# 数据包络分析（Data Envelopment Analysis）with deaR 软件包

## 使用指导

**1.0 版本（中文）**

**（2018 年 11 月）**

翻译：王莹*


**Vicente Coll-Serrano(1)**
**Rafael Benítez(2)**
**Vicente J. Bolós(3)**

**(1)应用经济系.** Vicente.Coll@uv.es

**文化计量与定量分析 (MC2)**

**(2) 商业数学系.** Benitez.Suarez@uv.es

**(3) 商业数学系.** Vicente.Bolos@uv.es

**（西班牙）瓦伦西亚大学经济学院**

*** 云南大学文化发展研究院助理研究员，瓦伦西亚大学在读博士.** wangyingynu@163.com

# 目录

# 1. 概述

deaR 是一种新的、灵活的 R 软件包（免费软件），它允许用户运行各种建立在数据包络分析（Data Envelopment Analysis）基础上的模型。

本指导并非 R[1]软件的使用说明，而是**帮助非 R 软件用户使用 deaR**。

**如果你是一名 R 软件使用者，可直接跳过本节，前往第 7 节**。

我们希望 **deaR** 软件包能够成为学者、实践者、教师、学生和数据包络分析方法用户的参考软件。为此，我们欢迎任何能够改进 **deaR** 的意见和建议。我们也期待大家的建议和推荐，在新版的 **deaR** 软件包中加入一些在当前版本中没有被包括进去的新模型和新特征，诸如：stochastic DEA，network DEA，Malmquist index（other decompositions and bootstrapping），负值以及未预见的 input/output 结果的拓展等。

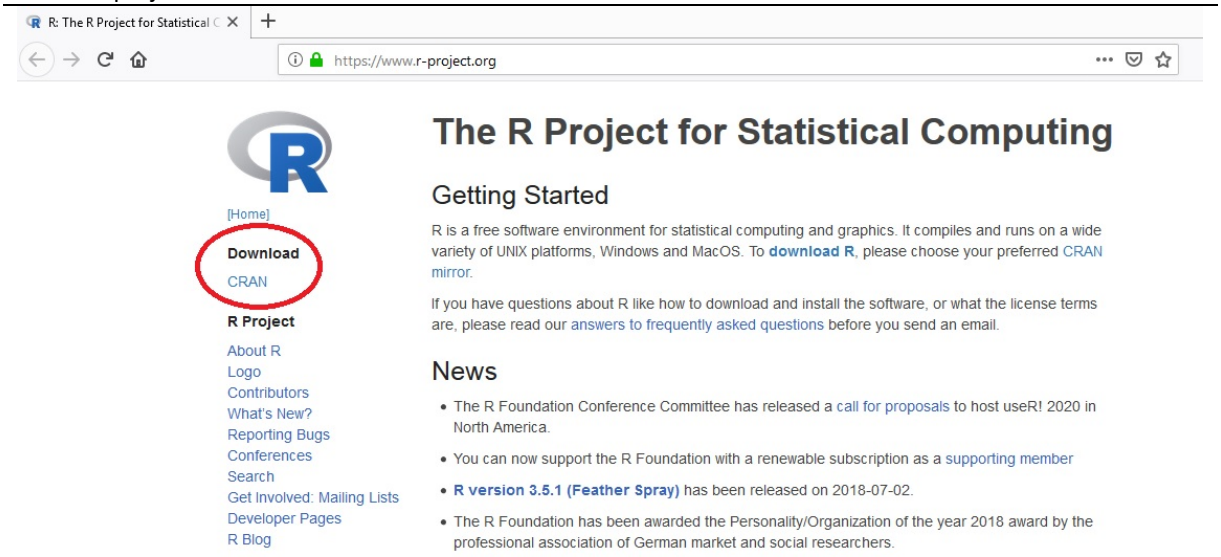
# 2. 下载并安装 R 和 RStudio

要使用 **deaR**，首先我们需要下载并安装 R 和 RStudio 软件。

## 2.1. 安装 R 软件

安装 R 软件，我们需要前往 R-project 的官方网站：http://www.r-project.org （参见图 1）

图 1：R-project 网站

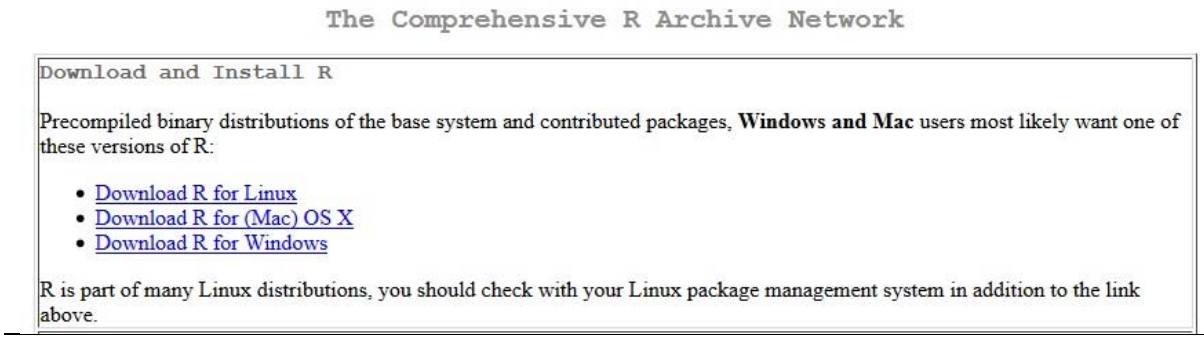



下载 R 软件，首先点击 *CRAN* 链接，选择距离我们所处位置最近的“镜像站点”（mirror site）。

其次，根据所使用计算机的操作系统选择适当的选项（参见图 2）。

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

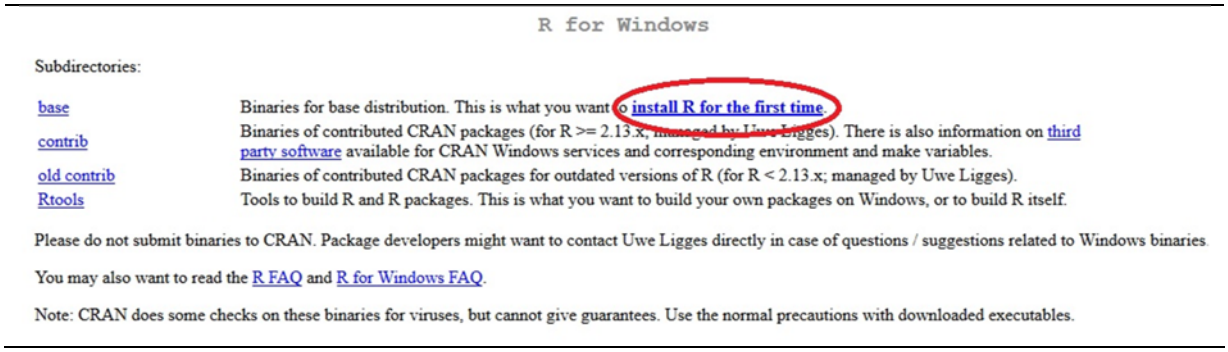
图 2：R 的不同版本




**a) 在 Windows 上安装 R**

点击 *Download R for Windows* 链接，我们将打开如下图所示的网页，然后点击 *install R for the first time*（参见图 3）

图 3： Windows 版的 R




接下来，点击 ***Download R 3.5.1 for Windows*** 并保存安装文件（参见图 4）。

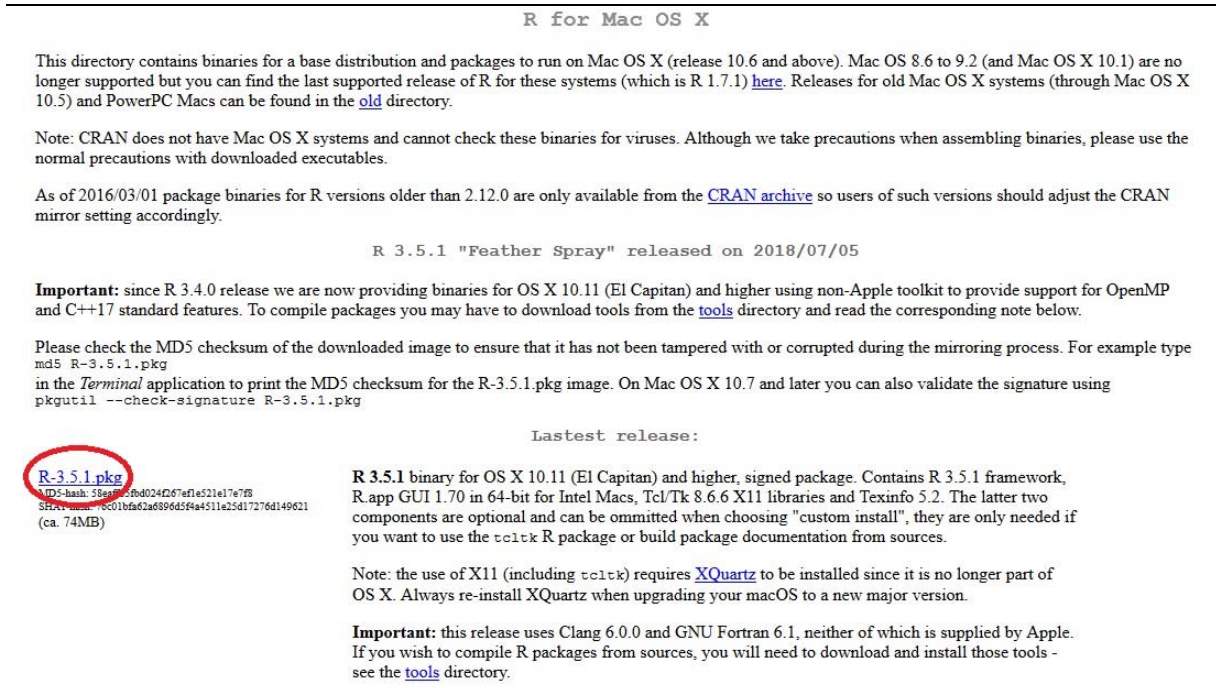
图 4：下载 Windows 版的 R


双击下载的文件以启动 R 的安装程序。

**b) 在 Mac 上安装 R**

点击 *Download R for (Mac) OS X*，打开如下图所示的网页，然后点击 ***R-3.5.1.pkg*** 下载安装文件（参见图 5）。

图 5：Mac 版的 R



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
SHA1-hash: 7bc01bfa62a6896d5f4a4511e25d17276d149621
(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.

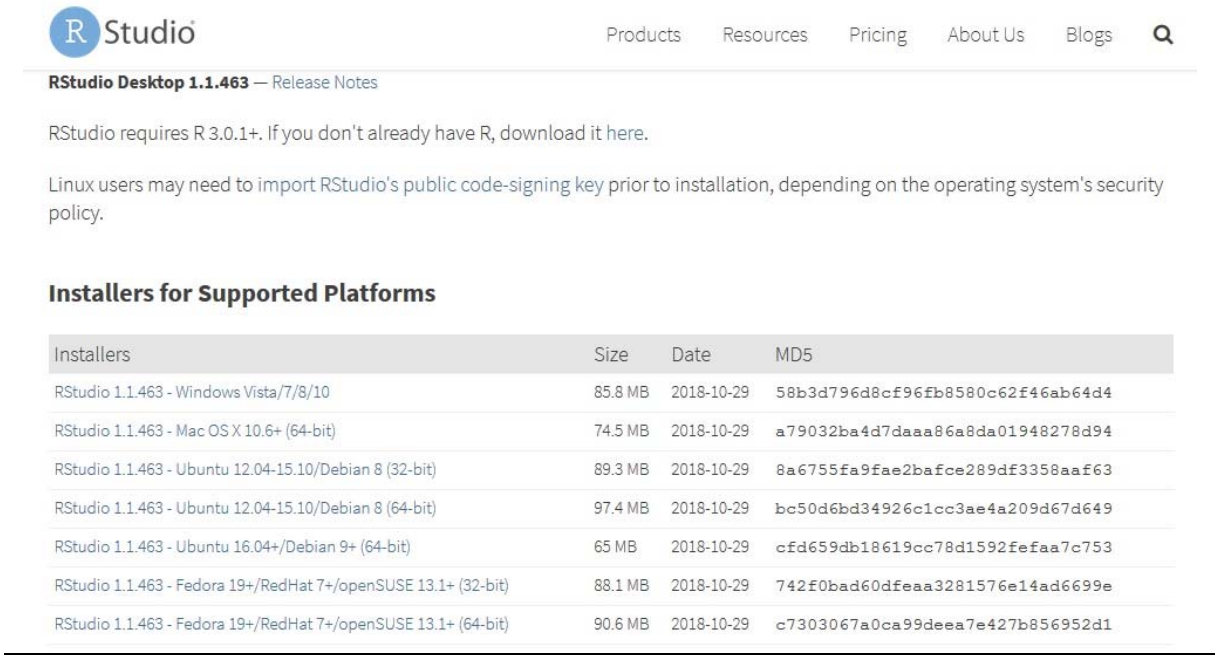
打开 *R-3.5.1.pkg* 并按照说明安装 R。

## 2.2. 安装 RStudio

R 软件安装成功后，我们通过以下链接下载 RStudio（参见图 6）。

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

图 6：下载 RStudio



RStudio Products Resources Pricing About Us Blogs

**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 |

下载可执行文件时，我们需要根据操作系统选择与其对应的选项：

- RStudio 1.1.463 - Windows Vista/7/8/10

- RStudio 1.1.463 - Mac OS X 10.6+ (64-bit)

先保存可执行文件，然后按照说明安装 RStudio。

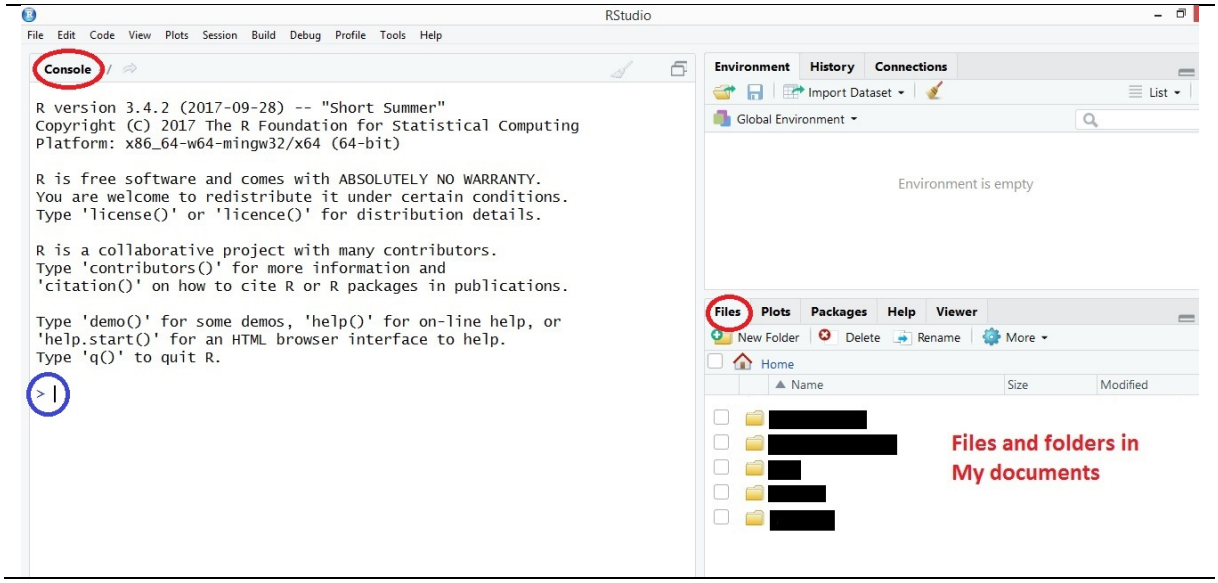
# 3. RStudio 入门

鉴于 RStudio 有着非常良好的交互界面，一般情况下，我们使用 RStudio 而不是 R。

点击 RStudio 图标  ，启动软件。

打开 RStudio 后，我们可以看到如图 7 所展示的内容：

图 7：RStudio 操作界面




默认设置下，*Console*（控制台）位于面板左侧。在简短的信息文本之后，出现系统提示符（“>”）。你能看到闪烁的光标吗？在这里我们可以编写由 R 运行的命令和指令。要在 *Console* 中执行指令并获得结果，按 *Enter* 键。

## 3.1. 如何创建脚本

在控制台中工作非常受限，因为在这里指令通常都是被逐个编写和执行的。因此，我们通常使用脚本(**scripts**)或指令文件。这些文件的扩展名为“*.R*”。

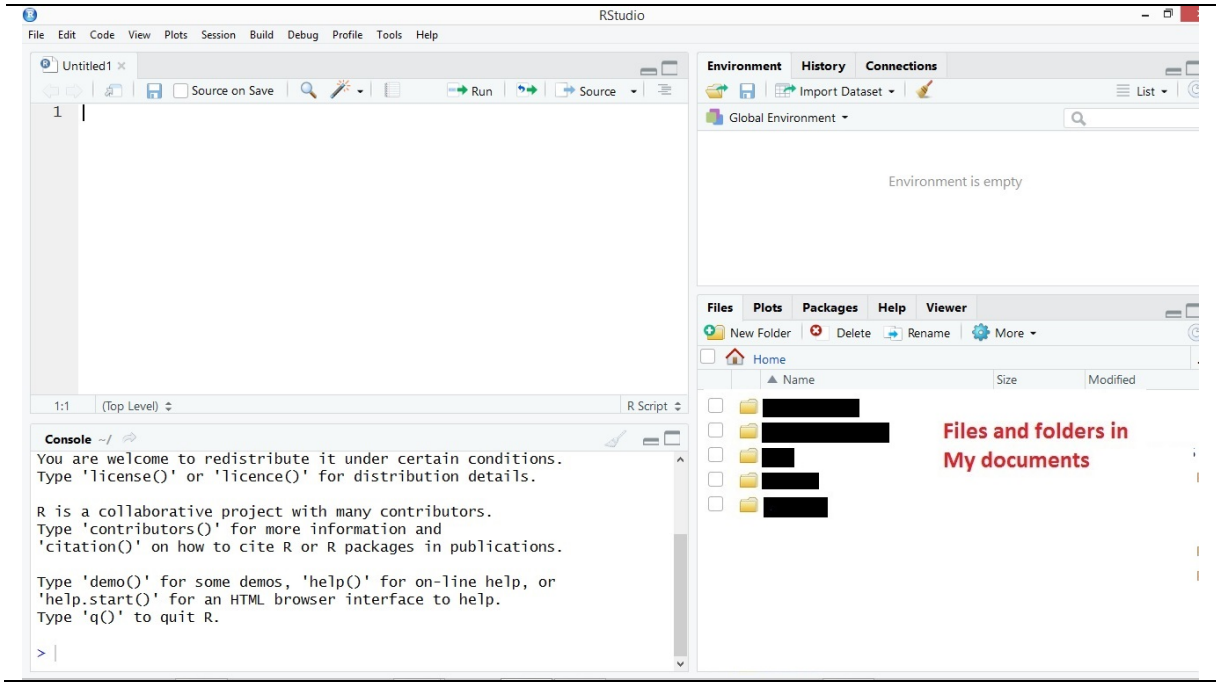
要创建脚本，我们选择 *File > New File > R Script*

现在，左上方是脚本面板，左下方是 *Console*（控制）面板。默认设置下，该脚本文件被命名为“*Untitled1*”（参见图 8）。

在脚本中，我们可以逐行编写指令。这些指令可以被逐一运行，我们也可以通过选择全部（或部分）指令来进行有针对性的操作。运行指令请点击 

。

图 8：脚本




# 4. 如何在 RStudio 中创建和使用项目（project）

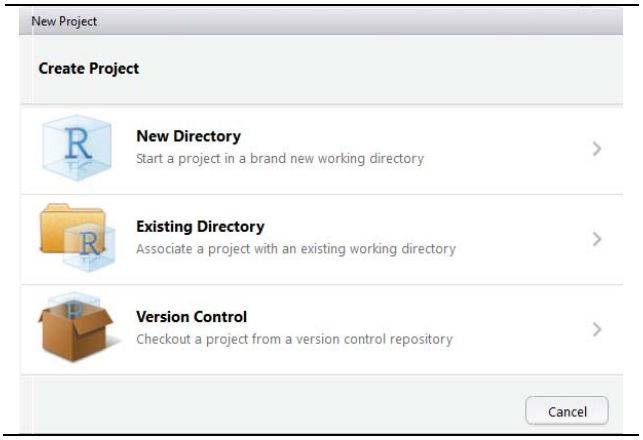
一旦启动 RStudio，我们就应该设置工作目录(working directory)，用于向 R 和 RStudio 指明数据、脚本等的所在路径。但是，我们认为，最好的选择是在 RStudio 中创建一个项目(project)，并使所有文件（数据、脚本等）以及目录都能直接链接到该项目。这样，项目中所有的工作都将包含在目录中。通过这种方式，我们可以轻松地与他人共享项目，或将项目复制粘贴到另一台计算机等。

> **重要提示：**
>
> 应当为每一个活动/工作创建一个项目（例如，为每篇文章/论文）。
>
> 为了恰当地管理项目，我们建议在项目中为原始数据、结果、文本文件等分别创建特定的目录。

## 4.1. 创建项目

要在 RStudio 中创建项目，我们选择 *File > New project...* 然后，可以看到如下图所展示的内容：

图 9：创建（一个）项目




要在新目录中创建项目，单击 *New Directory* 按钮，然后选择项目类型。在我们的示例中，选择 *New Project*。接下来，对所创建的目录（或文件夹）进行命名，而它也将是项目的名称。最后，点击 *Create Project*。以上过程完成后，我们将在电脑文档中找到新创建的文件：“*folder_name.Rproj*”。

要在现有目录中创建项目，则单击 *Existing Directory* 按钮，然后点击 *Browse…* 选择目录或文件夹，最后点击 *Create Project* 完成创建。

## 4.2. 打开项目

要打开项目，需要双击带有“*.Rproj*”扩展名的文件。我们也可以通过选择 RStudio 菜单上的 *File > Open Project…* 来执行此项操作。

项目的优势：我们在项目中工作时保存的任何文件都将储存在项目的目录中。

## 4.3. 补充信息

以下是两份介绍 RStudio 项目的可读资料，第二份还包括了如何创建项目的有关内容。

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

## 4.4. 如何创建项目

步骤 1：启动 RStudio

步骤 2：选择 *File > New Project*

步骤 3：选择 *New Directory*

步骤 4：*Project Type*（项目类型）选择 *New Project*

步骤 5：*Directory name*（目录名称）：Paper_1

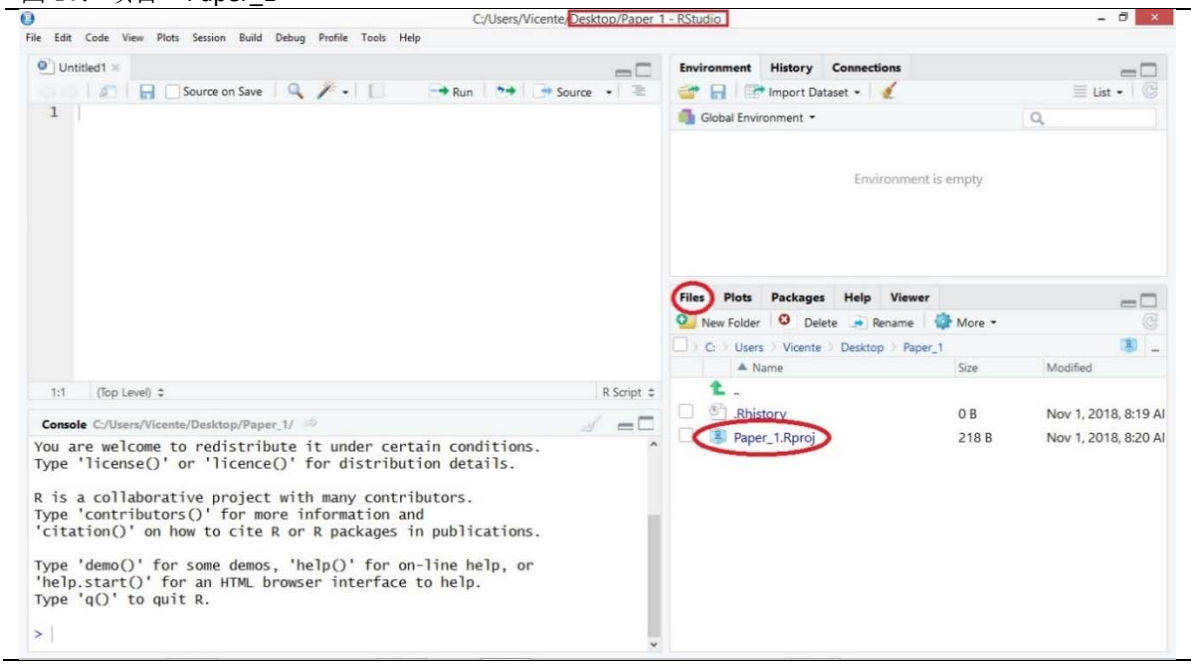
步骤 6：*Create project as subdirectory of*（创建项目为子目录）：点击 Browse 选择新建项目文件夹的路径。例如，选择“桌面”文件夹。

步骤 7：点击 *Create Project*

**结果：**文件夹“*Paper_1*”将被创建在桌面文件夹中。该文件夹包含以下两个文件（参见图 10）：

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

图 10：项目“Paper_1”




**重要提示：**

特定项目的工作会话框将处于始终被打开的状态，你可以：

双击带有扩展名为“*.Rproj*”的文件

或者

通过菜单 *File > Open Project* 选择项目

不建议同时运行多个项目。

# 5. 安装和加载 deaR

打开 RStudio 或者项目“*Paper_1*”

选择 *File > New File > R script*

## 5.1. 安装 deaR

要安装 **deaR**，我们在脚本中输入以下指令：

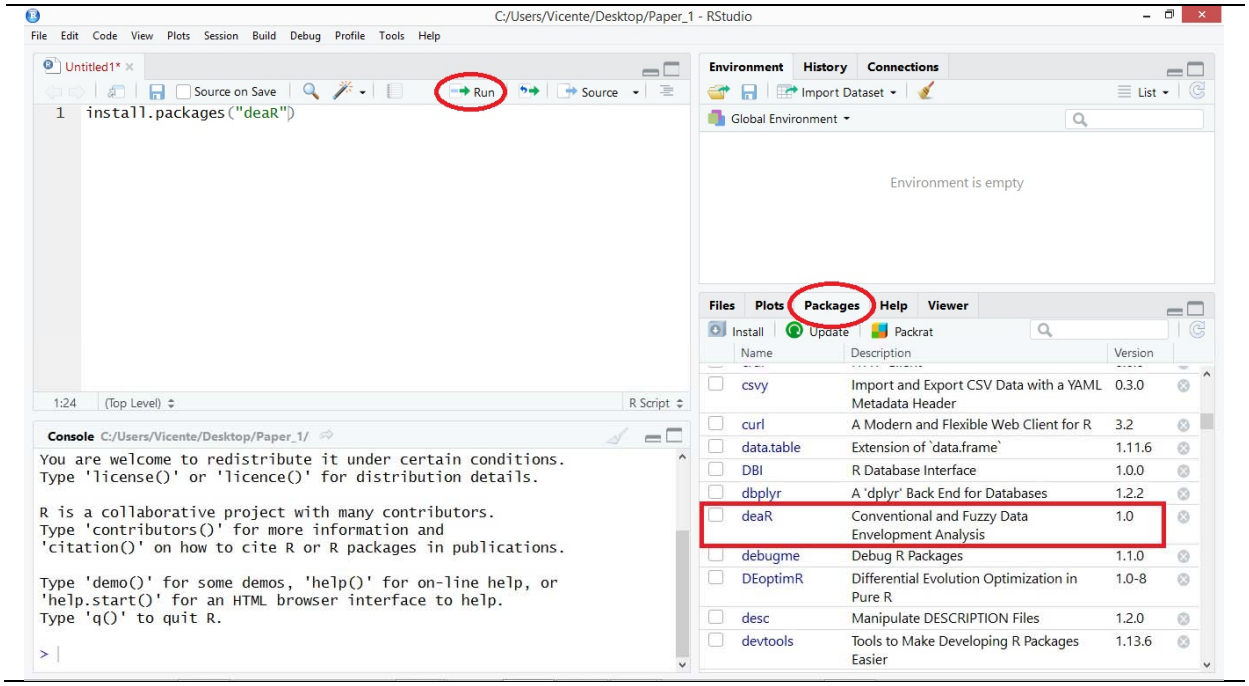
**install.packages(“deaR”)**

然后，点击按钮 Run 执行该指令（参见图 11）。

通过单击右下窗口中的 *Packages* 菜单，页面将会显示所有已安装软件包的列表。现在，**deaR** 软件包应该位列其中。

> **重要提示:**
>
> 我们只需安装 **deaR** 一次即可。此后可以通过更新软件包获得 **deaR** 的最新版本。

图 11：安装 **deaR**




## 5.2. 加载 deaR

安装了 **deaR** 或其他任何 R 软件包后，首先需要对其进行加载才能正常使用（参见图 12）。为此，我在脚本写入以下指令：

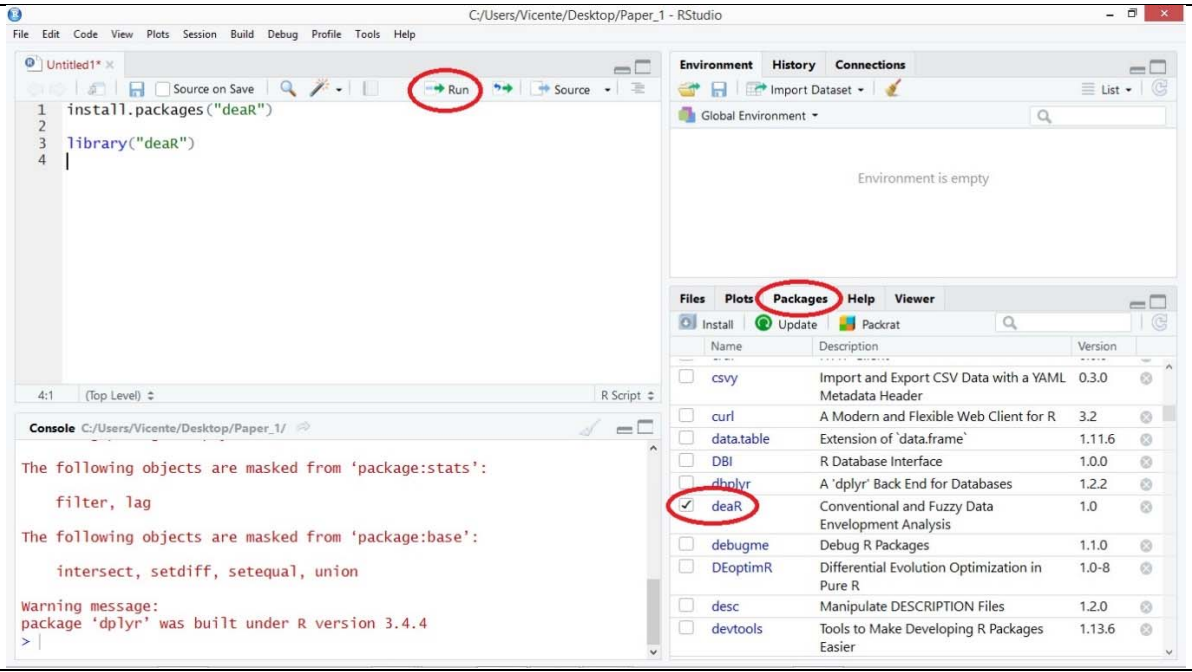
**library(“deaR”)**

然后点击按钮 Run 执行该指令。

**重要提示：**

每个工作会话都必须加载 **deaR** 软件包。

图 12：加载 **deaR**




## 6. 保存脚本并关闭工作会话

要保存脚本，选择 *File > Save as...*

现在，我们命名文件（如：session_1）后单击 *Save* 保存文件。默认设置下，脚本将被保存在名为“*Paper_1*”的项目文件夹中，这也正是在工作中使用项目的一个优势。

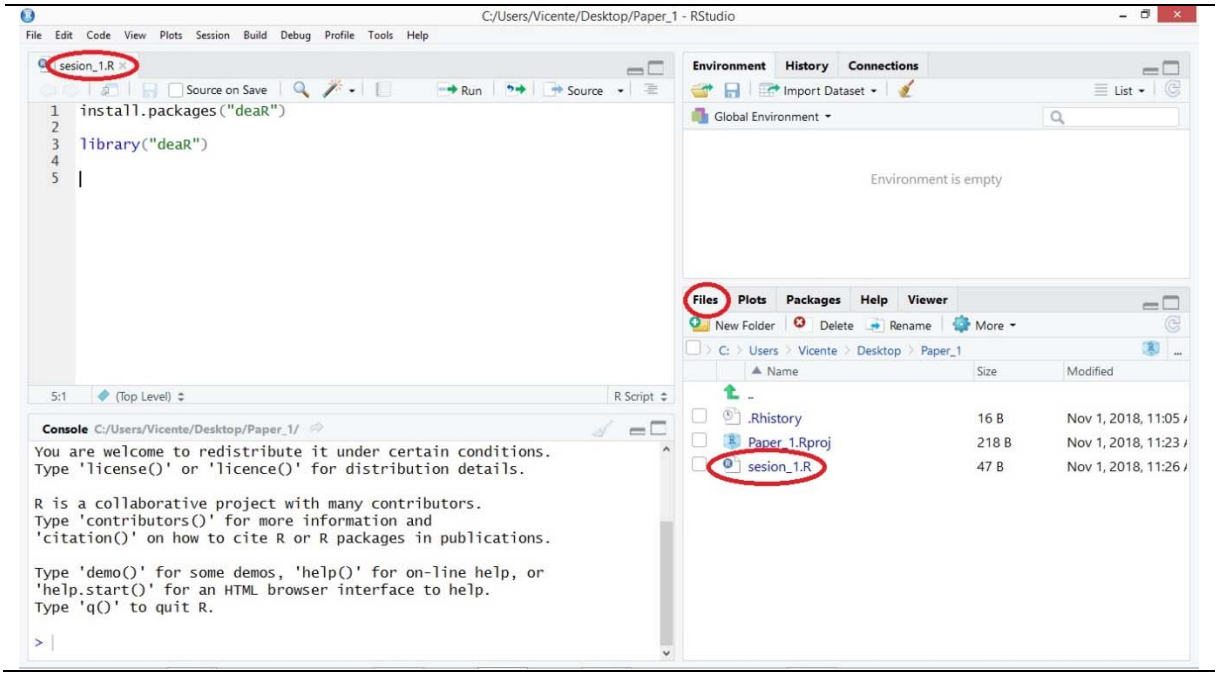
请注意，此时 R 文件（即脚本）的名称是“*session_1.R*”而不是“*Untitled1*”（参见图 13）。该文件会显示在属于 Paper_1 项目的文件列表中。

提示：第一个脚本的内容仅作为示例举出。重要的是要记住我们可以在同一个脚本文件中保存一组指令。

**重要提示：**

脚本文件的名称要能反映他们的内容。

图 13：保存脚本




如果想要关闭当前项目保留 Rstudio，请选择 *File > Close project*

如果想要同时关闭当前项目和 Rstudio，请选择 *File > Quit Session*

**现在，我们保存*“session_1.R”*，关闭项目*“Paper_1”*并退出 RStudio。**

## 7. 使用 deaR 进行数据包络分析（Data Envelopment Analysis）

首先打开项目*“Paper_1”*。为此，我们双击文件“*Paper_1.Rproj*”，打开 RStudio 和该项目。请注意，你也可以先打开 RStudio，然后通过 *File > Open Project* 打开指定项目。

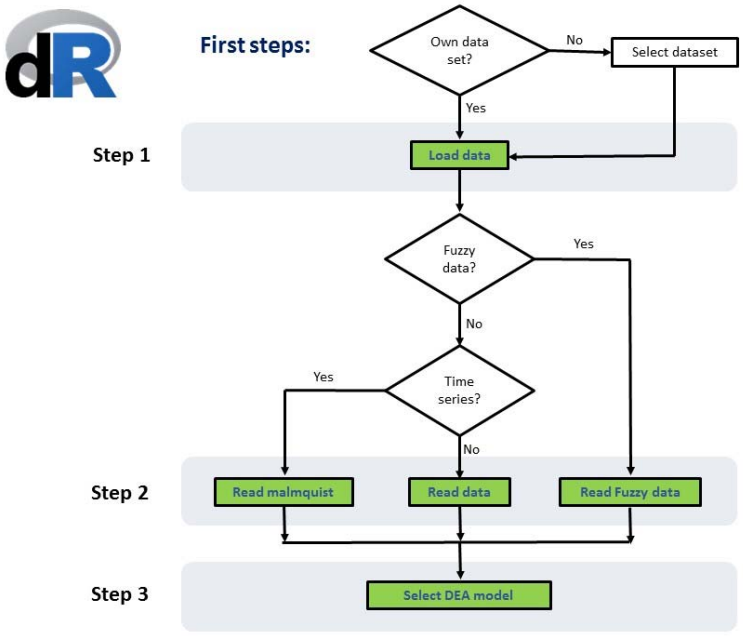
接下来，我们创建一个新的脚本（*File > New File > R Script*）。在这个新的脚本中，我们输入以下指令以加载 **deaR**：

**library(“deaR”)**

（提示：点击 *Run* 按钮 Run 执行指令）

在下面的流程图中（参见图 14），我们可以看到使用 **deaR** 执行任何 DEA（数据包络分析，以下简称 DEA）的步骤。

图 14：**deaR** 的使用步骤




### 7.1. 将数据导入 R

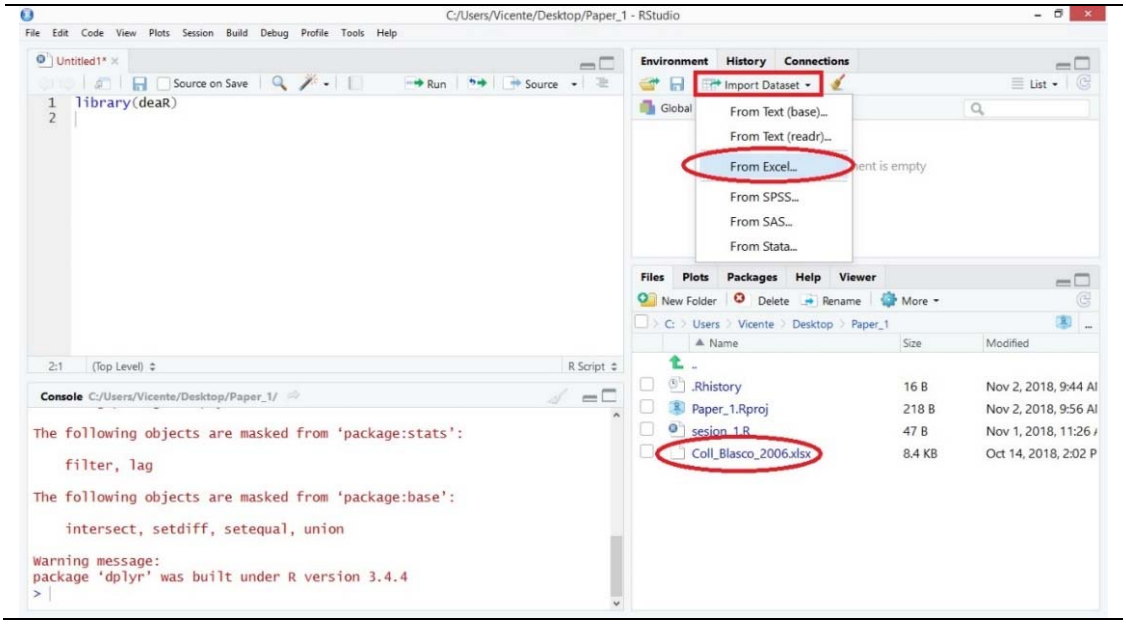
第一步是导入我们将要用于执行 DEA 分析的数据。非 R 用户可以使用 *Import dataset*（导入数据集）选项。示例 1 用于演示如何导入数据。

**示例 1：从 Excel 文件导入**

1. 点击 www.uv.es/vcoll/Coll_Blasco_2006.xlsx 下载示例数据，并将该文件保存至项目“*Paper_1*”的文件夹中。

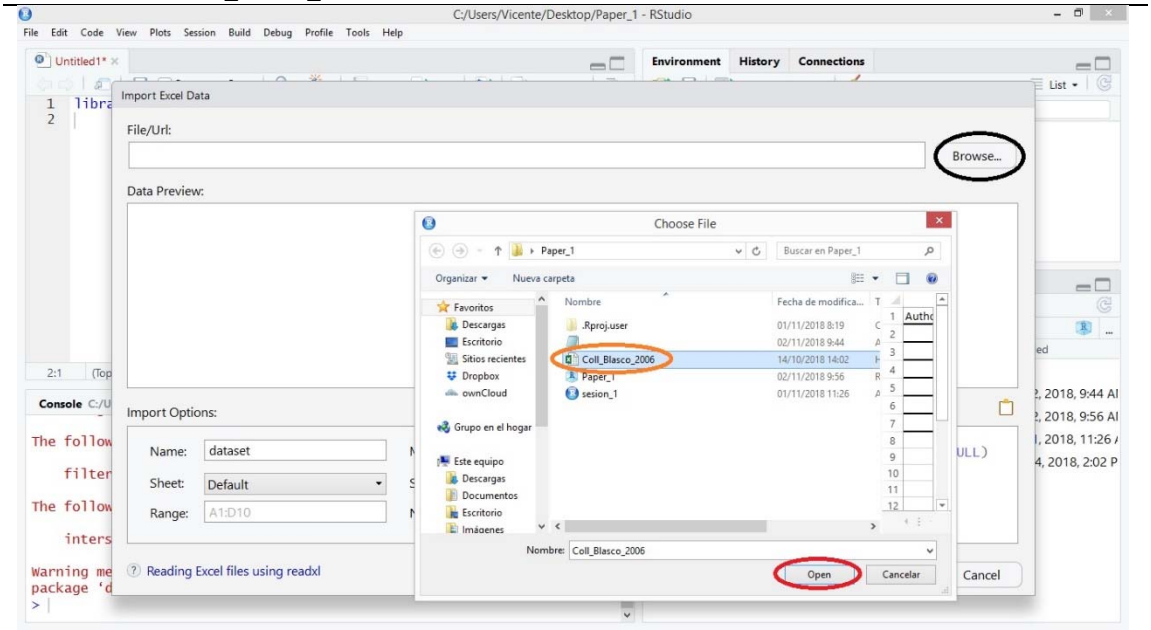
2. 在 RStudio 面板左上窗口的 *Environment* 菜单中，我们选择 *Import Dataset* < *From Excel*（参见图 15）。

图 15：从 Excel 导入数据




3. 待 *Import Excel Data* 窗口打开后，点击 *Browse* 选中 Excel 文件“*Coll_Blasco_2006.xlsx*”[2]，然后点击 *Open* 按钮（参见图 16）。

图 16：导入“Coll_Blasco_2006.xlsx”

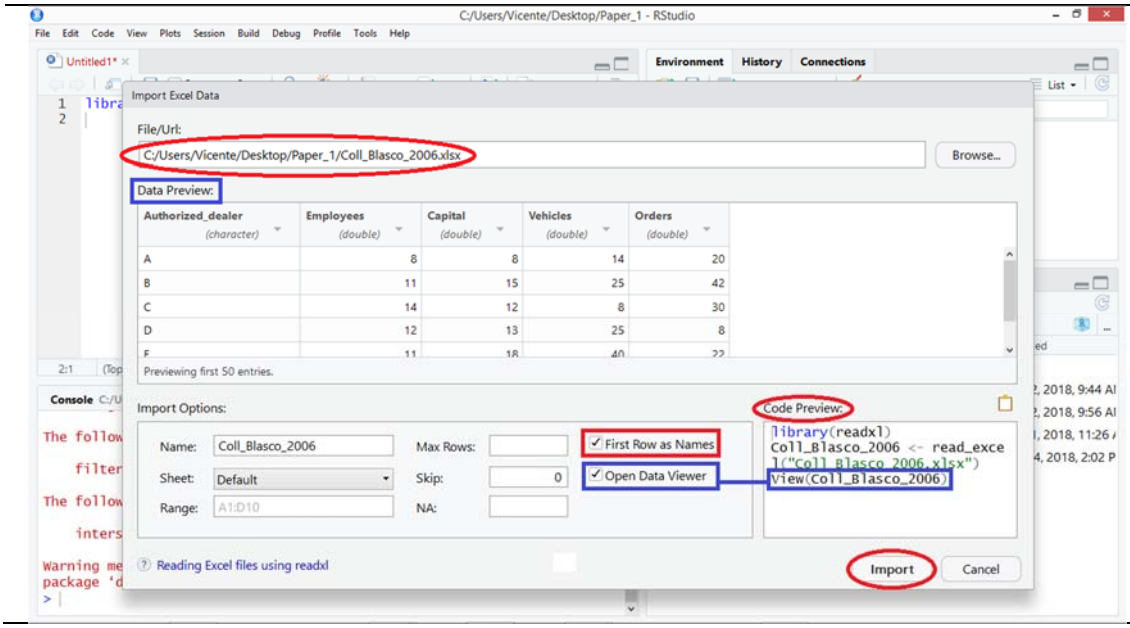



4. 现在，我们可以通过导入数据选项可视化“*Coll_Blasco_2006.xlsx*”。如下图所示，用于导入数据的 R 代码显示在窗口的右下角（参见图 17）。鉴于 *Open Data Viewer* 选项已经被选中[3]，数据集将在导入程序结束时被打开。然后，我们点击 *Import* 按钮。

[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/

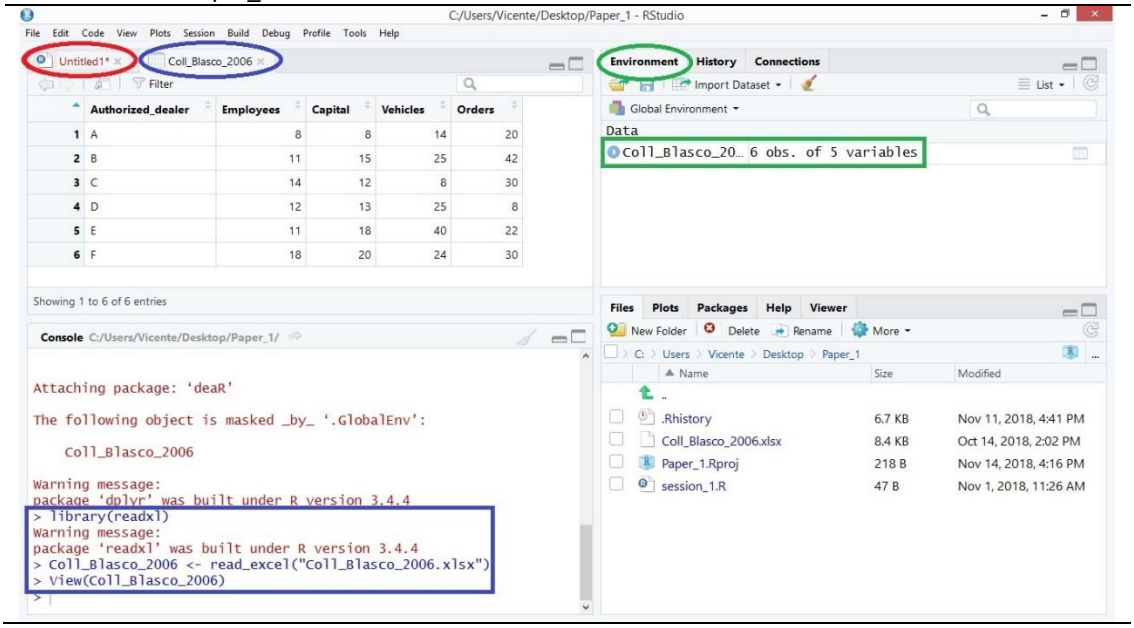
[3] 它对应于 R 代码: **View(Coll_Blasco_2006)**

图 17：数据预览




5. 单击 *Import* 按钮后，我们就会返回到项目“*Paper_1*”的主窗口，刚才被导入的数据则显示在新的工作表中（它与数据集的名称相同）（参见图 18）。同时，在 *Environment* 菜单（右上界面）下则会显示我们在 R 中创建的所有对象（object）[4]的列表。现在，我们只有一个对象，即“*Coll_Blasco_2006*”。实际上，这个对象是一个包含了 6 个观测值（observation）和 5 个变量（variable）的数据框（dataframe）。

图 18：项目“*Paper_1*”




在 *Console* 面板中，我们可以看到用于导入数据的 R 代码（参见图 18）：

[4] R 中的一切事项都被视为对象（object）。一个对象可以是一个数据集，一个函数，一条指令，一个矩阵模型，等等。

- **library(readxl)** → 加载“*readxl*”包
- **Coll_Blasco_2006 <- read_excel(“Coll_Blasco_2006.xlsx”)** → 函数“*read_excel*”（来自于 *readlx* 包）读取数据文件，并将该数据分配（“<-“）给对象“*Coll_Blasco_2006*”。

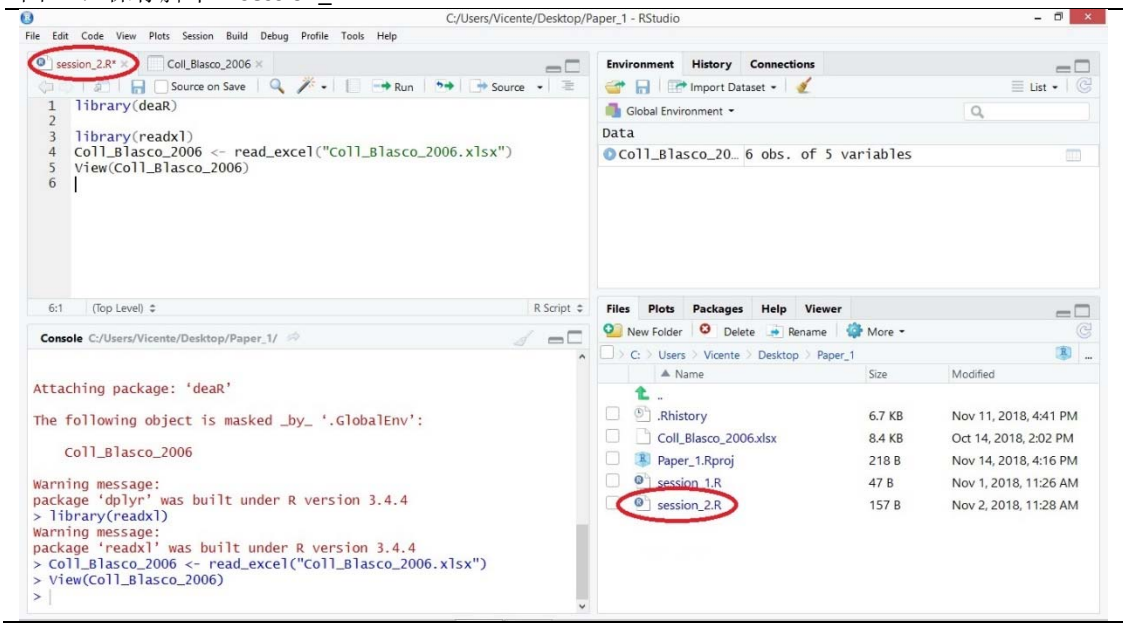
> **重要提示：**
>
> R 术语：**A <- B** 意指 B（无论 B 是一个数据集、一个函数、一个矩阵、或数学运算的结果等）被分配（<-）到对象 A。

- **View(Coll_Blasco_2006)** → 可视化对象“*Coll_Blasco_2006*”

6. 我们把以上代码复制到脚本“*Untitled1*”，将文件命名为“*session_2*”并保存文件。

这时，操作界面将会显示如下内容（参见图 19）。

图 19：保存脚本“session_2”




不需要在后续的所有工作中重复执行以上 1-5 步，因为我们已经把完整的代码储存在了文件“*session_2.R*”中。如果要再次从“*Coll_Blasco_2006.xlsx*”导入数据，我们只需直接执行“*session_2.R*”中的代码指令即可。

---

**deaR** 还提供庞大的数据集（dataset）。这些数据集来自于各类已经公开发表的文献，并用于重复这些文献的研究结果。我们认为，这是 **deaR** 为研究人员和从业者提供的一

个巨大的附加价值，因为它可以帮组和支持你教授和研习 DEA 的方法论。我们可以通过获取 **deaR** 的帮助信息[5]来了解它的数据结构和来源。

接下来，我们用示例 2 演示如何加载数据集（dataset）。

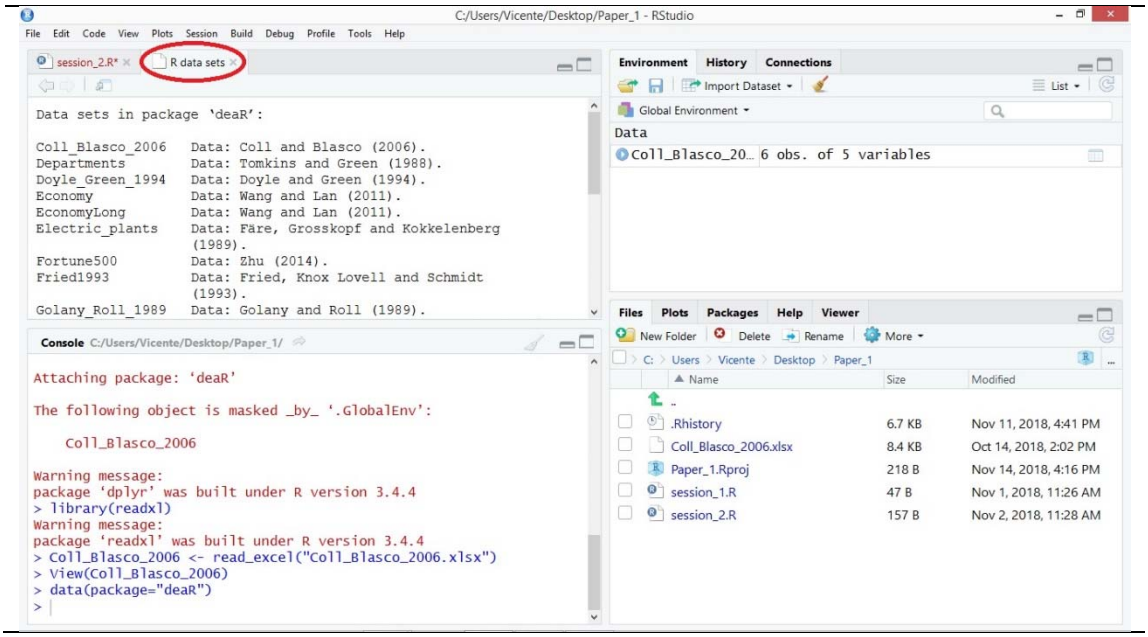
---

**示例 2：在 deaR 中加载可用的数据集**

要查看 **deaR** 中的可用数据集，我们写入并执行以下指令：

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

图 20：**deaR** 提供的数据集




加载数据集，我们需要使用 **data()**函数。例如，我们要从 Tomkins and Green (1988)[6]中加载数据，该数据集的名称是“*Departments*”。为此，我们在脚本“*session2.R*”中写入以下指令（参见图 21）：

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

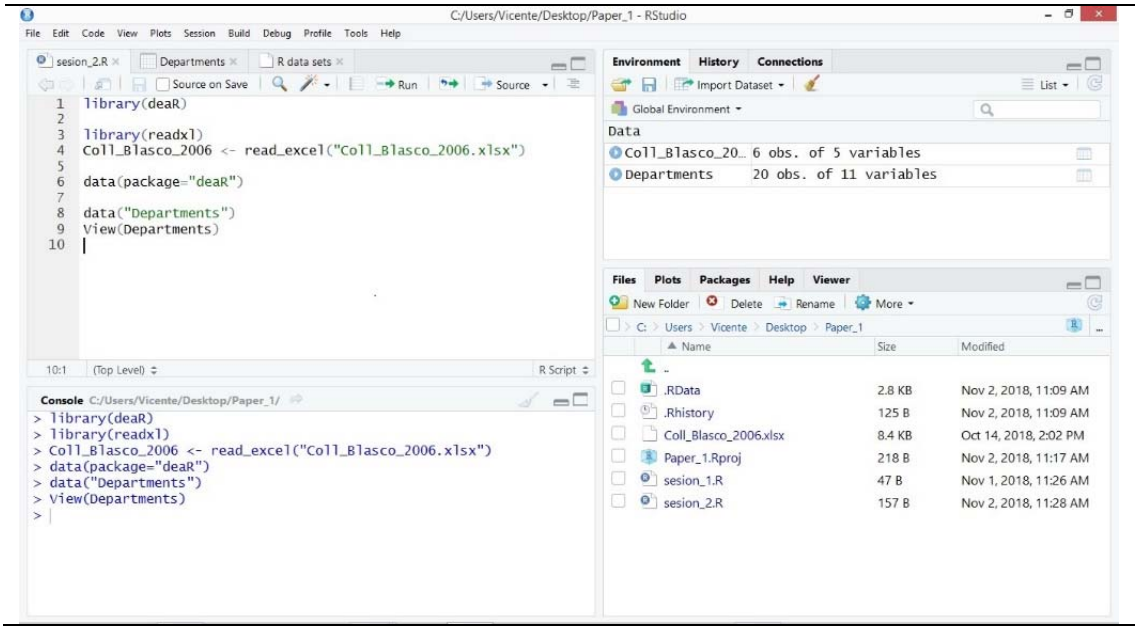
想要可视化数据集，指令如下（参见图 21）：

```
View(Departments)
```

[5] 获取 R 的帮助信息，请点击链接： http://www.r-project.org/help.html

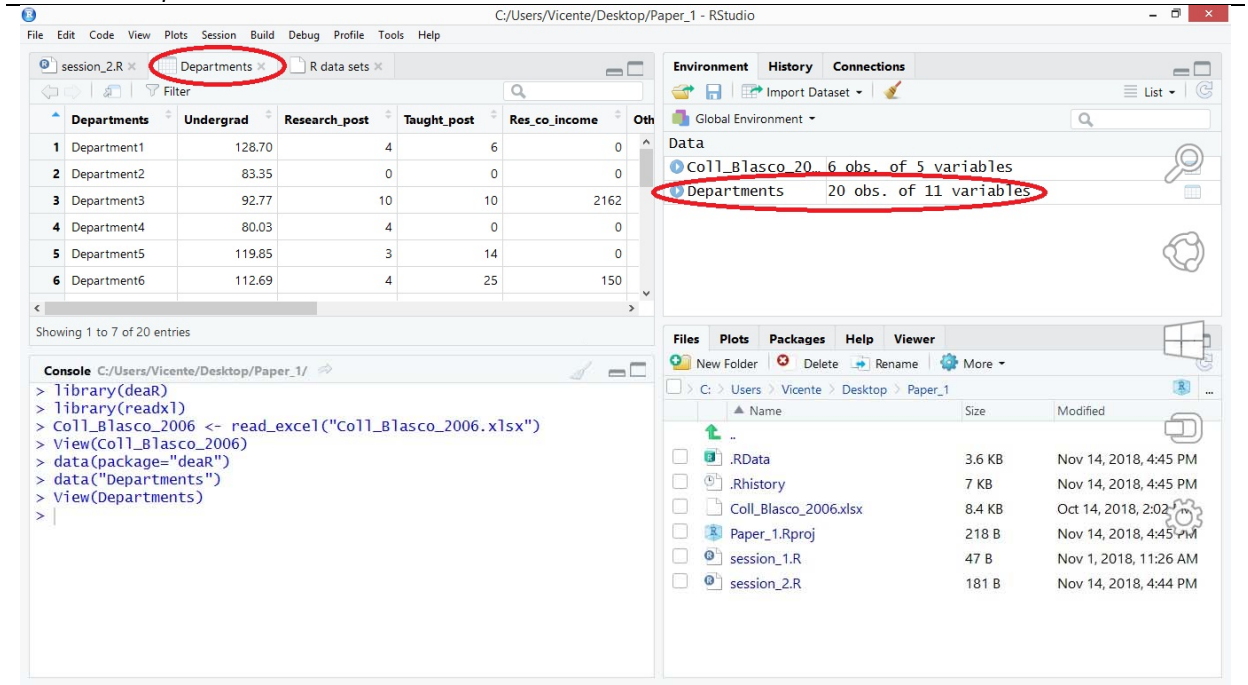
[6] Tomkins, C.; Green, R. (1988). "An Experiment in the Use of Data Envelopment Analysis for Evaluating the Efficiency of UK University Departments of Accounting", Financial Accountability and Management, 4(2), 147-164. https://doi.org/10.1111/j.1468-0408.1988.tb00296.x

图 21：加载和查看数据




点击 Run 按钮运行指令（参见图 22）。

图 22：“*Departments*”




提示：我们现在有两个对象：“*Coll_Blasco_2006*”和“*Departments*”，其中“*Departments*”是一个包含了 20 个观测值（observation，即 DMU：决策单位）和 11 个变量（variable）的数据框。

现在，我们保存“***session_2.R***”，关闭项目“***Paper_1***”并退出 **RStudio**。

## 7.2. 根据 deaR 调整数据

加载数据后，需要对其格式进行调整，以便于 **deaR** 能够顺利读取它们。**deaR** 有三种不同的数据读取函数，每一个函数都与一个特定的 DEA model 相对应。这些数据读取函数是：

**Read_data()**: 用于运行 conventional DEA model（传统 DEA 模型）。

**Read_malmquist()**: 用于执行 Malmquist productivity index（Malmquist 生产力指数）运算。

**Read_data_fuzzy()**: 用于运行含有不确定数据（uncertain data）的 DEA model（fuzzy DEA）

### 7.2.1. 从 deaR 获取帮助[7]

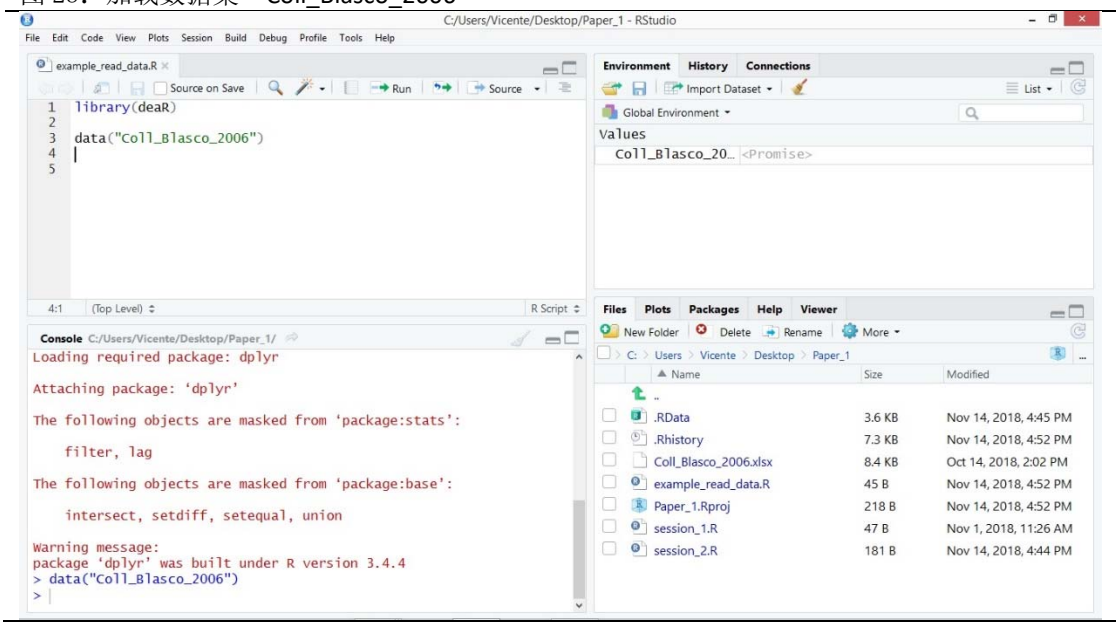
我们可以通过 **deaR** 的 help（帮助）功能学习如何使用特定的函数，并阅读关于每个函数的不同参数和使用示例。我们用示例 3 演示如何使用 deaR 的 help 函数。

---

**示例 3：获取 deaR 帮助**

1. 打开项目“*Paper_1*”，创建一个新的脚本，并将其命名为“*example_read_data*”。
2. 加载 **deaR**
3. 加载由 **deaR** 提供的数据集（dataset）“*Coll_Blasco_2006*”（参加图 23）。

图 23：加载数据集“Coll_Blasco_2006”



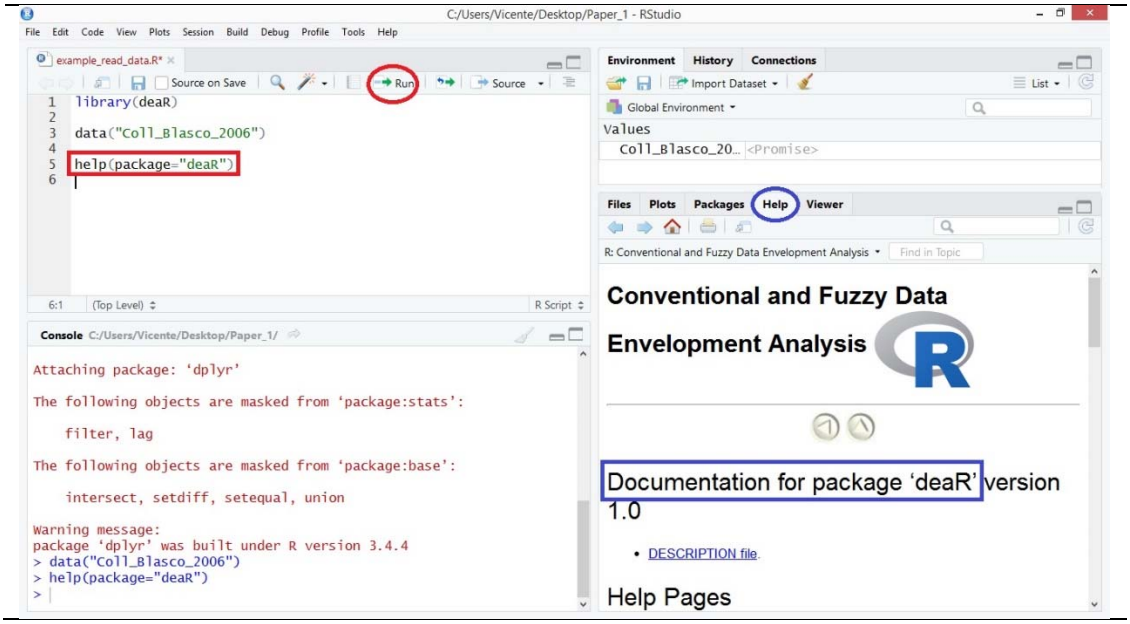

要访问 **deaR** 软件包函数文档，我们使用 **help()**函数。写入：

```
help(package="deaR")
```

[7] 获取关于 R 的帮助，请点击链接： https://www.r-project.org/help.html

在 *Help* 菜单中（右下窗口）将显示 **deaR** 中的所有函数和数据集（dataset）列表。它也是 **deaR** 软件包的文档（参见图 24）。

图 24：从 **deaR** 获取帮助



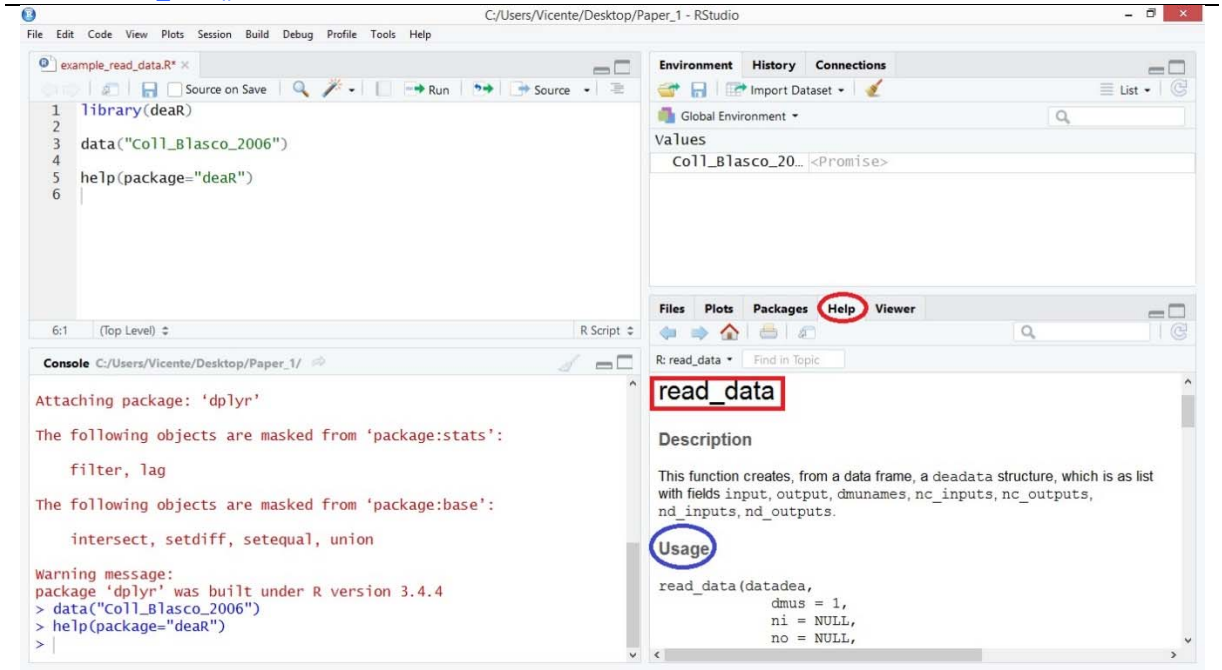

### 7.2.2. read_data()函数

访问了 **deaR** 的（help）文档后，我们可以通过单击“*read_data*”链接获得有关此函数的特定帮助。我们也可以通过在脚本中写入并执行以下指令获得相同的结果：

**help(read_data)**（或者 **?read_data**）

*Usage* 部分列出了 read_data()函数的所有参数，此外，我们还可以在 *Arguments* 部分读到关于这些参数的简要说明（参见图 25）。

图 25：read_data()函数的帮助信息



read_data()函数采用以下参数：

- *datadea*: 它是一个数据集（且必须是一个数据框）.
- *dmus*: DMU 所在列的编号。默认设置下，**deaR** 把 DMU 置于数据集的第一列。
- *ni*: inputs （输入）的数量
- *no*: outputs （输出）的数量
- *inputs*: 用于指明 input 所在的列的编号，而不是 input 的数量
- *Outputs*: 用于指明 output 所在的列的编号，而不是 output 的数量
- *nc_inputs*: 如果有 non-controllable inputs（不可控输入），可用于指明它们是哪些
- *nc_outputs*: 如果有 non-controllable outputs（不可控输出），可用于指明它们是哪些
- *nd_inputs*: 如果有 non-discretionary inputs（非目录化输入），可用于指明它们是哪些
- *nd_outputs*: 如果有 non-discretionary outputs（非目录化输出），可用于指明它们是哪些
- *ud_inputs*: 如果有 undesirable (bad) inputs（不良/坏输入），可用于指明它们是哪些
- *ud_outputs*: 如果有 undesirable (bad) outputs（不良/坏输出），可用于指明它们是哪些

示例 4 演示了如何使用 read_data()函数。**deaR** 文档则提供了软件包里所有函数的使用示例。

---

**示例 4：使用 read_data()函数**

目前，我们已加载了数据集“*Coll_Blasco_2006*”。该数据集有 6 个 DMU（第 1 列）、2 个 input（第 2 列和第 3 列）及 2 个 output（第 4 列和第 5 列）。

假设我们想要通过运用 BCC DEA model 来测量这些 DMU 的效率。

由于 BCC BCC model 是一个 conventional DEA model（它不是 fuzzy DEA model），我们必须使用 read_data()函数使原始数据集符合 **deaR** 的读取格式。

我们在脚本“*example_read_data*”中写入以下指令（参见图 26）：

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

**非常重要：阅读并理解指令**

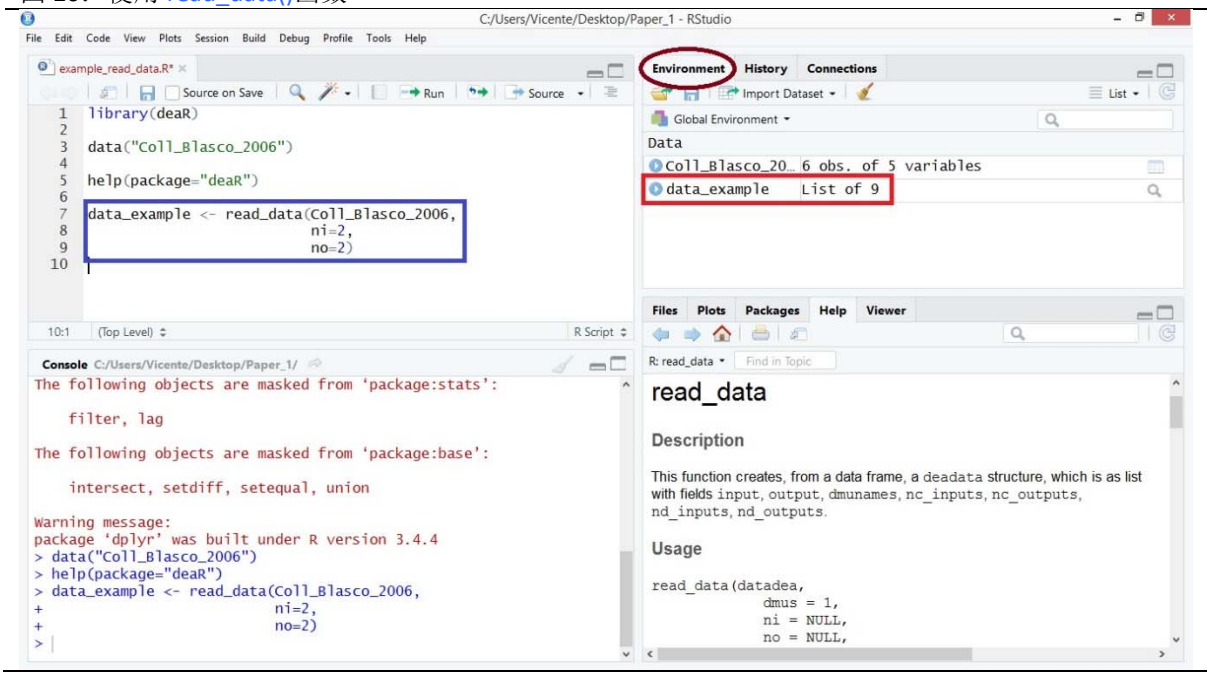
赋值符号（<-）右侧的表达式表示：读取 Coll_Blasco_2006 中的数据（read_data）；它有 2 个 input（ni=2）和 2 个 output（no=2）。DMU 在第 1 列（dmus=1），但在这里此参数不会出现，因为它是默认值。

然后，将此函数的结果被分配（<-）到对象 *data_example*。

点击按钮执行指令（ Run ）

请注意，此时 *Environment* 菜单列出了“*Coll_Blasco_2006*”和“*data_example*”两个对象，并且“*data_example*”是一个包含了 9 个组件的列表（参见图 26）。

图 26：使用 read_data()函数




现在，数据已经做好准备与 conventional DEA model 一起使用了（参见第 7.3 节）。为此，我们应该使用我们之前创建的对象：“*data_example*”。

建议：请用文档中的示例来练习使用 read_data()函数

---

**Save “*example_read_data.R*”.**

### 7.2.3. read_malmquist()函数

如果我们有时间序列数据（time series data）并且想要测量 Malmquist productivity index（Malmquist 生产力指数），就必须使用 read_malmquist()函数以使数据符合 **deaR** 的读取格式。

**deaR** 能够读取两种不同格式的时间序列数据（time series data）：

- **Wide format**: DMU 按列列出，input 和 output 分别根据不同的时间段按列列出。示例：请参看 **deaR** 中的“*Economy*”数据集[8]（参见图 27）。
- **Long format**: 时间按列列出，DMU、input 和 output 按照时间分组后按列列出。示例：请参看 **deaR** 中的“*EconomyLong*”数据集（参见图 28）。

我们将在接下来的示例中演示以上说明。

[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

**示例 5：Wide and Long data formats.**

现在，我们逐一展示这两种不同的数据格式。因此：

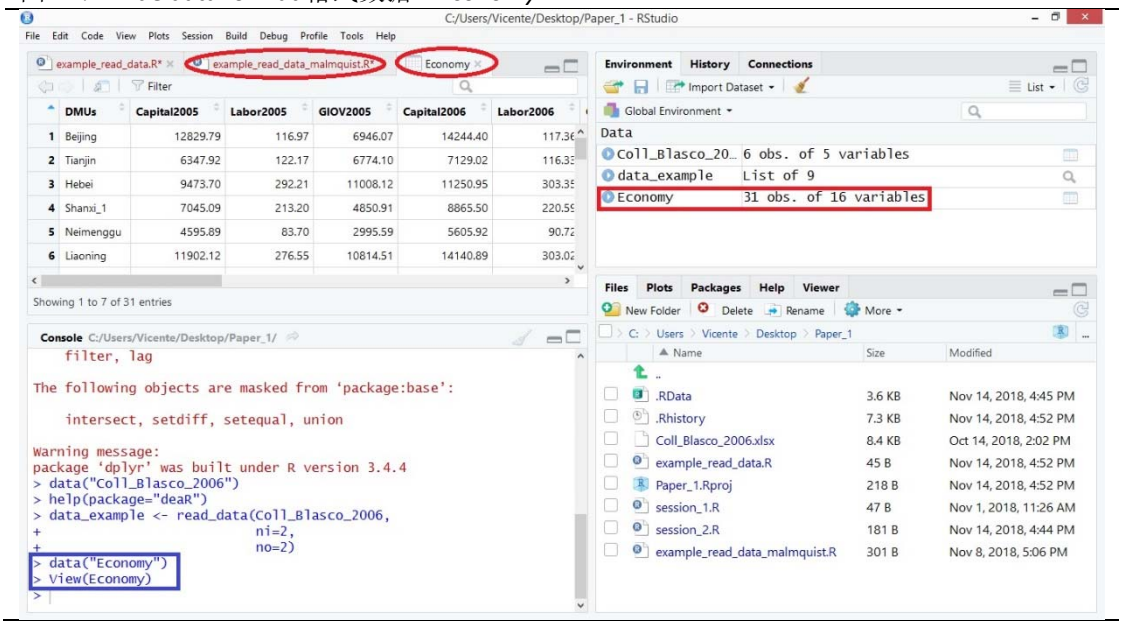
1. 创建一个新的脚本，将其命名为“*example_read_malmquist*”。

   提示：如果已经关闭了工作会话窗口，请打开项目“*Paper_1*”，然后创建新的脚本，同时也需要加载 **deaR**。

2. 在 **deaR** 中加载数据集“*Economy*”（参见图 27）。

3. 可视化“*Economy*”，可以看到这个对象（数据集）有 31 个 DMU、2 个 input（Capital 和 Labor）及 1 个 output（GIOV），时间段为 5 年（2005 至 2009）。所以，“*Economy*”是一个具有 wide forma 格式的数据集。

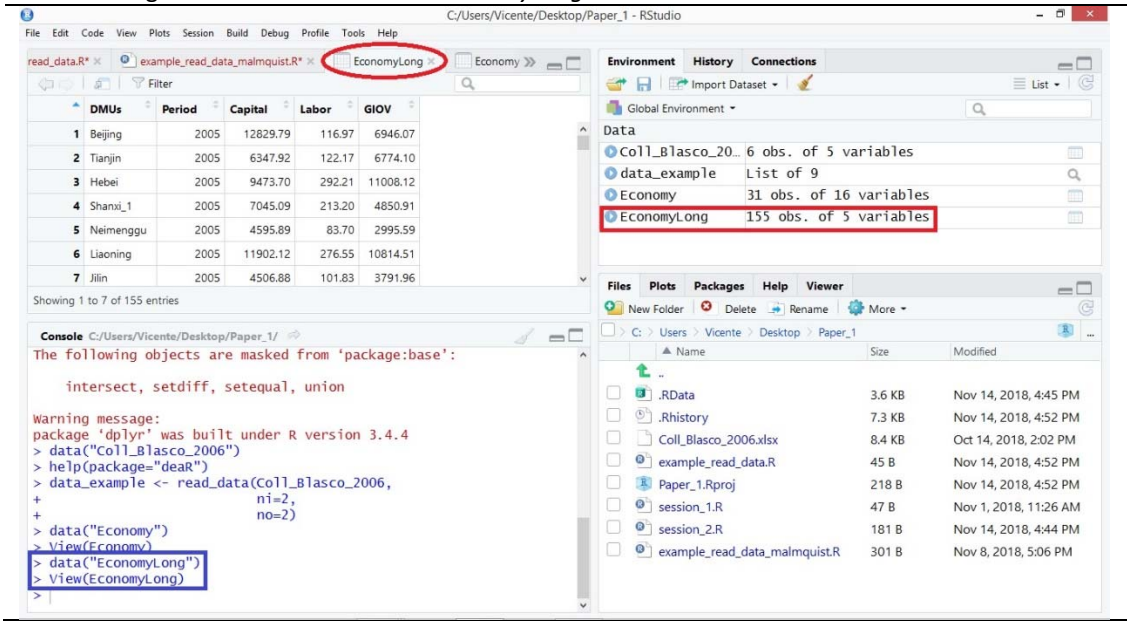
图 27：Wide data format 格式数据“*Economy*”




4. 接下来，加载数据集“*EconomyLong*”

5. 可视化“*EconomyLong*”后可以看到（参见图 28），这个新的 R 对象（实际上它是一个数据集）有 155 个观测值（31DMU x 5 年）、2 个 input（Capital 和 Labor）和 1 个 output（GIOV）；时间栏中显示的时间段是 2005 年至 2009 年。

图 28：Long data format 格式数据“*EconomyLong*”




提示：在工作会话中创建的所有对象都会显示在 *Environment* 菜单中（右上窗口）。

---

read_malmquist()函数采用以下参数：[9]

- *datadea*: 它是一个数据集（且必须是一个数据框）.
- *nper*: 年数（数据格式：wide format）
- *percol*: 变量时间所在的列的编号（数据格式：long format）
- *arrangement*: wide format 格式数据的“水平排列”(horizontal) 或者 long format 格式数据的“垂直排列”(vertical)
- *dmus*: DMU 所在列的编号。默认设置下，**deaR** 把 DMU 置于数据集的第一列。
- *ni*: inputs （输入）的数量
- *no*: outputs （输出）的数量
- *inputs*: 用于指明 input 所在的列的编号，而不是 input 的数量
- *outputs:* 用于指明 output 所在的列的编号，而不是 output 的数量
- *nc_inputs*: 如果有 non-controllable inputs（不可控输入），可用于指明它们是哪些
- *nc_outputs*: 如果有 non-controllable outputs（不可控输出），可用于指明它们是哪些
- *nd_inputs*: 如果有 non-discretionary inputs（非目录化输入），可用于指明它们是哪些
- *nd_outputs*: 如果有 non-discretionary outputs（非目录化输出），可用于指明它们是哪些

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

- *ud_inputs*: 如果有 undesirable (bad) inputs（不良/坏输入），可用于指明它们是哪些
- *ud_outputs*: 如果有 undesirable (bad) outputs（不良/坏输出），可用于指明它们是哪些

当前版本的 **deaR** 无法运用 undesirable inputs/outputs 计算 Malmquist productivity index，但此功能将被写入未来的版本中。

示例 6 和示例 7 分别演示了如何针对 wide format 格式数据和 long format 格式数据运用 read_malmquist()函数

---

## 示例 6：wide format 格式的 read_malmquist()函数

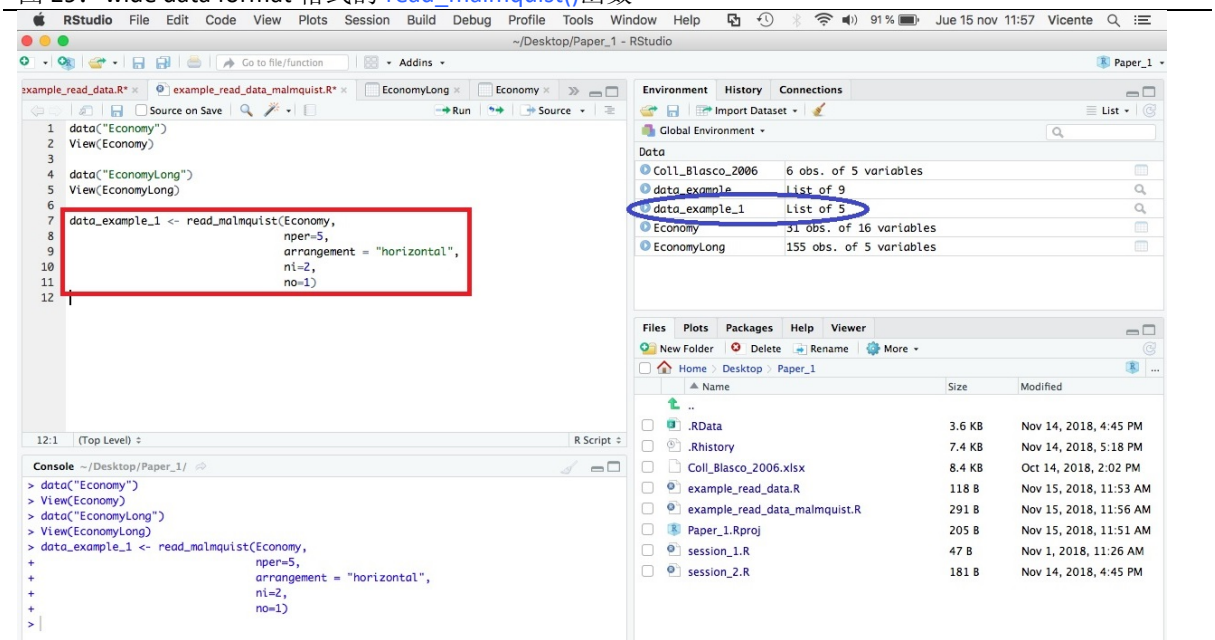
在这个示例中，我们将使用数据集“*Economy*”。该数据集已经被加载到了当前的工作会话中。

为了使“*Economy*”符合 **deaR** 的读取格式，我们在脚本“*example_read_malmquist*”中写入以下文本：

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

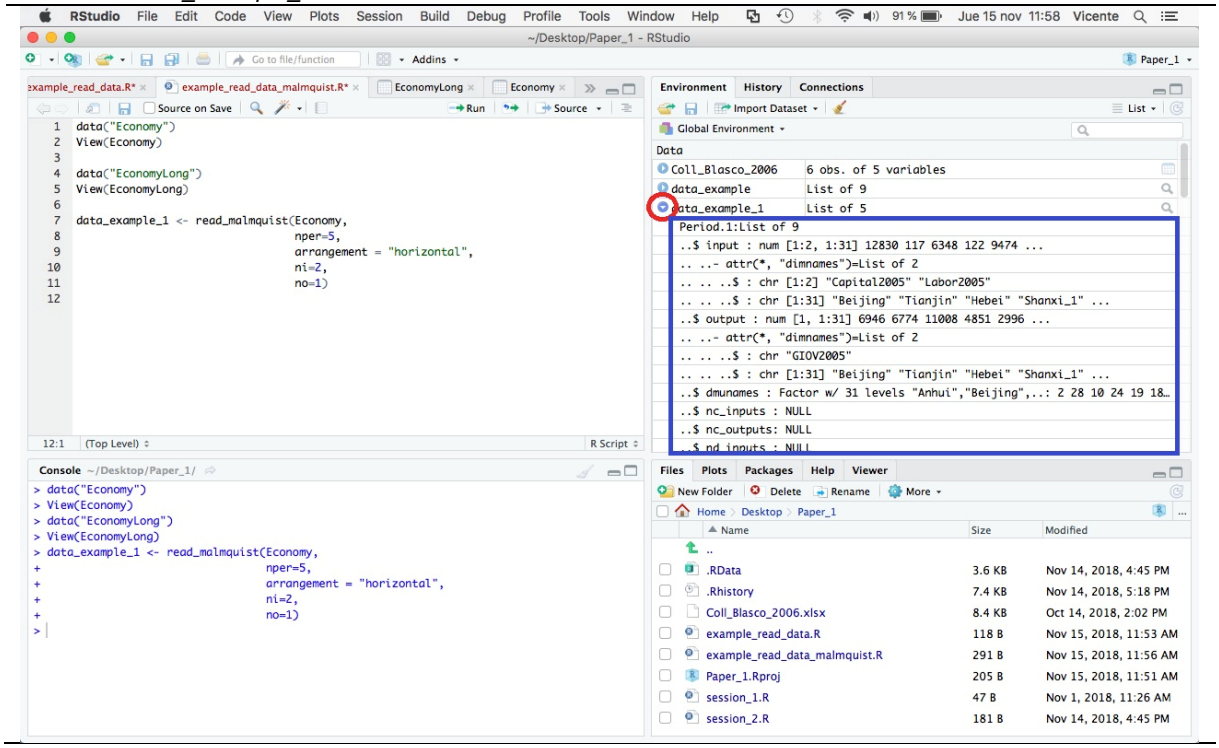
执行指令后，一个新的对象（“*data_example_1*”）被创建出来了，它显示在 *Environment* 菜单中（参见图 29）。

图 29：wide data format 格式的 read_malmquist()函数




我们可以看到，“*data_example_1*”是一个含有 5 个组件的列表。如果我们单击对象名称旁边的图标 ，还能看到该对象的具体结构（参见图 30）。

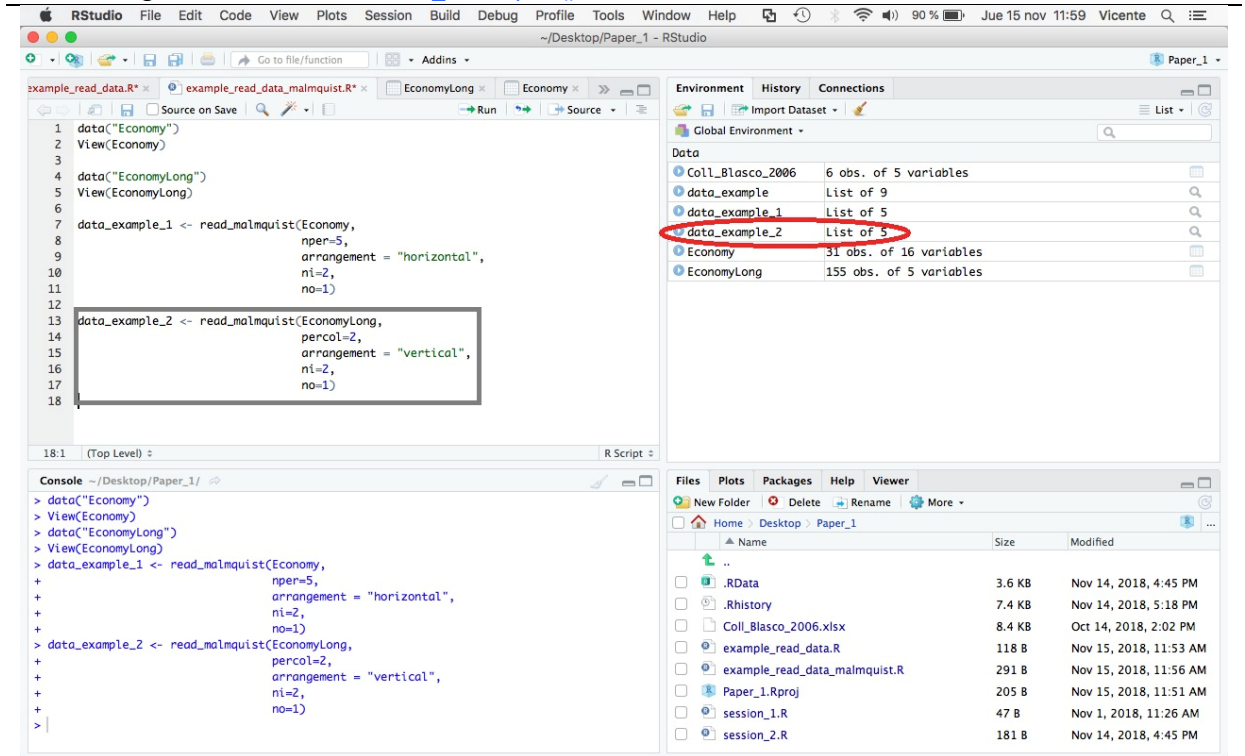
图 30：“*data_example_1*”结构




## 示例 7：long format 格式的 read_malmquist()函数

现在，我们使用数据集“*EconomyLong*”。图 31 显示了调整该数据集使其符合 **deaR** 读取格式的指令。

图 31：long data format 格式的 read_malmquist()函数



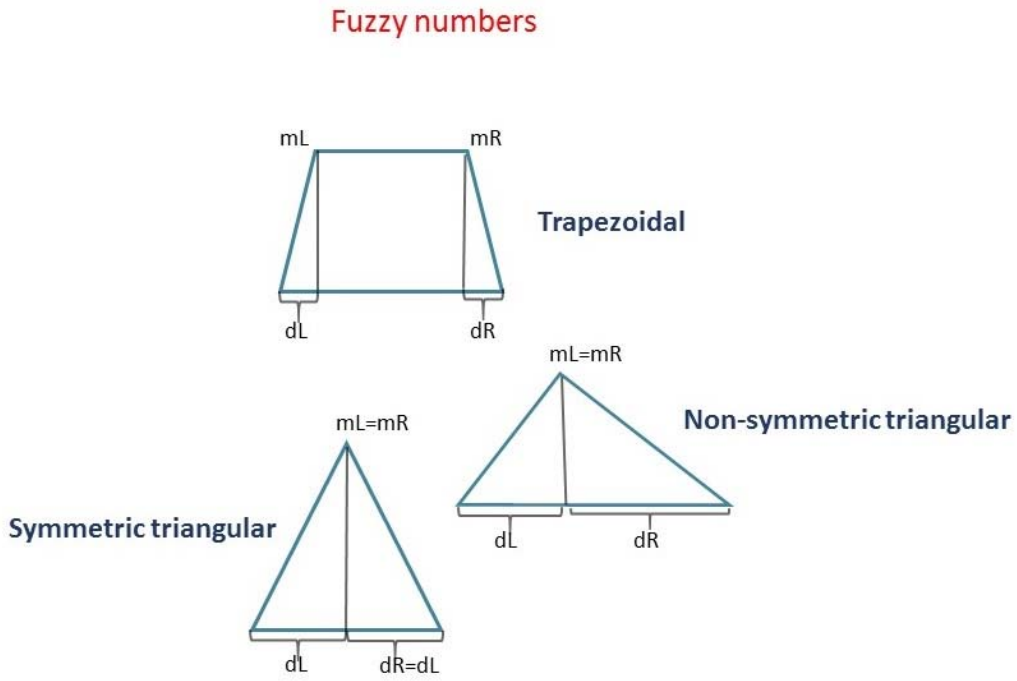
保存“example_read_malmquist.R”。

### 7.2.4. read_data_fuzzy()函数

如果想要运用 fuzzy DEA model 处理不确定数据（uncertain data）以测量效率，那么首先需要使用 read_data_fuzzy() 函数把数据调整为 **deaR** 的可读模式。

**deaR** 可以处理梯形（trapezoidal）、对称三角形（symmetric triangular）和不对称三角形（non-symmetric triangular）的模糊数（fuzzy number）（参见图 32）。

图 32：Fuzzy numbers




read_data_fuzzy()函数采用以下参数：

- *datadea*: 它是一个数据集（且必须是一个数据框）
- *dmus*: DMU 所在列的编号。默认设置下，deaR 把 DMU 置于数据集的第一列。
- *Inputs.mL*: input 的 mL value 数值所在的列的编号
- *Inputs.mR*: input 的 mR value 数值所在列的编号
- *Inputs.dL*: input 的 dL value 数值所在列的编号
- *Inputs.dR*: input 的 dR value 数值所在列的编号
- *Outputs.mL*: output 的 mL value 数值所在的列的编号
- *Outputs.mR*: output 的 mR value 数值所在列的编号
- *Outputs.dL*: output 的 dL value 数值所在列的编号
- *Outputs.dR*: output 的 dR value 数值所在列的编号

- *nc_inputs*: 如果有 non-controllable inputs（不可控输入），可用于指明它们是哪些
- *nc_outputs*: 如果有 non-controllable outputs（不可控输出），可用于指明它们是哪些
- *nd_inputs*: 如果有 non-discretionary inputs（非目录化输入），可用于指明它们是哪些
- *nd_outputs*: 如果有 non-discretionary outputs（非目录化输出），可用于指明它们是哪些
- *ud_inputs*: 如果有 undesirable (bad) inputs（不良/坏输入），可用于指明它们是哪些
- *ud_outputs*: 如果有 undesirable (bad) outputs（不良/坏输出），可用于指明它们是哪些

目前，**deaR** 仅能考量基于 BCC DEA Model 的 fuzzy DEA model 中的 undesirable inputs/outputs。

让我们来看示例 8。

---

**示例 8：read_data_fuzzy()函数**

1. 创建一个新的脚本，将其命名为“*example_read_data_fuzzy*”

   提示：如果已经关闭了工作会话窗口，我们先打开项目“*Paper_1*”，然后创建新脚本，同时需要加载 **deaR**。

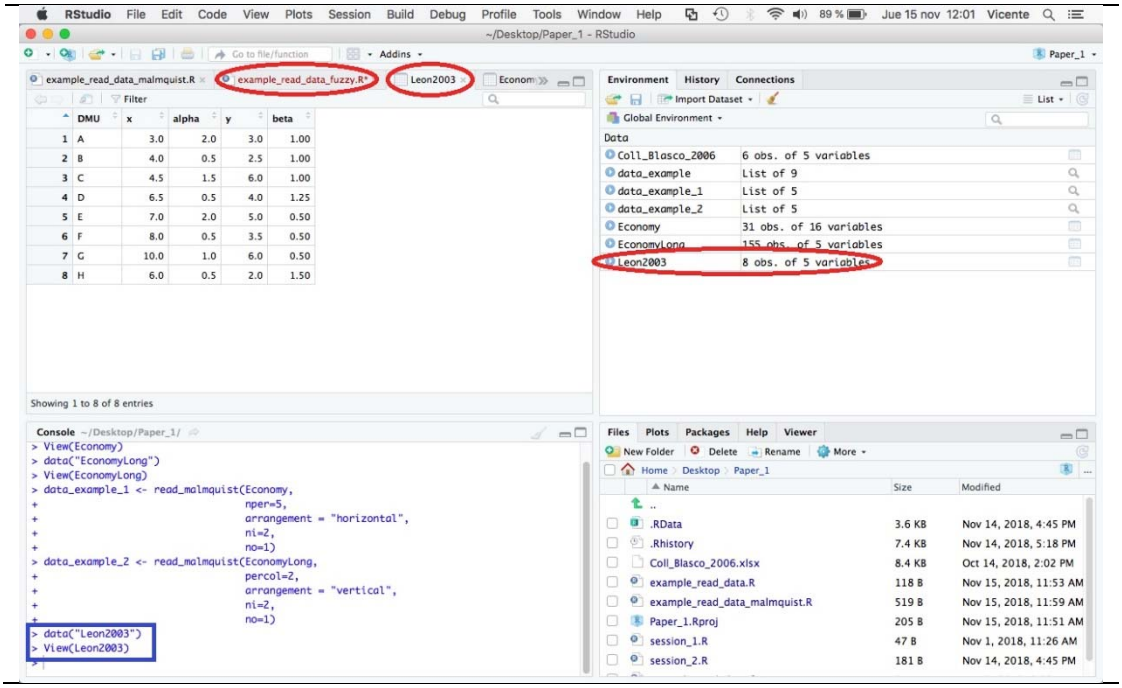
2. 加载数据集“*Leon2003*”[10]：

   **data(“Leon2003”).**

   该数据集（参见图 33）有 8 个 DMU、1 个 symmetric triangular fuzzy input（对称三角模糊输入）（alpha 是 input 的扩展）和 1 个 symmetric triangular fuzzy output（对称三角模糊输出）（beta 是 output 的扩展）

[10] 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

图 33：数据集 Leon2003

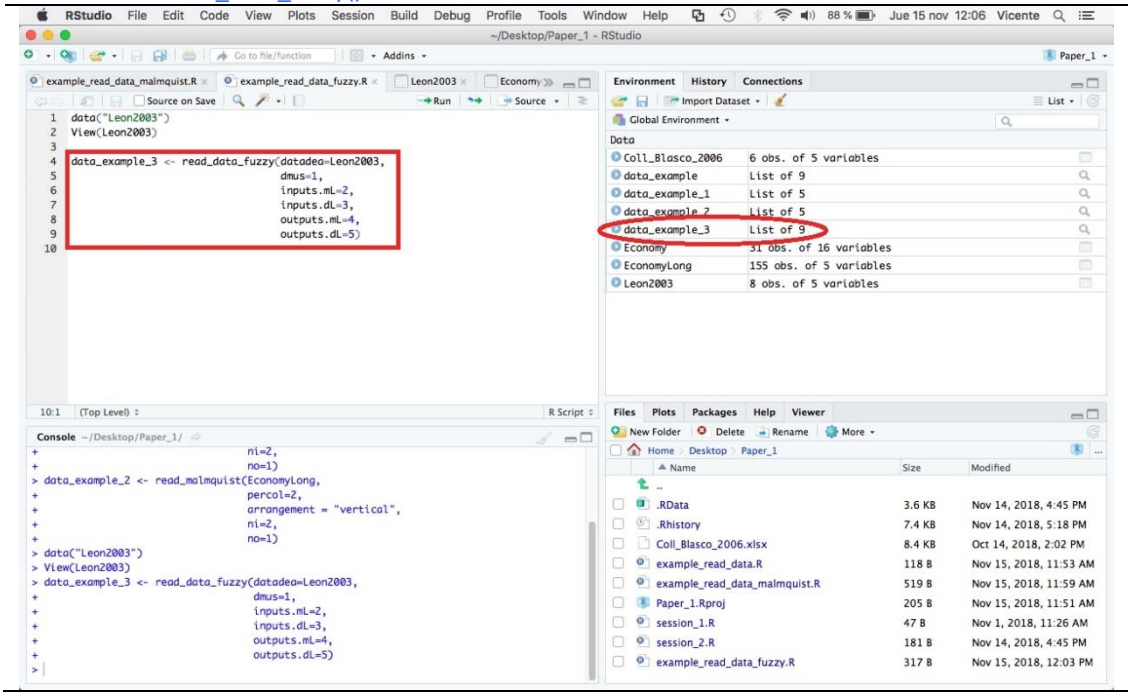


3. 现在，运行 read_data_fuzzy()函数用于调整数据。为此，我们写入并执行以下指令：

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

显示出的工作会话窗口应该如以下屏幕截图（参见图 34）：

图 34：使用 read_data_fuzzy()函数



在这个示例中，inputs.dL=alpha 并且 ouputs.dL=beta。由于我们有 symmetric triangular fuzzy numbers（对称三角模糊数），所以 inputs.dL=inputs.dR 并且 outputs.dL=outputs.dR。

现在，我们已经创建了一个新的 R 对象：“*data_example_3*”，它也是我们即将要运行的 fuzzy DEA model 的对象。

保存“*example_read_data_fuzzy.R*”，关闭项目并退出 RStudio。

### 7.3. 选择和运行 DEA Model

一旦我们将数据调整为 **deaR** 的可读格式，下一步就是选取 DEA Model 并运行它。

在当前版本的 **deaR**（版本 1.0）中，可以使用以下 DEA model：

| **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 |
| Cross-efficiency (crs[11] and vrs[12]) |
| Bootstrapping (Simar and Wilson algorithm) |
| FDH model |
| **Productivity** |
| Malmquist index |
| **Fuzzy DEA models** |
| Kao and Liu model[13] |
| Possibilistic model |
| Guo and Tanaka model |
| Fuzzy cross-efficiency[14] (only crs) |

[11] crs = constant returns-to-scale. crs =固定规模收益（constant returns-to-scale）

[12] vrs = variable returns-to-scale. vrs =变动规模收益（variable returns-to-scale）

[13] This Fuzzy DEA model has been extended to a several DEA models. See the help for the package: **help(“modelfuzzy_kaoliu”).** Fuzzy DEA model 已经被扩展成几个不同的 DEA model，详情请使用 **help(“modelfuzzy_kaoliu”)**获取软件包帮助信息。

[14] Based on Guo and Tanaka’s model. 基于 Guo and Tanaka’s model

在 **deaR** 文档中，我们可以找到如何应用这些模型函数的所有详细说明，以及将这些函数付诸实践的不同示例。

尽管在 **deaR** 中只提供了如上列举的 DEA model，但鉴于其编程的灵活性（这是 **deaR** 软件包的一个重要优势），用户可以自己试验、尝试和实施这些模型的不同变体。例如，如果用户在“*Additive model*”中适当地定义权重，就可以获得 MPI[15]模型或 RAM[16]模型。

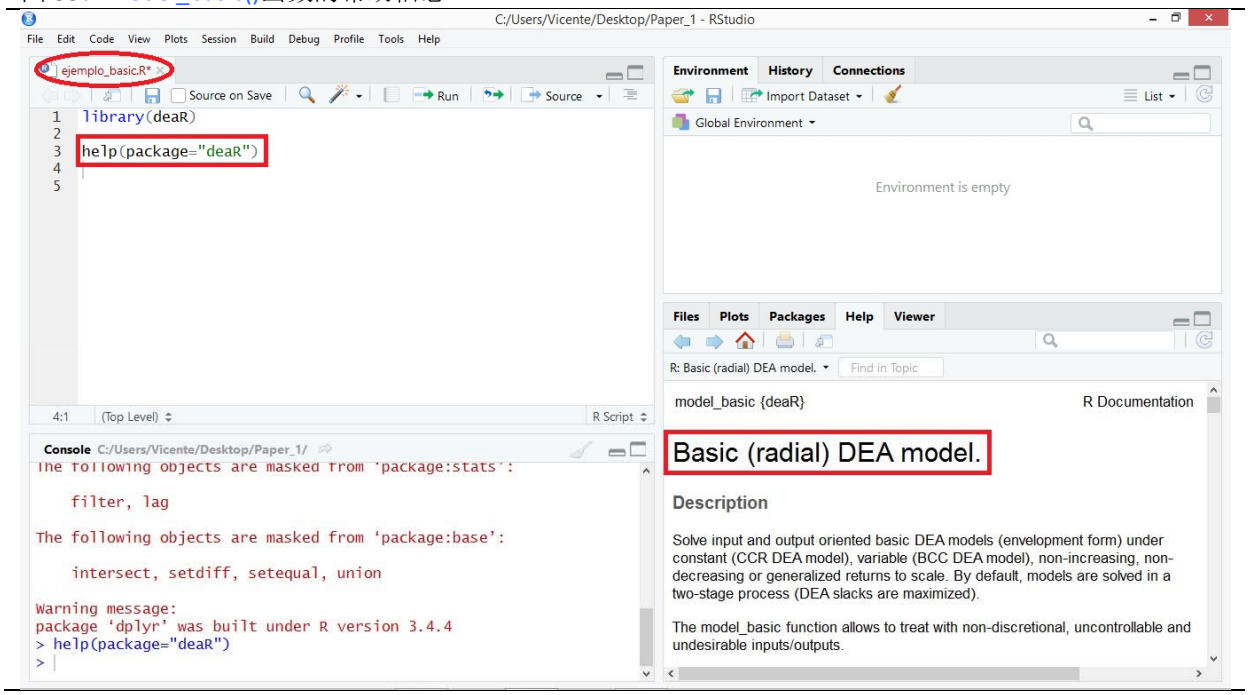
另一个用户通过定义权重获得不同模型的示例来自于“*Preference Structure Model*”(weighted non radial model)【“*偏好结构模型*”（加权非径向模型）】。在这种情况下，用户可以计算“*Cost Efficiency*”（“*成本效率模型*”）、“ *Revenue Efficiency*”（“*收入效率模型*”）或者“*Profit Efficiency*”（“*利润效率模型*”）。

现在，我们来看如何使用 model_basic()函数执行 basic DEA model：

- 打开项目“*Paper_1*”，创建一个新脚本，并将其命名为：“*example_basic*”。
- 加载 **deaR**
- 转到 **deaR** 的帮助页面
- 单击 *model_basic* 链接（参见图 35）。我们也可以在脚本中写入以下指令：

**help(“model_basic”)**

图 35：model_basic()函数的帮助信息



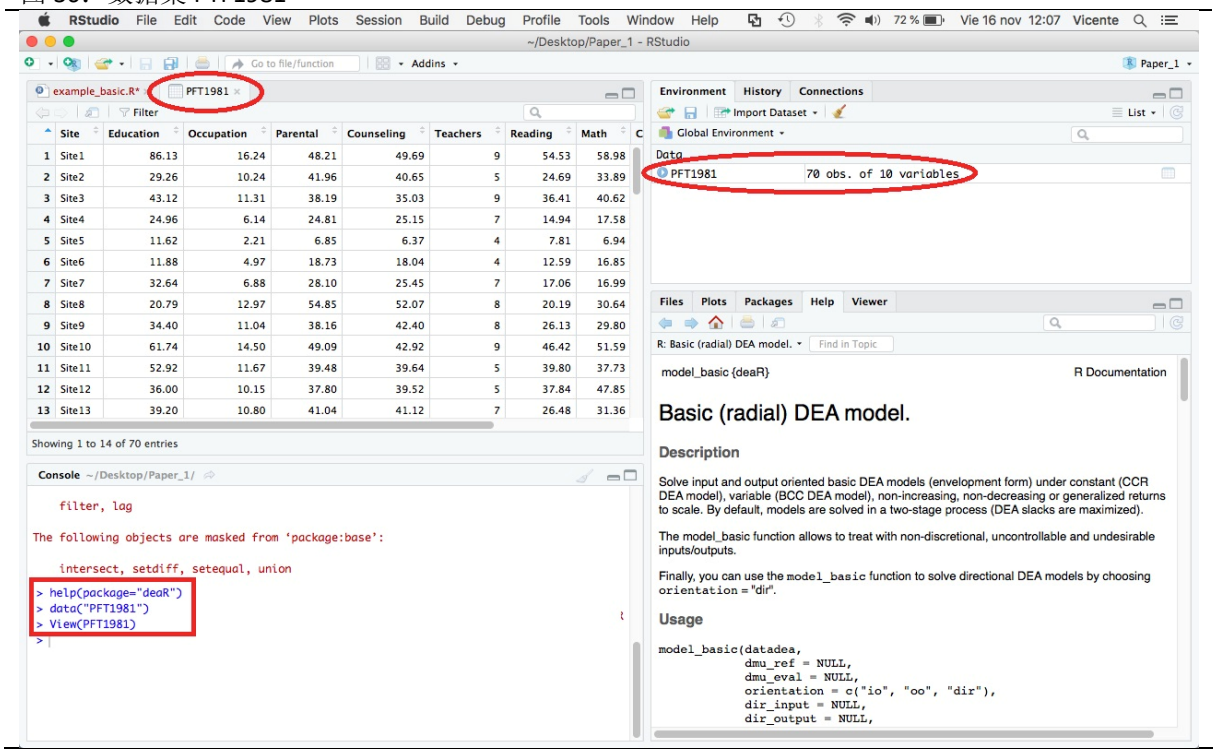

[15] Measure of Inefficiency Proportions (MPI)
[16] Range Adjusted Measure (RAM).

model_basic()函数采用以下参数：

- *datadea*: 它是一个数据集（且必须是一个数据框）
- *dmu_ref*: 选择 DMU 的子集
- *dmu_eval*: 从子集中选取的待评估的 DMU
- *orientation*: DEA model 的定位/定向（orientation）: input-oriented， output-oriented 或者 direccional
- *dir_input*: 在方向模型（directional models）中 input 的方向矢量（direction vector）
- *dir_output*: 在方向模型（directional models）中 output 的方向矢量（direction vector）
- rts: 规模收益（Returns to scale）【包括固定（constant）规模收益、变动（variable）规模收益、非递增（non-increasing）规模收益、非递减（non-decreasing）规模收益和普遍性（generalized）规模收益】
- *L*: 选择 generalized rts 时，lower bound.
- *U*: 选择 generalized rts 时，upper bound.
- *Maxslack*: 默认设置下， slack 数值在第二阶段最大化
- *weight_slack_i*: 用户可以在第二阶段设置 input slack 的权重以使其实现最大化
- *weight_slack_o*: 用户可以在第二阶段设置 output slack 的权重以使其实现最大化
- *vtrans_i*: 针对不良输入（undesirable inputs）和变动规模收益（variable returns-to-scale），用户可以设置平移矢量（translation vector）。默认设置下，平移矢量为(max + 1)
- *vtrans_o*: 针对不良输出（undesirable outputs）和变动规模收益（variable returns-to-scale，用户可以设置平移矢量（translation vector）。默认设置下，平移矢量为(max + 1)。
- *compute_targets*: 计算最大 slack 解决方案的目标
- *compute_multiplier*: 返回 multiplier form 的乘数（或权重）
- *returnlp*: 针对每一个 DMU, 此参数返回至第一阶段的线性问题（linear problem ）

现在，我们将通过示例 9 和示例 10 学习如何使用 model_basic()函数。首先，我们需要在 **deaR** 中加载数据集"*PFT1981*"[17]。这个数据集有 70 个 DMU（school site），5 个 input（*Education*, *Occupation*, *Parental*, *Counseling*, *Teachers*）和 3 个 output（*Reading*, *Math*, *Coopersmith*）。下图（图 36）显示了为加载该数据我们要在脚本"*example_basic*"中写入的指令。

图 36：数据集 PTF1981




## 示例 9： model_basic()函数

在"*PTF1981*"的 70 个 DMU 中，有 49 个位于 Project Follow Through (PFT)，21 个位于 Non-Follow Through (NFT)。

在这个示例中，我们使用 input-oriented CCR DEA moldel 来计算 PFT 中的 49 个 DMU 的效率。

首先，我们运用 read_data()函数把数据调整为 **deaR** 的可读模式（参见 7.2.2 节）。点击 Run 执行指令。

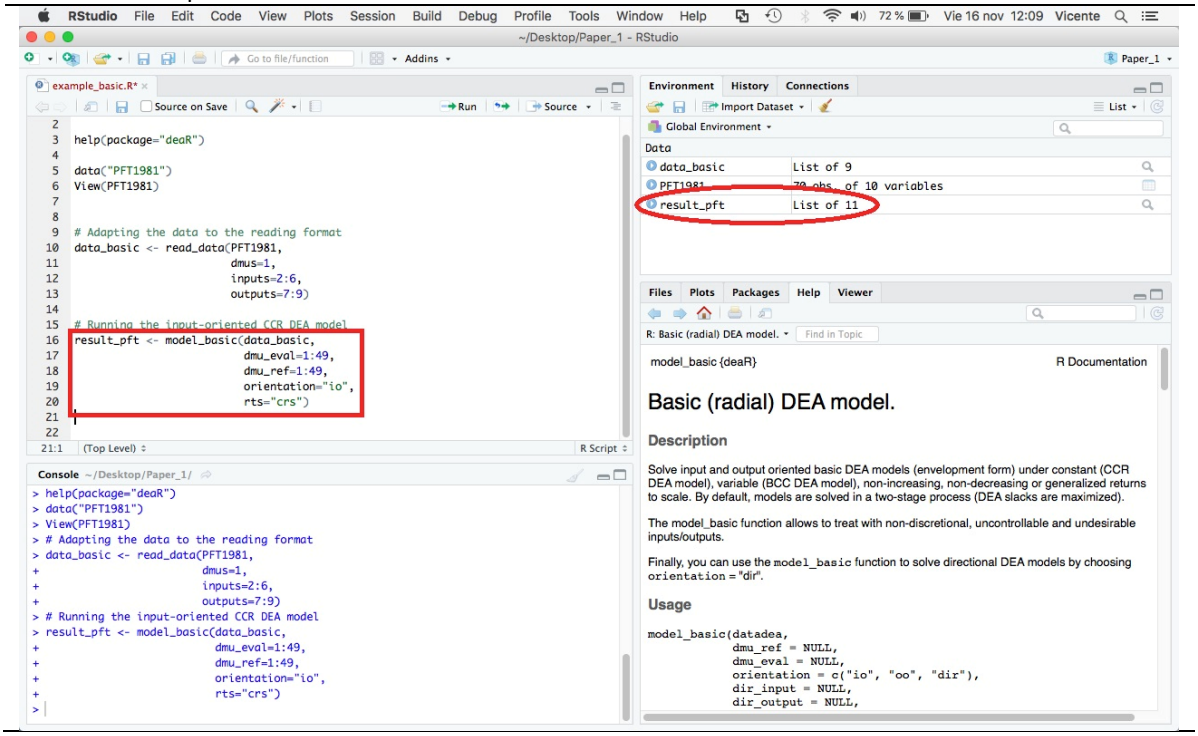
接下来，使用 model_basic()函数。因为我们希望通过使用 CCR DEA model（它是一个 conventional DEA model）来测量在 PFT 的 49 个 DMU 的效率，为此，我们使用参数 *dmu_ref=1:49*；如果要评估所有单位（units），则使用参数 *dmu_eval=1:49*。因此，我们在脚本"*example_basic*"中写入并执行以下指令（参见图 37）：

[17] 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")
```

提示：我们可以逐条运行指令，也可以全选后再运行它们。

图 37：运行 input-oriented CCR DEA model 测量 PFT 中的 DMU

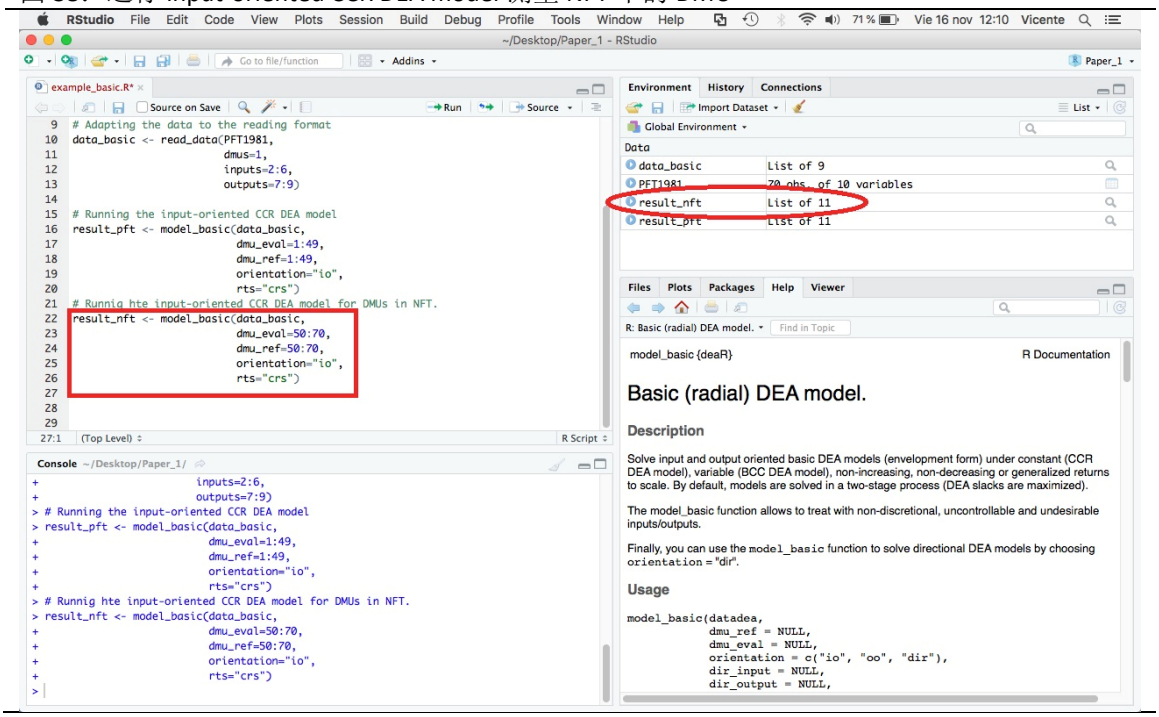



## 示例 10：model_basic()函数

现在，**你可以通过运用 CCR input-oriented DEA model 来计算位于 NFT 中的 DMU 的效率吗？**（NFT 中的 EMU 排列于 50-70）

*这个问题的答案将显示在下一页（参见图38）*

图 38：运行 input-oriented CCR DEA model 测量 NFT 中的 DMU

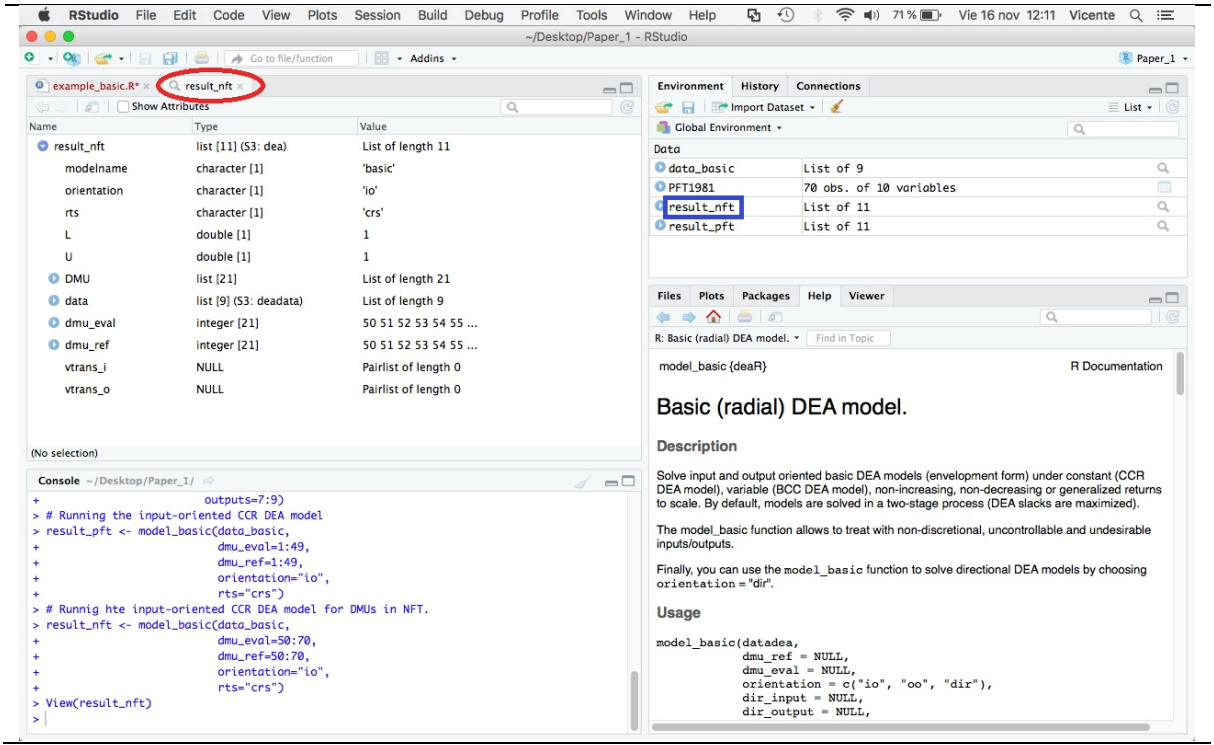



保存脚本“*example_basic*”

## 7.4. 提取主要结果

在示例 9 和示例 10 中，运行 model_basic()函数之后，所有的结果都被分别保存在对象“*result_pft*”和“*result_nft*”当中。可以看到，这两个对象都是分别包括 11 个组件的列表。我们可以通过单击对象的名称（参见图 39）来查看每个列表的内容，也可以通过单击蓝色箭头的图标（）来查看他们的结构。

图 39：basic DEA model 的结果及其结构



**deaR** 中有几个特定的函数用于获取（还是提取）DEA 分析的主要结果。这些函数是：

- ✓ **efficiencies()**: 提取效率分数（efficiency score）
- ✓ **slacks()**: 提取 input 和 output 的 slack 数值
- ✓ **targets()**: 提取 input 和 output 的目标值（target value）
- ✓ **lambdas()**: 提取 lambda 数值【或强度（intensity）】
- ✓ **references()**: 提取低效 DMU 的参考集（reference set）
- ✓ **rts()**: 提取规模收益（returns-to-scale）
- ✓ **multipliers()**: 提取 multiplier DEA form 中的乘数（或权重）

在接下来的示例 11 中，我们将练习使用这些函数。

---

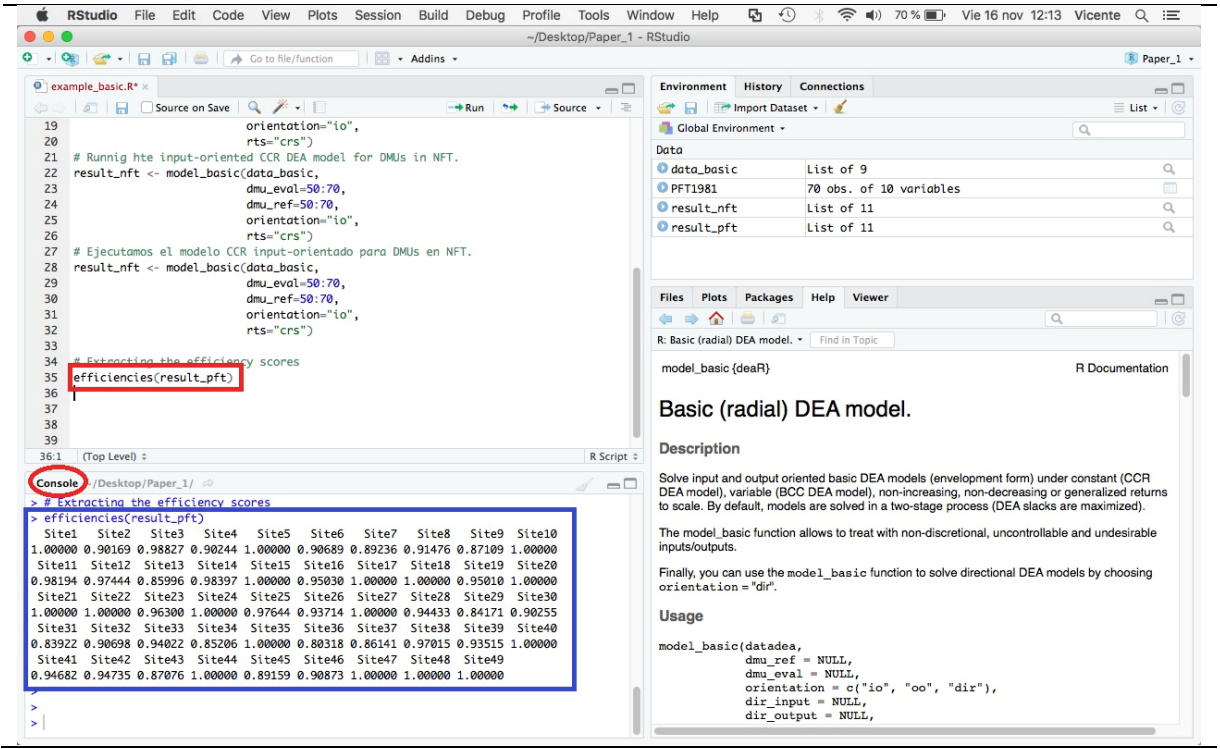
## 示例 11：提取结果

要从“*result_pft*”中提取效率分数（efficiency score），我们只需在脚本中写入并执行以下指令：

```
efficiencies(result_pft)
```

结果将会显示在 *Console* 面板中，如下图所示：

图 40：PFT 中的 DMU 的效率分数




同样地，要提取低效 DMU 的参考集（reference set），我们写入并执行以下指令：

```
references(result_pft)
```

其余的(slacks()函数、targets()函数、lambdas()函数、rts()函数和 multipliers()函数都以类似的方式运行。请尝试练习使用这些函数，并关注你获得的结果。

保存脚本“*example_basic..R*”，关闭项目并退出 RStudio。

### 7.5. 函数 summary()

除了上文列举的函数（如 efficiencies()、lambdas()等）外，**deaR** 还有 summary()函数，用于汇总 DEA 分析的全部结果。它同时适用于 conventional DEA model 和 fuzzy DEA model 的结果汇总。要使用该函数，我们需要了解它具有以下结构[18]：

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

其采用的参数分别为：

- *object*: 对象，我们向它分配了 conventional DEA model 或 fuzzy DEA model 的分析结果
- *exportExcel*: 该项参数的默认值为 TRUE。基于此，summary()函数将会自动在工作目录下生成包含主要结果的 Excel 文件。用户也可以选择不生成 Excel 文件（exportExcel = FALSE）
- *filename*: 该项参数的默认值为 NULL。基于此，生成的 Excel 文件的默认名称为“*ResutsDEAyear-month-dat_hour:minute*:second.xlsx”。当然，用户也可以自己给 Excel 文件命名。

正如前面提到的，在 conventional DEA model 和 fuzzy DEA model 中运用 summary()函数汇总 DEA 分析结果的方法都是相同的，其差异可以在函数生成的 Excel 文件中找到，即被创建的 Excel 工作表会根据分析模型的不同而有所差别。我们将在示例 12 和示例 13 中看到具体内容。

---

**示例 12：conventional DEA model**

请依照下面的步骤：

1. 打开项目“*Paper_1*”
2. 创建一个新脚本：“*Summary_DEA*”
3. 加载**deaR**

**假设**：加载 **deaR** 提供的数据集“*Hua_Bian_2007*”[19]。这个数据集有 30 个 DMU，2 个 input（*D-Input1*，*D-Input2*），2 个 output（*D-Output1*，*D-Output2*）和 1 个 undesirable output（*UD-Output1*）（其排列顺序与图片显示的顺序一致）

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

我们希望从 output-oriented BBC DEA model 中提取结果汇总。由于该数据中含有 1 个 undesirable output，为了在分析中把它考虑其中，我们使用 Hua and Bian (2007)当中用到的平移参数（translation parameter）：vtrans_o=1500

**非常重要：请牢记使用 deaR 的步骤**

步骤 1. 加载数据

步骤 2. 根据 **deaR** 的读取格式调整数据

步骤 3. 运行 DEA model

步骤 4. 提取结果

现在就试试吧！！！

（答案将显示在下一页）

[19] 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

**答案：**正如我们在图 41 中看到的，要解决示例 12 的问题，我们需要在脚本“Summary_*DEA*”中依次写入如下指令：

步骤 1. 加载数据：

```
data(“Hua_Bian_2007”)
```

步骤 2. 调整数据：

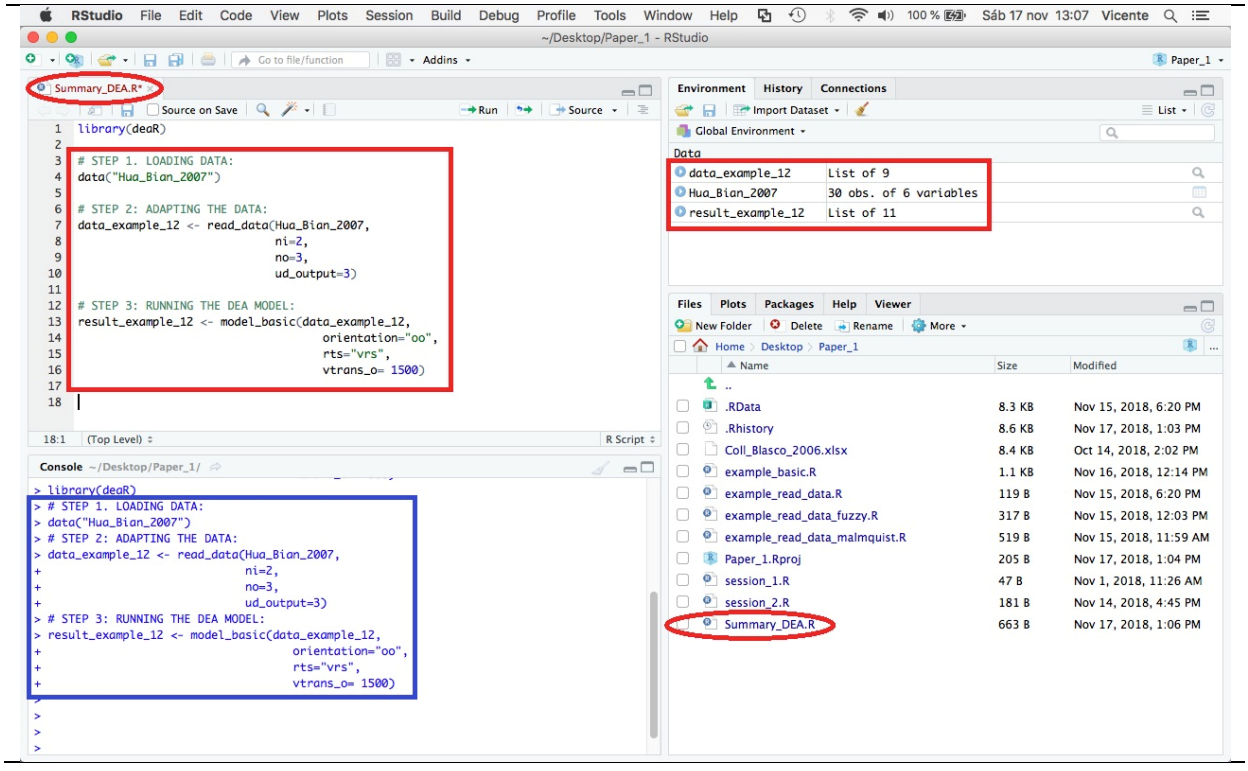
```
data_example_12 <- read_data(Hua_Bian_2007,
                             ni=2,
                             no=3,
                             ud_output=3)
```

**提示：**有 3 个 output（no=3），其中第 3 个是 undesirable output（ud_output=3）

步骤 3. 运行 DEA model：

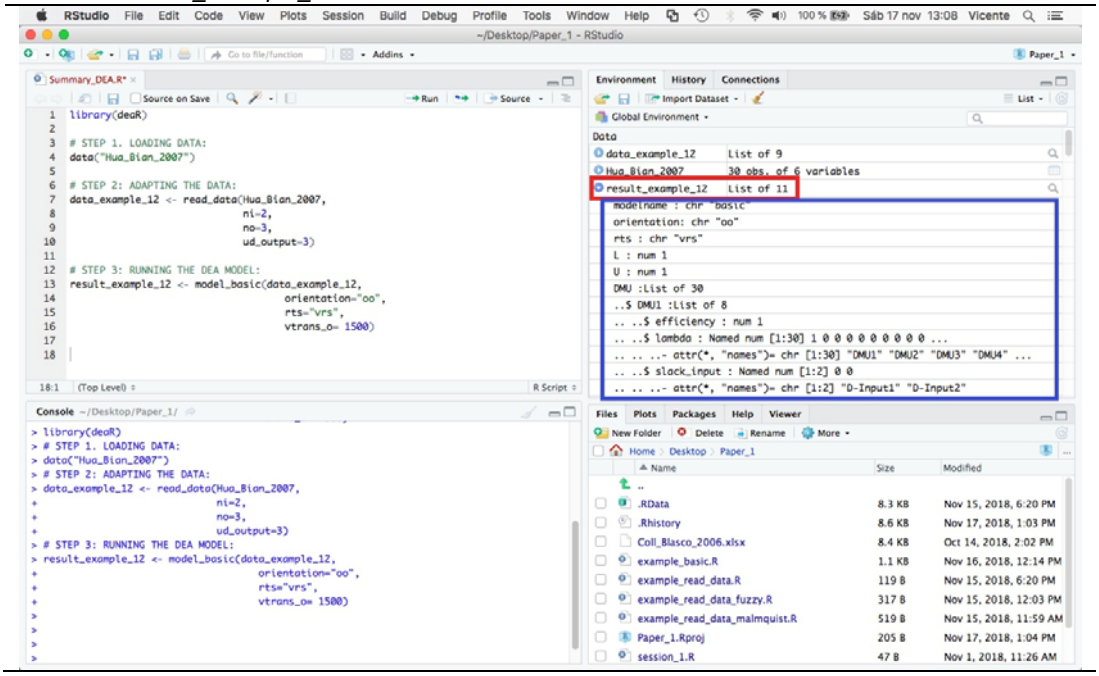
```
result_example_12 <- model_basic(data_example_12,
                                 orientation=”oo”,
                                 rts=”vrs”,
                                 vtrans_o= 1500)
```

图 41：示例 12 解答




现在，所有基于 output-oriented BCC DEA model 操作而获得的结果都可以在对象“*result_example_12*”中找到，这是一个包含了 11 个组件的列表（参见图 42）。

图 42：“*result_example_12*”的结构

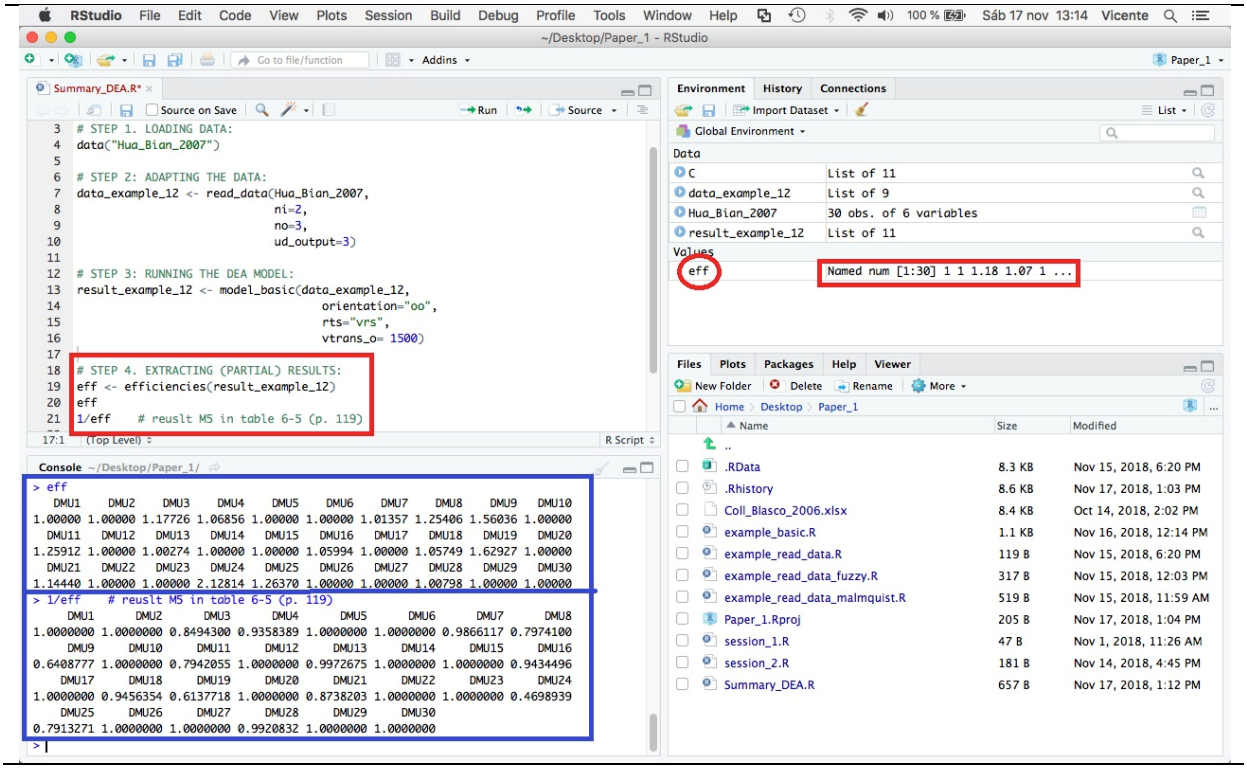


到这里，我们知道了如何通过使用函数 efficiencies()、lambdas()、multipliers()、rts()、references()、slacks()、targets()提取分析的部分结果，以及如何使用函数 summary()获得分析的结果汇总。

如图 43 所示，我们使用 efficiencies()函数来提取 DMU 的效率分数（efficiency score）。效率（efficiency）在 *Consol*e 列出并被分配给对象“*eff*”。要获得和 Hua and Bian (2007)研究同样的结果，我们在脚本中写入并执行以下指令：

```
1/eff
```

图 43：效率分数

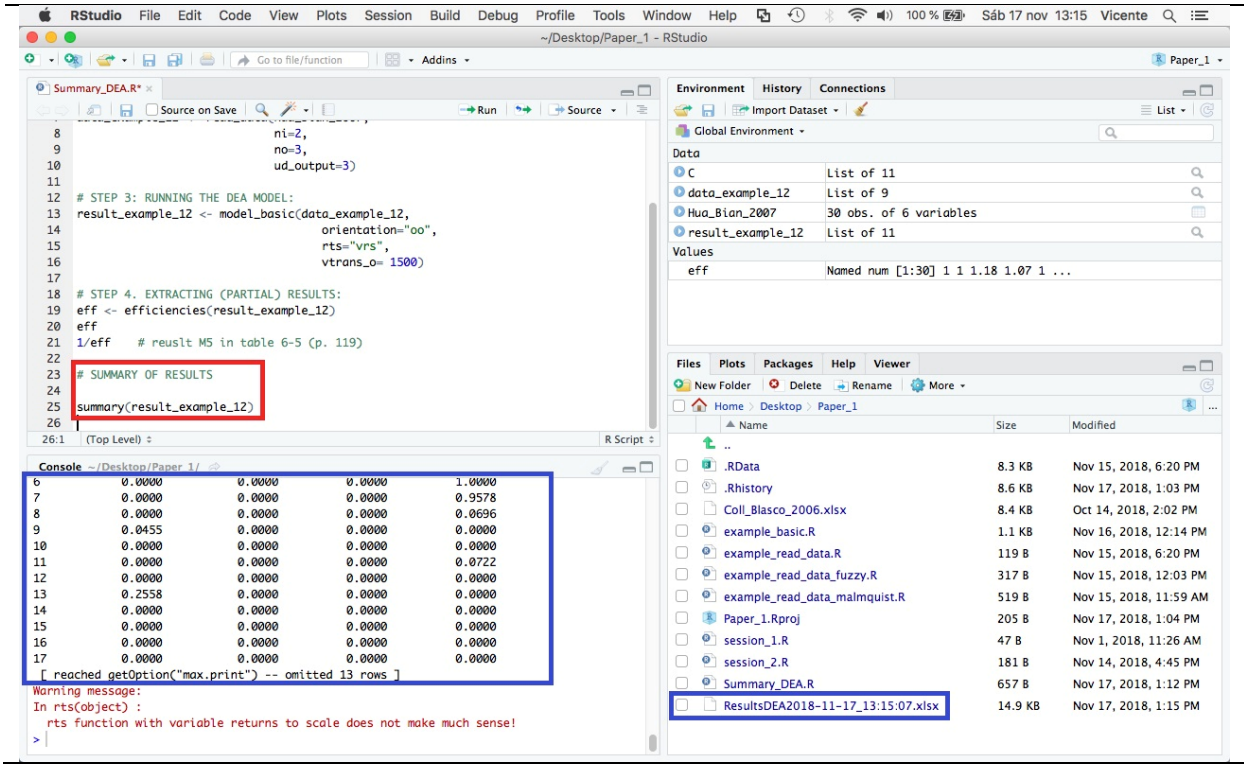


要获取基于 output-oriented BCC DEA model 的结果汇总，我们使用 summary()函数。在脚本“*Summary_DEA*”中写入以下指令：

```
summary(result_example_12)
```

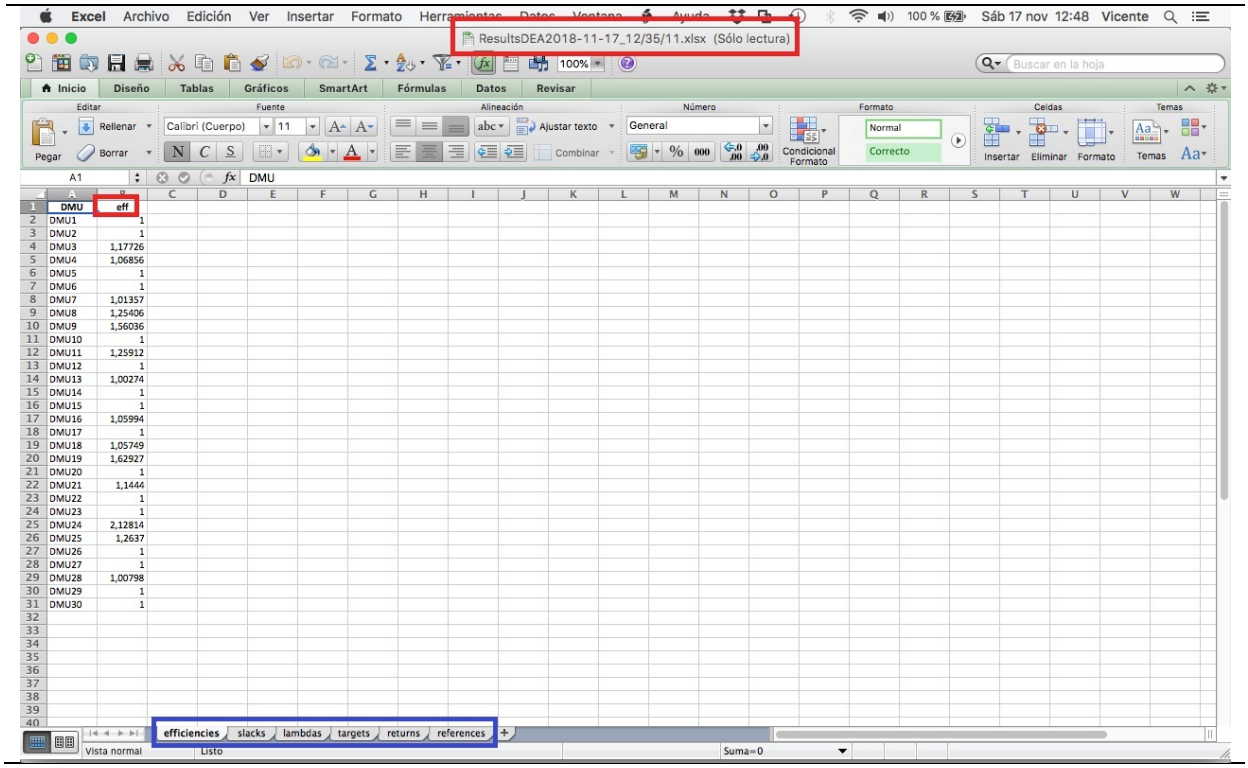
可以看到，所有的结果都在 *Console* 面板显示了出来。尽管没有多少 DMU（只提取到了 30 个），但结果却很多。另外，想要运行 summary()函数后直接在屏幕上查看所有结果的想法是不切实际的。正如我们在图 44 中看到的那样，**deaR** 也会自动生成一个包含了所有结果的 Excel 文件。请注意，默认设置下该文件的名称是预先给定的。

图 44：示例 12 的结果汇总




要打开 Excel 文件来查看所有结果（参见图 45），请单击文件“*ResutsDEAyear-month-dat_hour:minute*:second.xlsx”并选择“*View file*”选项即可。

图 44：Excel 文件中的汇总结果




保存脚本“*summary_DEA*”

接下来，我们将在示例 13 中重复 León, et al. (2003)[20]的研究结果。

**示例 13**：结果汇总：**Posibilistic fuzzy DEA model**

León et al.(2013)在包络数据分析模型 input-oriented BCC model 中应用 possibilistic 编程技术来测量效率。作者在文章中使用的数据可以在 **deaR** 提供的数据集“Leon2003”中找到。

首先我们创建一个新脚本：“*Summary_DEA_fuzzy*”。为了重现作者在表 2 中展示的研究结果，我们在脚本中写入：

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

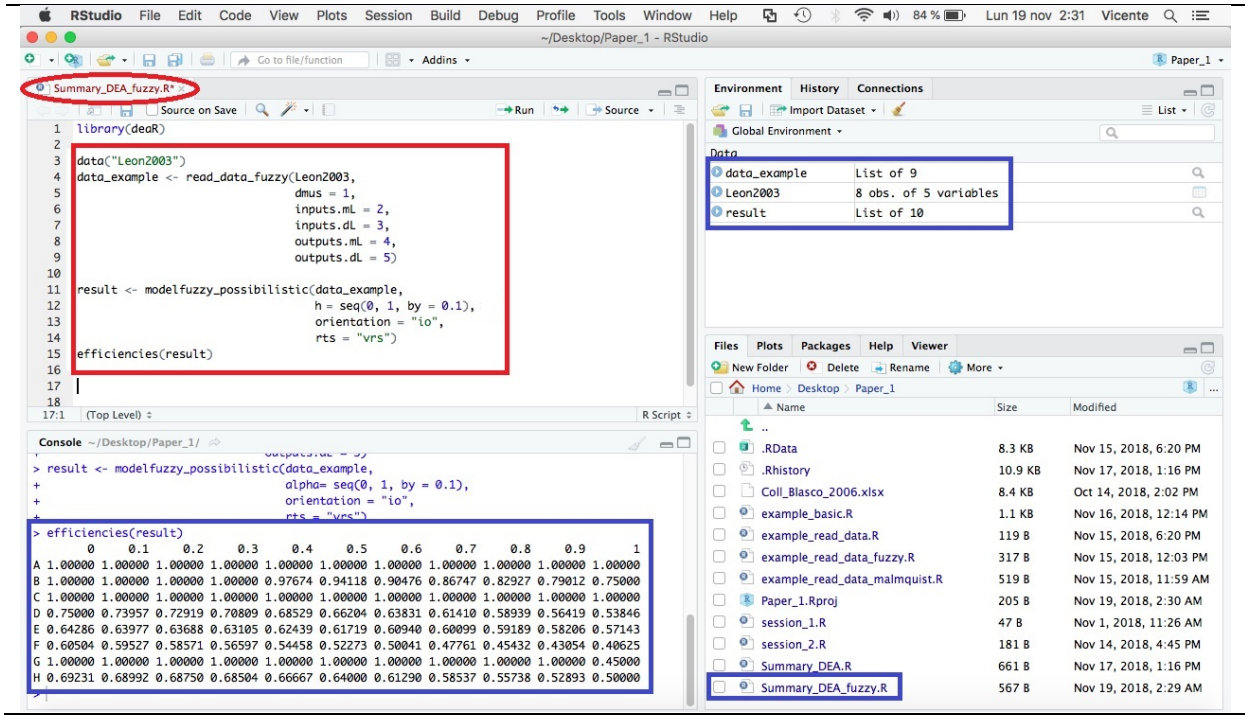
[20] Léon, T.; Liern, V. Ruiz, J.; Sirvent, I. (2003). "A Possibilistic Programming Approach to the Assessment of Efficiency with DEA Models", Fuzzy Sets and Systems, 139, 407–419. https://doi.org/10.1016/S0165-0114(02)00608-5

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

**提示：**h = seq (0, 1, by = 0.1)生成一个表示不同程度可能性的价值序列：h = (0, 0.1, 0.2, ..., 1)

执行命令后，针对不同程度（水平？）可能性 h 的 BCC input-oriented model 效率分数（efficiency score）将会显示在 *Console* 面板中（参见图 46）。

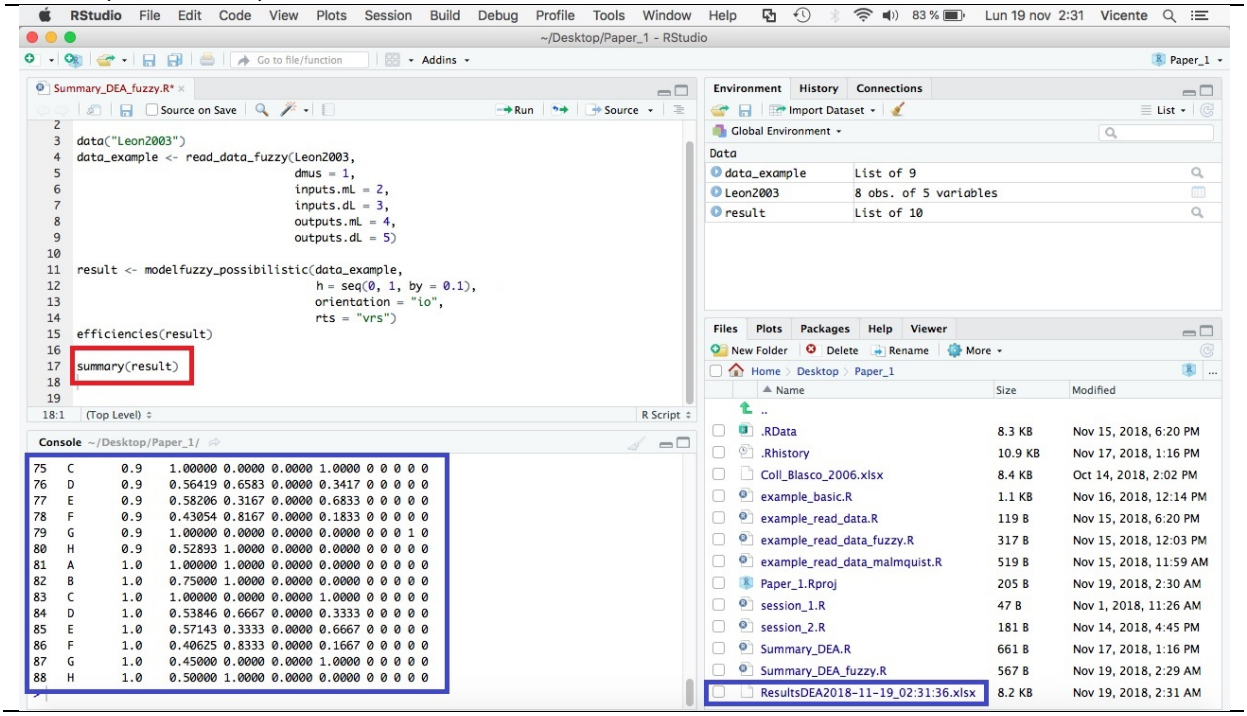
图 46：不同 h 层级的效率分数




想要在当前屏幕上获取结果汇总（参见图 47）并将其输出到 Excel 文件，我们在脚本中写入并执行以下指令：

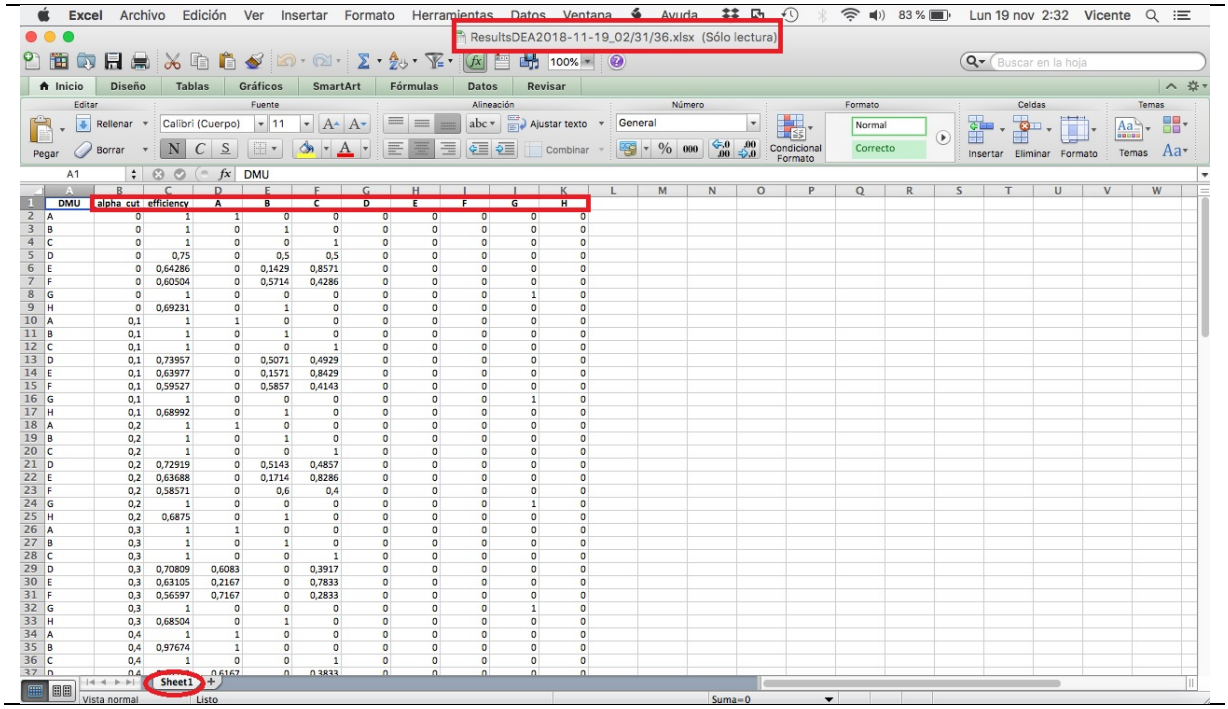
```
summary(result)
```

图 47：possibilistic input-oriented BCC DEA model 的汇总结果




就 possibilistic model 而言，汇总后的研究结果包括：（1）效率分数（efficiency score）和（2）参考集（reference set）

图 48：Excel 文件中的汇总结果




保存脚本“*Summary_DEA_fuzzy*”。

### 7.6. 图表结果：plot()函数

运用 plot()函数，我们可以在 conventional DEA model、Malmquist 生产力指数（Malmquist index）或 cross-efficiency model 获得一些结果的可视化图表。示例 14、示例 15 和示例 16 将分别对此进行演示说明。

plot()函数采取以下方式运行：

**plot(*x*)**

在这里，**x** 是储存了 DEA model 研究结果的对象。

运行 plot()函数后，我们将在 *Viewer* 标签下（右下窗口）获得绘图结果。我们可以通过单击 *Viewer* 的 *Export* 图标来保存绘图，图片格式可以是“*png*”、“*jpeg*”或者“*tiff*”。我们还可以把绘图复制到系统剪贴板，然后粘贴到 word 文档中。

**deaR** 的当前版本暂时尚未提供 fuzzy DEA models 的图表。

目前我们正在致力于改进 **deaR** 中的图表结果，它们将被包含在下一个版本中。

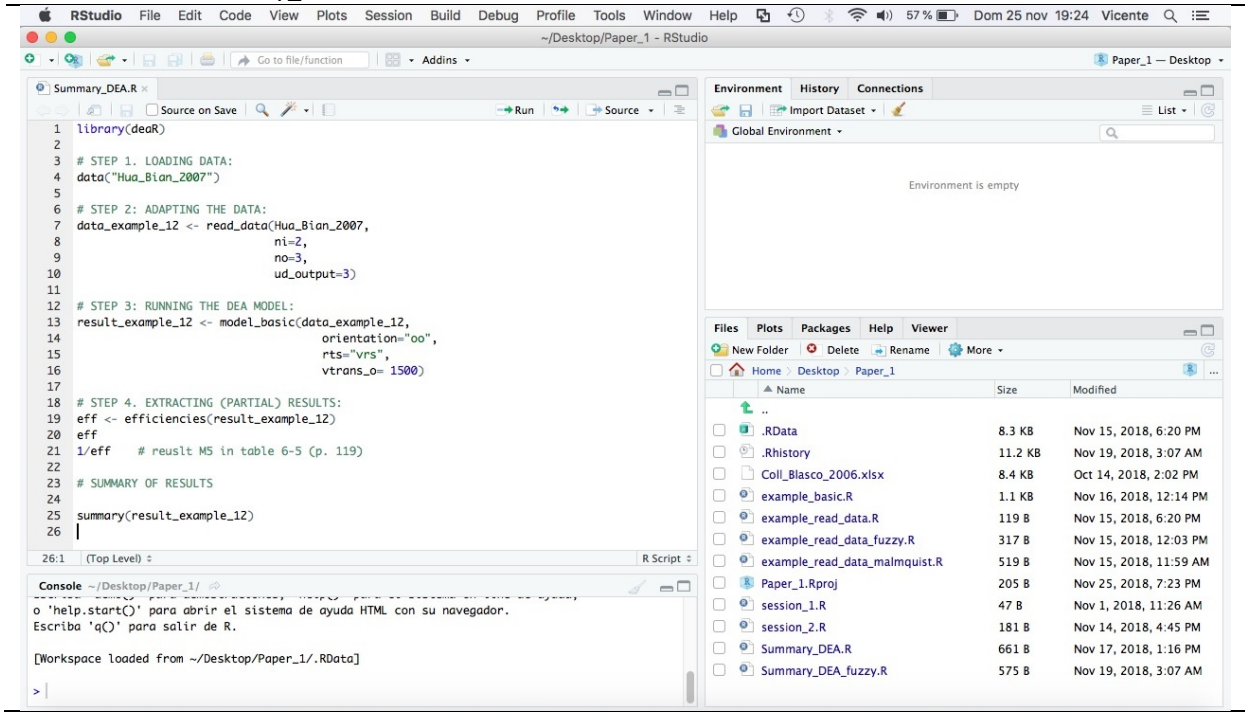
---

**示例 14：绘图：Basic DEA model**

打开脚本“*Summary_DEA*”。如果已经关闭了工作会话窗口，我们需要打开项目“*Paper_1*”，加载 **deaR**，然后再打开脚本。

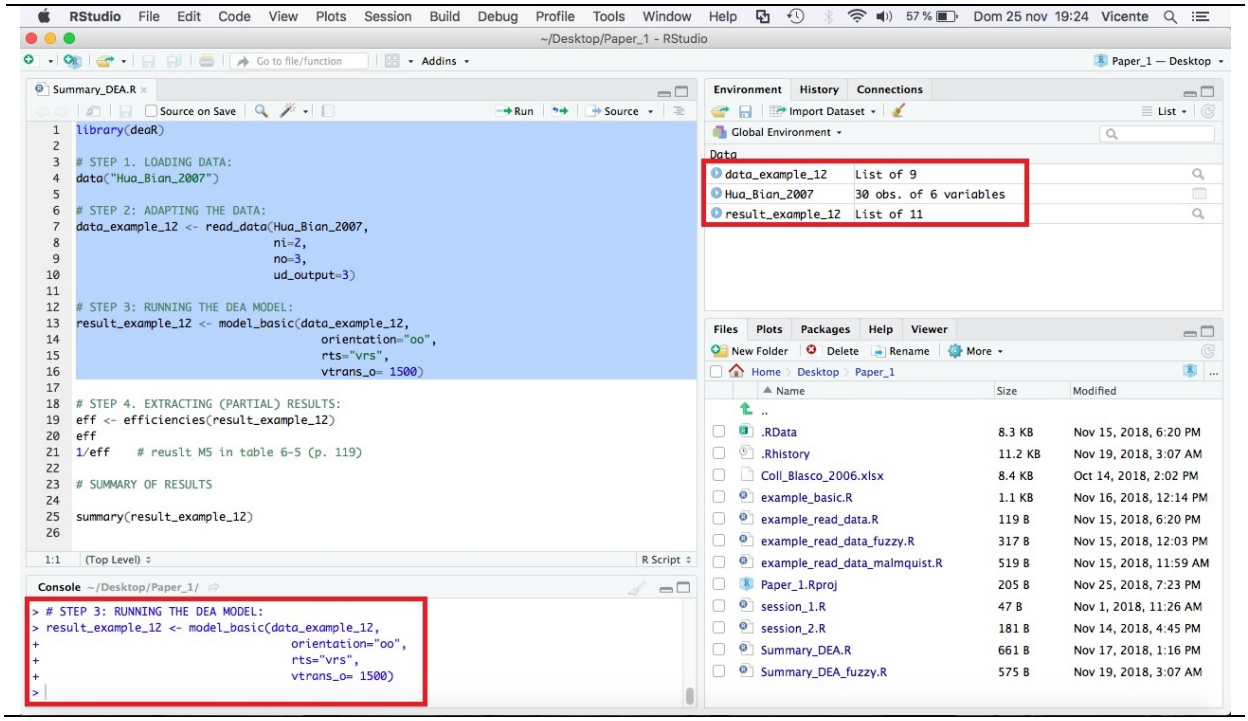
我们的工作会话窗口应该如图 49 所示：

图 49：脚本“*Summary_DEA*”




如图 50 所示，我们选中第 1 行 **library(deaR)**至第 16 行，然后运行被选中的指令。得到的结果是：3 个对象被列入了 *Environment* 菜单中，它们分别是“*data_example_12*”、“*Hua_Bian_2007*”和“*result_example_12*”。

图 50：执行脚本命令




DEA model 的结果被分配给（或被储存在）了对象“*result_example_12*”中。为了绘制其中一些结果，我们在脚本的第 27 行写下并运行以下指令：

```
plot(result_example_12)
```

之后 *Console* 面板会显示如下信息：

```
Press [enter] to continue
```

按下 *Enter* 键，在 *Viewer* 标签（右下窗口）下我们就会看到如图 51 所示的一个图表。

图 51：示例 12 之图 1

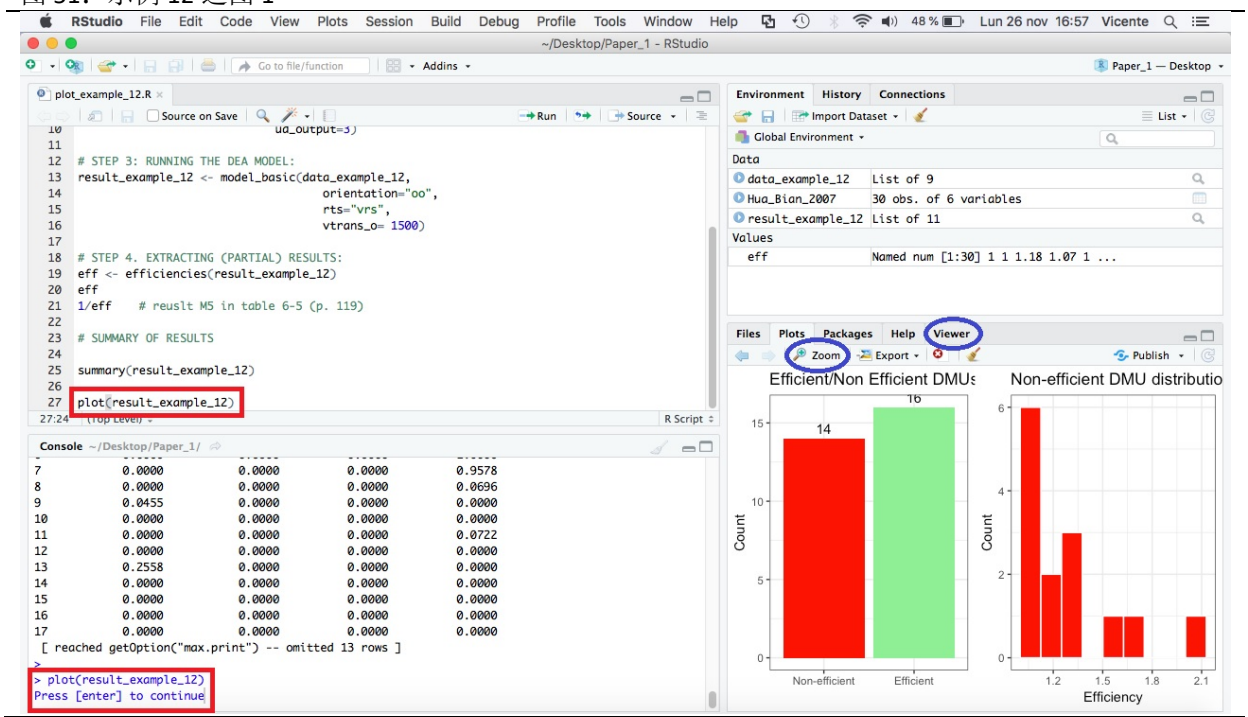
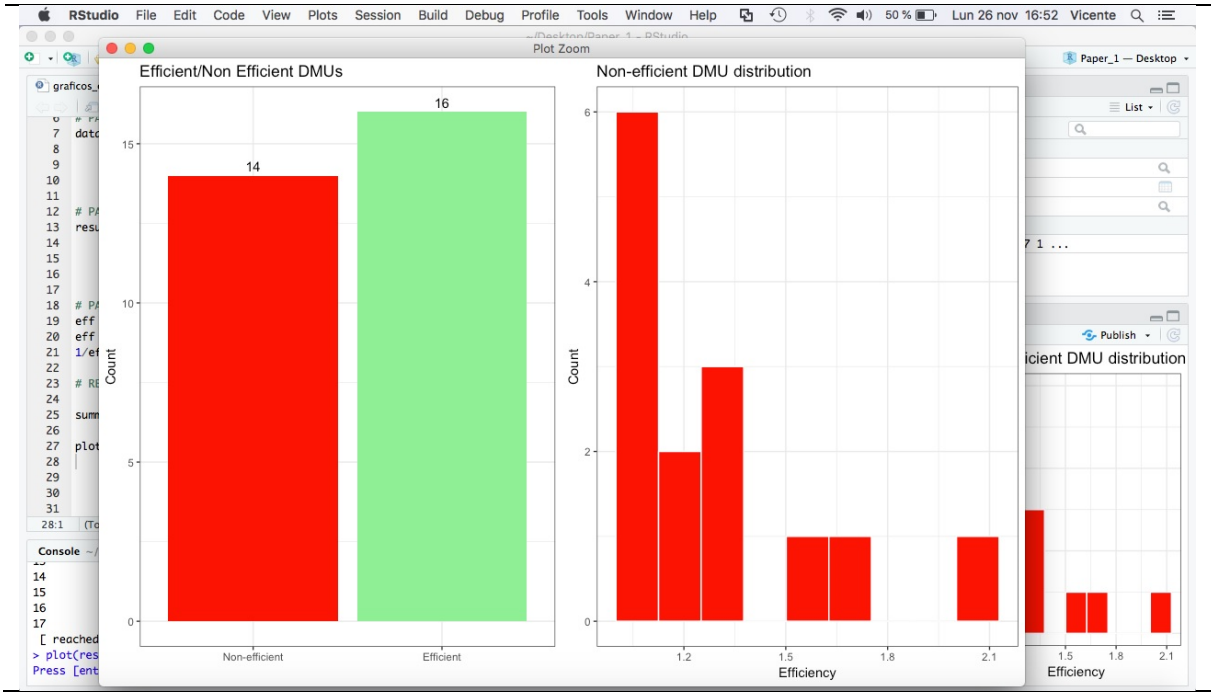


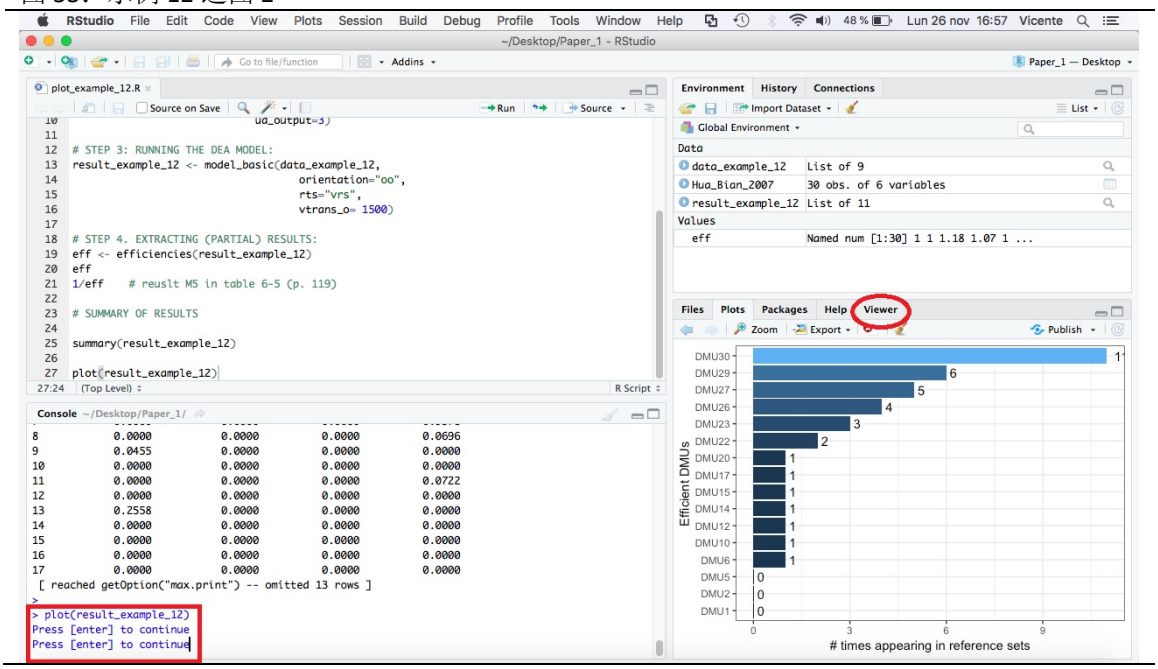
图 1 显示了有效 DMU（efficient）和低效 DMU（inefficient）的数量（右图），以及后者效率分数（efficiency score）的分布情况（左图）。如果我们点击 Zoom 按钮，图形可以被放大（参见图 52）。

图 52：图表放大



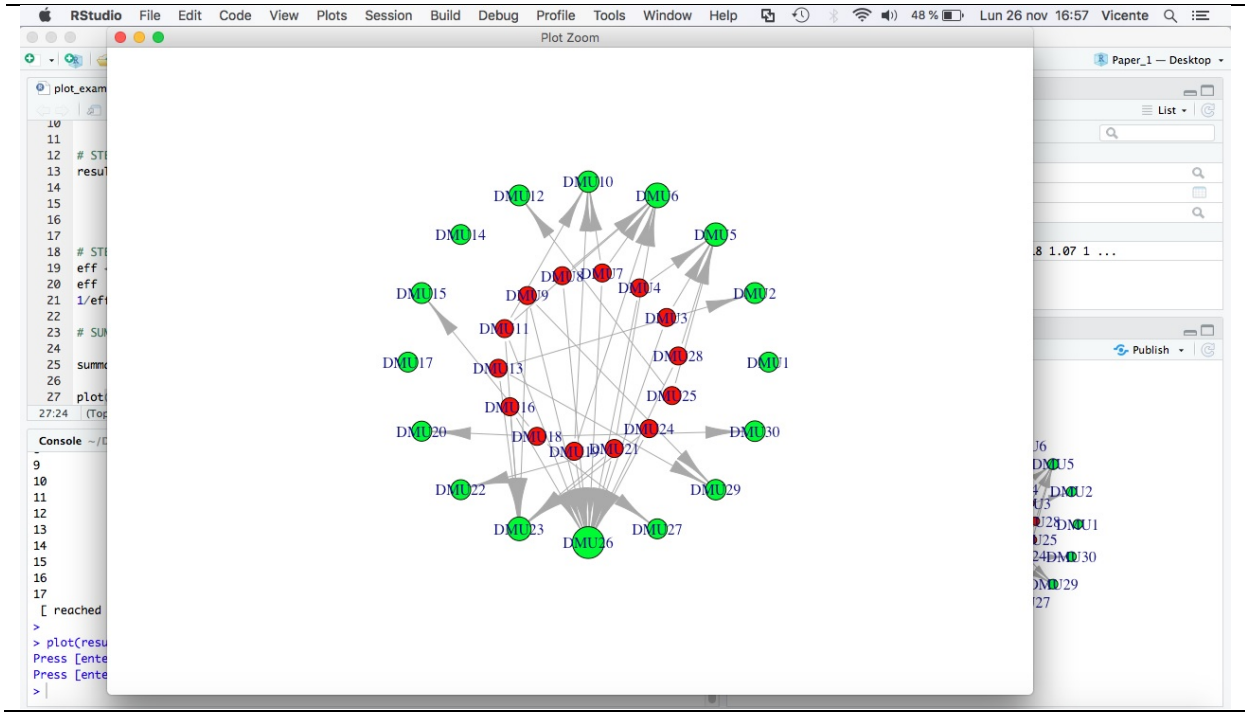

继续按 *Enter* 键，将会显示第二个图表。这个图表描述的是有效 EMU（efficient）被包括进低效 DMU（inefficient）参考集（reference set）的次数。

图 53：示例 12 之图 2



最后，再次按 *Enter* 键，我们可以看到最后一个图表（参见图 54）。

图 54：示例 12 之图 3




在图 54 中，我们可以看到一个网络图，其中绿色圆圈代表有效 DMU，红色圆圈代表低效 DMU。这个图形显示了低效 DMU 是如何与有效 DMU 相关联的，并且试图传递正是有效 DMU 构建了效率边界的观点。

需要注意的是，并非所有的绿色圆圈都大小相同。在这种情况下，圆圈的大小旨在传达有效 DMU 对于低效 DMU 集合至关重要的想法。

---

将该脚本保存为“*plot_example_12*”。

---

**示例 15：绘图：Malmquist index**

首先，创建一个新脚本并将其命名为“*plot_malmquist*”。如果已经关闭了工作会话窗口，我们需要打开项目“*Paper_1*”，加载 **deaR**，然后创建脚本。

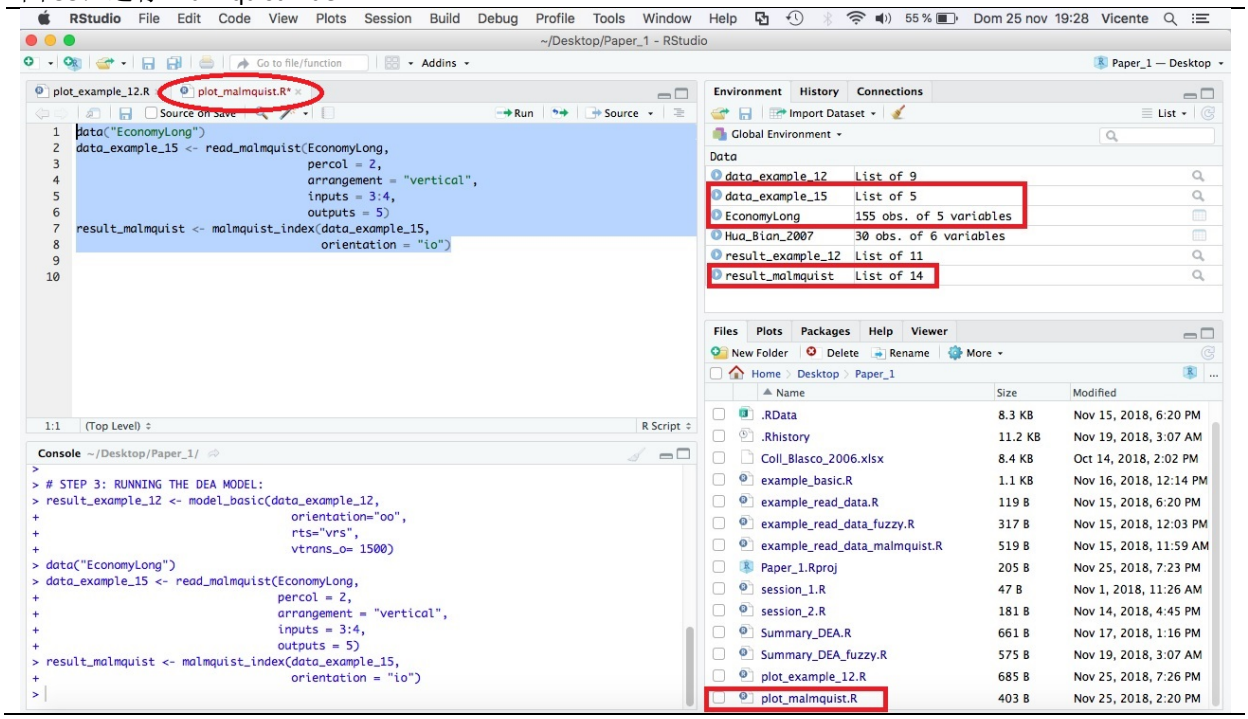
其次，运行 Malmquist index。为此，我们在脚本中写入：

```
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.

执行指令（参见图 52）。

图 55：运行 Malmquist index




Malmquist index 的结果将被保存在对象“*result_malmquist*”当中。接下来，我们在脚本中写入并执行以下指令，以获得绘图结果：

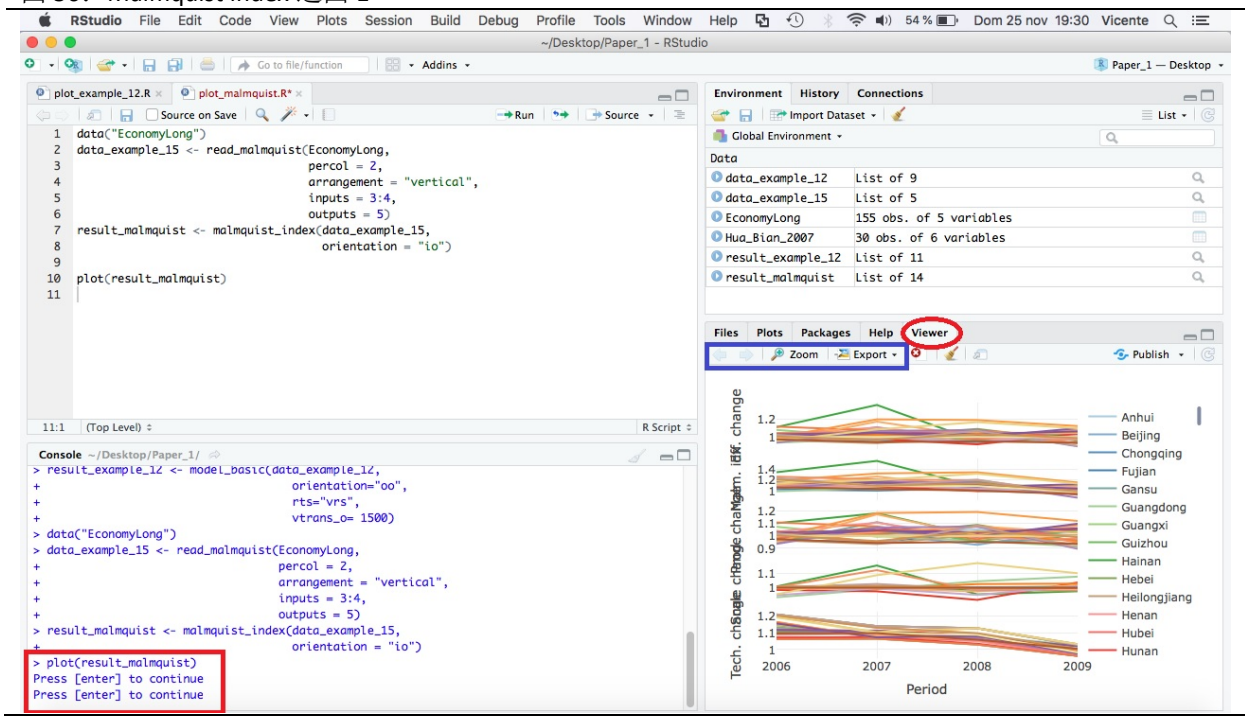
**plot(result_malmquist)**

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

**Press [enter] to continue**

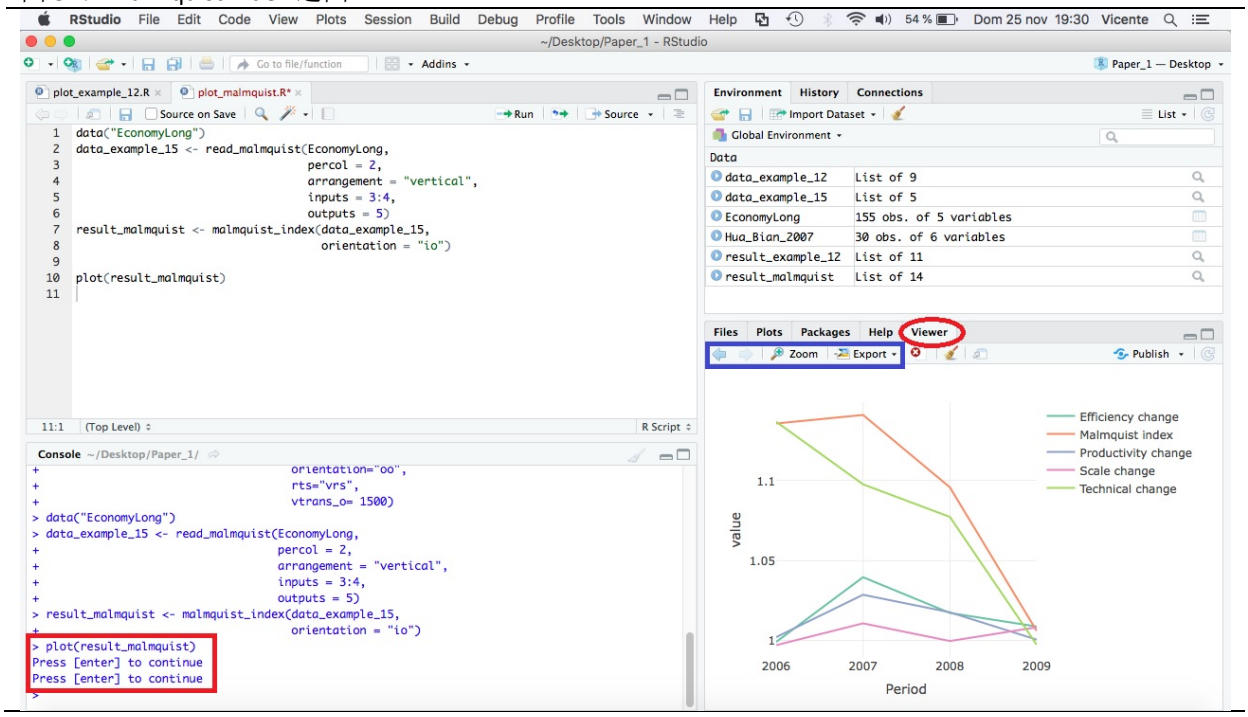
按 *Enter* 键后，图表将显示在 *Viewer* 标签中（右下窗口）（参见图 53）。我们可以单击 *Zoom* 运用缩放功能来进一步查看图表。第一图绘制了每个 DMU 在特定时间段内的 Malmquist index 及其组成部分。如果图表中的 DMU 数量太多以至于影响了显示效果，你可以单独选中你关注的 DMU 进行仔细查看（而其余未选中的 DMU 将会在图表中消失）。

图 56：Malmquist index 之图 1




如果继续按 *Enter* 键，将生成第二图。

图 57：Malmquist index 之图 2




第二图绘制了特定时间段内 Malmquist index 及其组成部分的几何平均数（geometric mean）。我们可以点击 *Zoom* 来放大图表，也可以点击箭头返回（或前进）来进行操作。该图还可以选择特定的 Malmquist index 以显示其索引组件。

保存“*plot_malmquist*”。

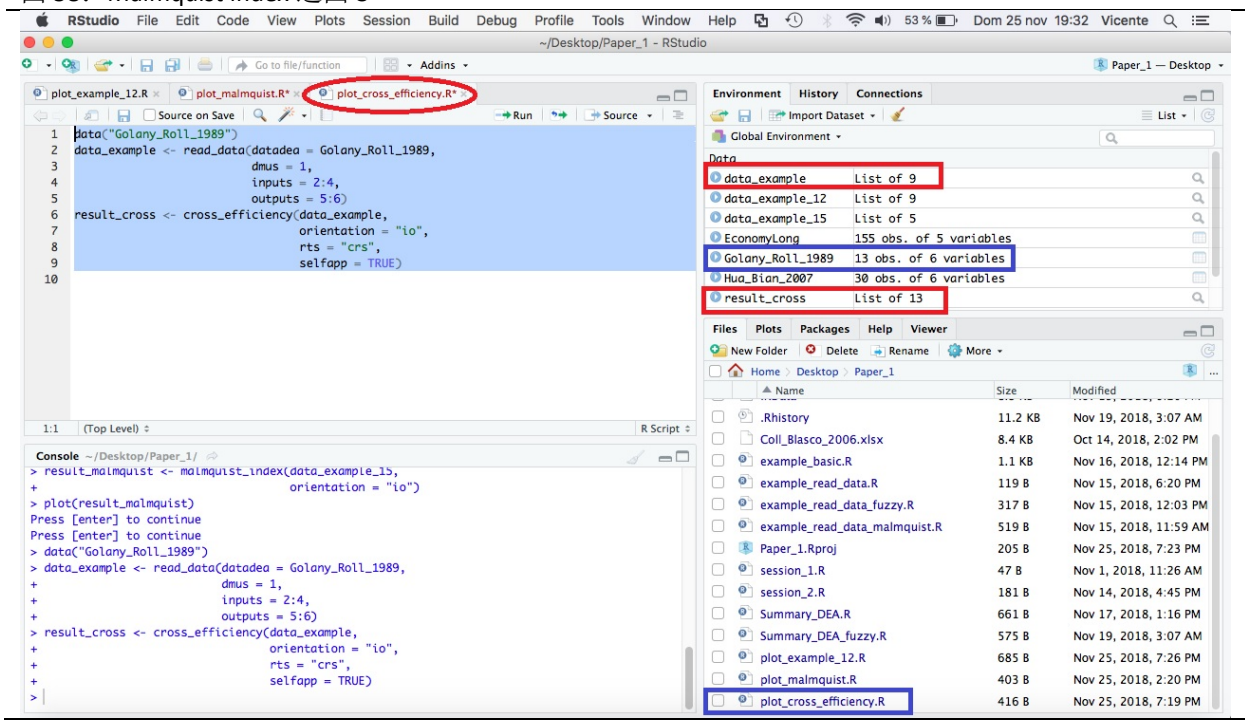
## 示例 16：绘图：Cross efficiency

首先，创建一个新脚本并将其命名为“*plot_cross_efficiency*”。如果已经关闭了工作会话窗口，我们需要打开项目“*Paper_1*”，加载 **deaR**，然后创建脚本

写入并执行以下指令（参见图 55）：

```
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)
```

图 58：Malmquist index 之图 3

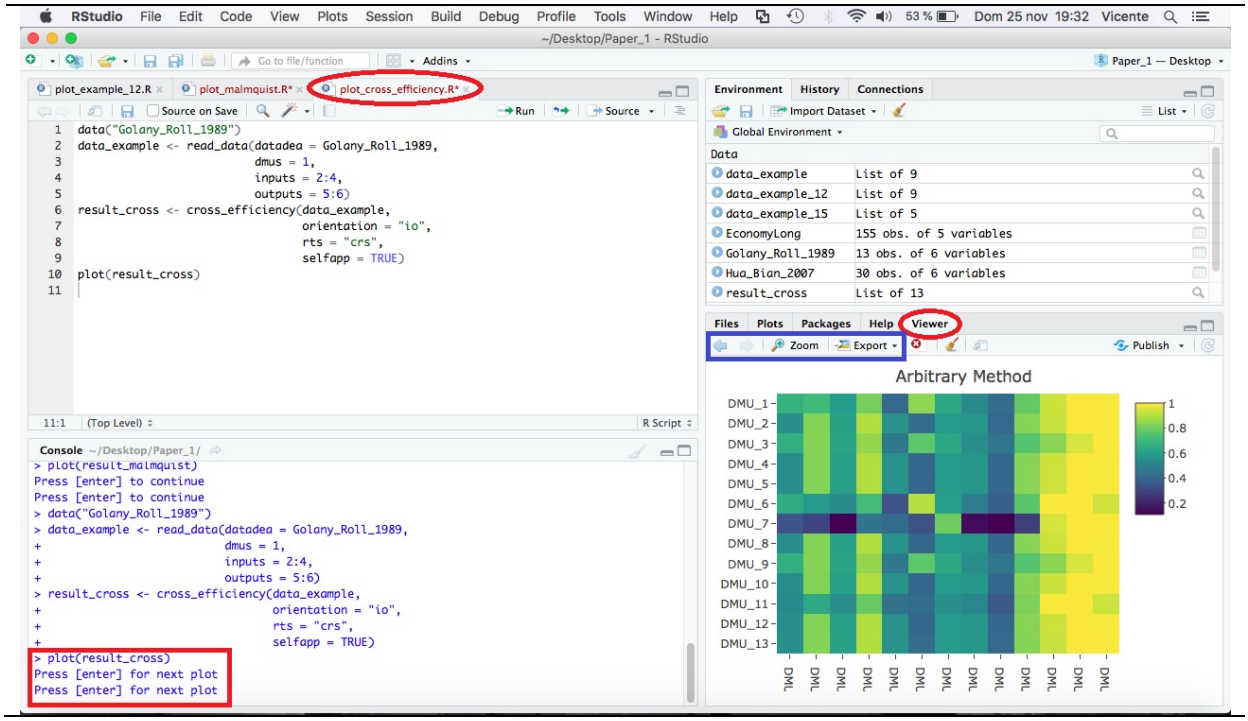



可以看到，在对象“*result_cross*”中保存了交叉效率（cross efficiency）的结果：arbitrary、benevolent 和 aggressive。所有这些结果可以通过使用 plot()函数被绘制成一幅热力图。为此，我们在脚本中写入：

```
plot(result_cross)
```

在 *Console* 面板中按 *Enter* 键以显示不同的图表。获得的结果应如图 56 所示：

图 59：交叉效率图




保存脚本发“*plot_cross_efficiency*”，关闭项目并退出 **RStudio**。