Coding Standards for R

Roger D. Peng, Associate Professor of Biostatistics
Johns Hopkins Bloomberg School of Public Health

Coding Standards for R

1. Always use text files / text editor

Coding Standards for R

1. Always use text files / text editor

2. Indent your code

Coding Standards for R

1. Always use text files / text editor

2. Indent your code

3. Limit the width of your code (80 columns?)

Indenting

- Indenting improves readability
- Fixing line length (80 columns) prevents lots of nesting and very long functions
- Suggested: Indents of 4 spaces at minimum; 8 spaces ideal

Coding Standards for R

1. Always use text files / text editor

2. Indent your code

3. Limit the width of your code (80 columns?)

4. Limit the length of individual functions

Introduction to Markdown

Roger D. Peng, Associate Professor of Biostatistics
Johns Hopkins Bloomberg School of Public Health

What is Markdown?

"Markdown is a text-to-HTML conversion tool for web writers. Markdown allows you to write using an easy-to-read, easy-to-write plain text format, then convert it to structurally valid XHTML (or HTML)."

- John Gruber, creator of Markdown

Markdown Syntax

Italics

```
*This text will appear italicized!*
```

This text will appear italicized!

Markdown Syntax

Bold

```
**This text will appear bold!**
```

This text will appear bold!

Markdown Syntax

Headings

```
## This is a secondary heading
### This is a tertiary heading
```

This is a secondary heading

This is a tertiary heading

Markdown Syntax

Unordered Lists

```
- first item in list
- second item in list
- third item in list
```

- first item in list
- second item in list
- third item in list

Markdown Syntax

Ordered Lists

```
1. first item in list
2. second item in list
3. third item in list
```

1. first item in list
2. second item in list
3. third item in list

Markdown Syntax

Links

```
[Johns Hopkins Bloomberg School of Public Health](http://www.jhsph.edu/)
[Download R](http://www.r-project.org/)
[RStudio](http://www.rstudio.com/)
```

Johns Hopkins Bloomberg School of Public Health
Download R
RStudio

Markdown Syntax

Advanced Linking

```
I spend so much time reading [R bloggers][1] and [Simply Statistics][2]!
[1]: http://www.r-bloggers.com/   "R bloggers"
[2]: http://simplystatistics.org/ "Simply Statistics"
```

I spend so much time reading R bloggers and Simply Statistics!

Markdown Syntax

Newlines

- Newlines require a double space after the end of a line.

```
First line
Second line
```

First line Second line

```
First line  
Second line
```

First line
Second line

Markdown Resources

- The Offical Markdown Documentation
- Github's Markdown Guide

R Markdown

Roger D. Peng, Associate Professor of Biostatistics
Johns Hopkins Bloomberg School of Public Health

What is Markdown?

- Created by John Gruber and Aaron Swartz
- A simplified version of "markup" languages
- Allows one to focus on writing as opposed to formatting
- Simple/minimal intuitive formatting elements
- Easily converted to valid HTML (and other formats) using existing tools
- Complete information is available at http://daringfireball.net/projects/markdown/
- Some background information at http://daringfireball.net/2004/03/dive_into_markdown

What is R Markdown?

- R markdown is the integration of R code with markdown
- Allows one to create documents containing "live" R code
- R code is evaluated as part of the processing of the markdown
- Results from R code are inserted into markdown document
- A core tool in **literate statistical programming**

What is R Markdown?

- R markdown can be converted to standard markdown using the knitr package in R
- Markdown can be converted to HTML using the markdown package in R
- Any basic text editor can be used to create a markdown document; no special editing tools needed
- The R markdown --> markdown --> HTML work flow can be easily managed using R Studio (but not required)
- These slides were written in R markdown and converted to slides using the slidify package

Literate Statistical Programming with knitr

Reproducible Research

Roger D. Peng, Associate Professor of Biostatistics
Johns Hopkins Bloomberg School of Public Health

Problems, 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

How Do I Make My Work Reproducible?

- Decide to do it (ideally from the start)
- Keep track of things, perhaps with a version control system to track snapshots/changes
- Use software whose operation can be coded
- Don’t save output
- 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)

What is knitr?

- 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
- Built right into RStudio for your convenience

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

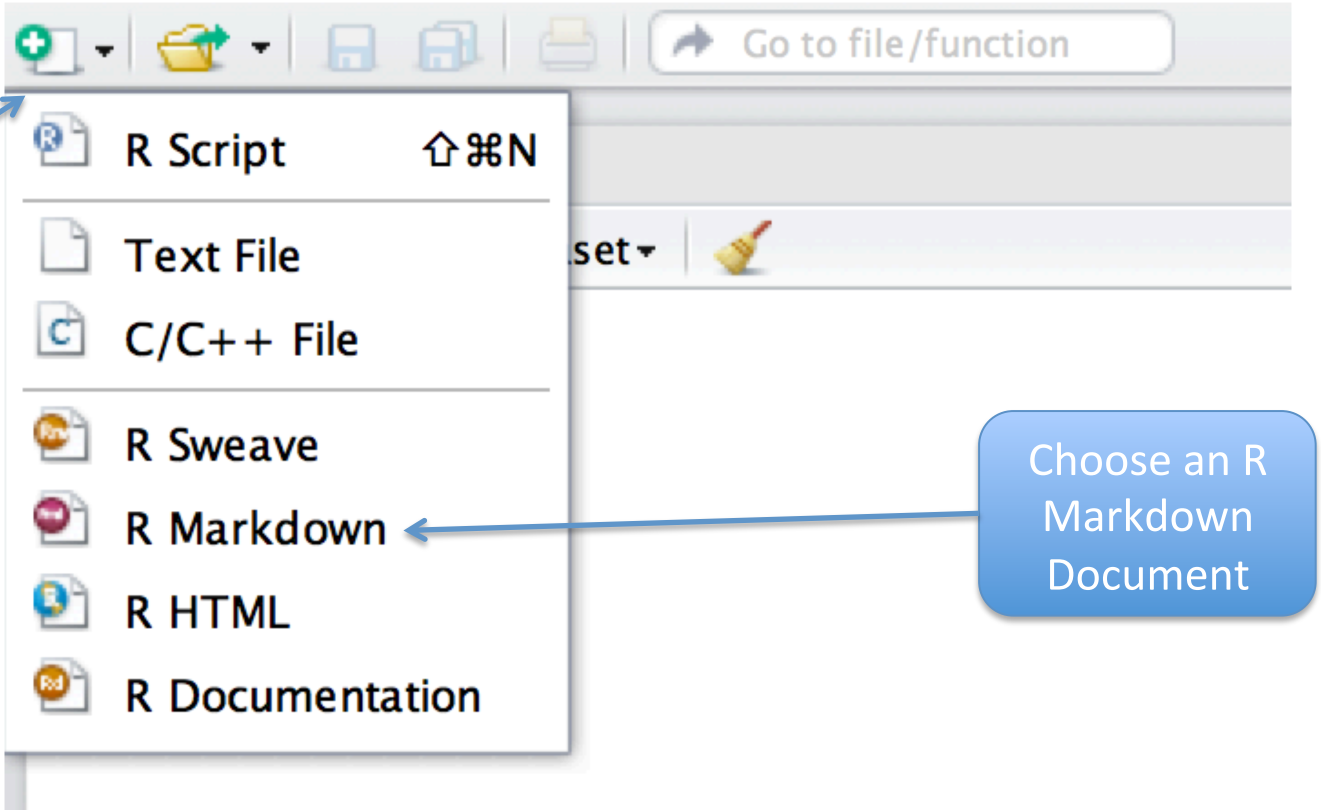
What is knitr Good For?

- Manuals
- 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

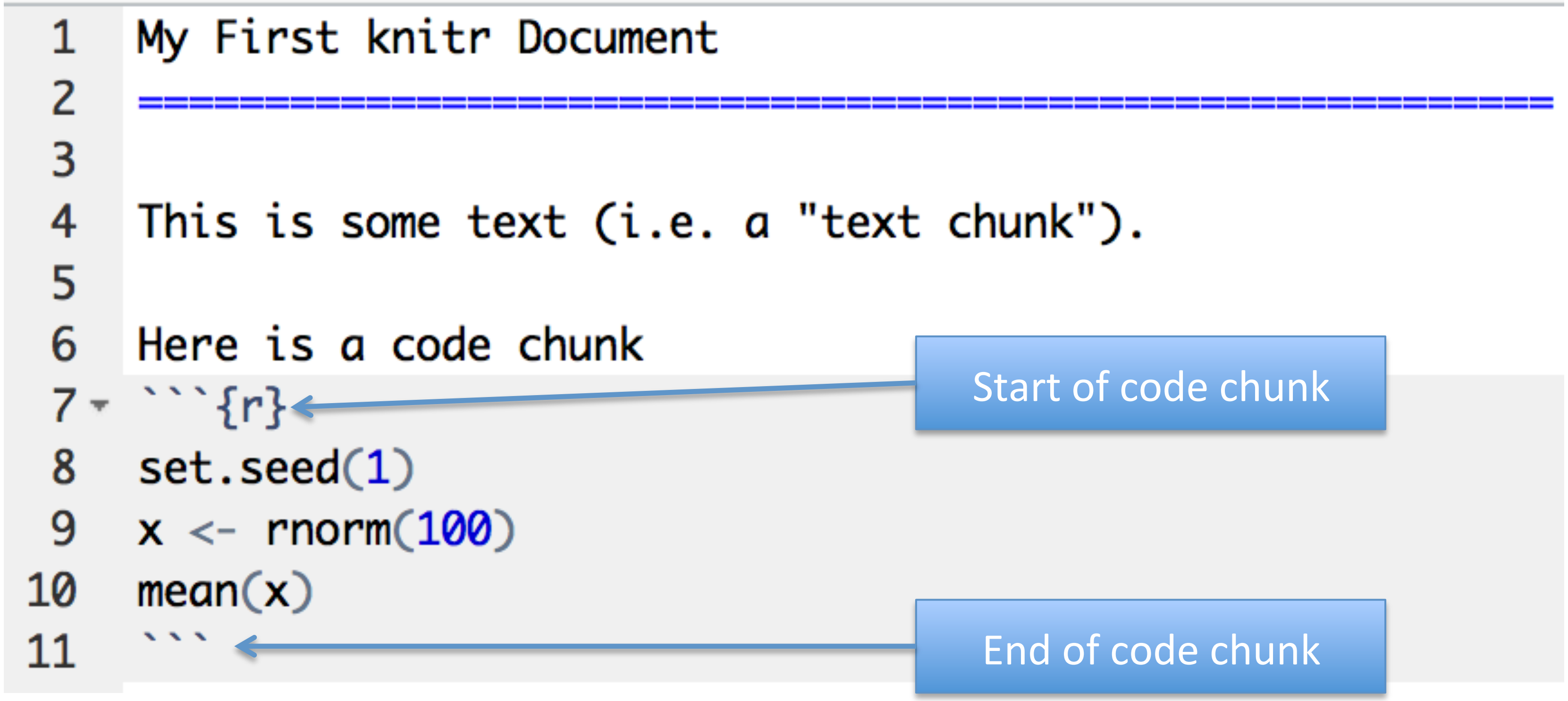
My First knitr Document



Create a new document

Choose an R Markdown Document

My First knitr Document

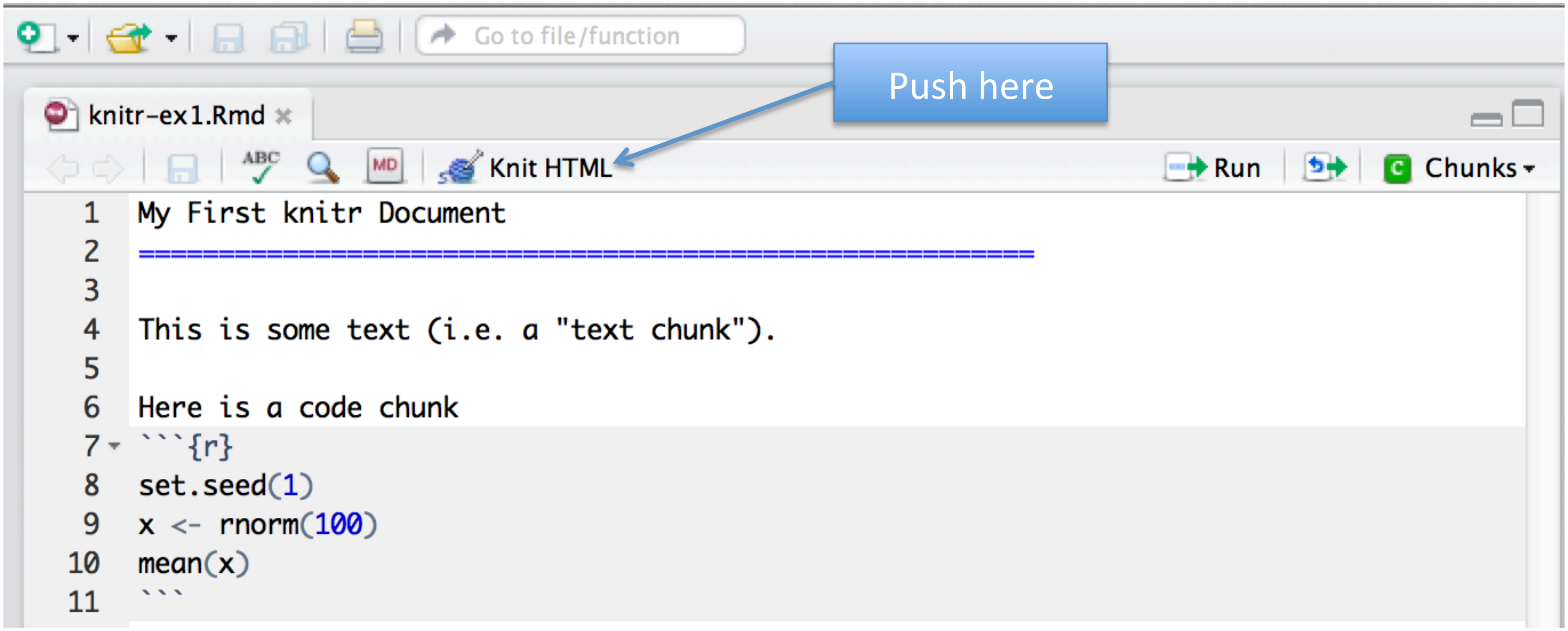


~~~
My First knitr Document
========================================================

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

Here is a code chunk
```{r}
set.seed(1)
x <- rnorm(100)
mean(x)
```
~~~

Start of code chunk

End of code chunk

Processing a knitr Document



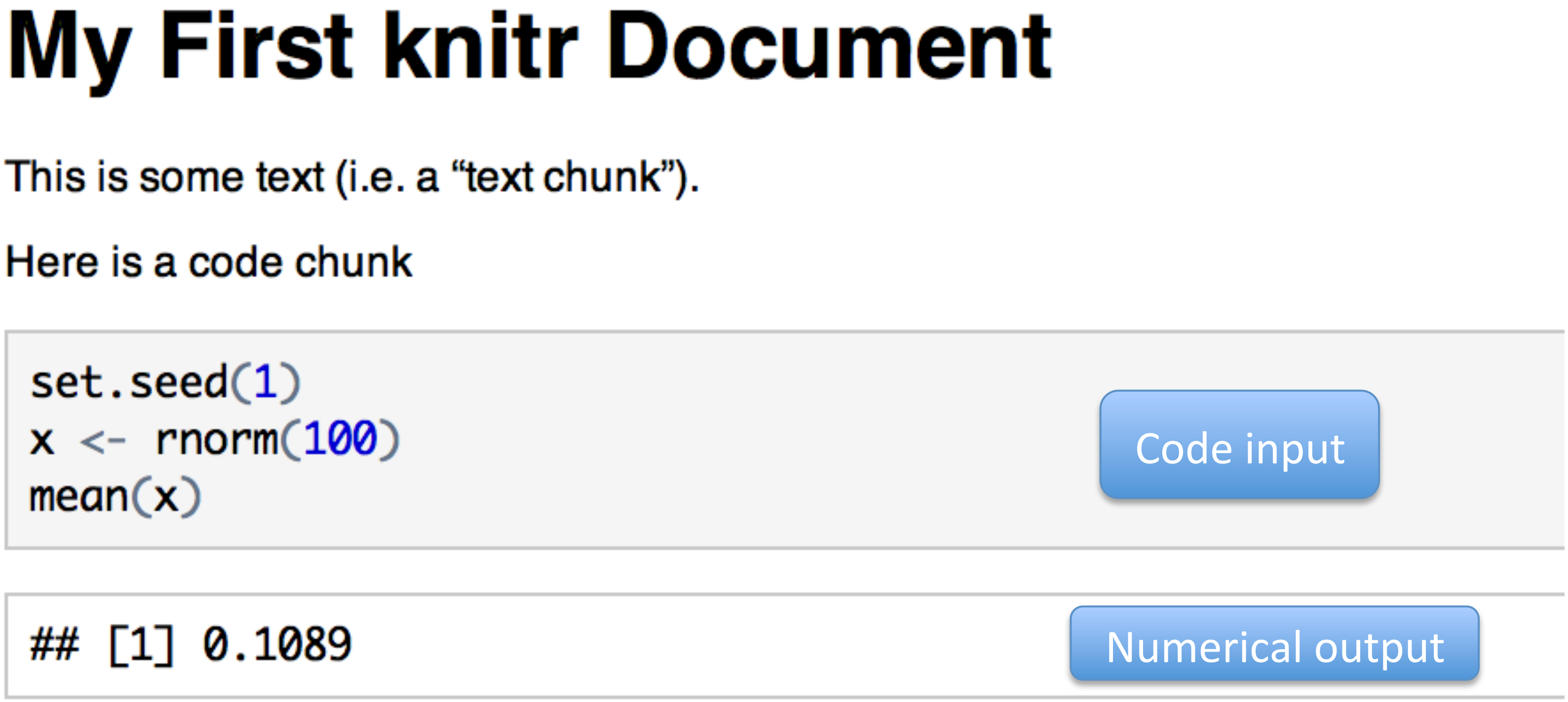
Push here

~~~
My First knitr Document
=======================================================

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

Here is a code chunk
```{r}
set.seed(1)
x <- rnorm(100)
mean(x)
```
~~~

More Complicated Way

```
library(knitr)
setwd(<working directory>)
knit2html(“document.Rmd”)
browseURL(“document.html”)
```

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)
```

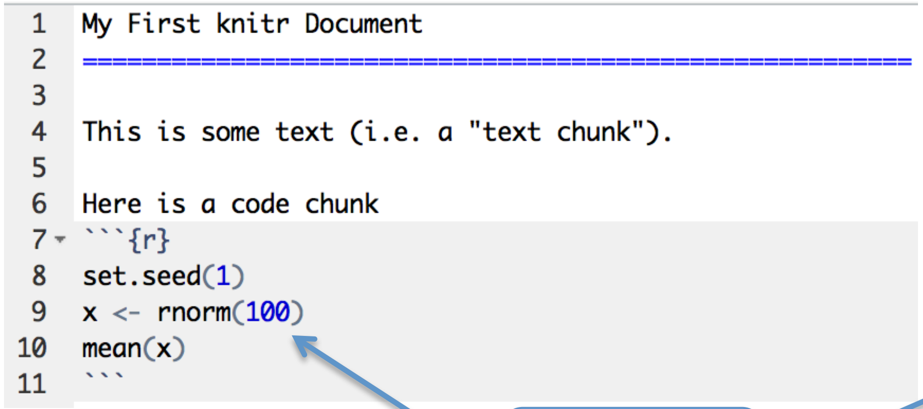
Code input

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

Numerical output

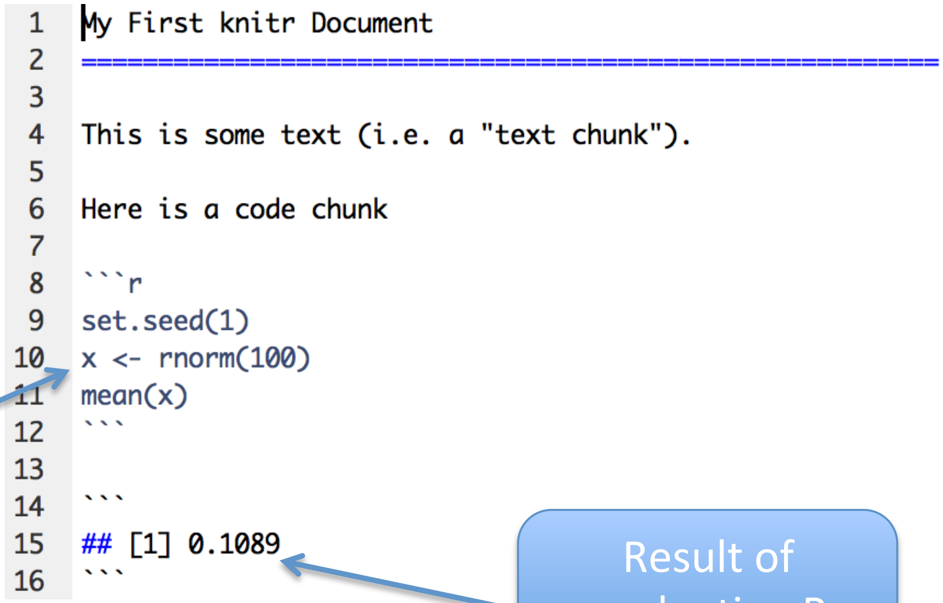
What knitr Produces: Markdown

RMarkdown Document



~~~
My First knitr Document
========================================================

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

Here is a code chunk
```{r}
set.seed(1)
x <- rnorm(100)
mean(x)
```
~~~

Markdown Document (generated)



~~~
My First knitr Document
========================================================

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

Here is a code chunk

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


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

Code is echoed

Result of evaluating R code

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

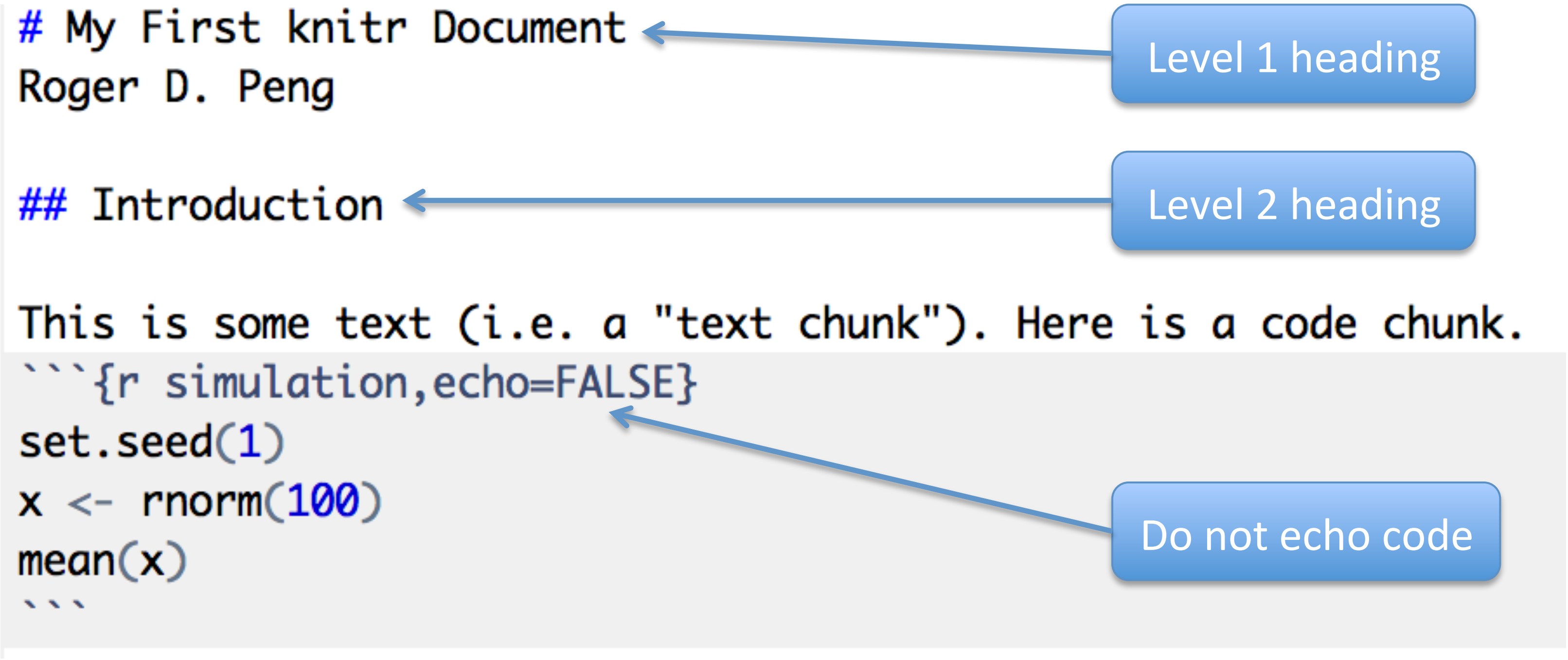
~~~
```{r firstchunk}
## R code goes here
```
~~~

- 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 (what happens under 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



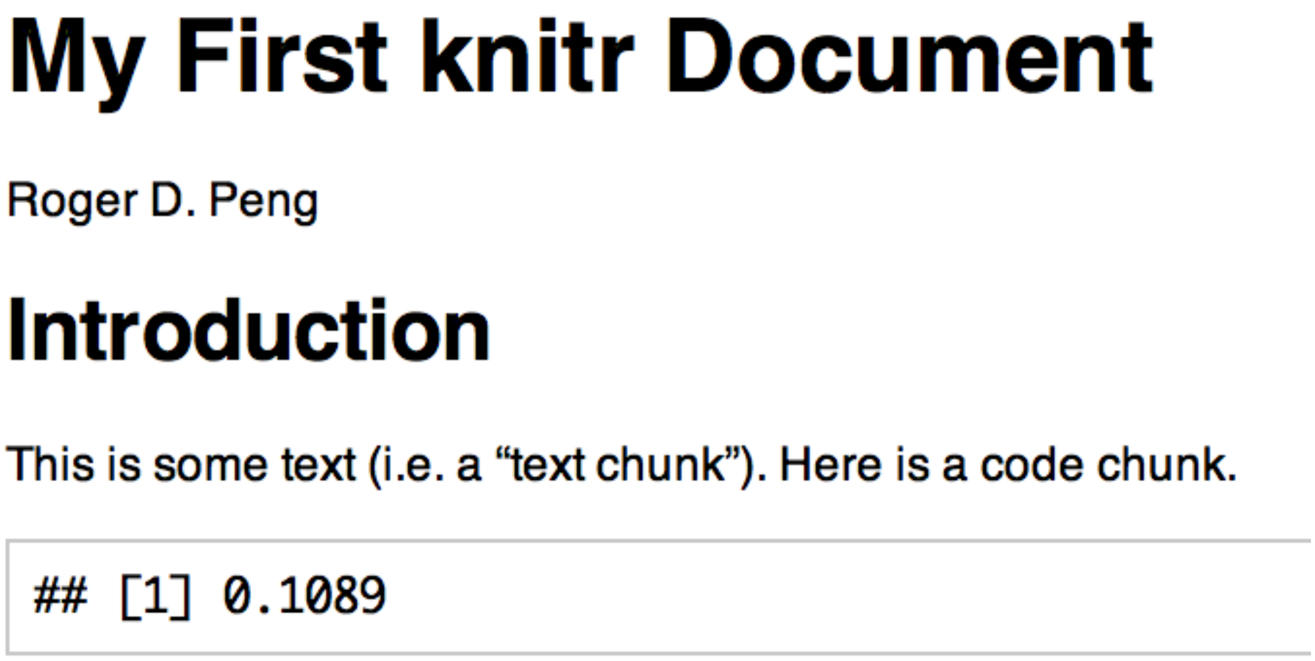
~~~
# My First knitr Document
Roger D. Peng

## Introduction

This is some text (i.e. a "text chunk"). Here is a code chunk.
```{r simulation,echo=FALSE}
set.seed(1)
x <- rnorm(100)
mean(x)
```
~~~

Level 1 heading

Level 2 heading

Do not echo code

Output



My First knitr Document

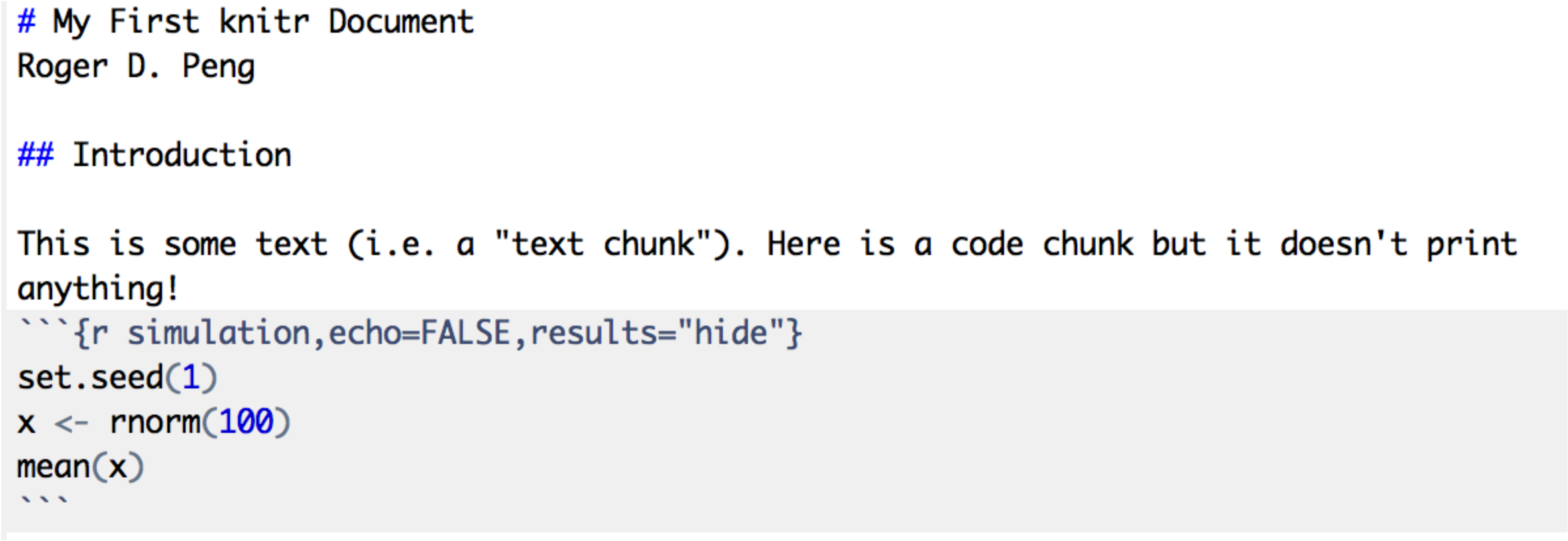
Roger D. Peng

Introduction

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

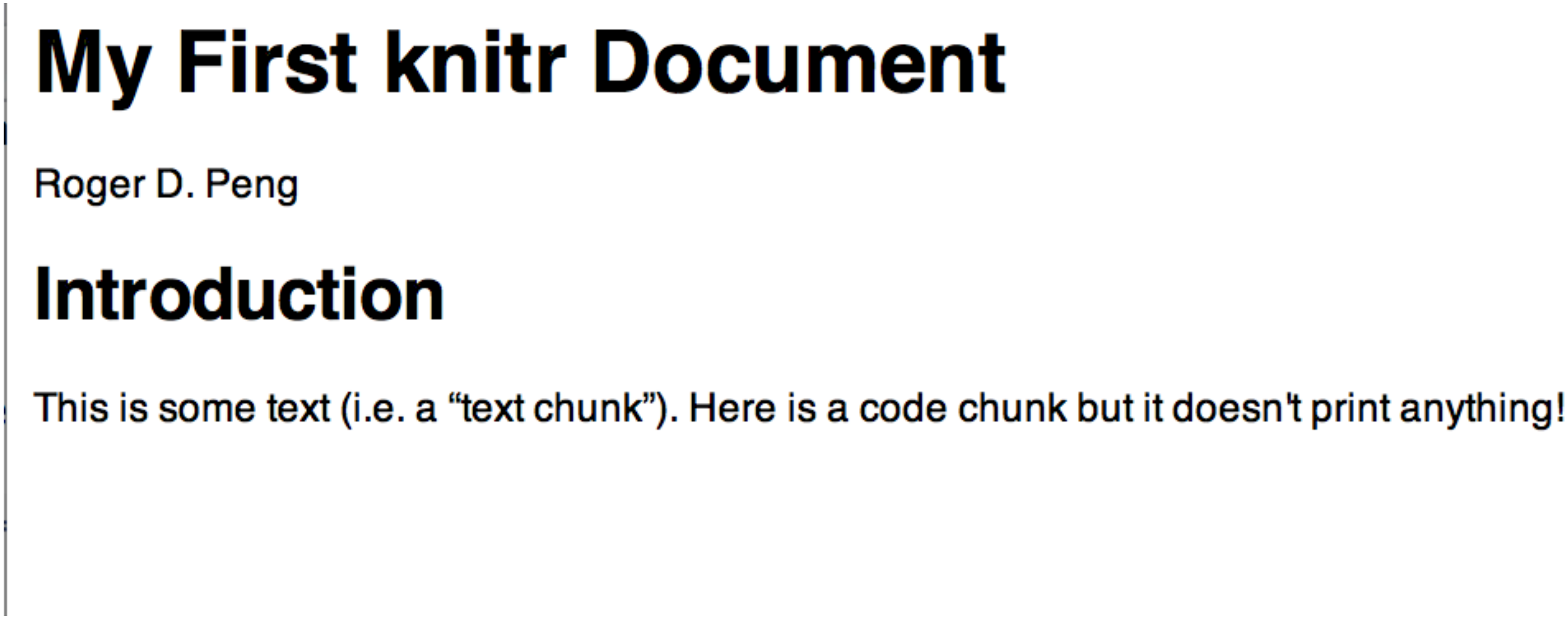
```
## [1] 0.1089
```

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



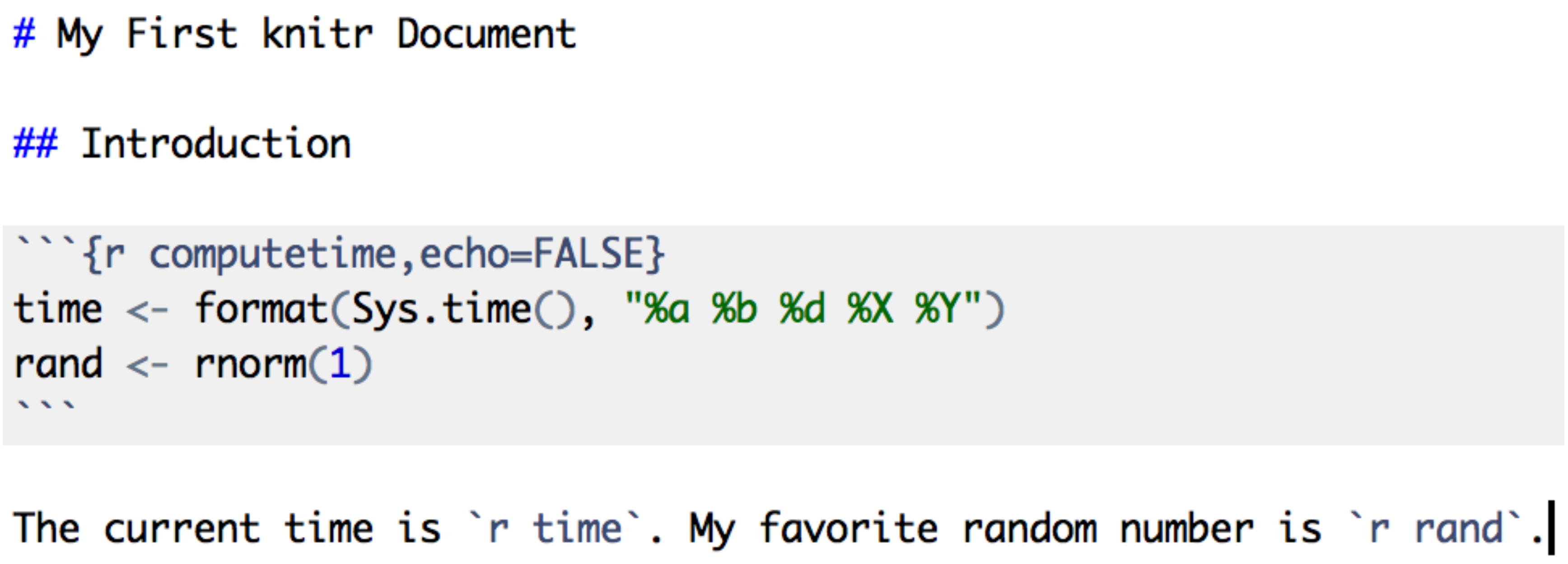
My First knitr Document

Roger D. Peng

Introduction

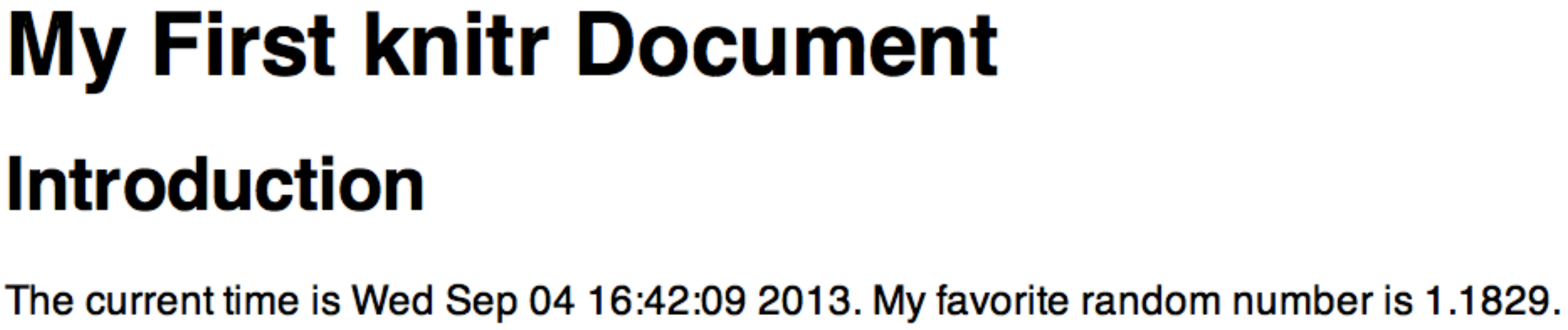
This is some text (i.e. a “text chunk”). Here is a code chunk but it doesn't print anything!

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

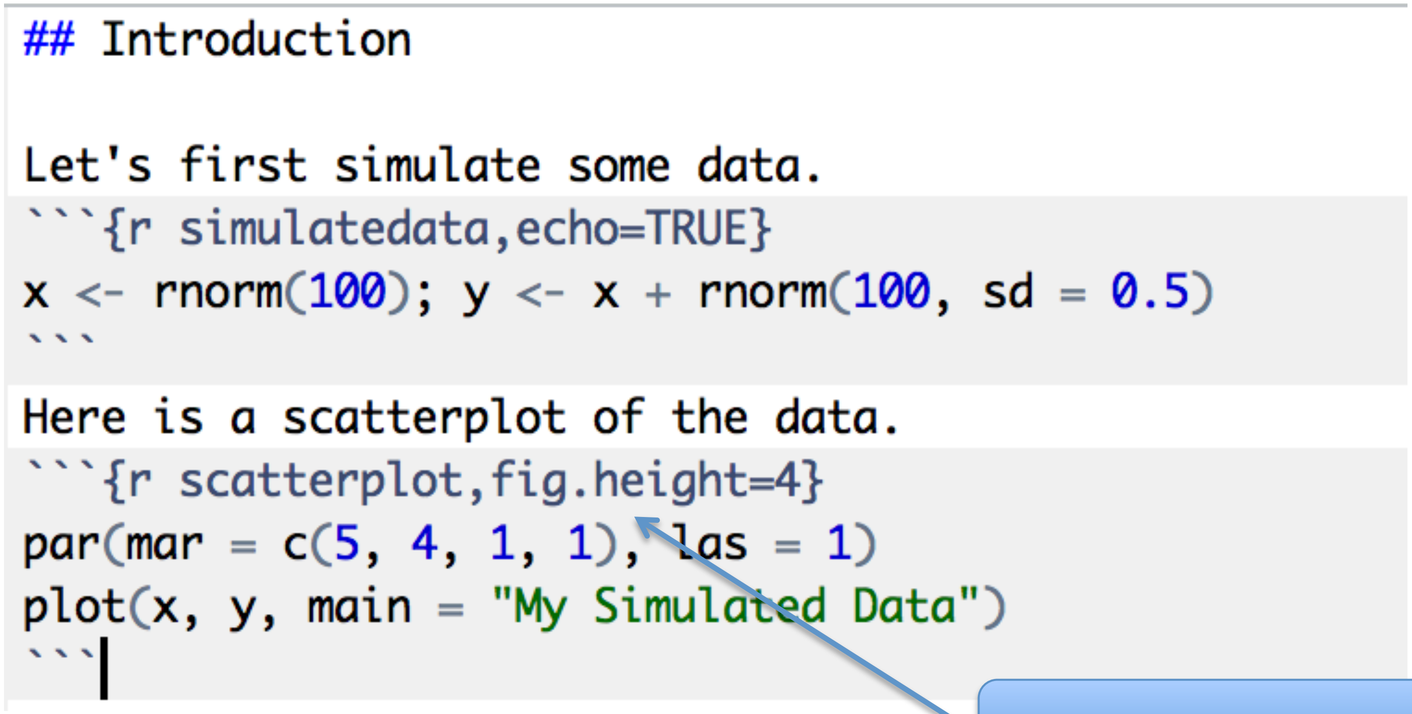


My First knitr Document

Introduction

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

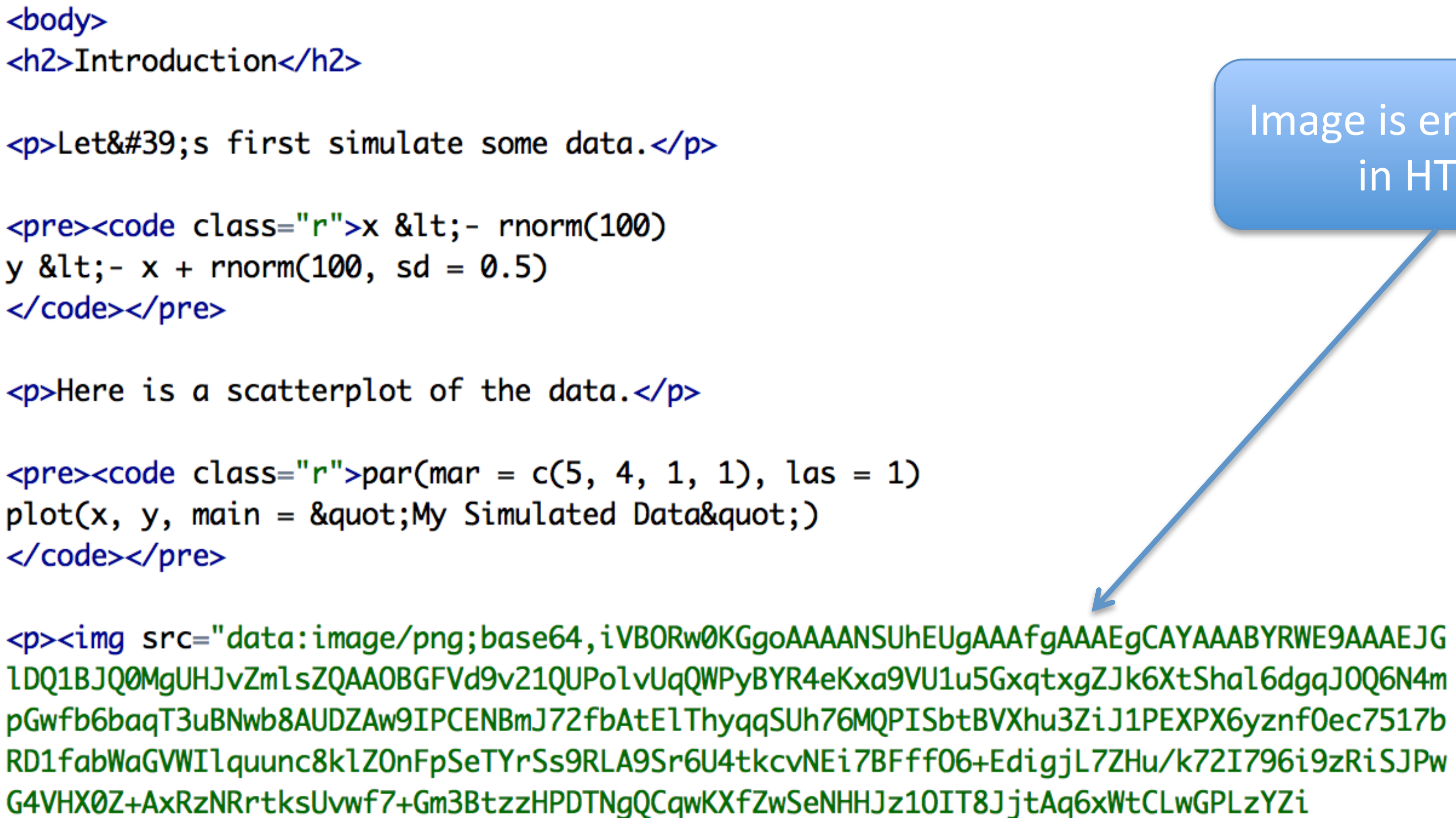
Incorporating Graphics



~~~
## 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

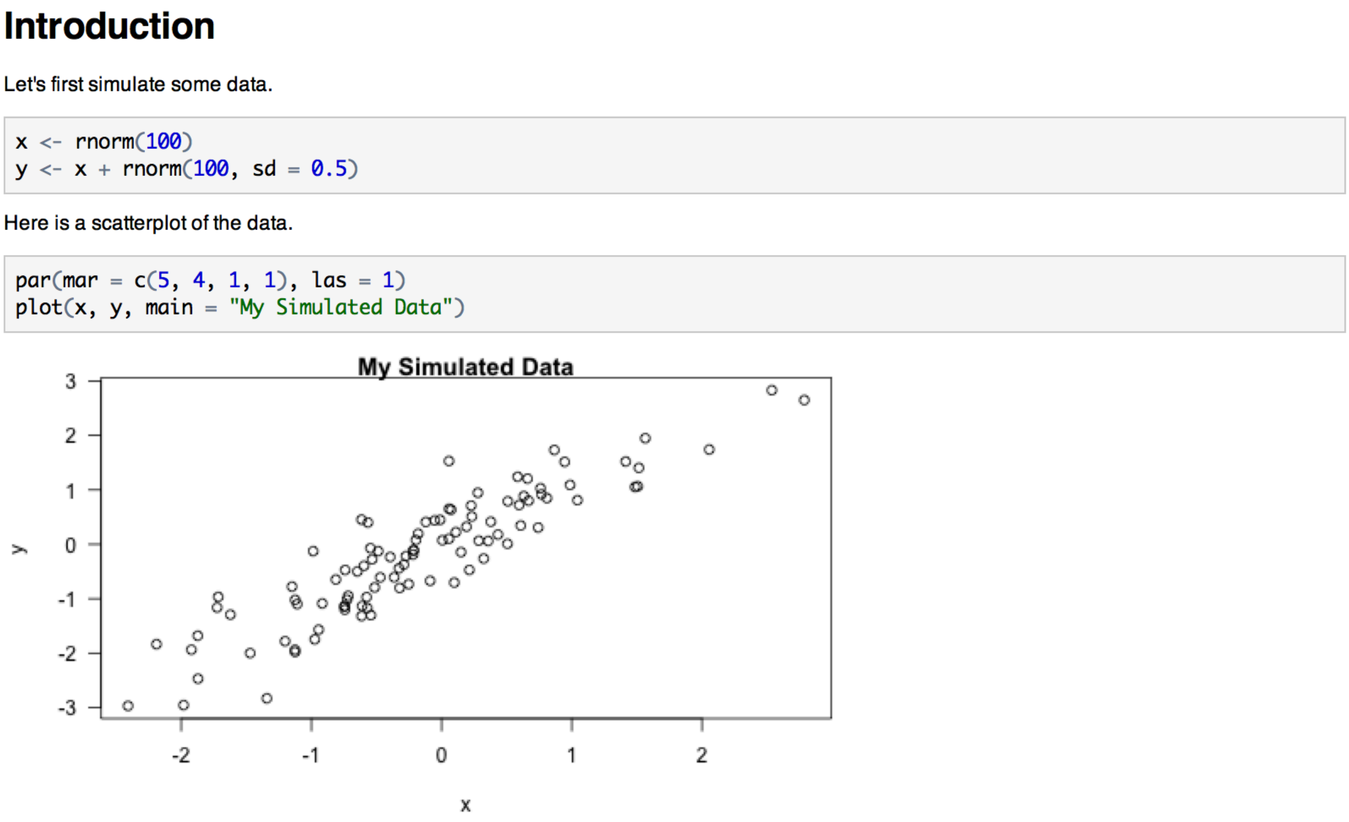
What knitr Produces in HTML



```
<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="data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAfgAAAEgCAYAAABYRWE9AAAEJG
lDQ1BJQ0MgUHJvZmlsZQAAOBGFVd9v21QUPolvUqQWPyBYR4eKxa9VU1u5GxqtxgZJk6XtShal6dgqJOQ6N4m
pGwfb6baqT3uBNwb8AUDZAw9IPCENBmJ72fbAtElThyqqSUh76MQPISbtBVXhu3ZiJ1PEXPX6yznfOec7517b
RD1fabWaGVWIlquunc8klZOnFpSeTYrSs9RLA9Sr6U4tkcvNEi7BFff06+EdigjL7ZHu/k72I796i9zRiSJPw
G4VHX0Z+AxRzNRrtksUvwf7+Gm3BtzzHPDTNgQCqwKXfZwSeNHHJz1OIT8JjtAq6xWtCLwGPLzYZi
```

Image is embedded in HTML

Incorporating Graphics



Introduction

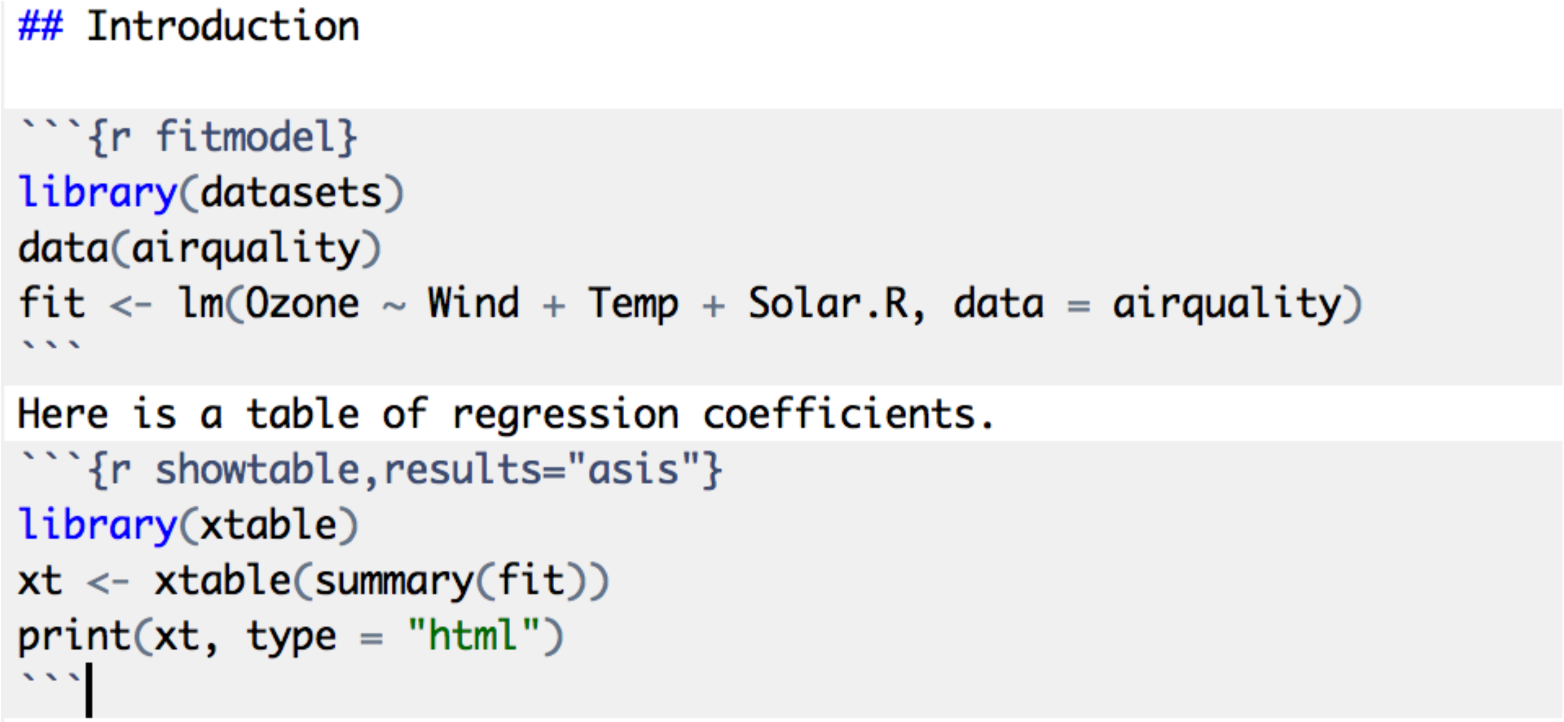
Let's first simulate some data.

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

Here is a scatterplot of the data.

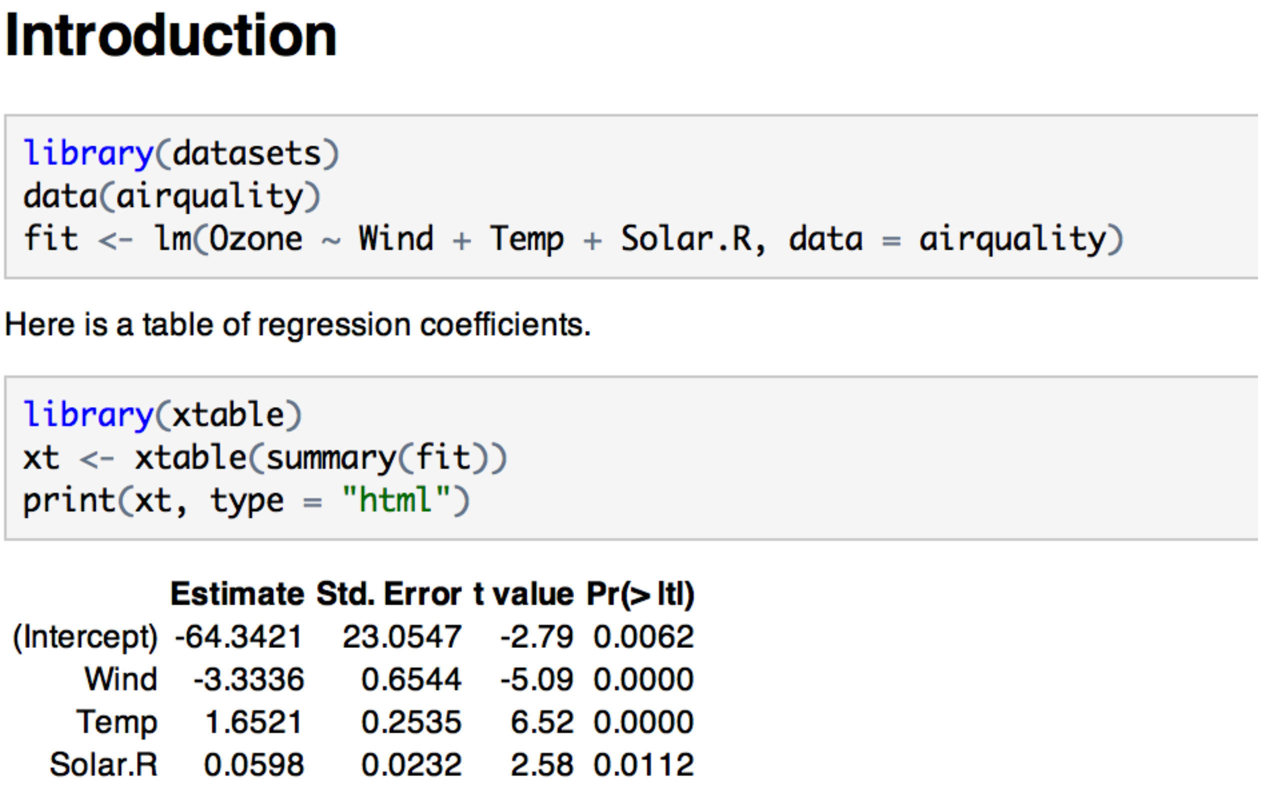
```
par(mar = c(5, 4, 1, 1), las = 1)
plot(x, y, main = "My Simulated Data")
```

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



Introduction

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

Here is a table of regression coefficients.

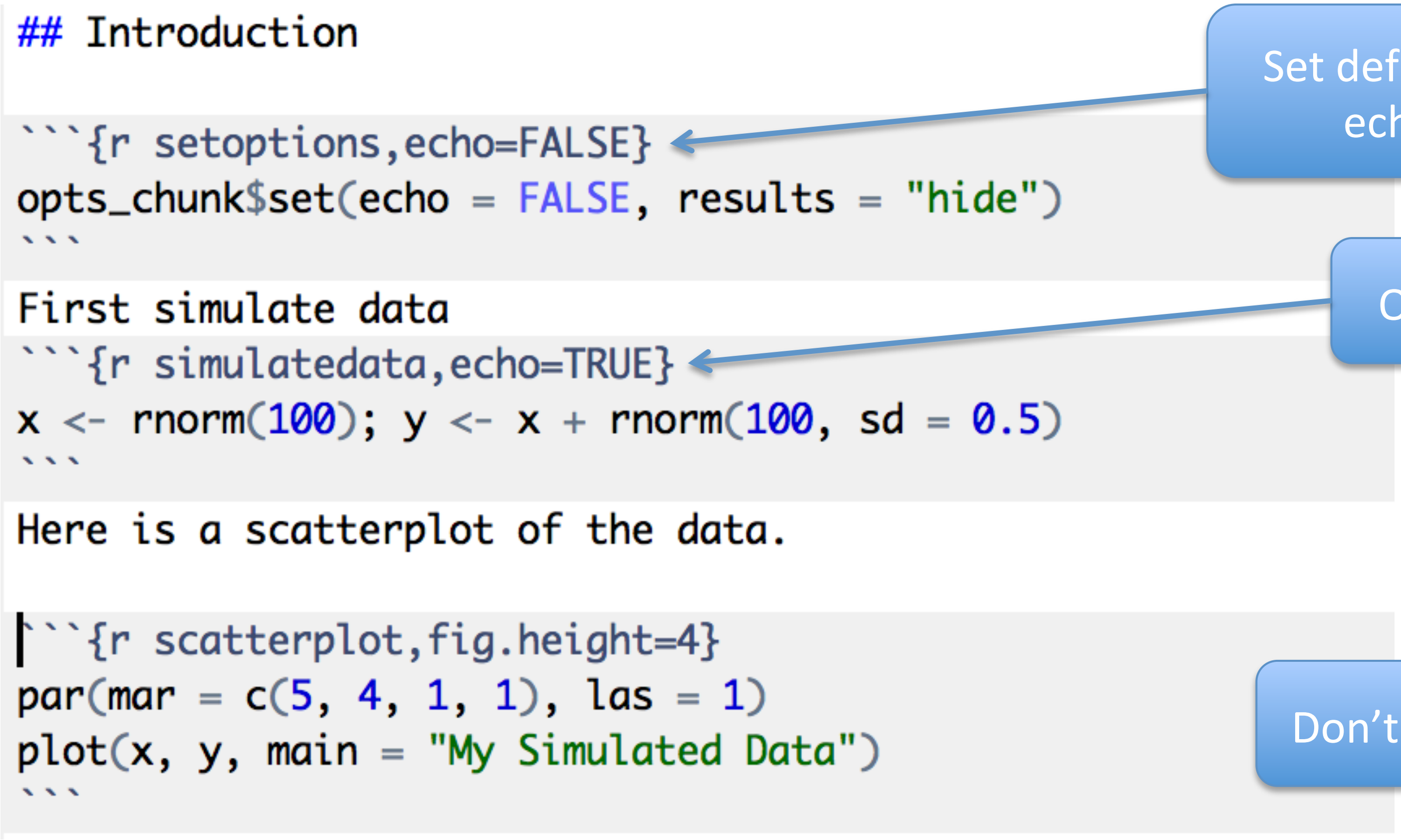
```
library(xtable)
xt <- xtable(summary(fit))
print(xt, type = "html")
```

	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



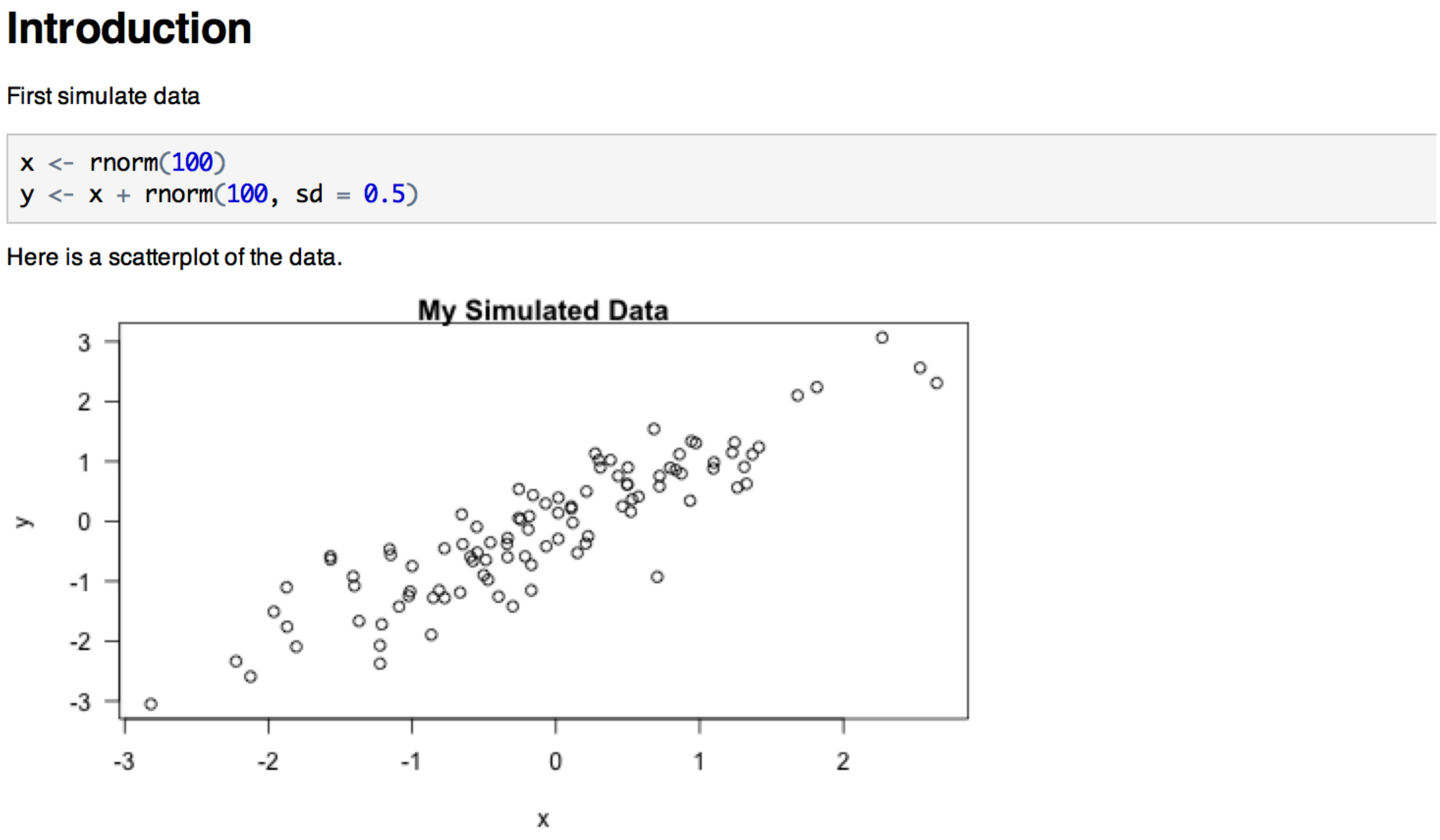
~~~
## Introduction

```{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



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 Computations

- 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