

A Brief Intro to R with R Markdown

Cheng Peng

West Chester University

Contents

1	Introduction	1
2	RStudio GUI	1
2.1	Console	2
2.2	Source Editor	2
2.3	Environment Window	2
2.4	System and Graphic files	2
3	R Markdown	2
3.1	Code Chunk	2
3.2	Graphics Generated from R Code Chunks	3
4	Collaborative Platforms	4
4.1	RPubs	4
4.2	GitHub Repository	4
4.3	Correct URL of A GitHub File	5

1 Introduction

This brief note will introduce the basics of Rstudio, R Markdown, and R.

- **RStudio** is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging, and workspace management.
- **R Markdown** is a file format for making dynamic documents with R. An R Markdown document is written in markdown (an easy-to-write plain text format) and contains chunks of embedded R code and the output generated from the R code. This note is written in R Markdown. This is also a tutorial showing how to use R Markdown to write an R Markdown report. – RStudio documentation.
- **R** is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

2 RStudio GUI

The RStudio interface consists of several windows. I insert an image of a regular RStudio GUI.

![RStudio GUI] (<https://datacarpentry.org/r-intro-geospatial/fig/01-rstudio.png>)

2.1 Console

We can type commands directly into the console, or write in a text file, and then send the command to the console. It is convenient to use the console if your task involves one line of code. Otherwise, we should always use an editor to write code and then run the code in the Console.

2.2 Source Editor

Generally, we will want to write programs longer than a few lines. The Source Editor can help you open, edit and execute these programs.

2.3 Environment Window

The Environment window shows the objects (i.e., data frames, arrays, values, and functions) in the environment (workspace). We can see the descriptive information such as types as the dimension of the objects in your environment. We also choose a data source from the environment to view in the source window like a spreadsheet.

2.4 System and Graphic files

The Files tab has a navigable file manager, just like the file system on your operating system. The Plot tab is where the graphics you create will appear. The Packages tab shows you the packages that are installed and those that can be installed (more on this just now). The Help tab allows you to search the R documentation for help and is where the help appears when you ask for it from the Console.

3 R Markdown

An R Markdown document is a text-based file format that allows you to include both descriptive text, code blocks, and code output. It can be converted to other types of files such as PDF, HTML, and WORD that can include code, plots, outputs generated from the code chunks.

3.1 Code Chunk

In R Markdown, we can embed R code in the code chunk defined by the symbol ````\n{}``` and closed by ````\n``. The symbol ```, also called **backquote** or **backtick**, can be found on the top left corner of the standard keyboard as shown in the following.



Figure 1: The location of backquote on the standard keyboard

There are two code chunks: executable and non-executable chunks. The following code chunk is non-executable since is no argument specified in the `{}`.

This is a code chunk

To write a code chunk that will be executed, we can simply put the letter `r` inside the curly bracket. If the code chunk is executable, you will the green arrow on the top-right corner of the chunk.

```
```{  
 }
This is a code chunk
```

Figure 2: Non-executable code chunk.

```
```{r}  
x = 5  
x  
```
```

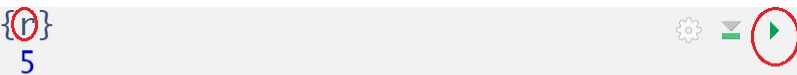


Figure 3: Executable code chunk.

We can define R objects with and without any outputs. In the above R code chunk, we define an R object under the name `x` and assign value 5 to `x` (the first line of the code). We also request an output that prints the value of `x`. The above executable code chunk gives output `[1] 5` in the Markdown document. The same output in the knit output files is in a box with a transparent background in the form `## [1] 5`.

```
[1] 5
```

We can also use an argument in the code chunk to control the output. For example, the following code chunk will be evaluated when knitting to other formats of files. But we can still click the green arrow inside the code chunk to evaluate the code.

### 3.2 Graphics Generated from R Code Chunks

In the previous sub-sections, we include images from external image files. In fact, we can use the R function to generate graphics (other than interacting with plots, etc.) in the markdown file & knit. For instance, we can generate the following image from R and include it in the Markdown document and the knitted output files.



In this class, all data sets you are going to use in your assignments and project are required to be uploaded to your specific repository so you can read your data sets directly from GitHub repository.

### 4.3 Correct URL of A GitHub File

The following figure shows how to find the correct URL of a file stored in a GitHub repository.

#### GitHub Repository URL

<https://github.com/{username}/{repo}/>

**Pages URL+filename → URL of a file in the GitHub repository**

<https://{username}.github.io/{repo}/filename.extension>



**Pages URL**