Using R to Analyse Key Surveys

UKDS

10/09/2021

## Introduction

The aim of this guide is to provide an introduction to analysing large UK surveys with the help of the R statistical software package. This document is targeted at two categories of users:

1. Those outside higher education, or who do not have access to one of the commonly used statistical packages such as Stata, SPSS or SAS (as R is free of charge) but who would like to conduct their own analysis beyond what is usually published by data producers such as the Office for National Statistics (for example statistics for specific groups of the population).
2. More advanced users who are already familiar with one of the aforementioned packages but would like to learn how to carry out their analyses in R. The guide, therefore, focuses on providing step-by-step examples of common operations most users carry out in the course of their research: how to open datasets, do basic data manipulation operations, produce simple descriptive statistics or weighted contingency tables. This is meant to provide the first category of users with a range of procedures that will help them produce straightforward and robust analyses tailored to their needs without spending too much time on learning the inner workings of R. The second category of users will find a number of familiar operations from which they will be able to further expand their R skills. It should be noted however that this guide is not an introduction to R. Beginners should use it in conjunction with one of the more comprehensive guides available online. Links and information about R resources are available at the end of this document.

Examples provided in this guide, use the Quarterly Labour Force Survey, January - March, 2016, which can be downloaded from the UK Data Service website. The website also has instructions on how to acquire and download large-scale government datasets.

### What is R

R is a free, user developed, advanced statistical and computing programme. It has a large audience in the programming and statistical community and is increasingly used in the academic world for teaching purposes. R can be downloaded from the Comprehensive R Archive Network (CRAN) website. Installation instructions as well as guides, tutorials and FAQ are available on the same website.

R is often described as an object-oriented statistical programming language rather than simply a statistical analysis package. It originates in the ‘S’ and ‘S Plus’ languages developed during the 1970s and 1980s. Anyone can download and use it without charge, and to some extent contribute to and amend the existing programme itself. It is particularly favoured by users who want to develop their own statistical application or implement the latest advances that are not yet available in commercial packages. The existence of a vast number (more than 3,600 at the time of writing this guide) of user written packages – which bear some resemblance to downloadable ado files in Stata – is one of the great strengths of R. Users who want to contribute should be aware that in order to be part of the R archive, a minimum set of rules need to be followed.

Although R can perform most of the tasks available in generalist statistical packages such as Stata, SPSS, or SAS, it has a broader potential since it can also be used for mapping or data mining. Being a language also means that there are often several ways to carry out analyses in R, each one with its pros and cons. Publication quality output from R can be obtained easily thanks to its integration with the LaTeX document presentation system, and R graphs can also be imported into MS Word documents.

### The Pros and Cons of R

Although R has advantages over other statistical analysis software, it has also a few downsides, both of which are summarised below. Users should be reminded that as an open-source software, R and its packages are developed by volunteers, which makes it a very flexible and dynamic project, but at the same time reliant on developers’ free time and good will.

|  |  |
| --- | --- |
| Pros | Cons |
| R is free, and allows users to perform almost any analysis they want. | The learning curve may be steep for users who do not have a reasonably robust background in statistics and programming. |
|  |  |
| R puts statistical analysis closer to the reach of individual citizens rather than specialists. | Problem solving (for both advanced and beginners) in R may be time-consuming, depending on how common the problem encountered is and may lead to more time spent solving technical rather than substantive issues. |
|  |  |
| Transparency of use and programming of the software and its routines, which improves the peer-reviewing and quality control of the software in many cases | Packages can stop being maintained without notice, and some of them have a short life span. Many people who design them are or will become busy academics, and at some point will not have the time to maintain them anymore. Others will take over in some, but not necessary in all the cases |
|  |  |
| Very flexible |  |
|  |  |
| Availability of a wide range of advanced techniques not provided in mainstream statistical software or only available in specialised packages |  |
|  |  |
| A very large user base provides abundant documentation, tutorials, and web pages |  |

## Using R;

I think we should merge the code fromm the this bit, and just staart showing how to do this in RStudio, and not R - what’d you think? Something like this…

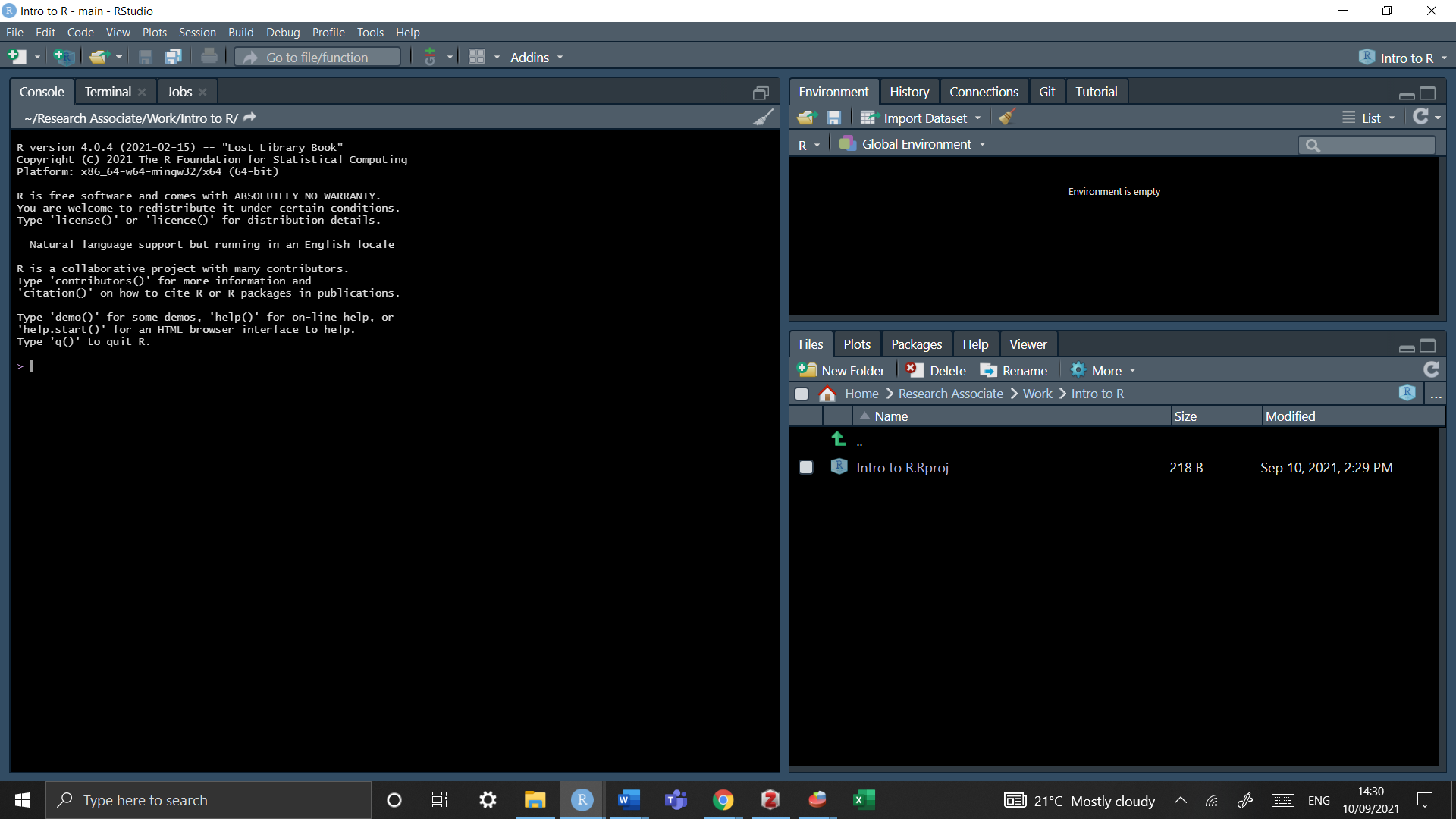
## Using R Studio;

There is a more user-friendly way of using R that is becoming increasingly popular among R users, which is R Studio. This software package is available to download from this website: <https://www.rstudio.com/>.

In R Studio, R syntax works exactly the same as with the traditional R environment, but the interface is more interactive, which makes it easier to use.

### R Projects

Once you open R studio you will be directed to a screen with three main panels;



What we are working is an ’empty interface. RStudio project is just a directory (a folder) with the name of the project, and a few files and folders created by R Studio for internal purposes.

Tip: If you want to change the aesthetics as seen below simply go to *tools* -> *global option* -> *appeareance* and choose your desired colour

To create a new project

Go to *File* -> *New Project* -> *New Directory* -> *New Project*

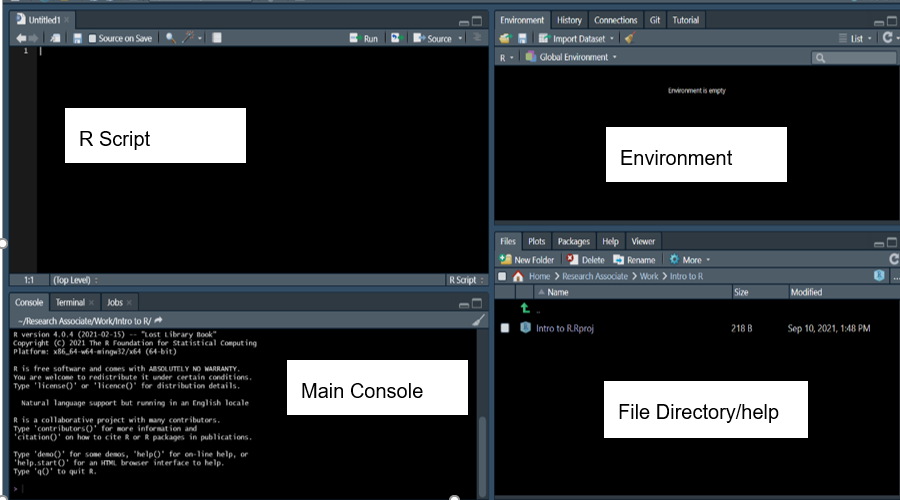
And then select a name and folder in which you want to save this. Now you have a wokring directory - any additionally data should be saved into this foldr (but we’ll cover this later)

#### In order to create a script

• File -> New File -> New script…

• Or just press the ‘Plus’ green button, as shown in the image below.

• You can either select the command or just place the cursor on it, then press ‘Run’ or type ‘Ctrl+R’



Now we are introduced to four panels 1. the script is wher you will code 2. the main console produces the results from the script 3. the environment saves any dataframes or objects that are being used 4. the file directory are where all packages are saved under CRAN, there is also a help menu

Practise talking to your computer; in the scrip type “My name is (your name), run the code by pressing CTRL+R (or at the top of the panel click”Run") and you will see this code in your main console - this is how R communicates!

print("My name is x")

## Installng and Loading Packages

Apart from a basic set of commands and functions, most of the tools offered by R are available in packages that are not provided during the main installation and need to be installed and downloaded separately from within R. For example, in order to install the foreign package which allows users to import Stata or SPSS datasets, one needs to type:

The following code downloads packages into R

install.packages("forigen")

In order to use packages they have to be LOADED via the library function

library(foreign)

Every time you restart up R you will have to load packages again but once a package is installed it will perminately stay in your file directory, unless deleted.

## Basic function

### Objects

An object is a dataframe that can be created and will exist in your global environment. To create an object you use the assignemnt operator which is just an arrow (<-). For example

x <- 5

In this example we’ve assinged the value 5 to an object call x - can you see it in your environment to the right. Run just x and you will the value appear in your console

x  
  
## [1] 5

## Deleting objects in R

To delte an object you just use the rm() function

rm(x)

Now its disseapered from your global environemtn - great!

## Saving objects and more in R

When working with data, it is very likely that the user will edit the original dataset, either by recoding variables or creating new ones, etc. In those cases, saving the progress made in the data used is crucial to avoid repeating every single operation in the next session working with the data. There are several ways of accomplish this, depending on the format in which the data will be stored. The line of command used to save the data frame used called “mydata” is:

save(mydata, file="mydata.Rda")

This command will save the data into a format that can be read by R. The first part of the command is referring to the data frame used in the current R session, while the section file=“mydata.Rda”, is referring to the data that will be saved in the working directory. The name of the saved file can be changed, for instance:

save(mydata, file="mydata\_Jan17.Rda")

To load the saved .Rda data:

load("mydata\_Jan17.Rda")

This command will work only if the working directory where the data is stored is defined in advance (see section 2). Alternatively, the path to the folder where the data is saved can be specified.

load("c:/mydocuments/mydata\_Jan17.Rda")

Another option to save the data is using the “foreign” package, so data can be exported to several formats, such as .txt, .cvs, .dta, which can be used in other software packages. The following example shows how to export data from R to a comma delimited format (.csv) that can be read in excel, Stata and SPSS.

library(foreign)  
write.csv(mydata, “mydata.csv”, row.names=FALSE)

Another example is to export the data as a Stata file, using the foreign package previously loaded:

write.dta(mydata, "mydata.dta")

The newly created files will be stored in the working directory defined earlier on.

Some users will want to save the whole R project in which they are working. This would include functions, variables, data (in R it is possible to load and work with more than one dataset at a time). This option comes very handy, especially when working with several datasets.

Thus, another approach is to save the current session or workspace as an image, using the save.image command, specifying the path where the workspace will be saved. For example, to save ‘my current session’ in R, one needs to write the following command, making sure to include the .RData extension.

save.image("c:/Folder/my\_current\_session.RData")

This will open a window to save the workspace in a particular folder, which can be different from the current working directory. The name needs to be specified in the ‘File name’ section. The workspace will be saved in the chosen folder as an .RData file.

The data can be retrieved using the load function

load("c:/Folder/my\_current\_session.RData")

Or using graphical interface as follows:

*File* -> *load workspace*

## Opening UK dataservice in R

In this guide, we use the Quarterly Labour Force Survey, January - March, 2016 (LFS), which can be downloaded from the UK Data Service website. The website also has instructions on how to acquire and download large-scale government datasets.

### Which versions of the UKDS can I use in R

pay attention to what type of file you have downloaded (whether csv, spss or stata) will affect how you import data into R

bsa <- read.dta("bsa\_2017")   
bsa <- read.spss("bsa\_2017")   
bas <- read.csv(("bsa\_2017"))