formatting

Andy Clifton

2019-10-17

Introduction

something something reproducible research mumble, grumble

Linking analysis and publication workflows

Anecdotally, the separation of the publication from the analysis process has been a barrier to reproducible research, as it is impossible to ensure the link between source data and final publication.

This document demonstrates the concept of Literate Programming. Literate programming means that the program documentation is complete and contained within the program itself. It is important to note that the documentation is effectively a publication, and thus it is possible to combine data analysis with the creation of a publication in the same file. The use of literate programming therefore mitigates this barrier to reproducible research.

Furthermore, this project has been structured so that the data required for this publication are in a subdirectory of the project. This means that all of the files required to reproduce the analysis results can be included in a repository.

How Literate Programming was used to write this document

In this example, an output PDF document and results are generated from a file called *main.rmd*. *main.rmd* is an R markdown file. R markdown is a flavor of markdown that can be processed by the R programming language to run code (i.e, do analysis) and create documentation from the same document. This is done using a package called *knitr*. Instructions for how to run *knitr* are included in the *howto.md* file in this repository.

The markdown document contains a mixture of documentation and code `chunks`. The code chunks can be configured so that their outputs are echoed to the document (or not), which in turn allows the output PDF to show only those parts of the data processing that are relevant.

I suggest reading the PDF together with the .rmd file and possibly the *knitr* instructions [1]. This will greatly help in understanding what is done in the processing and what makes it to the publication.

An example of Literate Programming in action

Like most scripts, *main.rmd* includes a few variables that the user must set to run the analysis.

- The *project.root* variable defines the location of the files required for this analysis.
- The *made.by* variable forms part of a label that will be added to the plots.

An advantage of *knitr* is that we can simply execute the code and show the code and results inline:

```r
# Where can files be found?
project.root <- file.path('/Users/andyc/Documents/public/GitHub/LiterateDemo')
project.root

# Who ran this script
made.by = "A. Clifton"
made.by
```

[1] See https://yihui.name/knitr/

```
## [1] "/Users/andyc/Documents/public/GitHub/LiterateDemo"
## [1] "A. Clifton"
```

We can also show the value of those variables in the documentation using backticks around the variable names in the markdown.

- *project.root* is /Users/andyc/Documents/public/GitHub/LiterateDemo
- *made.by* is A. Clifton.

The directory structure

working.dir is the root directory of the project. There are several important subdirectories:

- /**code** contains functions required for the analysis
- /**data** contains the data files to be analyzed.

We'll also create a new directory:

- /**analysis** contains the results of the analysis.

A note on Packages

Packages are required to supplement base R functions. For example, this script requires the *ggplot2*, *grid*, *knitr*, *RColorBrewer*, *rgdal*, and *stringr* packages to run. These are called from the script using the *require()* function. This assumes that the packaages are available on your system.[2]

N.B. It is likely that the use of packages makes data processing non-repeatable, because the function and output of the packages may change over time.

Implementing a coupled analysis and publication workflow

An analysis and publication workflow usually follows a similar path:

Loading our own routines

Every data processing workflow requires its own scripts or functions to run. In this example, they are included in the *codes* directory and sourced during the preparation of this document.

```r
# source these functions
code.files = dir(code.dir, pattern = "\\.R$")
for (file in code.files){
  source(file = file.path(code.dir,file))
}
```

Load the data

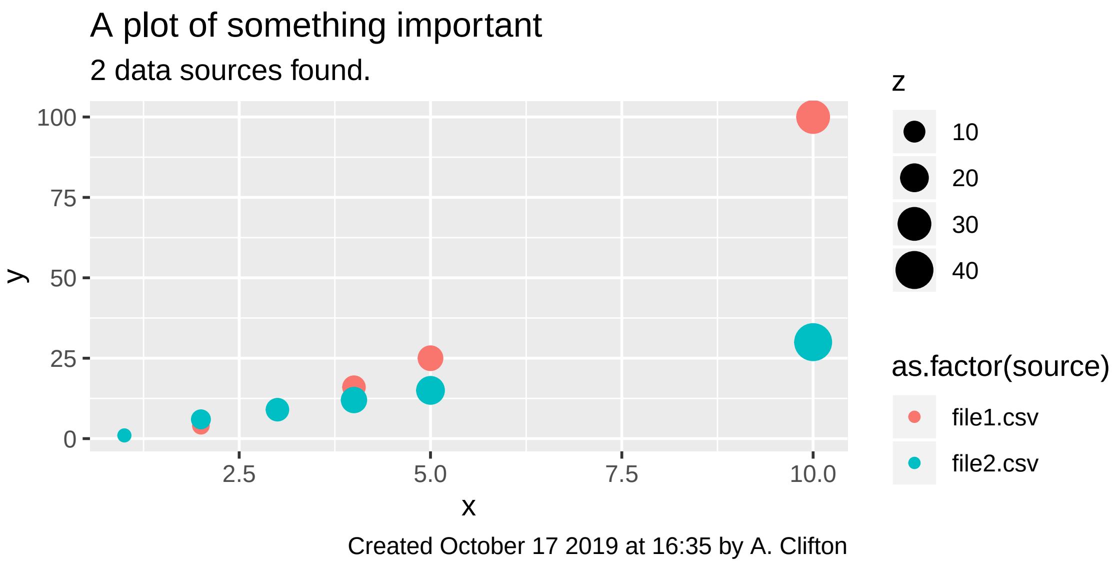
We now analyse the data from the simple data set. In this case, code has been written to load all of the files in the *data.dir* directory (/Users/andyc/Documents/public/GitHub/LiterateDemo/data). I'm also going to map the three columns in the data files to the variables x, y, and z.[3]

[2]For details of how to install packages, see the RStudio help.

[3]See https://www.calvin.edu/~rpruim/courses/s341/S17/from-class/MathinRmd.html for more information about including maths in R markdown

Plot input data

The next step is to plot the input data. In this case we plot all of the input data together in one plot, but there are many different possibilities. Figures can also be given a consistent look and feel through ggplot's themes.



For convenience, we'll also save a copy of the figure as a .png file to the *analysis* directory.

Operate on the data

Plot the results

Connect processing with publication

So far we have demonstrated that we can import and manipulate data and plot results. Another important part of a publication is the ability to generate statistics from data and include that in our text.

To demonstrate that, I can calculate that the maximum value of y in the input data sets was 100. This can be confirmed by checking the input data files and

Save the processed data

We now write our data to file.