Stat 405/705
Class 14
There are no Class 13 notes

Statistical computing with R

Richard P. Waterman

Wharton

Table of contents I

Today's module

Topics to be covered in this module:

- Last time
- Living documents:
  - Reproducible research
  - Recurring/dynamic reporting
  - Markdown and Markup
  - Sweave/Knitr (.Rhtml, .Rnw)
  - R-markdown (.Rmd)
- Next time

Last time

- Low-level graphics
- Basic graphics
- High-level presentation graphics

Replicable research

- A crisis of science.
- It's bad that people can't replicate an experiment: http://www.apa.org/monitor/2015/10/share-reproducibility.aspx
- The Science article said that out of 100 experimental correlation psychology studies published in three top-tier journals, only 39 were successfully replicated.
- It is not just a problem in Psychology. It is a problem in all of the empirical sciences.
- Reasons include:
  1. Publication bias in favor of "big/exciting" results.
  2. Small study sizes can produce larger effects by chance variation.
  3. Multiplicity not accounted for.

Reproducible research

- It's perhaps even worse if from the same experiment, with the same set of data, you can't even **reproduce** the original numerical results.
- Why might this happen? Reasons include:
  1. Unstated treatment of outliers.
  2. Conditional analyses (looking at subsets of the data).
  3. Revisions to datasets over time (the government expects to revise data like GDP).
  4. Code that worked once but was later changed (package revisions).
  5. Randomized components of the analysis (e.g. bootstrap) for which the random number seed was not set explicitly.

Replicable v. reproducible research

From Roger Peng (BioStats at Hopkins):

> *As I made clear in the commentary, I define* **replication** *as independent people going out and collecting new data and* **reproducibility** *as independent people analysing the same data.*

Reproducible research

- This is a broad criticism of *Science*.
- But a lack of reproducibility is as likely to be as big an issue in any business related project/study (perhaps even more so, given the proprietary nature of most work).
- So approaches to try and "fix" Science are equally valid in the business environment.

Reproducible research

- One approach to enhance reproducibility is to tightly bind the text, analytics and data, essentially in the same document: https://rpubs.com/marschmi/105639
- If all the elements of the analysis are in one place, then it is more likely that the results can be reproduced.
- The most fullproof approach is to not only save code and data but also the complete computing environment. See "Docker" containers: https://meetingorganizer.copernicus.org/EGU2018/session/28650

The main idea is binding text/narrative with analysis

Mix together the text and the programming code in a *literate* source file:

```
Text chunk 1

Code chunk 1

Text chunk 2

Code chunk 2

Text chunk 3
```

Literate programming

- From a *literate* source file we extract two representations:
  1. The *tangled code*: extract just code chunks.
  2. The *woven result*: keep text, but replace code with its output.

Tangled

Just the [1]code:

```
Code chunk 1
```

```
Code chunk 2
```

[1]This is how I make the .R file that goes with each class.

Woven

Mix [2]together the text and the output from the code:

```
Text chunk 1
```

```
Results chunk 1
```

```
Text chunk 2
```

```
Results chunk 2
```

```
Text chunk 3
```

[2]This is how I make the slide .PDF for each class.

No web

- The source of the weaving and tangling ideas is from "Noweb": https://en.wikipedia.org/wiki/Noweb
- This implements the *literate* programming ideas of Donald Knuth.
- The ideas are implemented in R with the packages `sweave` or `knitr`.

Sweave and Knitr

- The weaving and tangling paradigm was originally implemented in an R a package called `Sweave`.
- The `Knitr` package is a more recent replacement.
- You can use different languages for the text chunk elements:
  1. HTML (.Rhtml)
  2. LaTeX(.Rnw)
  3. R markdown (.Rmd)
- The identification of the code blocks depends on the language of the written chunk component.

HTML example

The file `example_01.Rhtml` contains a mix of HTML and R code. Note the mixed file extension `.Rhtml`, a mix of `.R` and `.html`.
R code in the document is delineated with the identifiers:

```
<!--begin.rcode
```

and

```
    end.rcode-->
```

A minimal HTML example

```
<html>
<body>
<h1> Here's how it works </h1>
<!--begin.rcode code-chunk-1
set.seed(19390909) # For reproducibility
a <- rnorm(100) # Some data for analysis
result <- mean(a)
   end.rcode-->
The mean of the 100 numbers was <!--rinline result -->
</body>
</html>
```

Weaving

To *weave* the .Rhtml file into the .html file read in the `knitr` library and the command is `knit`:

```
my.location <- "C:\\Users\\richardw\\Dropbox (Penn)\\Teaching\\705s2019\\No
my.infile <- "example_01.Rhtml"  # INFILE
my.outfile <- "example_01.html"  # OUTFILE

library(knitr) # Read in the knitr package
knit(input = paste(my.location,my.infile,sep=""), # Input file
     output = paste(my.location,my.outfile,sep="") # output file
     )
```

```
## [1] "C:\\Users\\richardw\\Dropbox (Penn)\\Teaching\\705s2019\\Notes\\exa
```

Tangling

To *tangle* the .Rhtml file into the .R file use the `purl` command:

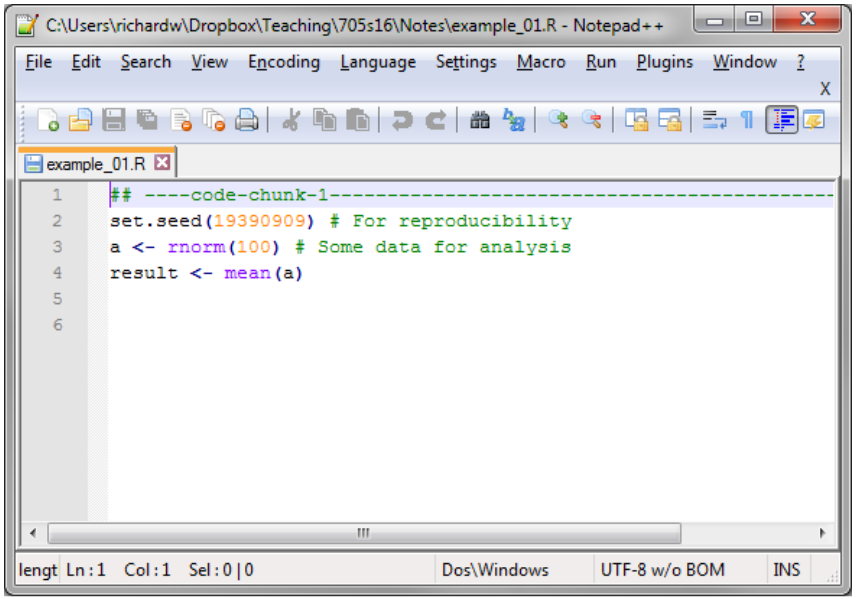
```
my.location <- "C:\\Users\\richardw\\Dropbox (Penn)\\Teaching\\705s2019\\No
my.infile <- "example_01.Rhtml"  # INFILE
my.outfile <- "example_01.R"     # OUTFILE

purl(input = paste(my.location,my.infile,sep=""), # infile
     output = paste(my.location,my.outfile,sep=""), #outfile
     documentation = 1) # Control documentation level
```

```
## [1] "C:\\Users\\richardw\\Dropbox (Penn)\\Teaching\\705s2019\\Notes\\exa
```

The results

- We now have two new files:
  - `example_01.html` with the html output
  - `example_01.R` with just the R code



Controlling the output

- The code chunk can be given options to control the output.
- The format is tag = value.
- See http://yihui.name/knitr/options/ for all of the available options
- Options can be set globally, so that they apply to all chunks.

For example:

```
echo=TRUE will echo the R source code
echo=-1 would echo all the source code except the first line
eval=FALSE will stop evaluation of the code chunk
fig.path="my/path" is a path to a directory to save figures to
fig.align='center' will align a figure
results='hide' would hide results
warnings=TRUE will preserve warnings
error=FALSE would stop execution on an error
```

Controlling the output

- There are code decoration options, to tidy up the code etc.
- There are caching options. Caching allows you to not re-execute code that may have taken a long time to run, but the previously cached output is still available.
- There are many figure related options, for example to control width and height.

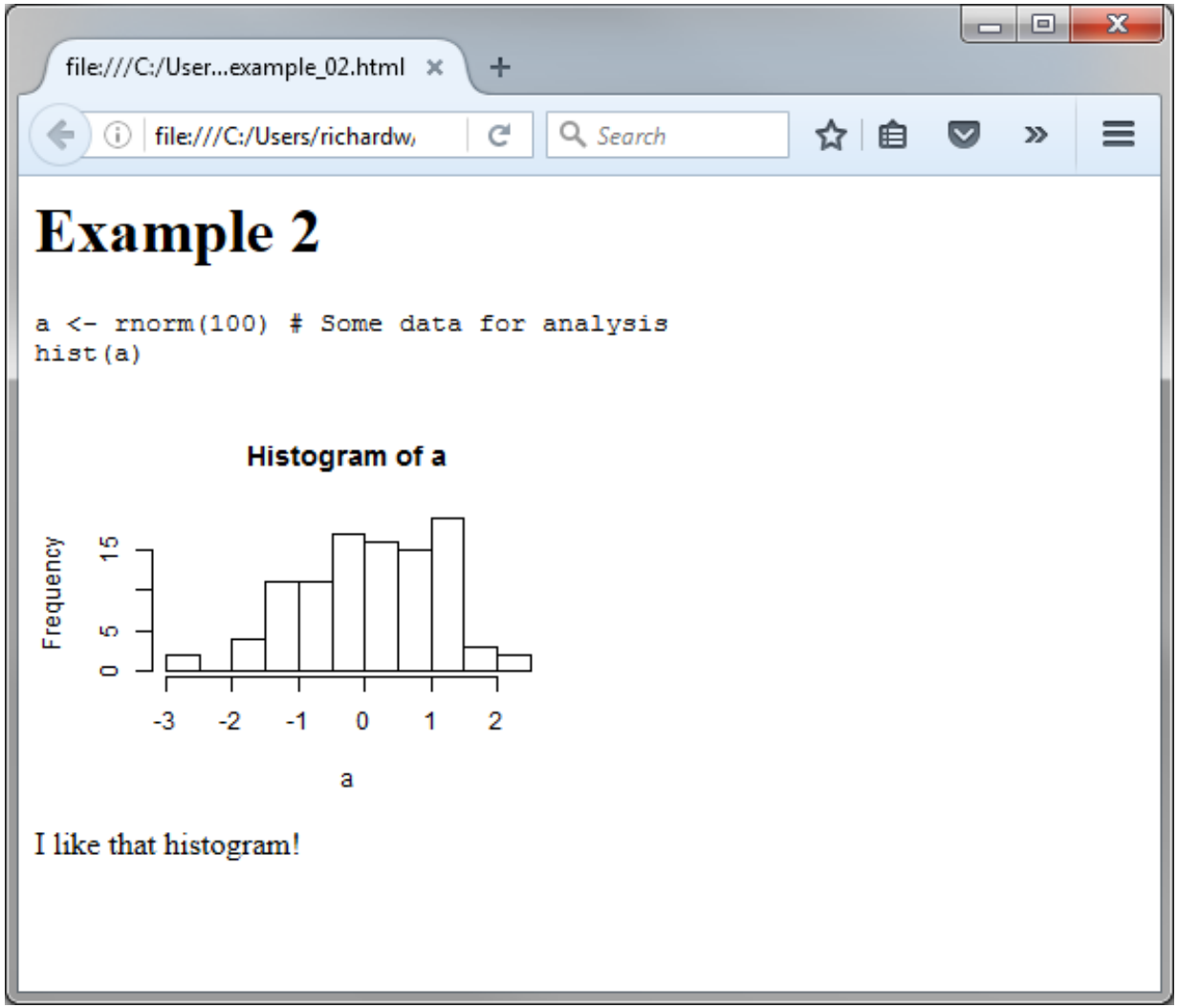
A second example with options

Set the R working directory to the directory that has the .Rhtml file so that the figure can be found by the html file. (Use the RStudio, Session menu.)

```
<html>
<body>
<h1> Example 2</h1>
<!--begin.rcode code-chunk-2, fig.width=4,fig.height=3,echo=-1
set.seed(19390909) # For replication
a <- rnorm(100) # Some data for analysis
hist(a)
   end.rcode-->
I like that histogram!
</body>
</html>
```

The results

- After running `knit` on this file we now have a new file: `example_02.html` with the html output which includes a graphic:



An example report generated with `knitr`

An example report using R, LaTeX (instead of HTML) and `knitr`.

Markdown languages

- Markdown is a plain text formatting language.
- The intention is to write one document and have it be convertible to many output formats, such as HTML, PDF, LaTeX, ePUB3 etc.
- Think of it as a lowest common denominator markup language.
- If you only had an old fashioned ASCII terminal, how would you indicate text attributes?
- Markdown formalizes this into a language.

An example markdown document (see Wikipedia)

```
Heading
=======

Sub-heading
-----------

### Another deeper heading

Paragraphs are separated
by a blank line.

Two spaces at the end of a line leave a
line break.

Text attributes _italic_, *italic*, __bold__, **bold**, `monos
```

R's Markdown language

- RStudio has it's own dialect of markdown, and you can interweave Markdown and R code.
- See: http://rmarkdown.rstudio.com
- This is the same weaving concept as we saw before, but just replacing the HTML code with R markdown code.
- The benefit is that we can create output for different formats.
- In RStudio start with File, New File, R Markdown.
- Choose Document or Presentation.
- A template document is created.
- You have control over which chunks are processed.
- Knit the HTML.
- Open in the Browser.

R's Markdown language

We will review an example R markdown document.
There are two ways of including R code in the markdown document:

- You can include an entire chunk/block of code using the three back-ticks notation:

  ```
  ```{r codechunk_01}
  summary(cars)
  ```
  ```

- or you can insert R output *in-line* using the singe back-tick construct:

  `The number of cars in the dataset is `r nrow(cars)`.`

Synopsis

- This set of reporting tools allows you to create dynamic documents in a business setting.
- They allow raw R code to be inserted into the initial document and then the results from the code to be woven into the output document.
- As a scientist, these tools move toward more reproducible research.

Module summary

Topics covered today include:

- Reproducible research.
- Recurring/dynamic reporting.
- Mark-down and Mark-up.
- Sweave/Knitr (.Rhtml, .Rnw).
- R-markdown (.Rmd).

Next time

- There isn't!

Today's function list

Do you know what each of these functions does?

```
knit
purl
```