A demonstration of combined scientific data processing and publication using Literate Programming in R

Andy Clifton

2019-10-17

Contents

1 Introduction

Something something reproducible research, mumble, grumble, get off my lawn, grumble.

1.1 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.

1.2 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 [R Core Team, 2017] 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. A far more detailed guide to writing using R markdown can be found in Xie et al. [2019].

The markdown document contains a mixture of documentation – written in markdown or LaTeX - and so-called "code chunks", which here are written in R. The output is a PDF.

- This document is written in Pandoc markdown. It is possible to use other versions, including GitHub flavored markdown.
- 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. You can thus completely hide the data operations in your output PDF and just concentrate on displaying the results.
- It is possible to use other programming languages (see "But I hate R")
- There are a lot of different possible output formats, including PDF, HTML, Notebooks, and others. See [Xie et al., 2019] or https://bookdown.org/yihui/rmarkdown/documents.html for more information.

I suggest reading this PDF together with the R markdown file (*main.rmd*) 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.

1.3 But I hate R

You don't mean that. But just in case you can't handle learning yet another new language, this next statement might interest you.

> "A less well-known fact about R Markdown is that many other languages are also supported, such as Python, Julia, C++, and SQL. The support comes from the knitr package, which has provided a large number of language engines.""
>
> — Xie et al. [2019]

The currently available language engines are:

```r
names(knitr::knit_engines$get())
```

```
##  [1] "awk"         "bash"        "coffee"      "gawk"        "groovy"
##  [6] "haskell"     "lein"        "mysql"       "node"        "octave"
## [11] "perl"        "psql"        "Rscript"     "ruby"        "sas"
## [16] "scala"       "sed"         "sh"          "stata"       "zsh"
## [21] "highlight"   "Rcpp"        "tikz"        "dot"         "c"
## [26] "fortran"     "fortran95"   "asy"         "cat"         "asis"
## [31] "stan"        "block"       "block2"      "js"          "css"
## [36] "sql"         "go"          "python"      "julia"       "sass"
## [41] "scss"        "theorem"     "lemma"       "corollary"   "proposition"
## [46] "conjecture"  "definition"  "example"     "exercise"    "proof"
## [51] "remark"      "solution"
```

So, you have no excuse. You can simply write your code in any of the 52 languages, and off you go.

2 Implementing a coupled analysis and publication workflow

An analysis and publication workflow usually follows a similar path:

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

1. Set up the computing environment
2. Load our own data processing routines
3. Import data
4. Plot it
5. Do some operations
6. Plot some more
7. Write.
8. Format for a journal
9. Iterate around items 1-9 for a while
10. Submit

Fortunately, all of this can be captured in a markdown document.

2.1 Setting up the computing environment

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:

```
# 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] "/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.

We want to change our working directory (*working.dir*) to the root directory of the project. We've already set up several important subdirectories:

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

Let's tell the code where these are.

We'll also create a new directory for the results of the analysis. In this case it can be found at **/Users/andyc/Documents/public/GitHub/LiterateDemo/analysis/all**.

Note: Packages are required to supplement base functions in R and many other languages. For example, this script requires the *bookdown*, *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] The use of packages represents a challenge to reproducable and repeatable research as it is possible that the function and output of the packages may change over time.

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

2.2 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. I have included output below to show these codes being called.

```
# source these functions
code.files = dir(code.dir, pattern = "\\.R$")
for (file in code.files){
  source(file = file.path(code.dir,file))
  print(paste0("Sourcing ", file, "."))
}
```

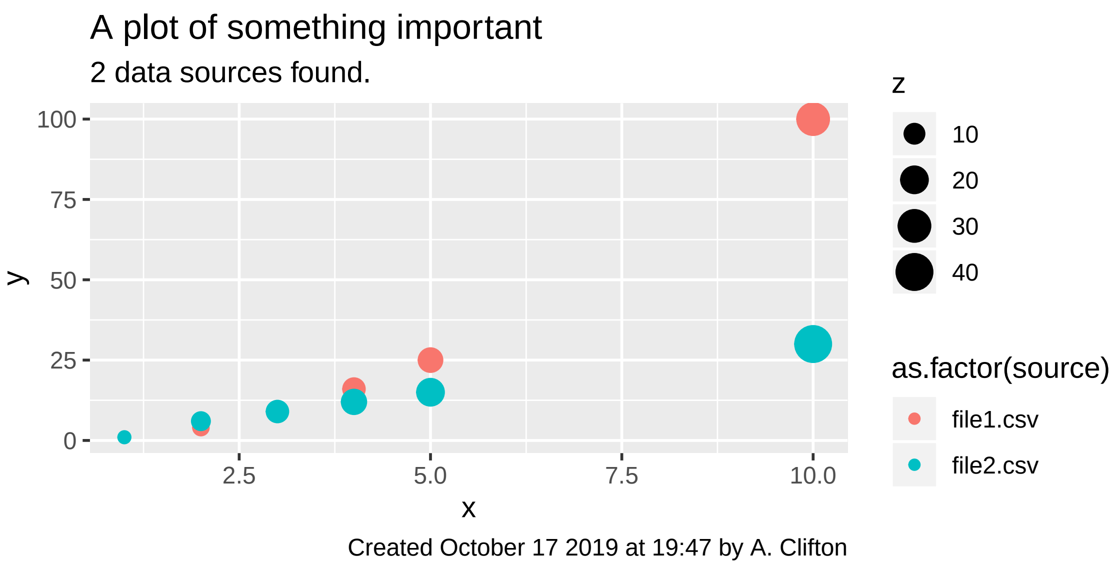
```
## [1] "Sourcing cleanPlot.R."
## [1] "Sourcing plotInfoLabel.R."
## [1] "Sourcing plotSomething.R."
## [1] "Sourcing theme_Literate.R."
```

2.3 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]

2.4 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.

2.5 Operate on the data

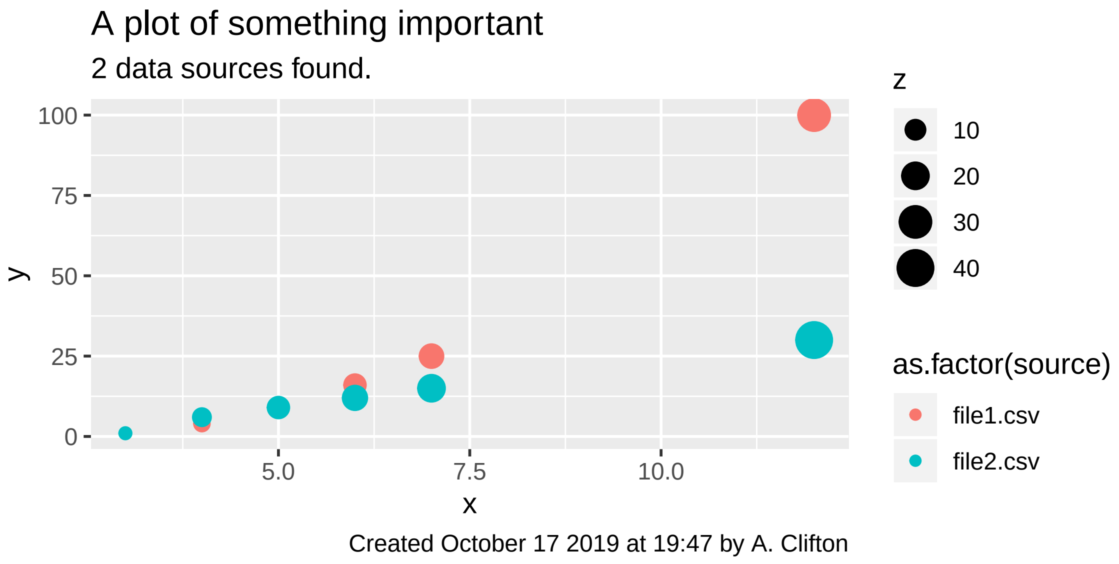
At this point we can do any number of operations on the data. For sake of demonstration, let's add 2 to all x values.

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

```
df.all <- df.in
df.all$x <- df.in$x + 2.0
```

2.6 Plot the results

Let's run that *plotSomething* routine again.



And, as we can see, the data have shifted along x by a small amount.

2.7 Connect processing with publication

So far we have demonstrate that we can import and manipulate data and plot results. Another important part of a publication is the ability to generate statistics or summary information 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. I could also include more complex logic in these statements, for example to say if one statistic is bigger or larger than another.

We sometimes need to include formatted tables in documents. This can be done using the *kable* function (Table 1).

```
knitr::kable(df.all,
             booktabs=TRUE,
             caption = "The df.all data frame.")
```

2.8 Save the processed data

We now write our processed data to file.

```
# save the data
save(list = c("project.root",
              "made.by",
              "df.all"),
       file = file.path(output.dir,"Data.RData"),
       envir = .GlobalEnv)
```

In R it is also possible to save the whole workspace. We can do that here as well:

Table 1: The df.all data frame.

x	y	z	source
3	1	3	file1.csv
4	4	6	file1.csv
5	9	9	file1.csv
6	16	12	file1.csv
7	25	15	file1.csv
12	100	30	file1.csv
3	1	4	file2.csv
4	6	8	file2.csv
5	9	12	file2.csv
6	12	16	file2.csv
7	15	20	file2.csv
12	30	40	file2.csv

```
# save the workspace
save.image(file=file.path(output.dir,"workspace.RData"))
```

2.9 Saving packages

2.10 Applying Journal formating

Scientific Journals often have their own formatting requirements. These requirements can still be met using markdown. The mechanics of such a process are beyond the scope of this paper and should probably be done as the last step in the publishing process. The reader is suggested to look at the *rticles* package and to use the detailed instructions in section 13 of the R Markdown Guide [Xie et al., 2019].

3 Conclusions

It is possible to write a single document that captures all of the process of preparing and analysing data and creating a publication to describe that data.

Referencing this document

This document has been assigned the Digital Object Identifier 10.5281/zenodo.3497450.

DOI 10.5281/zenodo.3497450

Acknowledgements

Many thanks to Nikola Vasiljevic at DTU for prompting me to get this done.

Bibliography

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL https://www.R-project.org/.

Yihui Xie, J. J. Allaire, and Garrett Grolemund. *R Markdown: The Definitive Guide*. 2019. URL https://bookdown.org/yihui/rmarkdown/.