Reproducible science: Module5

Literate statistical programming knitr

Gbadamassi G.O. Dossa

Xishuangbanna Tropical Botanical Garden, XTBG-CAS

2021/10/1 (updated: 2022-06-28)

Acknowledgements

The content of this module are based on materials from:

Roger D. Peng's materials

Problems

- Authors must undertake considerable effort to put data/results on the web
- Readers must download data/results individually and piece together which data go with which code sections, etc.
- Authors/readers must manually interact with websites
- There is no single document to integrate data analysis with textual representations; i.e. data, code, and text are not linked

Literate statistical Programming

- Original idea comes from Don Knuth
- An article is a stream of text and code
- Analysis code is divided into text and code "chunks"
- Presentation code formats results (tables, figures, etc.)
- Article text explains what is going on
- Literate programs are weaved to produce human-readable documents and tangled to produce machine-readable documents

Literate Statistical Programming

- Literate programming is a general concept. We need
  - A documentation language
  - A programming language
- The original Sweave system developed by Friedrich Leisch used LaTeX and R
- knitr supports a variety of documentation languages

When to decide to work reproducibly?

- Decide to do it (ideally from the start)
- Keep track of things, perhaps with a version control system to track snapshots/changes [later in the workshop]
- Use software whose operation can be coded [R]
- Don't save output [R]
- Save data in non-proprietary formats

Literate programming: Pros

- Text and code all in one place, logical order
- Data, results automatically updated to reflect external changes
- Code is live--automatic "regression test" when building a document

Literate programming: Cons

- Text and code all in one place; can make documents difficult to read, especially if there is a lot of code
- Can substantially slow down processing of documents (although there are tools to help)

Knitr: Definition & usages

- An R package written by Yihui Xie (while he was a grad student at Iowa State)
    - Available on CRAN
- Supports RMarkdown, LaTeX, and HTML as documentation languages
- Can export to PDF, HTML, Word
- Built right into RStudio for your convenience

Knitr: Requirements

- A recent version of R
- A text editor (the one that comes with RStudio is okay)
- Some support packages also available on CRAN
- Some knowledge of Markdown, LaTeX, or HTML
- We will use Markdown here

What is Markdown?

- A simplified version of "markup" languages
- No special editor required
- Simple, intuitive formatting elements
- Complete information available at http://goo.gl/MUt9i5

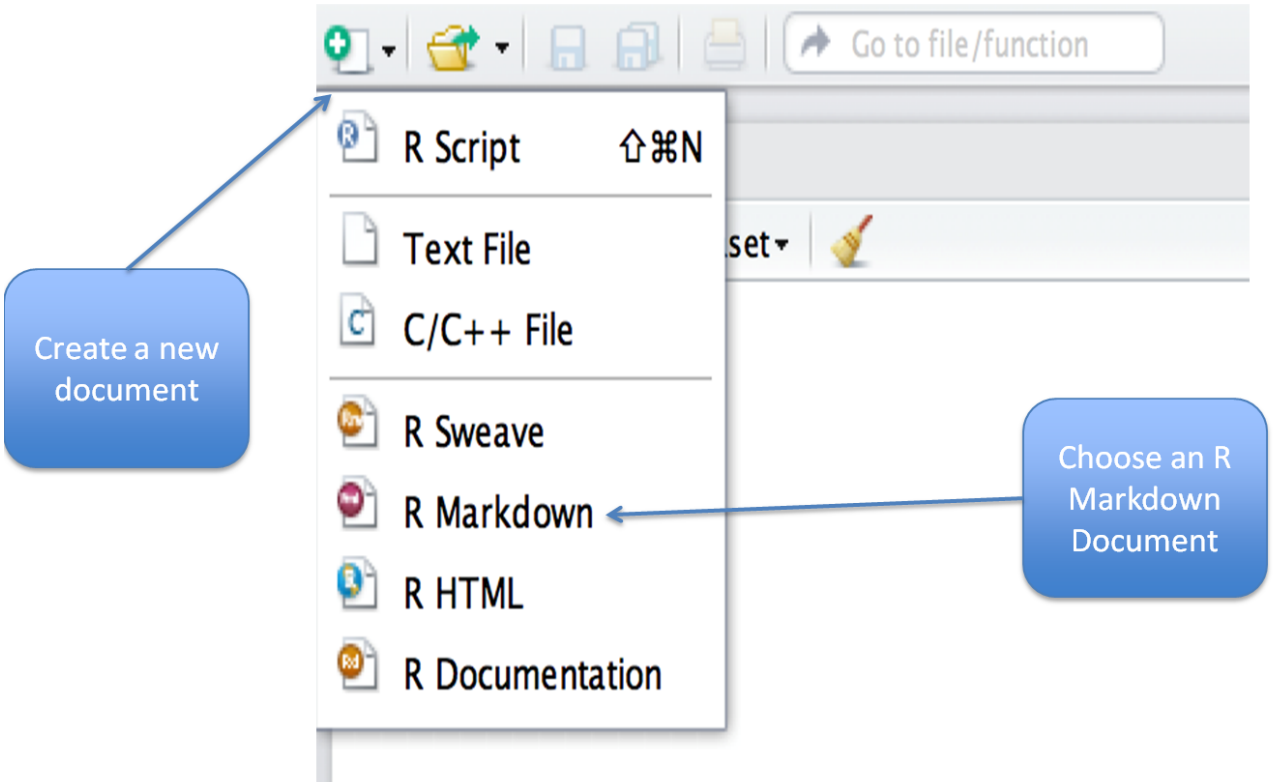
What is knitr good For?

- Writing manuals/manuscript
- Short/medium-length technical documents
- Tutorials
- Reports (esp. if generated periodically)
- Data preprocessing documents/summaries

What is knitr not good for?

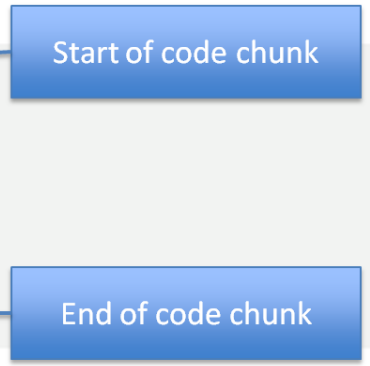
- Very long research articles
- Complex time-consuming computations
- Documents that require precise formatting & complicated formatting

How to create knitr document

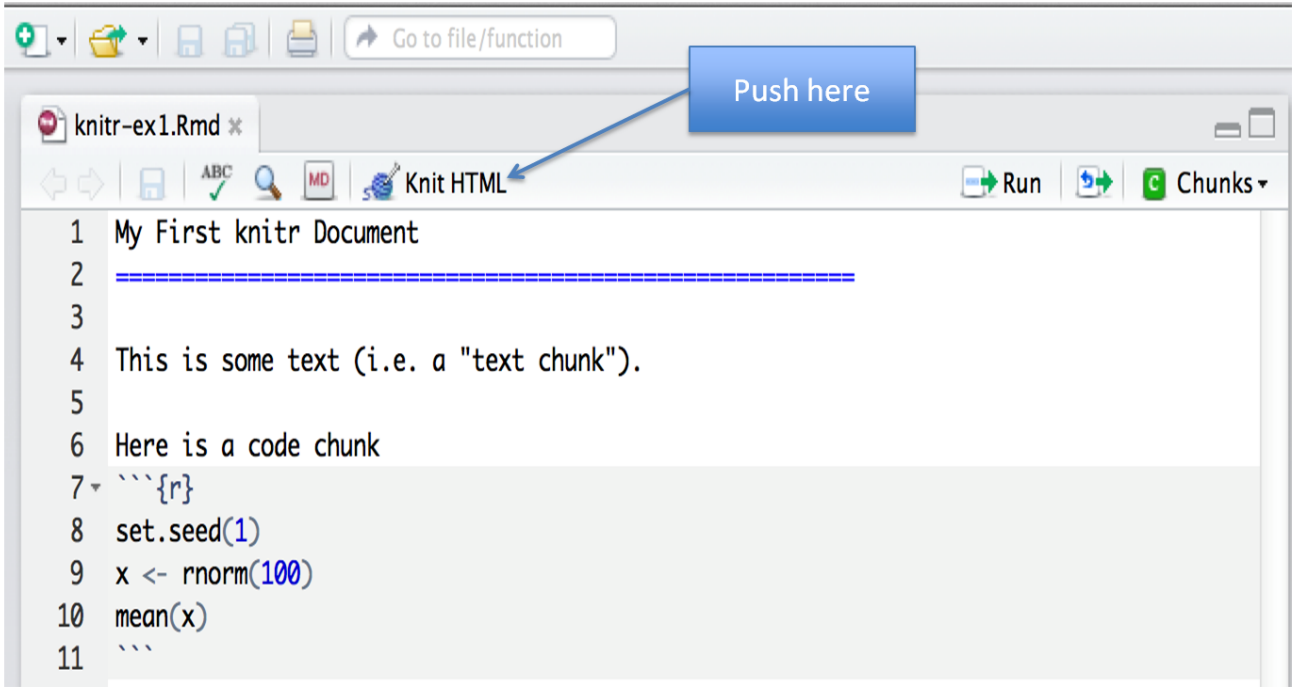


First knitr document as example

````
1  My First knitr Document
2  ==========================================================
3
4  This is some text (i.e. a "text chunk").
5
6  Here is a code chunk
7  ```{r}
8  set.seed(1)
9  x <- rnorm(100)
10 mean(x)
11 ```
````



Processing a knitr document: one click



Processing a knitr document: one click

- library(knitr)setwd()
- knit2html(“document.Rmd”)
- browseURL(“document.html”)

Knitr to HTML Output

My First knitr Document

This is some text (i.e. a "text chunk").

Here is a code chunk

```
set.seed(1)
x <- rnorm(100)
mean(x)
```

Code input

```
## [1] 0.1089
```

Numerical output

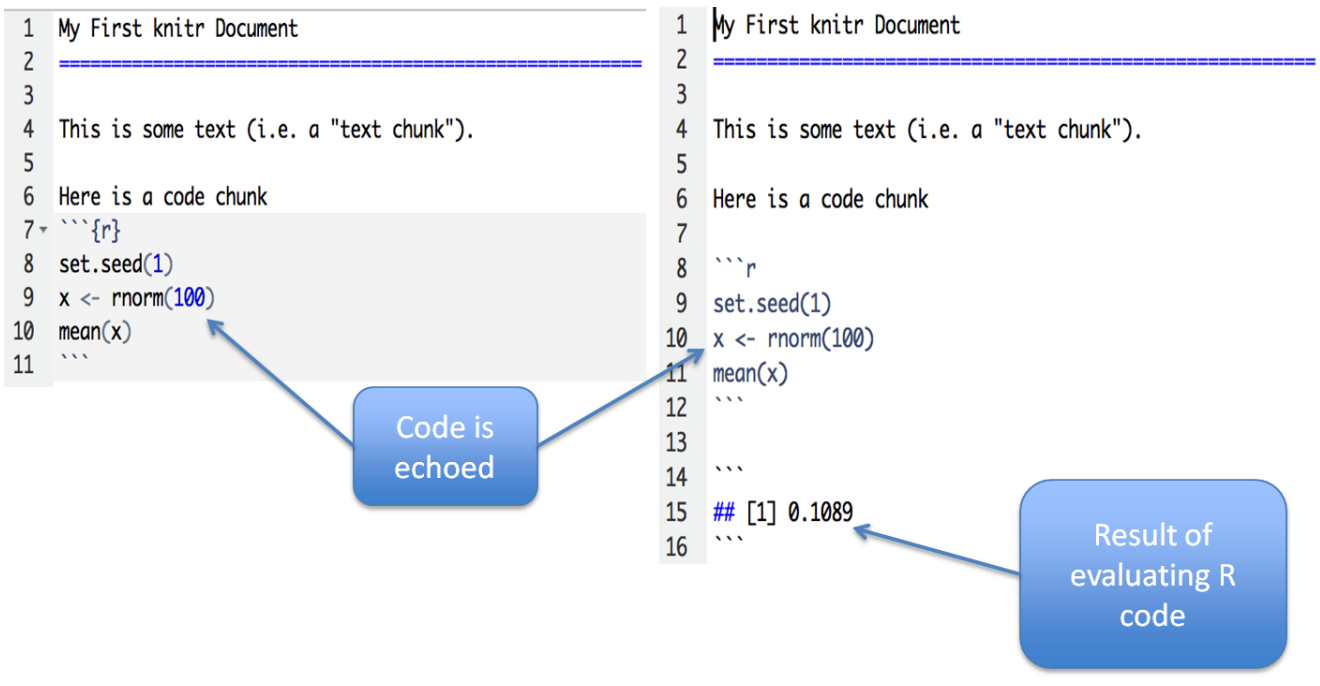
What knitr produces: Markdown

RMarkdown Document

```
1  My First knitr Document
2  ==========================================================
3
4  This is some text (i.e. a "text chunk").
5
6  Here is a code chunk
7  ```{r}
8  set.seed(1)
9  x <- rnorm(100)
10 mean(x)
11 ```
```

Markdown Document (generated)

```
1  My First knitr Document
2  ==========================================================
3
4  This is some text (i.e. a "text chunk").
5
6  Here is a code chunk
7
8  ```r
9  set.seed(1)
10 x <- rnorm(100)
11 mean(x)
12 ```
13
14 ```
15 ## [1] 0.1089
16 ```
```



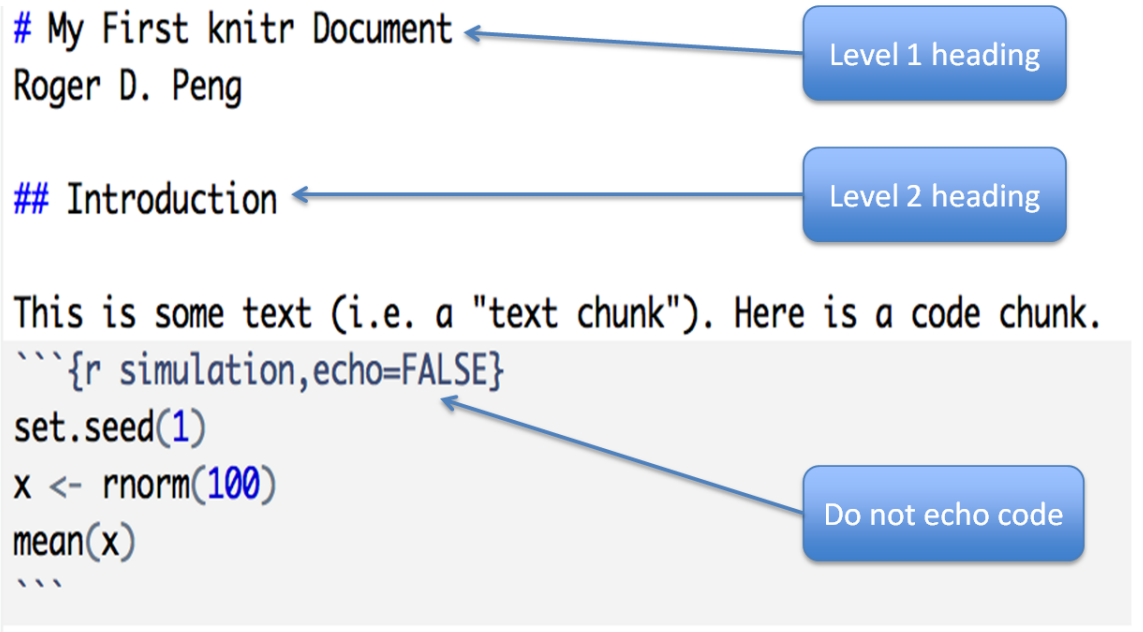
A few notes

- knitr will fill a new document with filler text; delete it
- Code chunks begin with `{r}` and end with `
- All R code goes in between these markers
- Code chunks can have names, which is useful when we start making graphics ```{r firstchunk}␣## R code goes here
- By default, code in a code chunk is echoed, as will the results of the computation (if there are results to print)

Processing of knitr Documents (behind the hood)

- You write the RMarkdown document (.Rmd)
- knitr produces a Markdown document (.md)
- knitr converts the Markdown document into HTML (by default) .Rmd --> .md -->.html
- You should NOT edit (or save) the .md or .html documents until you are finished

Another Example



Output

My First knitr Document

Roger D. Peng

Introduction

This is some text (i.e. a “text chunk”). Here is a code chunk.

```
## [1] 0.1089
```

Hiding Results

````
# My First knitr Document
Roger D. Peng

## Introduction

This is some text (i.e. a "text chunk"). Here is a code chunk but it doesn't print
anything!
```{r simulation,echo=FALSE,results="hide"}
set.seed(1)
x <- rnorm(100)
mean(x)
```
````

Output

My First knitr Document

Roger D. Peng

Introduction

This is some text (i.e. a “text chunk”). Here is a code chunk but it doesn't print anything!

Inline Text Computations

````
# My First knitr Document

## Introduction

```{r computetime,echo=FALSE}
time <- format(Sys.time(), "%a %b %d %X %Y")
rand <- rnorm(1)
```

The current time is `r time`. My favorite random number is `r rand`.
````

Inline Text Computations (output)

My First knitr Document

Introduction

The current time is Wed Sep 04 16:42:09 2013. My favorite random number is 1.1829.

Incorporating Graphics

````
## Introduction

Let's first simulate some data.
```{r simulatedata,echo=TRUE}
x <- rnorm(100); y <- x + rnorm(100, sd = 0.5)
```
Here is a scatterplot of the data.
```{r scatterplot,fig.height=4}
par(mar = c(5, 4, 1, 1), las = 1)
plot(x, y, main = "My Simulated Data")
```
````

Adjust figure height

What knitr Produces in HTML

```
<body>
<h2>Introduction</h2>

<p>Let&#39;s first simulate some data.</p>

<pre><code class="r">x &lt;- rnorm(100)
y &lt;- x + rnorm(100, sd = 0.5)
</code></pre>

<p>Here is a scatterplot of the data.</p>

<pre><code class="r">par(mar = c(5, 4, 1, 1), las = 1)
plot(x, y, main = &quot;My Simulated Data&quot;)
</code></pre>

<p><img src="
lDQ1BJQ0MgUHJvZmlsZQAAOBGFVd9v21QUPolvUqQWPyBYR4eKxa9VU1u5GxqtxgZJk6XtShal6dgqJOQ6N4m
pGwfb6baqT3uBNwb8AUDZAw9IPCENBmJ72fbAtElThyqqSUh76MQPISbtBVXhu3ZiJ1PEXPX6yznfOec7517b
RD1fabWaGVWIlquunc8klZOnFpSeTYrSs9RLA9Sr6U4tkcvNEi7BFffO6+EdigjL7ZHu/k72I796i9zRiSJPw
G4VHX0Z+AxRzNRrtksUvwf7+Gm3BtzzHPDTNgQCqwKXfZwSeNHHJz1OIT8JjtAq6xWtCLwGPLzYZi
```

Incorporating Graphics

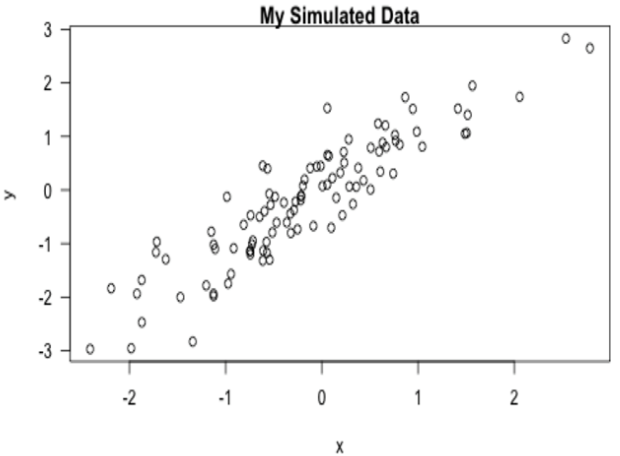
Introduction

Let's first simulate some data.

```
x <- rnorm(100)
y <- x + rnorm(100, sd = 0.5)
```

Here is a scatterplot of the data.

```
par(mar = c(5, 4, 1, 1), las = 1)
plot(x, y, main = "My Simulated Data")
```



Making Tables with xtable

````
## Introduction

```{r fitmodel}
library(datasets)
data(airquality)
fit <- lm(Ozone ~ Wind + Temp + Solar.R, data = airquality)
```

Here is a table of regression coefficients.

```{r showtable,results="asis"}
library(xtable)
xt <- xtable(summary(fit))
print(xt, type = "html")
```
````

Making Tables with xtable (output)

Introduction

```
library(datasets)
data(airquality)
fit <- lm(Ozone ~ Wind + Temp + Solar.R, data = airquality)
```

Here is a table of regression coefficients.

```
library(xtable)
xt <- xtable(summary(fit))
print(xt, type = "html")
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-64.3421	23.0547	-2.79	0.0062
Wind	-3.3336	0.6544	-5.09	0.0000
Temp	1.6521	0.2535	6.52	0.0000
Solar.R	0.0598	0.0232	2.58	0.0112

Setting Global Options

- Sometimes we want to set options for every code chunk that are different from the defaults
- For example, we may want to suppress all code echoing and results output
- We have to write some code to set these global options

Setting Global Options

````
## Introduction

```{r setoptions,echo=FALSE}
opts_chunk$set(echo = FALSE, results = "hide")
```

First simulate data
```{r simulatedata,echo=TRUE}
x <- rnorm(100); y <- x + rnorm(100, sd = 0.5)
```

Here is a scatterplot of the data.

```{r scatterplot,fig.height=4}
par(mar = c(5, 4, 1, 1), las = 1)
plot(x, y, main = "My Simulated Data")
```
````

Set default to NOT echo code

Override default

Don't echo code here

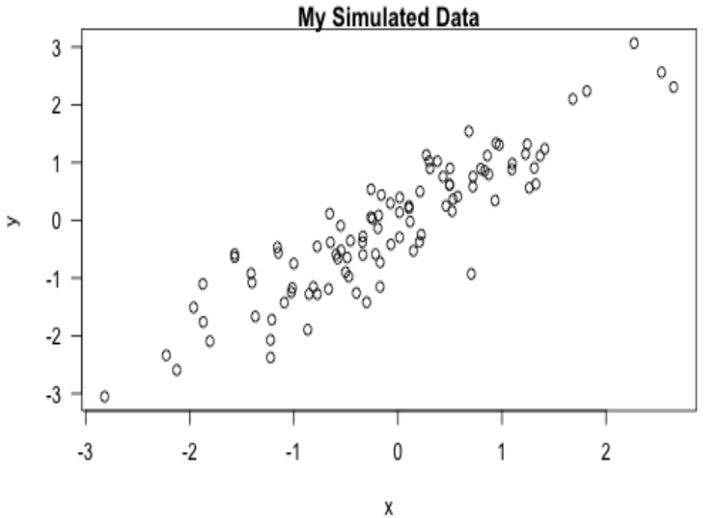
Setting Global Options (output)

Introduction

First simulate data

```
x <- rnorm(100)
y <- x + rnorm(100, sd = 0.5)
```

Here is a scatterplot of the data.



Some common options

- Output
  - results: “asis”, “hide”
  - echo: TRUE, FALSE
- Figures
  - fig.height: numeric
  - fig.width: numeric

Caching computation

- What if one chunk takes a long time to run?
- All chunks have to be re-computed every time you re-knit the file
- The cache=TRUE option can be set on a chunk-by-chunk basis to store results of computation
- After the first run, results are loaded from cache

Caching caveats

- If the data or code (or anything external) changes, you need to re-run the cached code chunks
- Dependencies are not checked explicitly
- Chunks with significant side effects may not be cacheable

summary

- Literate statistical programming can be a useful way to put text, code, data, output all in one document
- knitr is a powerful tool for integrating code and text in a simple document format

Thank you for listening!

Any questions now or email me at **dossa@xtbg.org.cn**

Slides created via the R package **xaringan**.

The chakra comes from remark.js, **knitr**, and R Markdown.