home > Desktop > Paper_1

Name	Size	Modified
.RData	8.3 KB	Nov 15, 2018, 6:20 PM
.Rhistory	11.2 KB	Nov 19, 2018, 3:07 AM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
example_basic.R	1.1 KB	Nov 16, 2018, 12:14 PM
example_read_data.R	119 B	Nov 15, 2018, 6:20 PM
example_read_data_fuzzy.R	317 B	Nov 15, 2018, 12:03 PM
example_read_data_malmquist.R	519 B	Nov 15, 2018, 11:59 AM
Paper_1.Rproj	205 B	Nov 25, 2018, 7:23 PM
session_1.R	47 B	Nov 1, 2018, 11:26 AM
session_2.R	181 B	Nov 14, 2018, 4:45 PM
Summary_DEA.R	661 B	Nov 17, 2018, 1:16 PM
Summary_DEA_fuzzy.R	575 B	Nov 19, 2018, 3:07 AM
plot_example_12.R	685 B	Nov 25, 2018, 7:26 PM
plot_malmquist.R	403 B	Nov 25, 2018, 2:20 PM

The results of the Malmquist index are stored in the object “*result_malmquist*”. Therefore, to get the plots we write into the script:

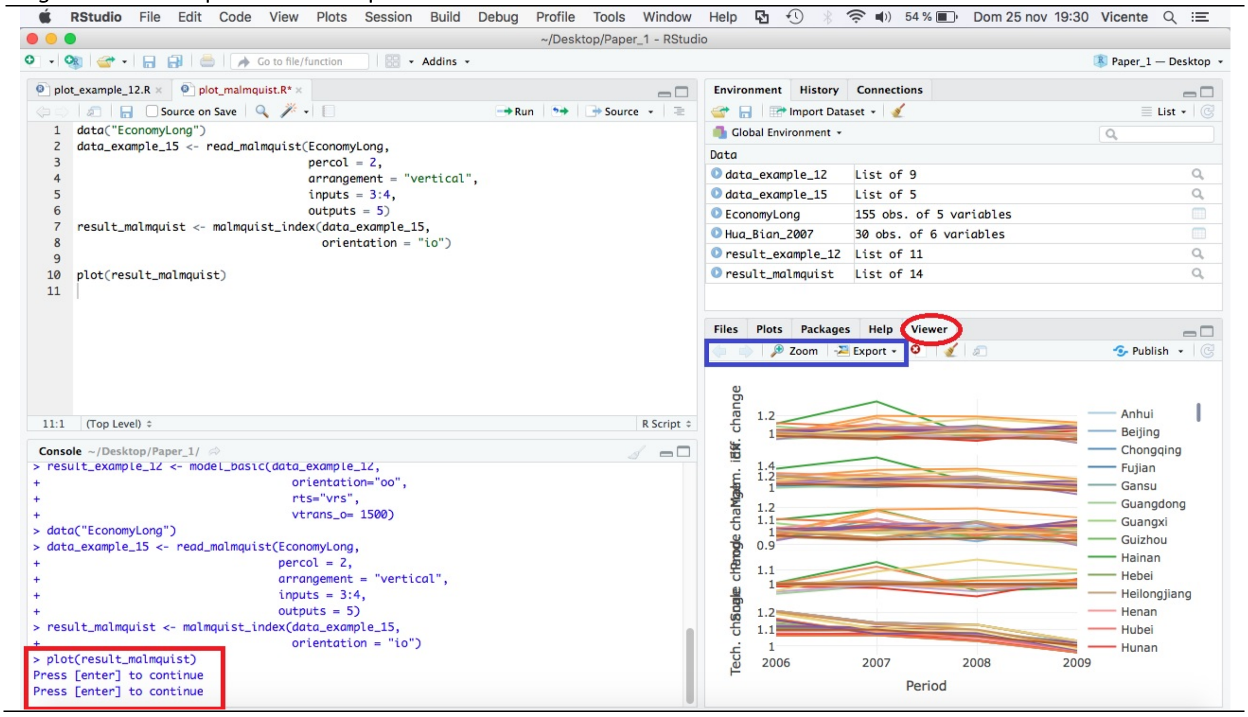
```
plot(result_malmquist)
```

and run the instruction. The following message will be displayed in the *Console*:

Press [enter] to continue

By pressing the *Enter* key a plot will be shown in the *Viewer* (right bottom window) (see Figure 56). We can click on the *Zoom* to see the plot better. In this first plot, for each DMU the Malmquist index and its components are drawn over the years. If there are many DMUs you will not probably see them very well. However, it is possible to select the DMUS the DMUs of your interest to focus on them (and the other DMUs in the graph will disappear).

Figure 56. Malmquist index: Graphic 1.



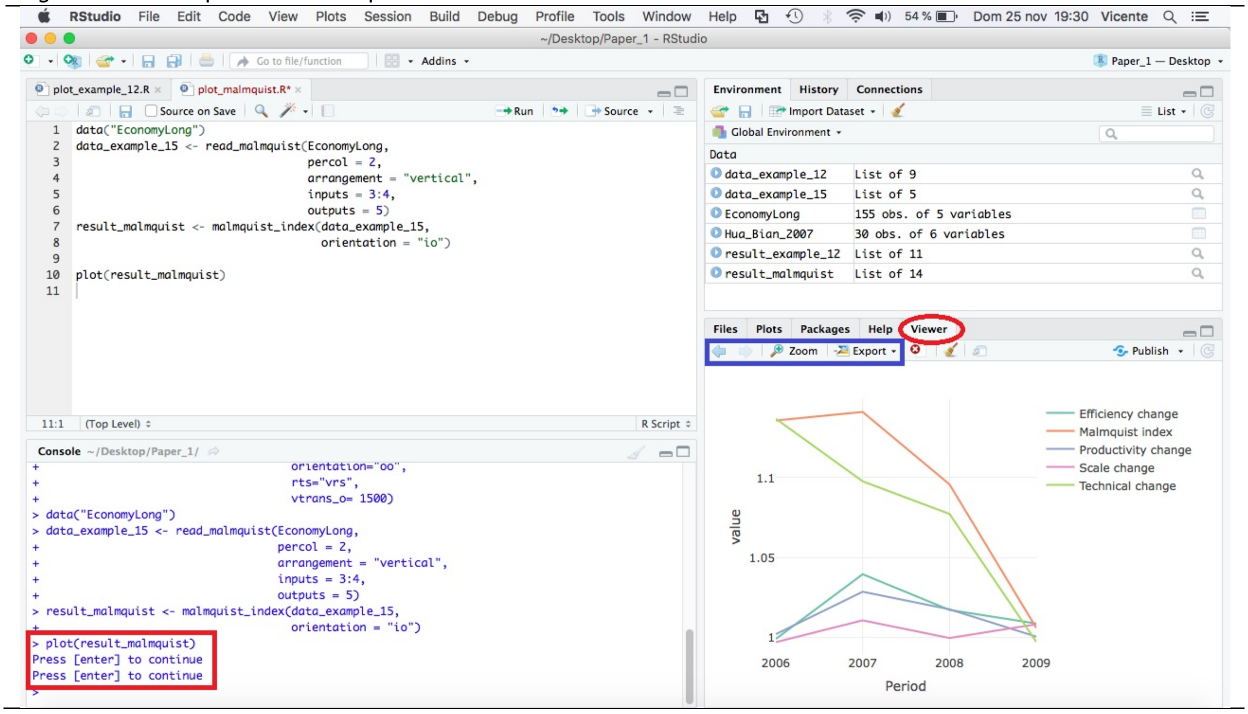
```
data("EconomyLong")
data_example_15 <- read_malmquist(EconomyLong,
                                  percol = 2,
                                  arrangement = "vertical",
                                  inputs = 3:4,
                                  outputs = 5)
result_malmquist <- malmquist_index(data_example_15,
                                    orientation = "io")

plot(result_malmquist)
```

data_example_12	List of 9
data_example_15	List of 5
EconomyLong	155 obs. of 5 variables
Hua_Bian_2007	30 obs. of 6 variables
result_example_12	List of 11
result_malmquist	List of 14

```
> result_example_12 <- model_basic(data_example_12,
+                                  orientation="oo",
+                                  rts="vrs",
+                                  vtrans_o= 1500)
> data("EconomyLong")
> data_example_15 <- read_malmquist(EconomyLong,
+                                   percol = 2,
+                                   arrangement = "vertical",
+                                   inputs = 3:4,
+                                   outputs = 5)
> result_malmquist <- malmquist_index(data_example_15,
+                                     orientation = "io")
> plot(result_malmquist)
Press [enter] to continue
Press [enter] to continue
```

If we press the *Enter* key (in the *Console*) again, a second plot is created (see Figure 57).

Figure 57. Malmquist index: Graphic 2.



```
> result_example_12 <- model_basic(data_example_12,
+                                  orientation="oo",
+                                  rts="vrs",
+                                  vtrans_o= 1500)
> data("EconomyLong")
> data_example_15 <- read_malmquist(EconomyLong,
+                                   percol = 2,
+                                   arrangement = "vertical",
+                                   inputs = 3:4,
+                                   outputs = 5)
> result_malmquist <- malmquist_index(data_example_15,
+                                     orientation = "io")
> plot(result_malmquist)
Press [enter] to continue
Press [enter] to continue
>
```

On this occasion, the geometric means of Malmquist index and its components are drawn over the years. We can enlarge the plot by clicking on the *Zoom* and go back (or go forward) by clicking on the arrows. Here, it is possible to select the Malmquist index components to be shown.

We save "*plot_malmquist*".

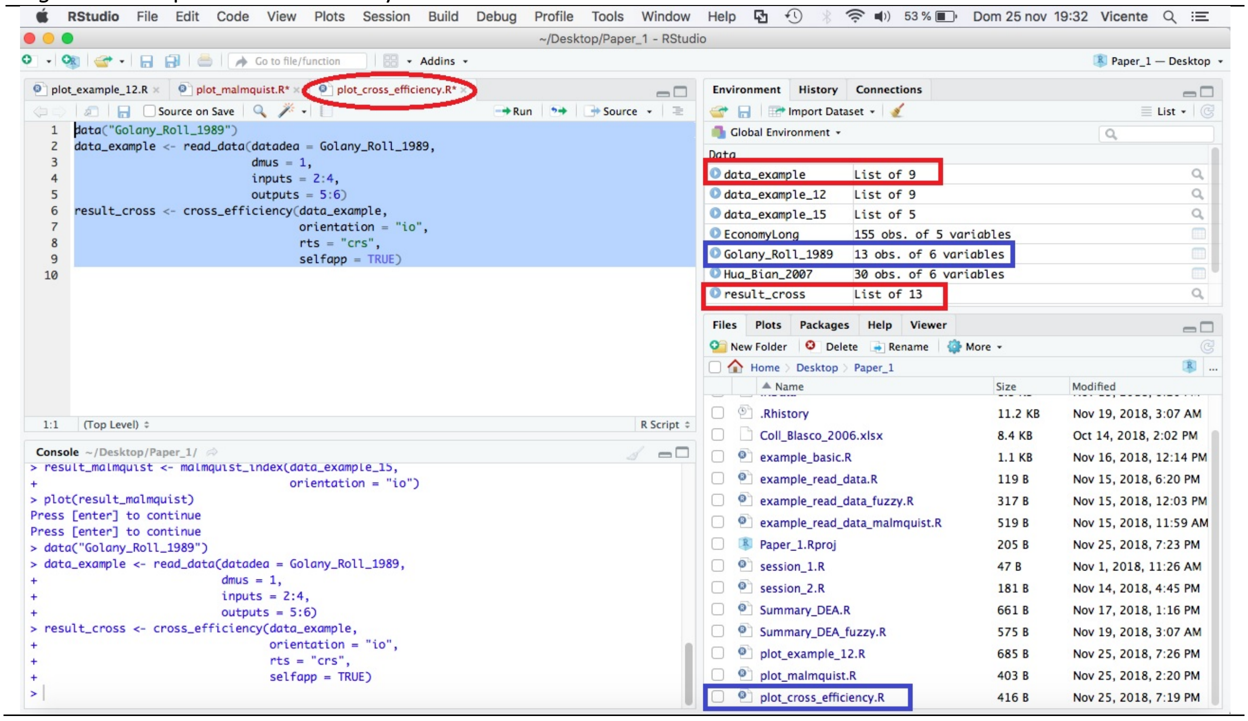
Example 16. Plot: Cross efficiency.

We create the new script "*plot_cross_efficiency*". If we have closed the working session, we need to open the project "*Paper_1*", load **deaR**, and create the script.

We write and run the followings instructions (see Figure 58):

```
data("Golany_Roll_1989")

data_example <- read_data(datadea = Golany_Roll_1989,
                          dmus = 1,
                          inputs = 2:4,
                          outputs = 5:6)

result_cross <- cross_efficiency(data_example,
                                 orientation = "io",
                                 rts = "crs",
                                 selfapp = TRUE)
```

Figure 58. Script of cross efficiency.



RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
53 % Dom 25 nov 19:32 Vicente
~/Desktop/Paper_1 - RStudio
Go to file/function
Addins
Paper_1 — Desktop
plot_example_12.R
plot_malmquist.R*
plot_cross_efficiency.R*
Source on Save
Run
Source
Environment History Connections
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home › Desktop › Paper_1

```
data("Golany_Roll_1989")
data_example <- read_data(datadea = Golany_Roll_1989,
                          dmus = 1,
                          inputs = 2:4,
                          outputs = 5:6)
result_cross <- cross_efficiency(data_example,
                                 orientation = "io",
                                 rts = "crs",
                                 selfapp = TRUE)
```

1:1 (Top Level) R Script

Console ~/Desktop/Paper_1/

```
> result_malmquist <- malmquist_index(data_example_15,
+                                     orientation = "io")
> plot(result_malmquist)
Press [enter] to continue
Press [enter] to continue
> data("Golany_Roll_1989")
> data_example <- read_data(datadea = Golany_Roll_1989,
+                           dmus = 1,
+                           inputs = 2:4,
+                           outputs = 5:6)
> result_cross <- cross_efficiency(data_example,
+                                  orientation = "io",
+                                  rts = "crs",
+                                  selfapp = TRUE)
> |
```

Data

data_example	List of 9
data_example_12	List of 9
data_example_15	List of 5
EconomyLong	155 obs. of 5 variables
Golany_Roll_1989	13 obs. of 6 variables
Hua_Bian_2007	30 obs. of 6 variables
result_cross	List of 13

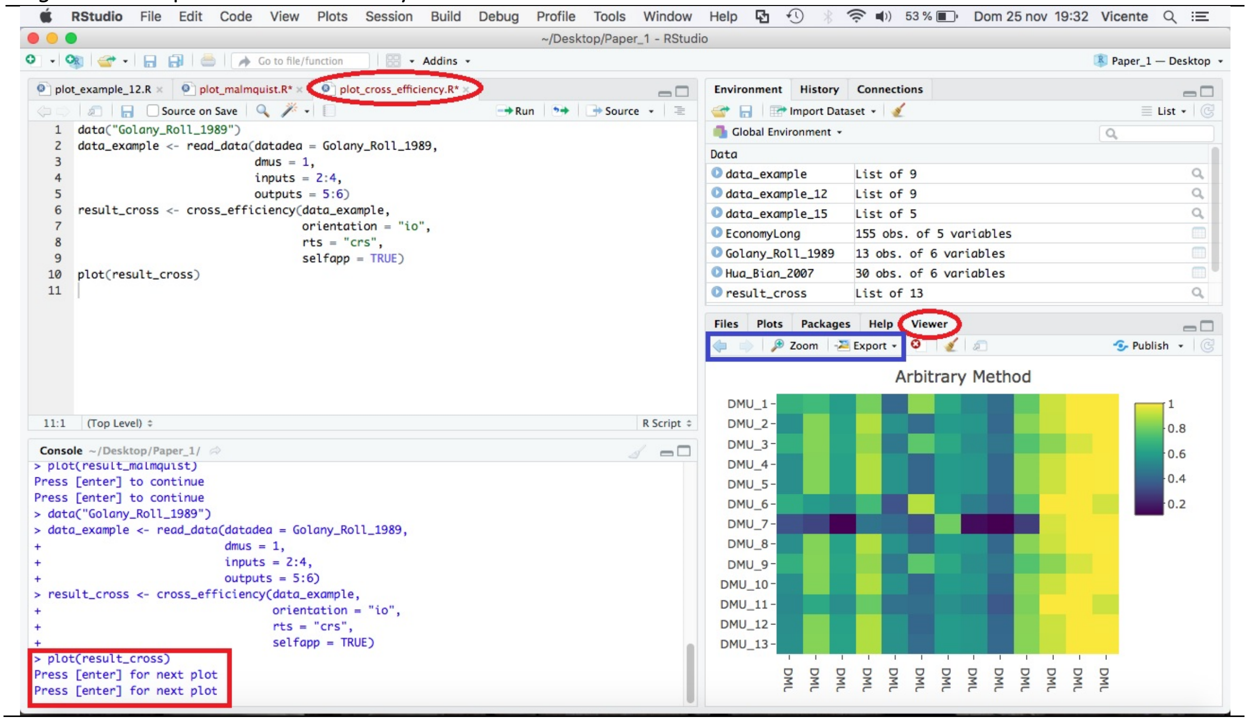
Name	Size	Modified
.Rhistory	11.2 KB	Nov 19, 2018, 3:07 AM
Coll_Blasco_2006.xlsx	8.4 KB	Oct 14, 2018, 2:02 PM
example_basic.R	1.1 KB	Nov 16, 2018, 12:14 PM
example_read_data.R	119 B	Nov 15, 2018, 6:20 PM
example_read_data_fuzzy.R	317 B	Nov 15, 2018, 12:03 PM
example_read_data_malmquist.R	519 B	Nov 15, 2018, 11:59 AM
Paper_1.Rproj	205 B	Nov 25, 2018, 7:23 PM
session_1.R	47 B	Nov 1, 2018, 11:26 AM
session_2.R	181 B	Nov 14, 2018, 4:45 PM
Summary_DEA.R	661 B	Nov 17, 2018, 1:16 PM
Summary_DEA_fuzzy.R	575 B	Nov 19, 2018, 3:07 AM
plot_example_12.R	685 B	Nov 25, 2018, 7:26 PM
plot_malmquist.R	403 B	Nov 25, 2018, 2:20 PM
plot_cross_efficiency.R	416 B	Nov 25, 2018, 7:19 PM

In the object "*result_cross*" are stored the results of the cross efficiency corresponding to the models: arbitrary, benevolent and aggressive. All these results are drawn like a heat map by using the plot() function. For that, we write into the script:

```
plot(result_cross)
```

and press *Enter* in the *Console* to display the different graphs. The result should be similar to the one shown in Figure 59.

Figure 59. Graphs of cross efficiency.



```
data("Golany_Roll_1989")
data_example <- read_data(datadea = Golany_Roll_1989,
                          dmus = 1,
                          inputs = 2:4,
                          outputs = 5:6)
result_cross <- cross_efficiency(data_example,
                                 orientation = "io",
                                 rts = "crs",
                                 selfapp = TRUE)
plot(result_cross)
```

```
> plot(result_malmquist)
Press [enter] to continue
Press [enter] to continue
> data("Golany_Roll_1989")
> data_example <- read_data(datadea = Golany_Roll_1989,
+                           dmus = 1,
+                           inputs = 2:4,
+                           outputs = 5:6)
> result_cross <- cross_efficiency(data_example,
+                                  orientation = "io",
+                                  rts = "crs",
+                                  selfapp = TRUE)
> plot(result_cross)
Press [enter] for next plot
Press [enter] for next plot
```

```
Data
data_example       List of 9
data_example_12    List of 9
data_example_15    List of 5
EconomyLong        155 obs. of 5 variables
Golany_Roll_1989   13 obs. of 6 variables
Hua_Bian_2007      30 obs. of 6 variables
result_cross       List of 13
```

We save the script "*plot_cross_efficiency*", close the project and quit RStudio.

Acknowledgements

The authors wish to thank Dra. Cristina Pardo-Garcia for her inestimable assistance reviewing this tutorial.

We hope you find **deaR** useful and you use it in your future research and teaching.

Any comments and suggestions to improve **deaR** will be really appreciated.