Reproducible Research and Literate Statistical Programming (with knitr)

Roger D. Peng, PhD

Department of Biostatistics

Johns Hopkins Bloomberg School of Public Health

University of Minnesota

April 2013

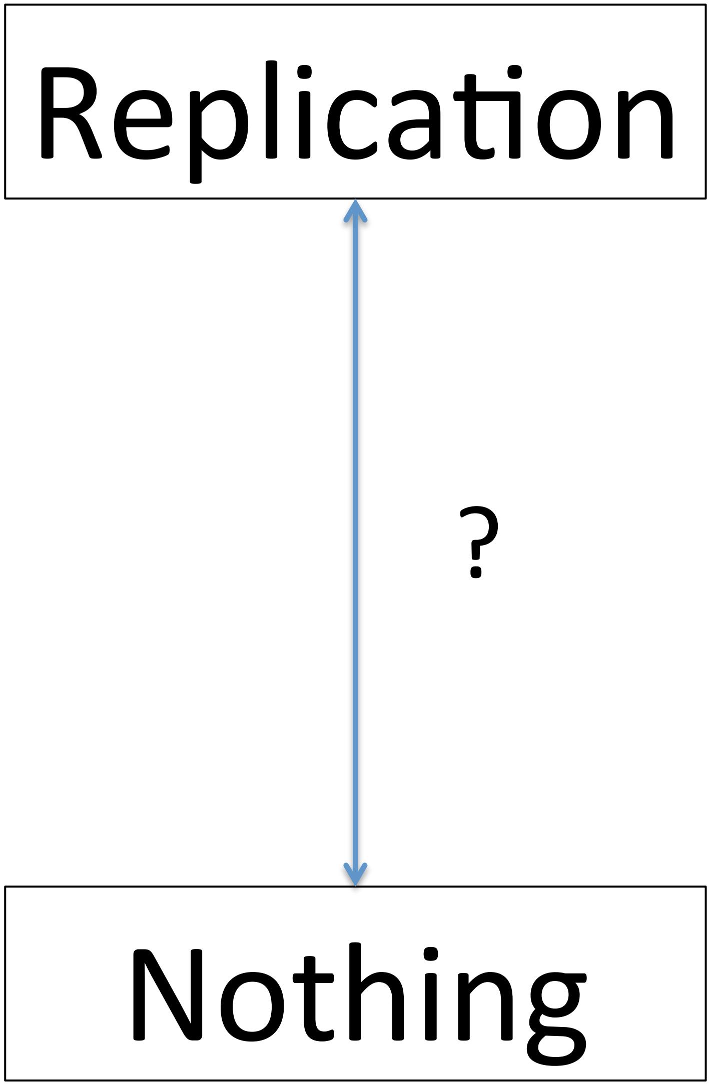
Replication

- The ultimate standard for strengthening scientific evidence is replication of findings and conducting studies with independent
  - Investigators
  - Data
  - Analytical methods
  - Laboratories
  - Instruments
- Replication is particularly important in studies that can impact broad policy or regulatory decisions

What's Wrong with Replication?

- Some studies cannot be replicated
  - No time, opportunistic
  - No money
  - Unique
- Reproducible Research: Make analytic data and code available so that others may reproduce findings

How Can We Bridge the Gap?

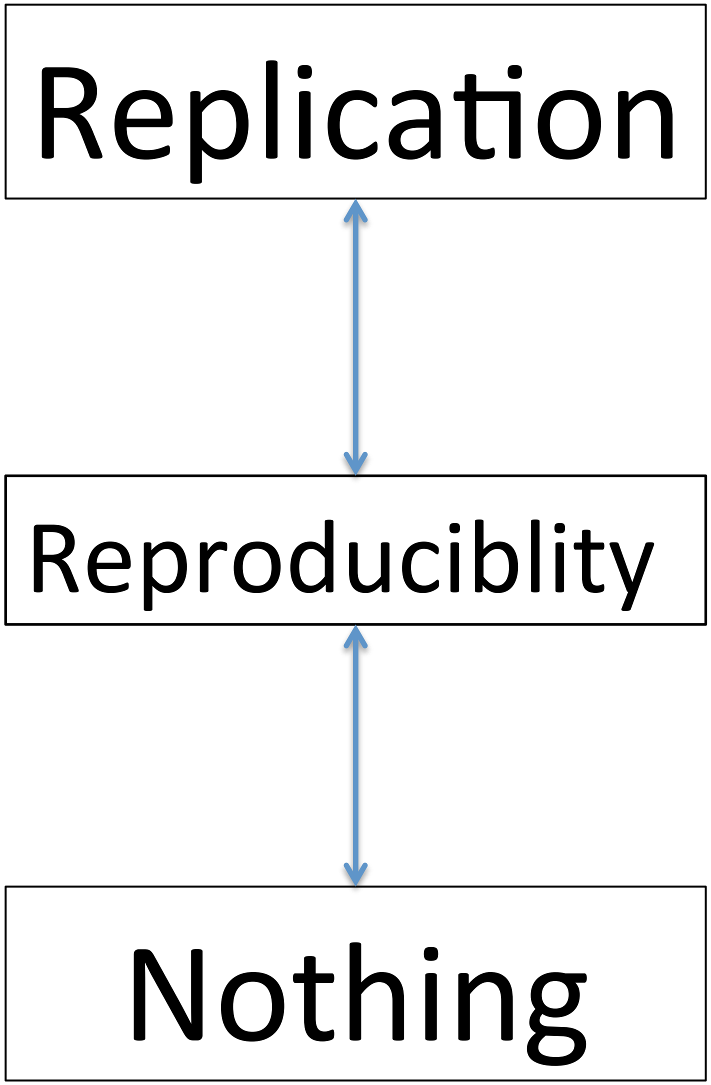


Replication

?

Nothing

How Can We Bridge the Gap?



Replication

Reproduciblity

Nothing

Why Do We Need Reproducible Research Now?

- New technologies increasing data collection throughput; data are more complex and extremely high dimensional
- Existing databases can be merged into new "megadatabases"
- Computing power is greatly increased, allowing more sophisticated analyses
- For every field "X" there is a field "Computational X"

Example: Reproducible Air Pollution and Health Research

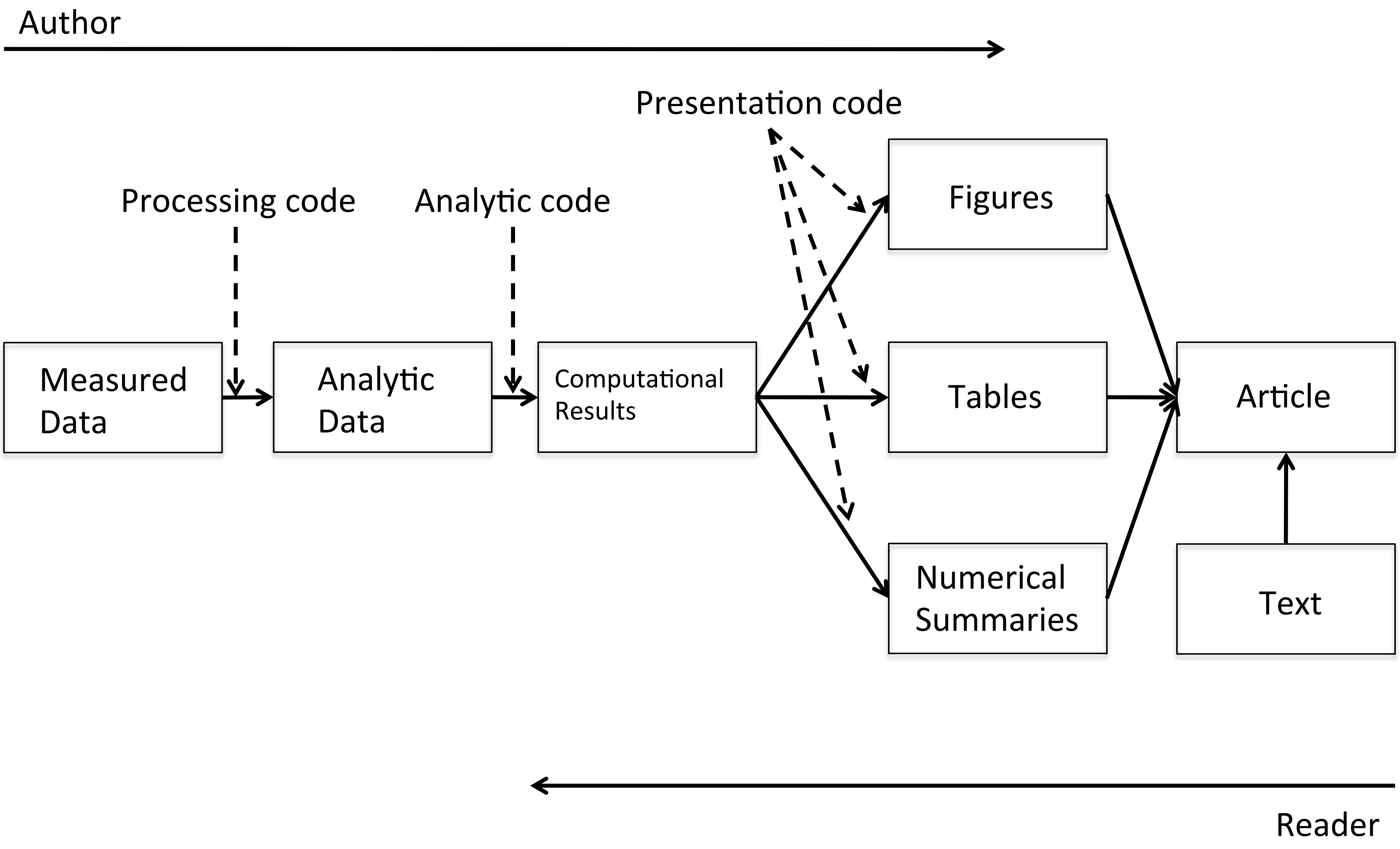
- Estimating small (but important) health effects in the presence of much stronger signals
- Results inform substantial policy decisions, affect many stakeholders
  - EPA regulations can cost billions of dollars
- Complex statistical methods are needed and subjected to intense scrutiny

Research Pipeline

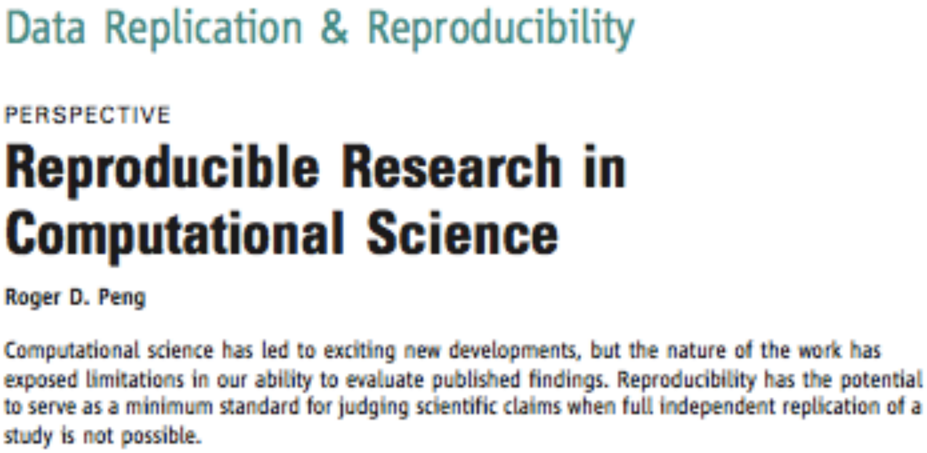
Article

Reader

Research Pipeline



Recent Developments in Reproducible Research



Data Replication & Reproducibility

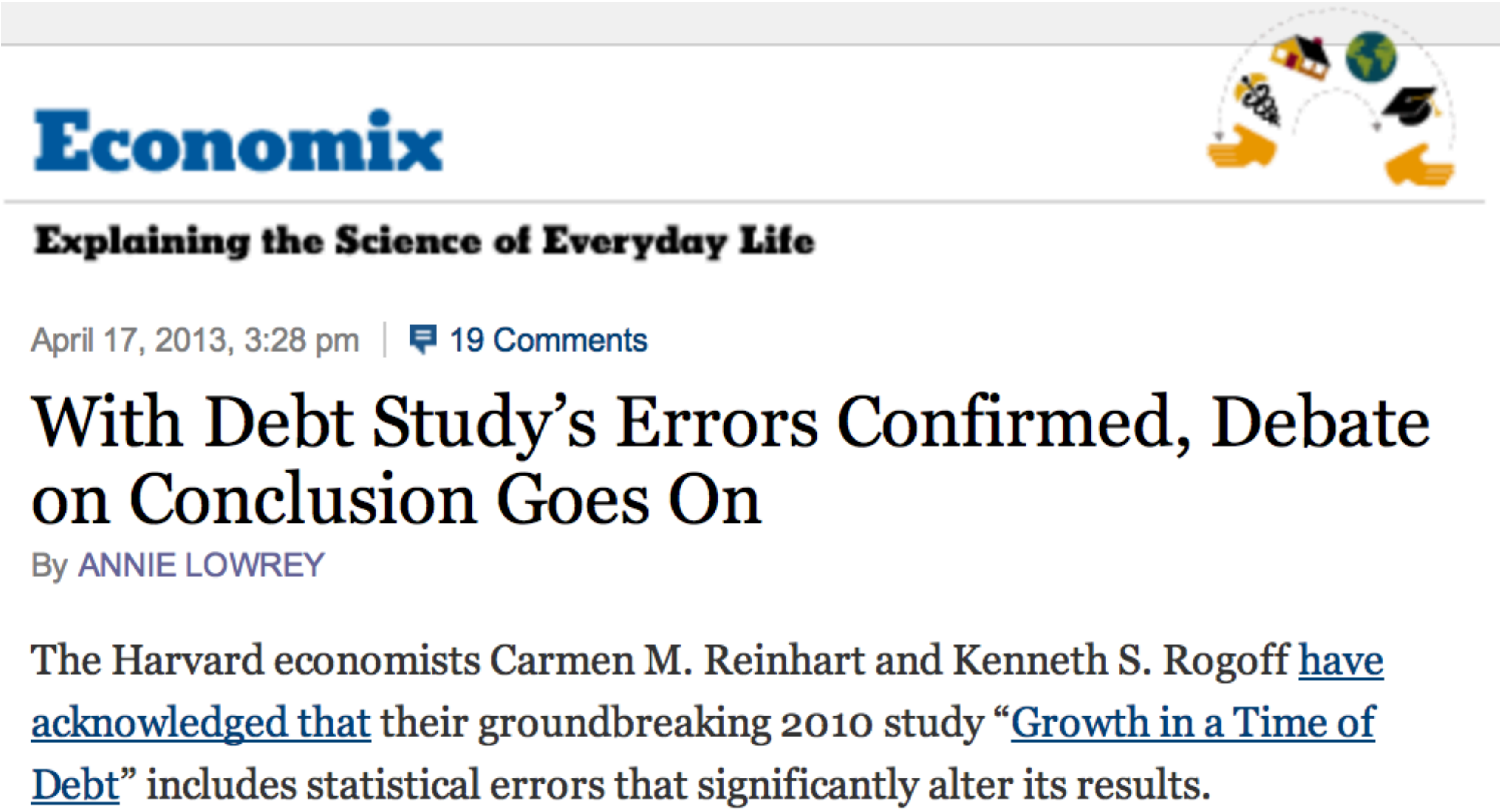
PERSPECTIVE

Reproducible Research in Computational Science

Roger D. Peng

Computational science has led to exciting new developments, but the nature of the work has exposed limitations in our ability to evaluate published findings. Reproducibility has the potential to serve as a minimum standard for judging scientific claims when full independent replication of a study is not possible.

Recent Developments in Reproducible Research



Economix

Explaining the Science of Everyday Life

April 17, 2013, 3:28 pm | 19 Comments

With Debt Study's Errors Confirmed, Debate on Conclusion Goes On

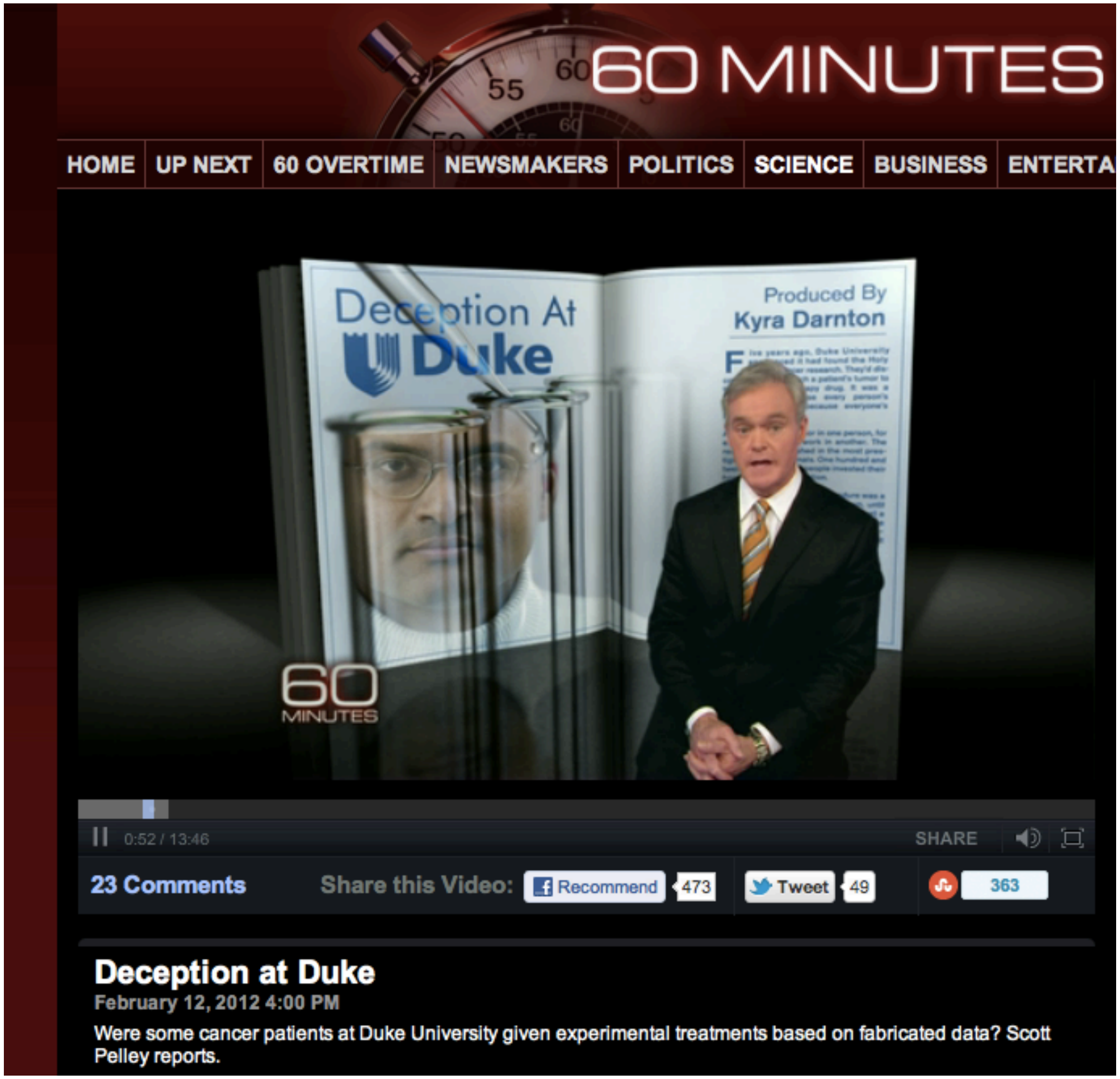
By ANNIE LOWREY

The Harvard economists Carmen M. Reinhart and Kenneth S. Rogoff have acknowledged that their groundbreaking 2010 study "Growth in a Time of Debt" includes statistical errors that significantly alter its results.

Recent Developments in Reproducible Research

The Duke Saga

http://goo.gl/hijaN



Deception at Duke

February 12, 2012 4:00 PM

Were some cancer patients at Duke University given experimental treatments based on fabricated data? Scott Pelley reports.

Recent Developments in Reproducible Research

REPORT BRIEF MARCH 2012

INSTITUTE OF MEDICINE
OF THE NATIONAL ACADEMIES

Advising the nation • Improving health

For more information visit www.iom.edu/translationalomics

Evolution of Translational Omics
Lessons Learned and the Path Forward

EVOLUTION OF TRANSLATIONAL
OMICS
Lessons Learned and the Path Forward

http://goo.gl/ZllJa

The IOM Report

In the Discovery/Test Validation stage of omics-based tests:

- **Data/metadata** used to develop test should be made publicly available
- The **computer code** and fully specified computational procedures used for development of the candidate omics-based test should be made sustainably available
- “Ideally, the computer code that is released will **encompass all of the steps of computational analysis**, including all data preprocessing steps, that have been described in this chapter. All aspects of the analysis need to be transparently reported.”

When Can Research be Reproducible?

- Analytic data are available
- Analytic code are available
- Standard means of distribution
- Documentation of code and data

Literate (Statistical) Programming

- An article is a stream of **text** and **code**
- Analysis code is divided into text and code "chunks"
- Each code chunk loads data and computes results
- Presentation code formats results (tables, figures, etc.)
- Article text explains what is going on
- Literate programs can be **weaved** to produce human-readable documents and **tangled** to produce machine-readable documents

Literate (Statistical) Programming

- Literate programming is a general concept that requires
  1. A documentation language (human readable)
  2. A programming language (machine readable)
- Sweave uses $\LaTeX$ and R as the documentation and programming languages
- Sweave was developed by Friedrich Leisch (member of the R Core) and is maintained by R core
- Main web site: `http:// www.statistik.lmu.de/ ˜leisch/Sweave`

Sweave Limitations

- Sweave has many limitations
- Focused primarily on LaTeX, a difficult to learn markup language used only by weirdos
- Lacks features like caching, multiple plots per chunk, mixing programming languages and many other technical items
- Not frequently updated or very actively developed

Literate (Statistical) Programming

- **knitr** is an alternative (more recent) package
- Brings together many features added on to Sweave to address limitations
- knitr uses R as the programming language (although others are allowed) and variety of documentation languages
  - LaTeX, Markdown, HTML
- knitr was developed by Yihui Xie (graduate student in statistics at Iowa State)
- See http://yihui.name/knitr/

What is LSP good for?

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

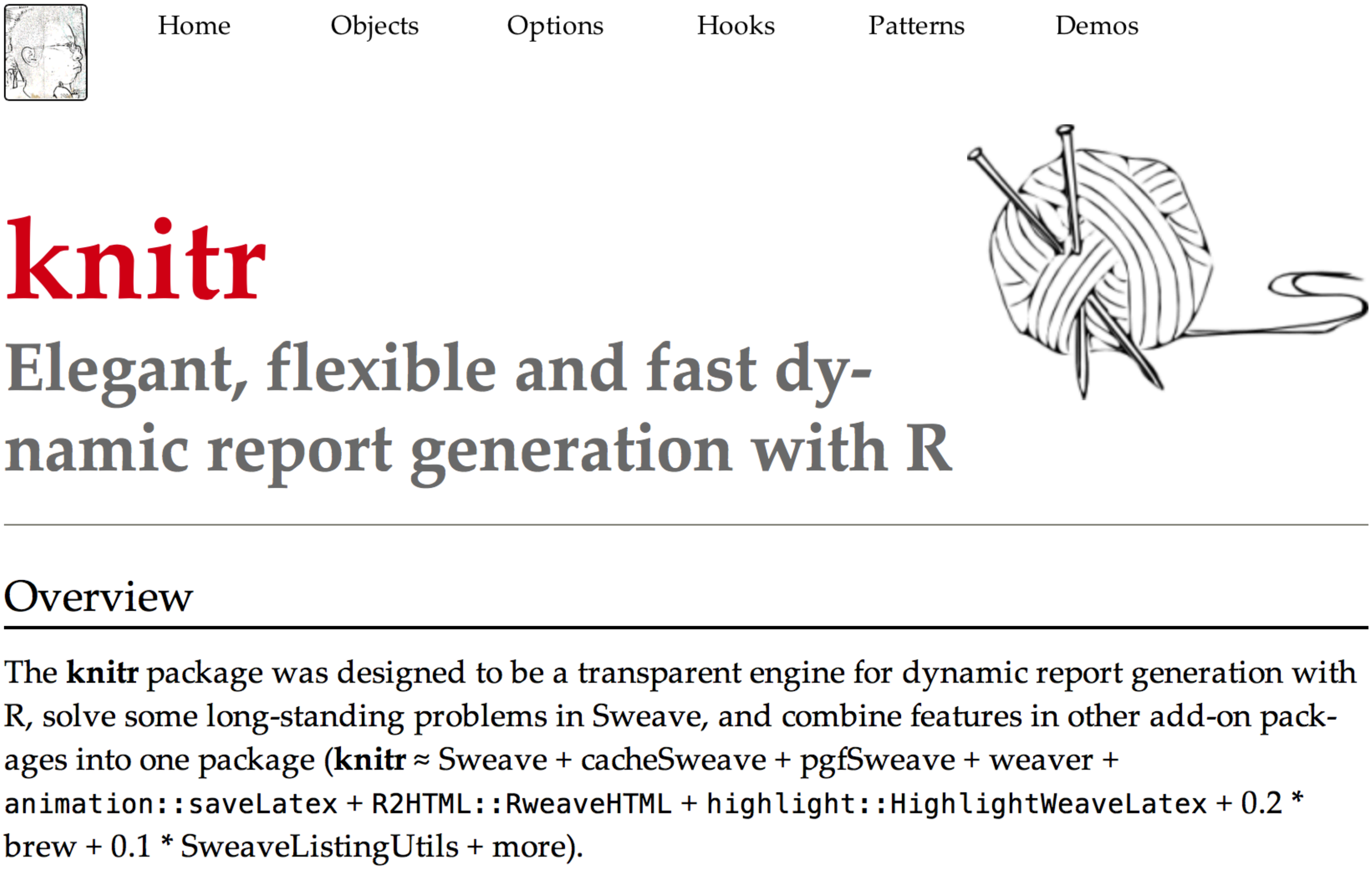
What is LSP NOT good for?

- Long research articles
- Complex time-consuming computations (i.e. long MCMC, optimizations)
- Documents that require precise formatting
- Books (although can be done with use of other tools)

Using knitr

- Toolchain
  - Text editor
  - R
  - knitr package (from CRAN via `install.packages`)
  - An appropriate document viewer (PDF viewer, web browser)
  - pandoc (optional)
- Easiest to use RStudio where knitr (and Sweave) are already integrated

knitr



knitr

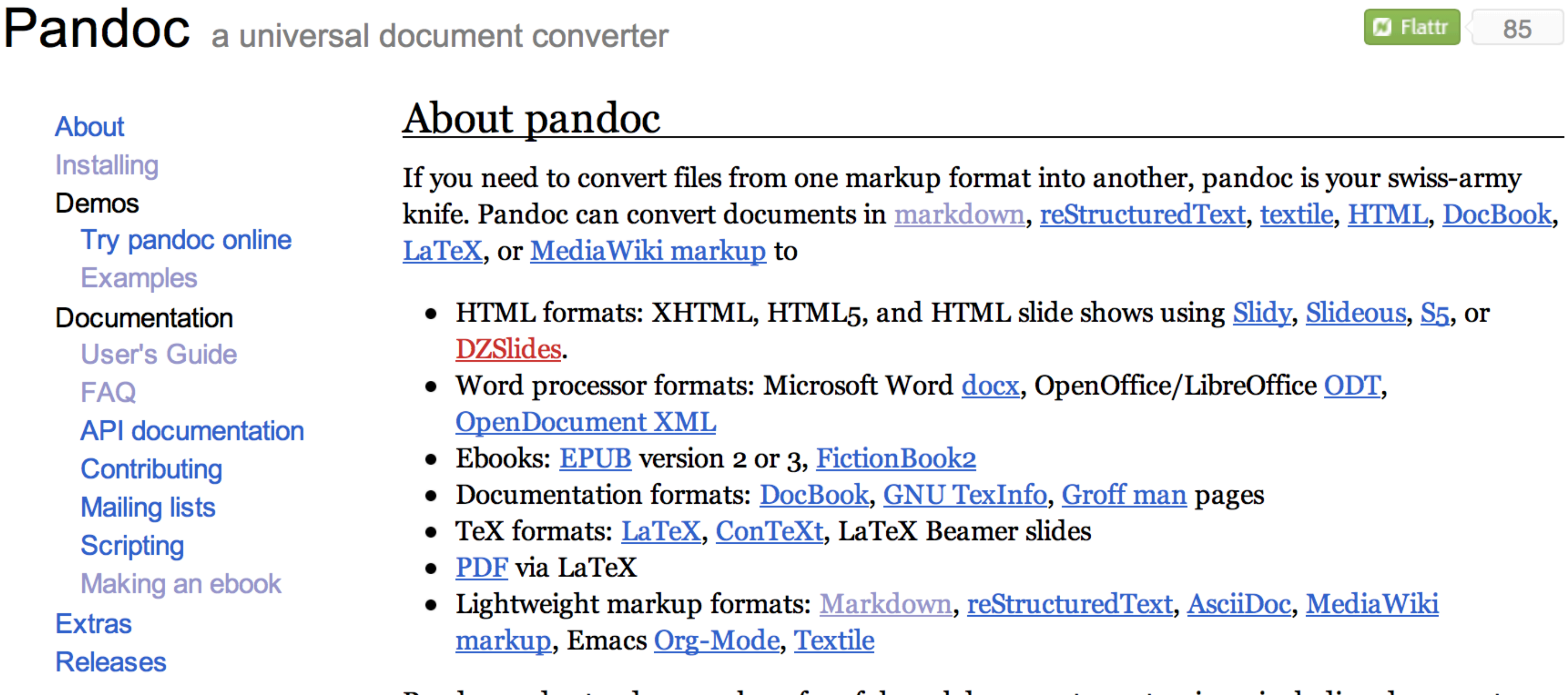
Elegant, flexible and fast dynamic report generation with R

Overview

The **knitr** package was designed to be a transparent engine for dynamic report generation with R, solve some long-standing problems in Sweave, and combine features in other add-on packages into one package (**knitr** ≈ Sweave + cacheSweave + pgfSweave + weaver + `animation::saveLatex` + `R2HTML::RweaveHTML` + `highlight::HighlightWeaveLatex` + 0.2 * brew + 0.1 * SweaveListingUtils + more).

http://yihui.name/knitr

Pandoc



Pandoc a universal document converter

- About
- Installing
- Demos
  - Try pandoc online
  - Examples
- Documentation
  - User's Guide
  - FAQ
  - API documentation
  - Contributing
  - Mailing lists
  - Scripting
  - Making an ebook
- Extras
- Releases

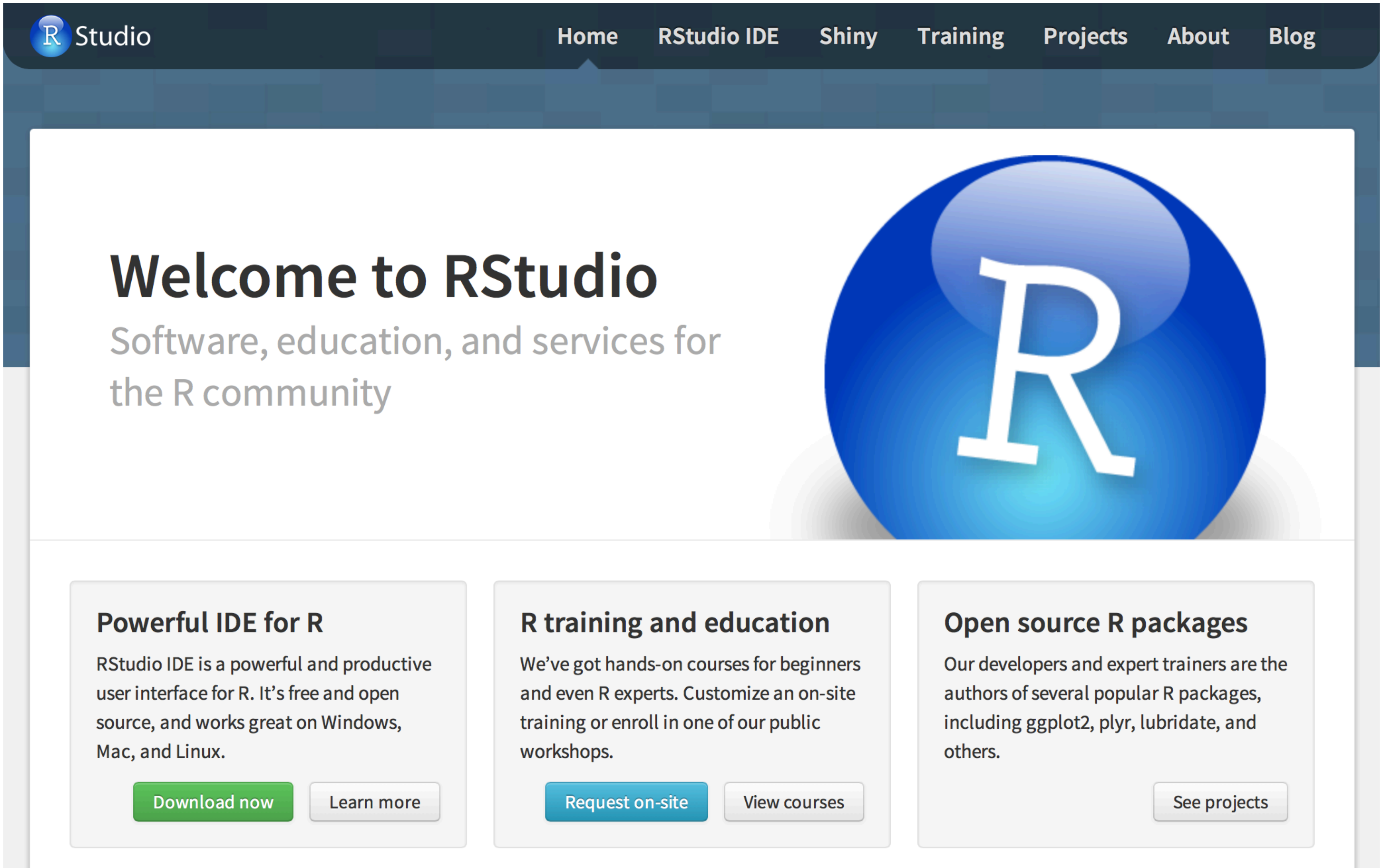
About pandoc

If you need to convert files from one markup format into another, pandoc is your swiss-army knife. Pandoc can convert documents in markdown, reStructuredText, textile, HTML, DocBook, LaTeX, or MediaWiki markup to

- HTML formats: XHTML, HTML5, and HTML slide shows using Slidy, Slideous, S5, or DZSlides.
- Word processor formats: Microsoft Word docx, OpenOffice/LibreOffice ODT, OpenDocument XML
- Ebooks: EPUB version 2 or 3, FictionBook2
- Documentation formats: DocBook, GNU TexInfo, Groff man pages
- TeX formats: LaTeX, ConTeXt, LaTeX Beamer slides
- PDF via LaTeX
- Lightweight markup formats: Markdown, reStructuredText, AsciiDoc, MediaWiki markup, Emacs Org-Mode, Textile

http://johnmacfarlane.net/pandoc/

RStudio



Welcome to RStudio

Software, education, and services for the R community

Powerful IDE for R

RStudio IDE is a powerful and productive user interface for R. It's free and open source, and works great on Windows, Mac, and Linux.

R training and education

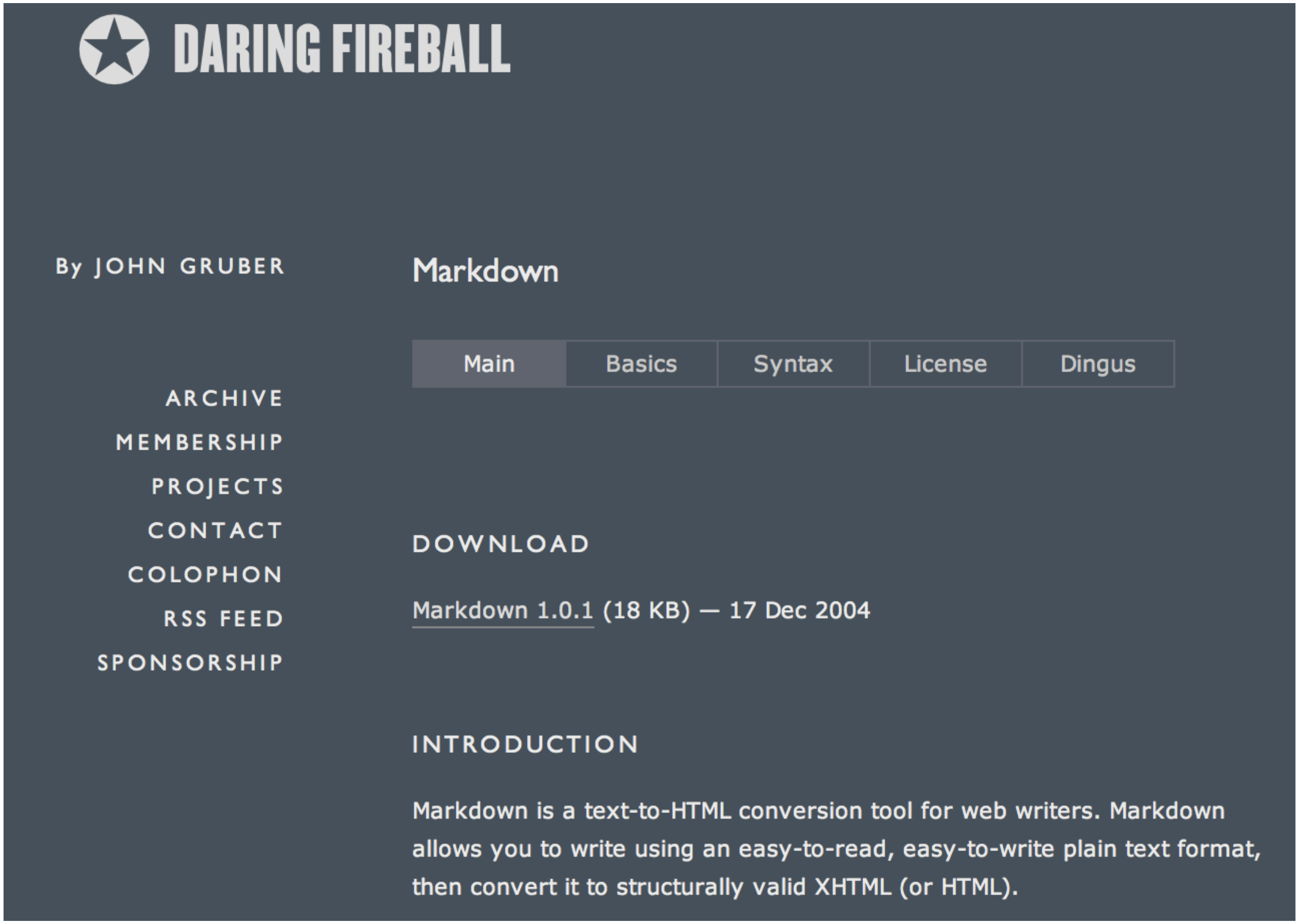
We've got hands-on courses for beginners and even R experts. Customize an on-site training or enroll in one of our public workshops.

Open source R packages

Our developers and expert trainers are the authors of several popular R packages, including ggplot2, plyr, lubridate, and others.

http://www.rstudio.com

Markdown



Markdown

DOWNLOAD

Markdown 1.0.1 (18 KB) — 17 Dec 2004

INTRODUCTION

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).

http://daringfireball.net/projects/markdown/

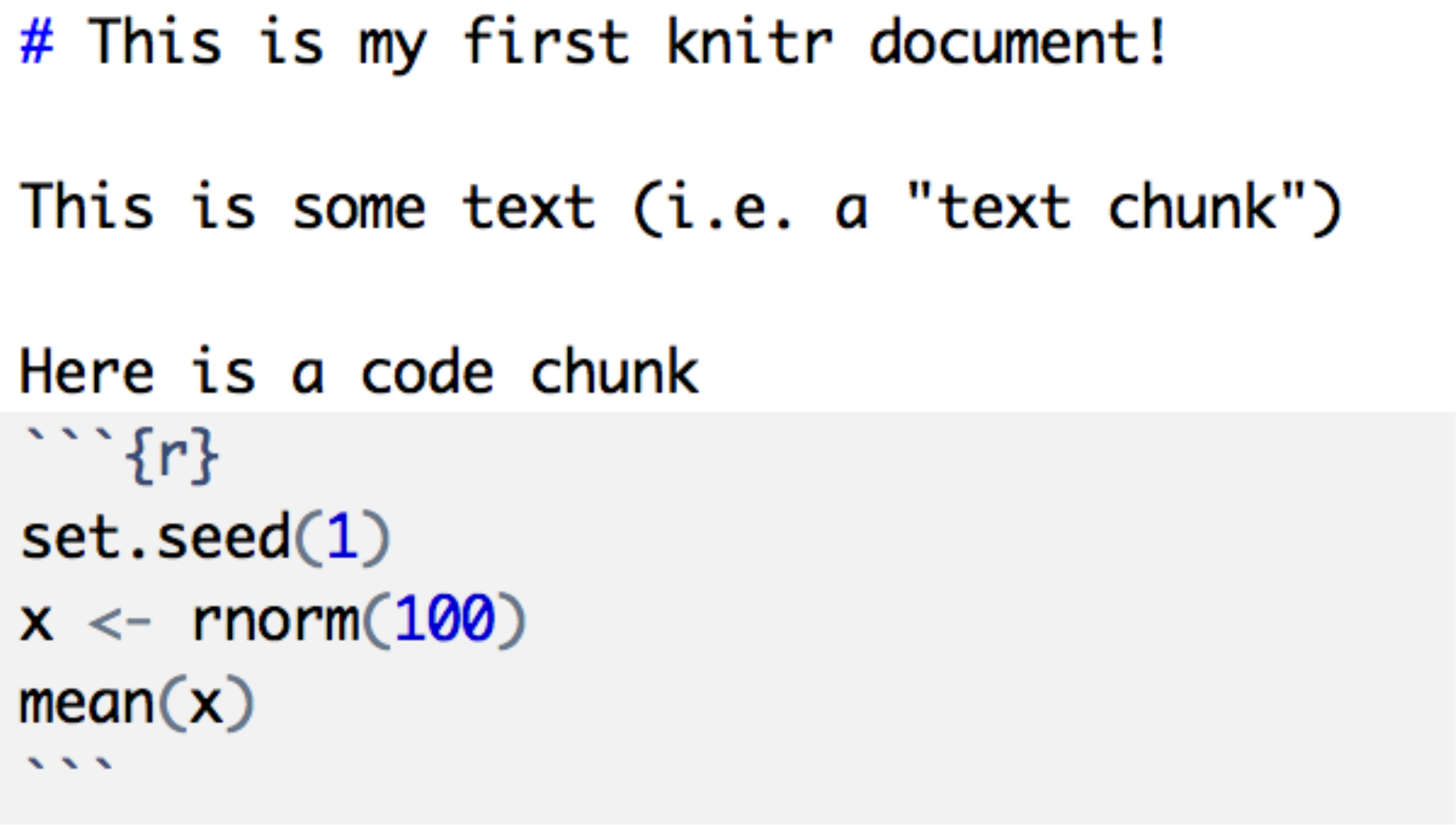
Markdown

- A simpler version of “markup” languages
- Written in simple text format
- Has minimal formatting indicators
- Tools support converting it to many other formats (pandoc)

Markdown

- # indicates a first-level header
- ## second-level header
- ### third-level header
- Indentation can be used to create an ordered (with numbers) or unordered (with non-numbers) list

Your First R Markdown Document

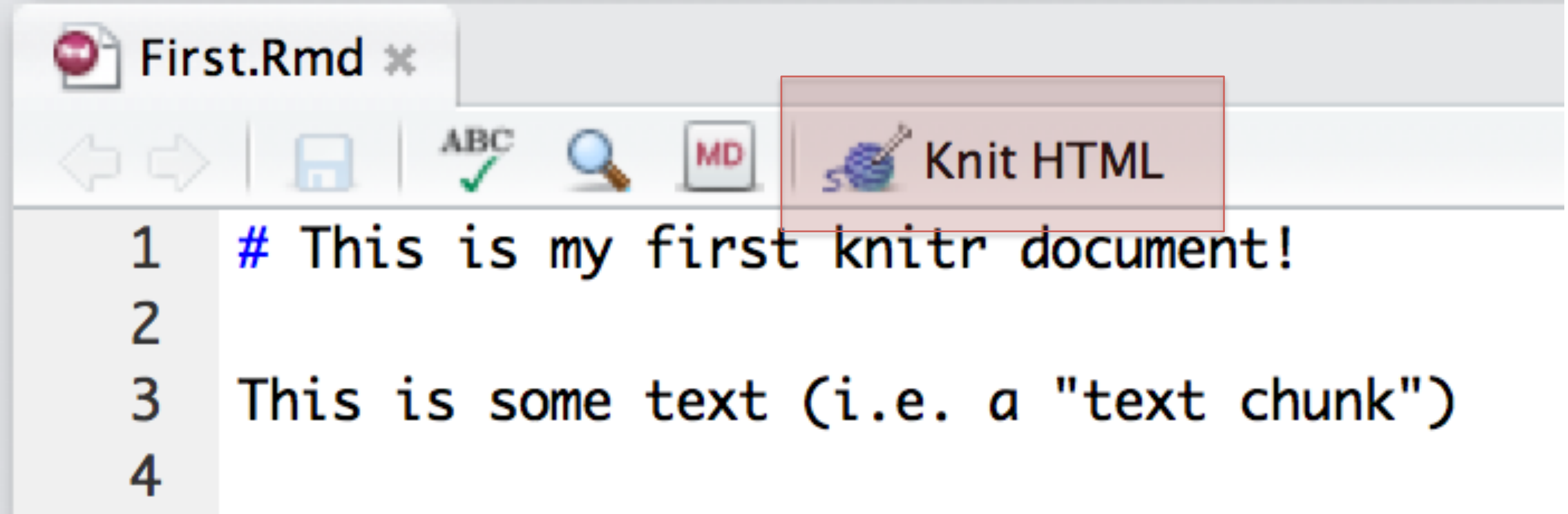


~~~~
# This is my first knitr document!

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

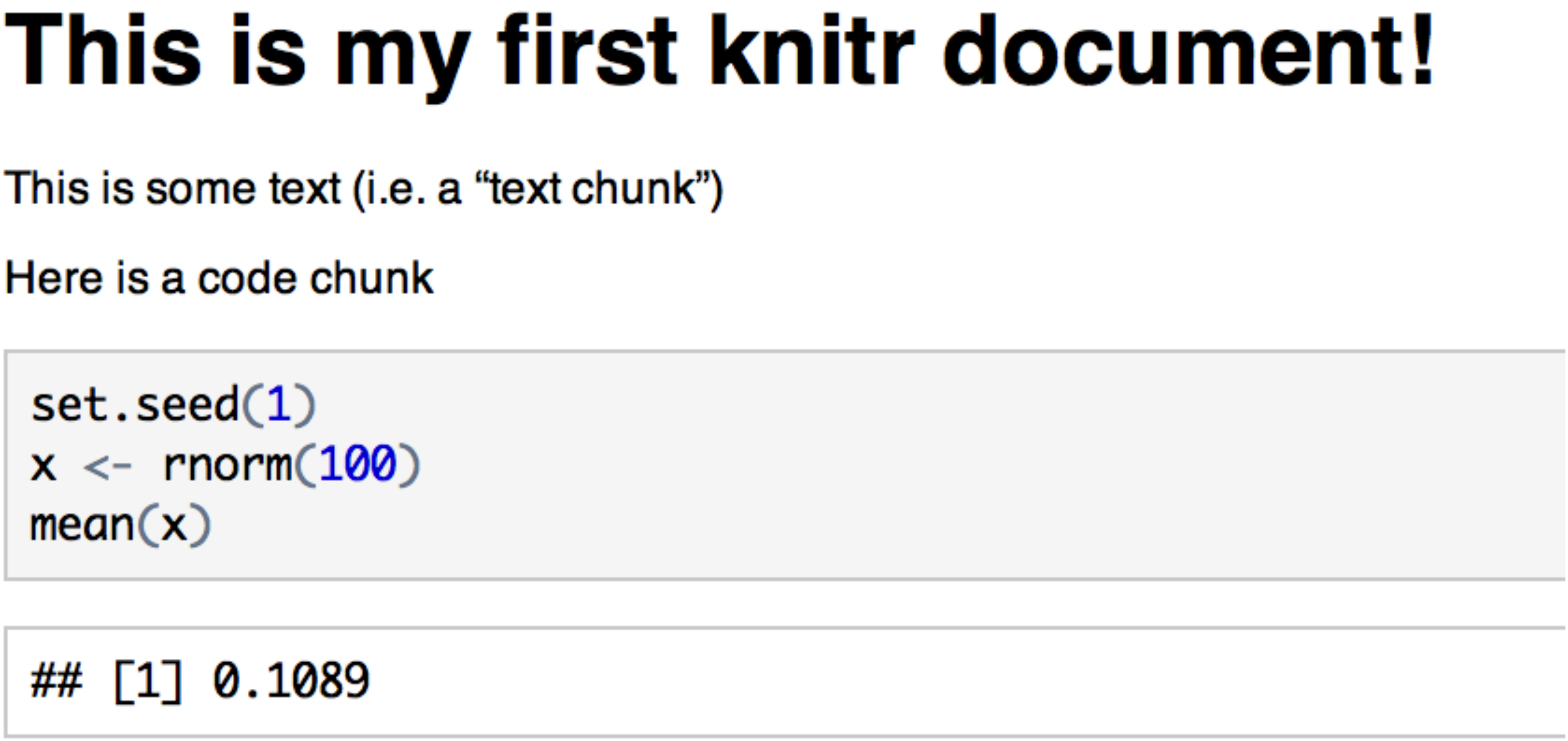
Here is a code chunk
```{r}
set.seed(1)
x <- rnorm(100)
mean(x)
```
~~~~

Processing R Markdown



```
# This is my first knitr document!

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

```

Resulting HTML Document



This is my first knitr document!

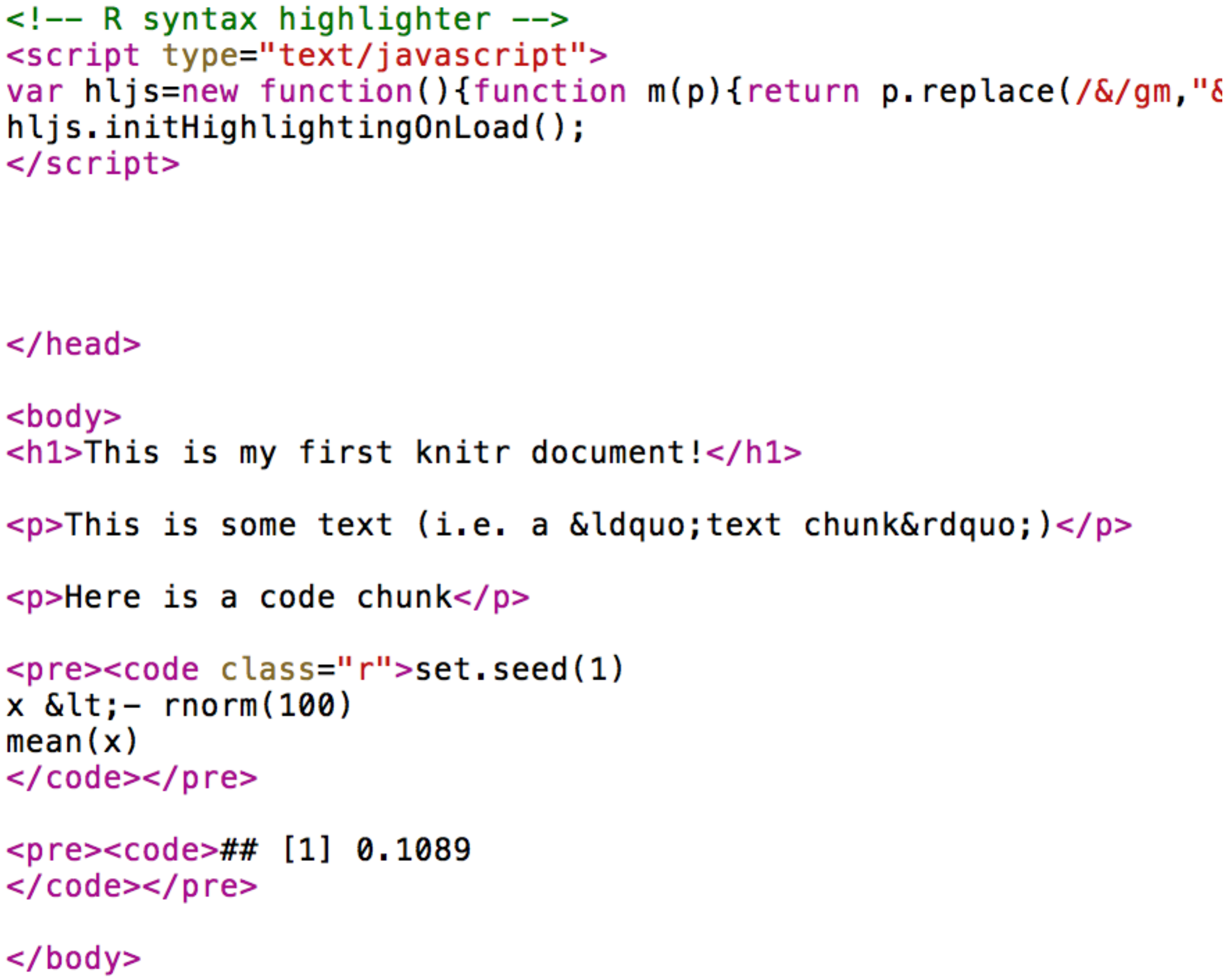
This is some text (i.e. a “text chunk”)

Here is a code chunk

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

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

Under the Hood



```
<!-- R syntax highlighter -->
<script type="text/javascript">
var hljs=new function(){function m(p){return p.replace(/&/gm,"&
hljs.initHighlightingOnLoad();
</script>




</head>

<body>
<h1>This is my first knitr document!</h1>

<p>This is some text (i.e. a &ldquo;text chunk&rdquo;)</p>

<p>Here is a code chunk</p>

<pre><code class="r">set.seed(1)
x &lt;- rnorm(100)
mean(x)
</code></pre>

<pre><code>## [1] 0.1089
</code></pre>

</body>
```

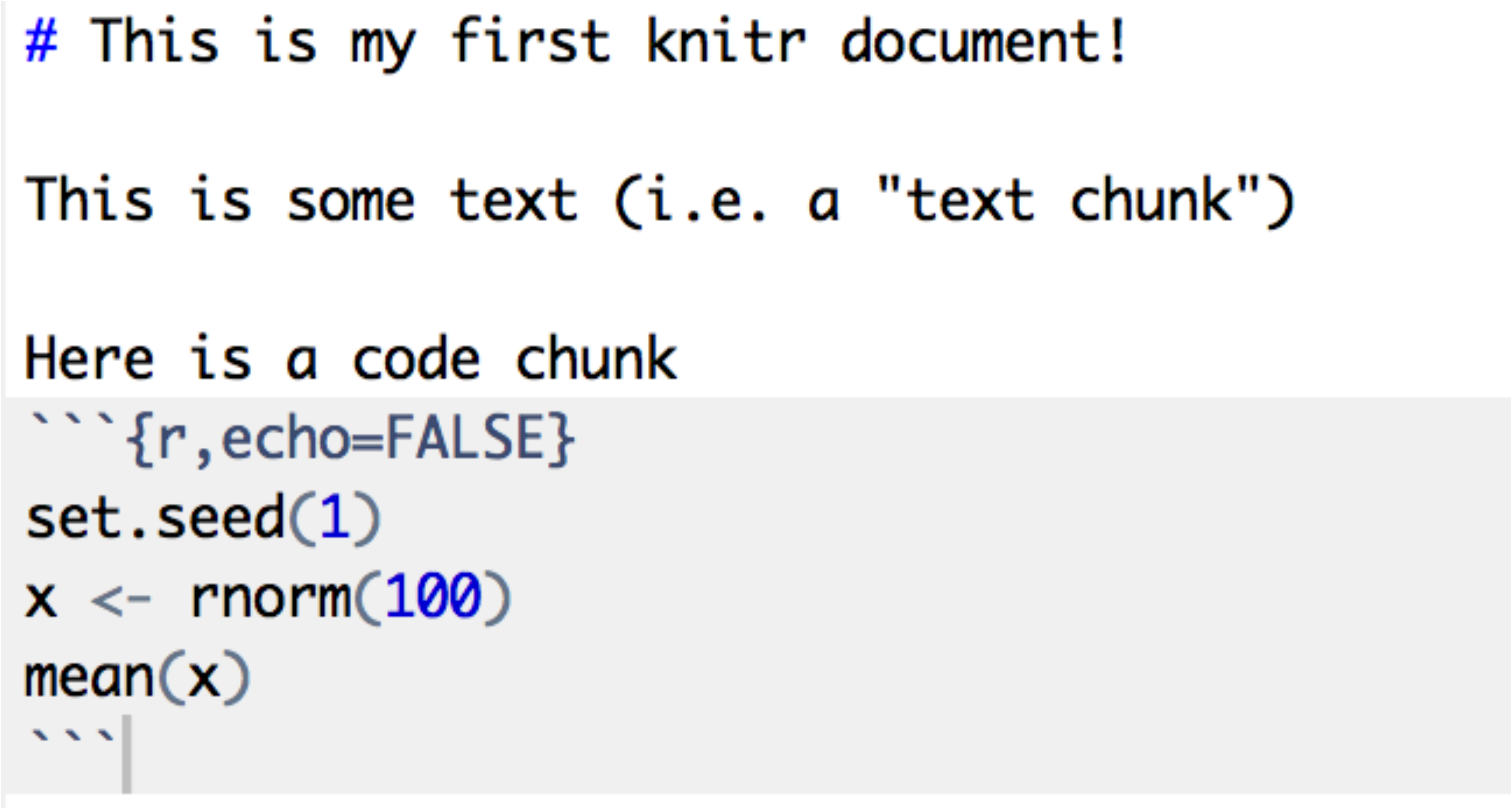
A Few Notes

- Code chunks begin with ```{r} on a line by itself and end with ``` on a line by itself
  - All R code goes in between
- Code chunks can have names, which is useful when making plots
- By default, code in a code chunk will be **echoed**, as will the **result** of the computation (if there is anything to print)

Processing R Markdown Documents

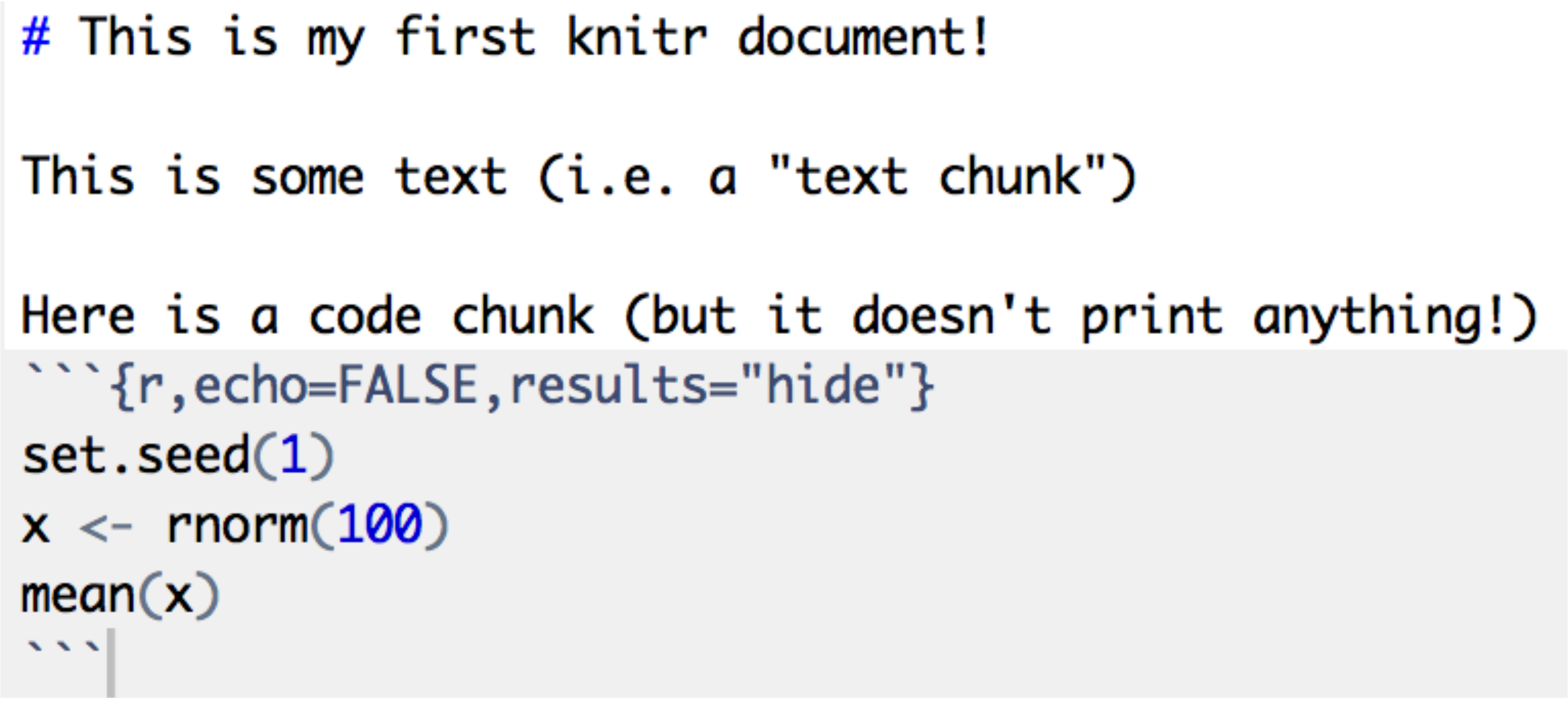
- The order is
  - First.Rmd (written by you!)
  - First.md (created by knitr)
  - First.html (created by knitr)
- The .md file (and the .html file) are not things we care about and **should not be edited**
- Always edit the .Rmd file

Chunk Options



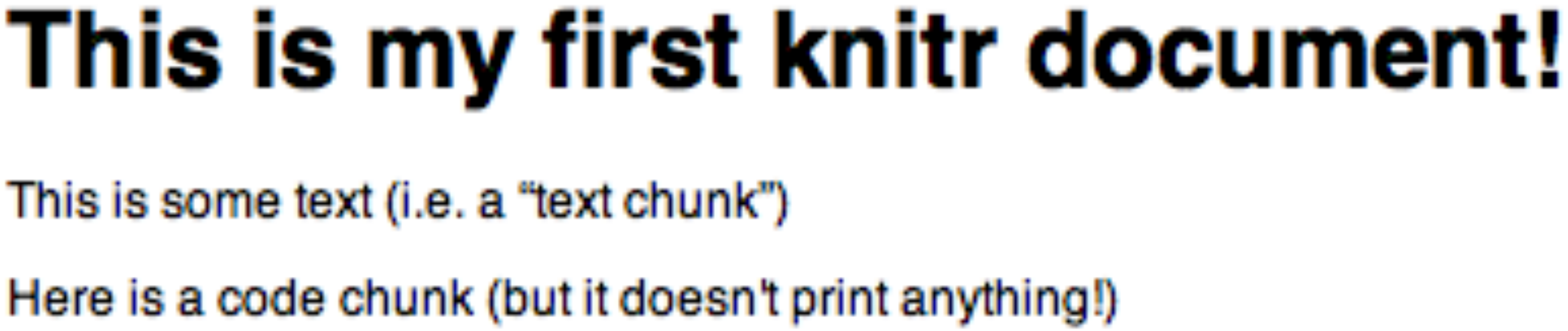
~~~
# This is my first knitr document!

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

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

Chunk Options



~~~
# This is my first knitr document!

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

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

Chunk Options

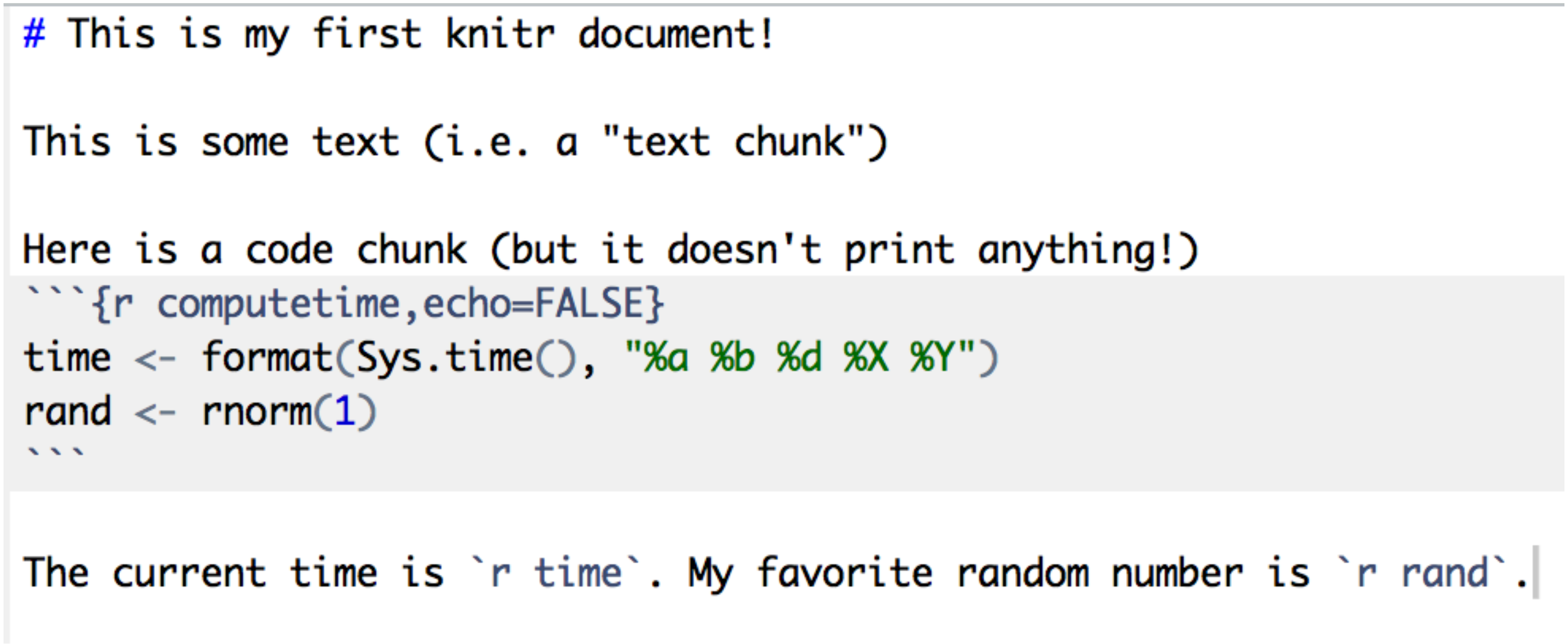


This is my first knitr document!

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

Here is a code chunk (but it doesn't print anything!)

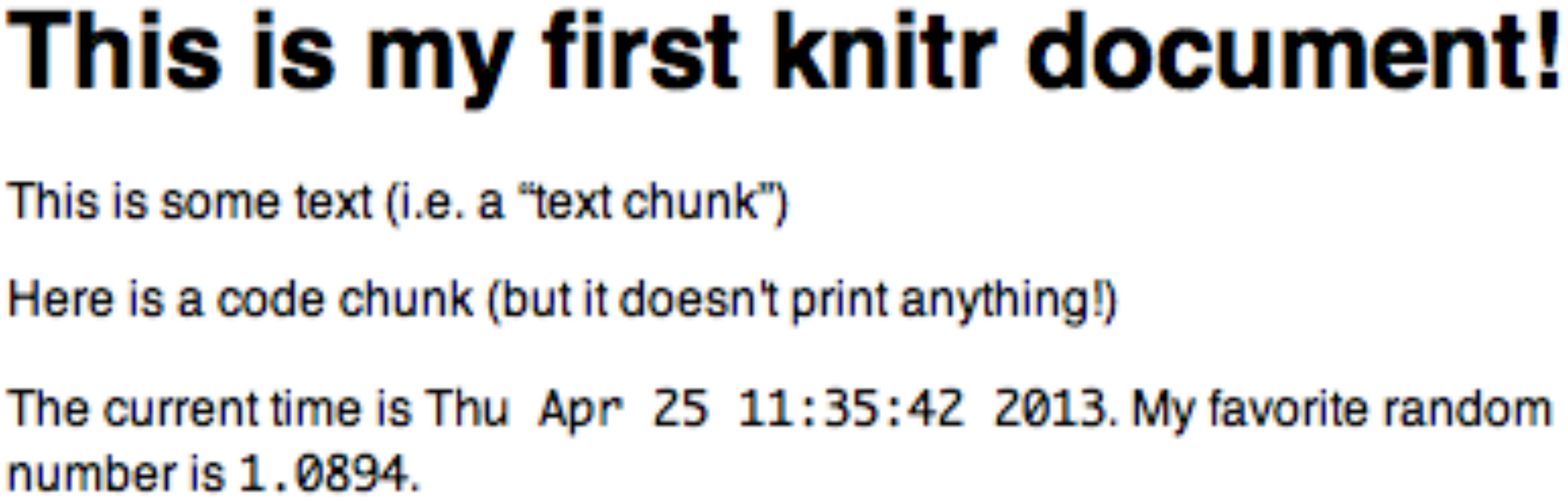
Inline Computation



~~~
# This is my first knitr document!

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

Here is a code chunk (but it doesn't print anything!)
```{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`.
~~~

(NOTE: Look at ?strptime for a list of all the date/time format codes.)

Inline Computation



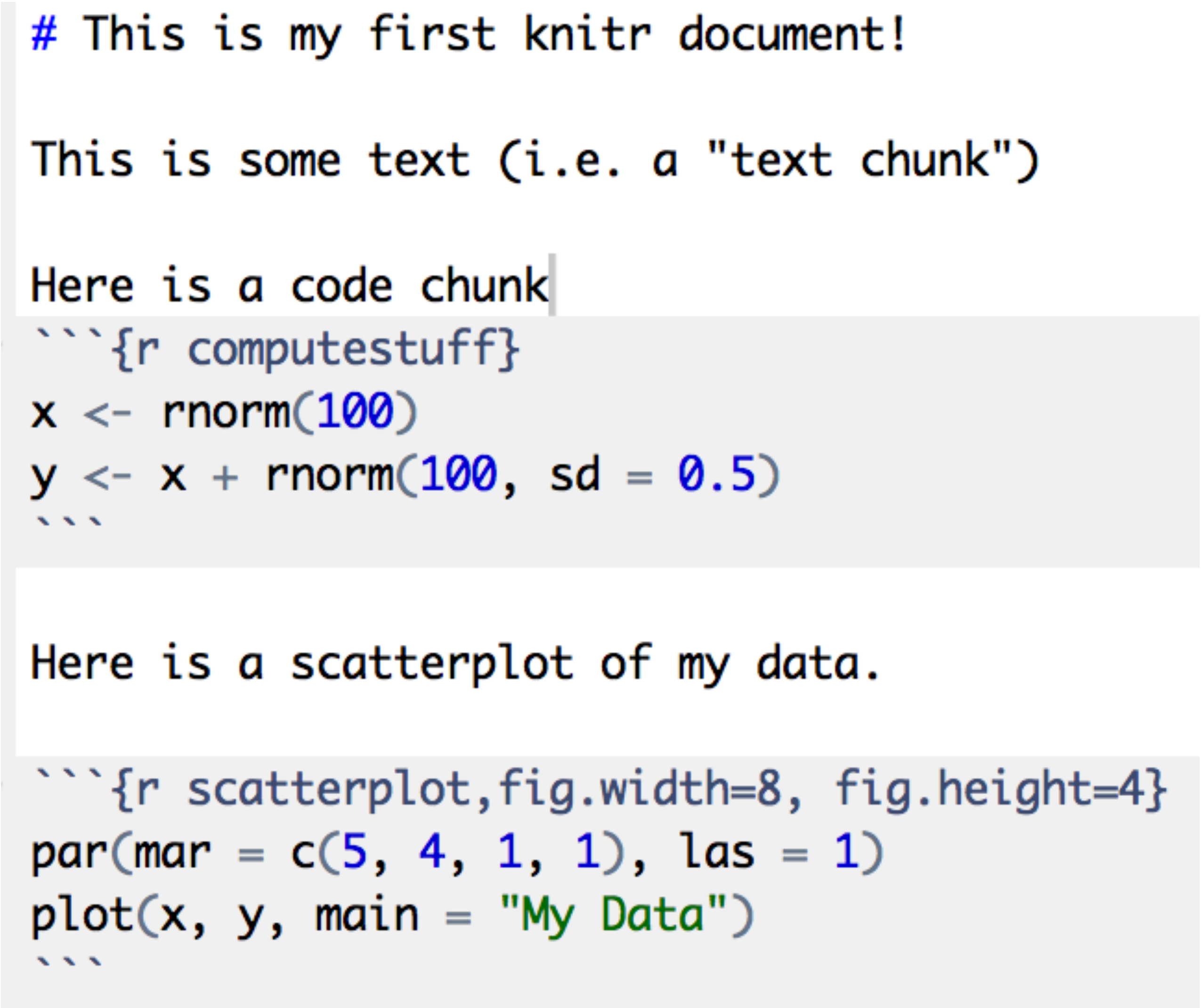
This is my first knitr document!

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

Here is a code chunk (but it doesn't print anything!)

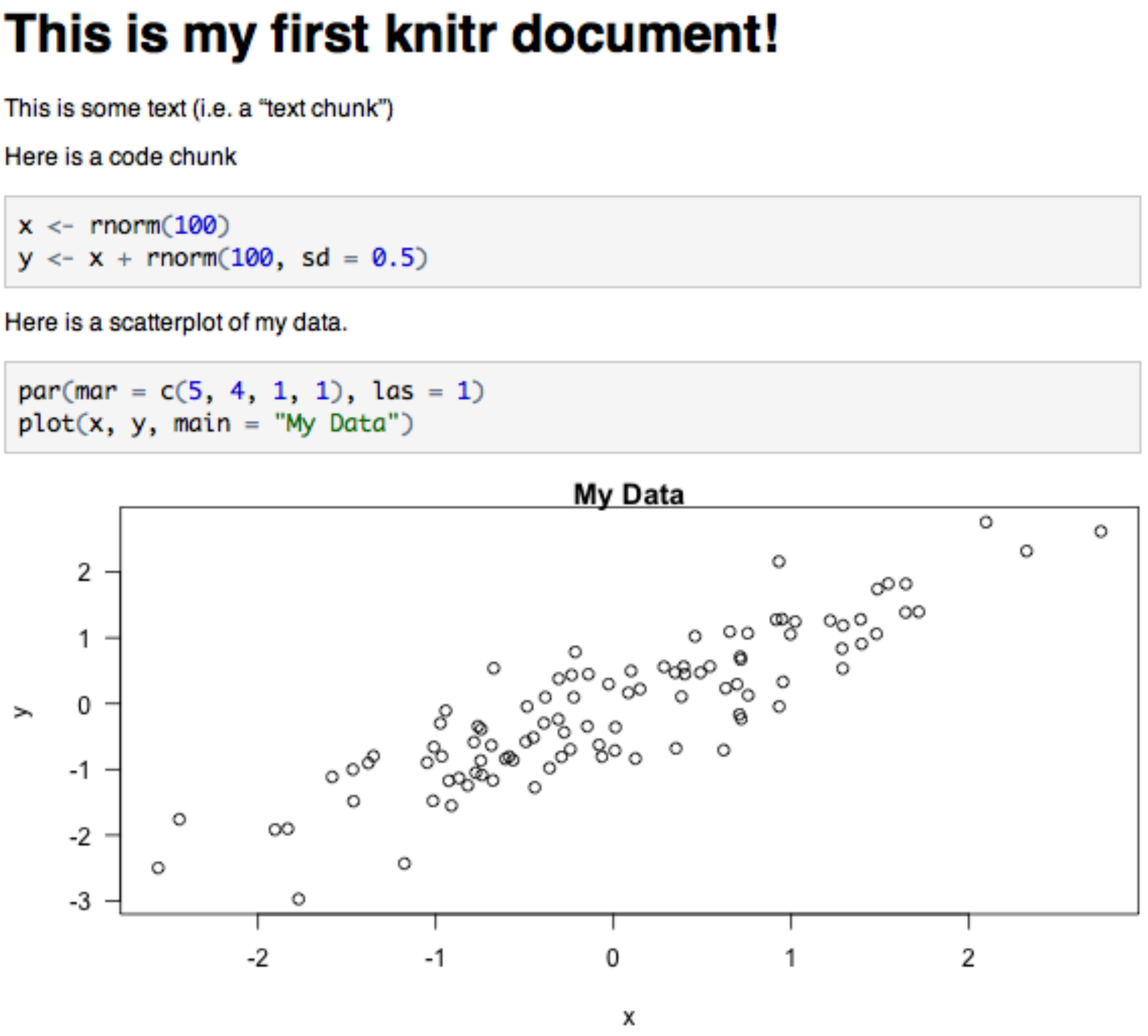
The current time is Thu `Apr 25 11:35:42 2013`. My favorite random number is `1.0894`.

Graphics



~~~
# This is my first knitr document!

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

Here is a code chunk
```{r computestuff}
x <- rnorm(100)
y <- x + rnorm(100, sd = 0.5)
```

Here is a scatterplot of my data.

```{r scatterplot,fig.width=8, fig.height=4}
par(mar = c(5, 4, 1, 1), las = 1)
plot(x, y, main = "My Data")
```
~~~

Graphics



This is my first knitr document!

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

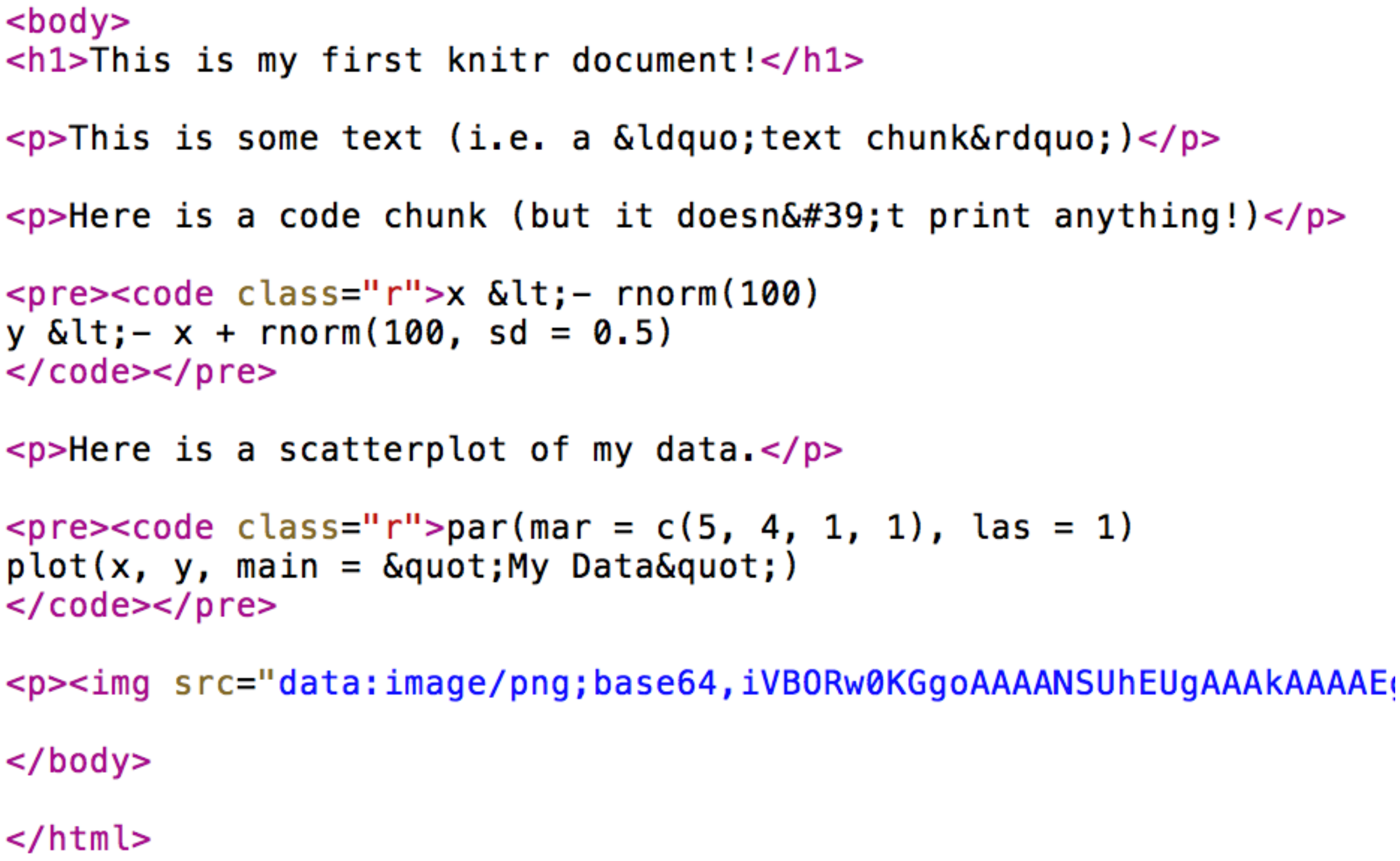
Here is a code chunk

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

Here is a scatterplot of my data.

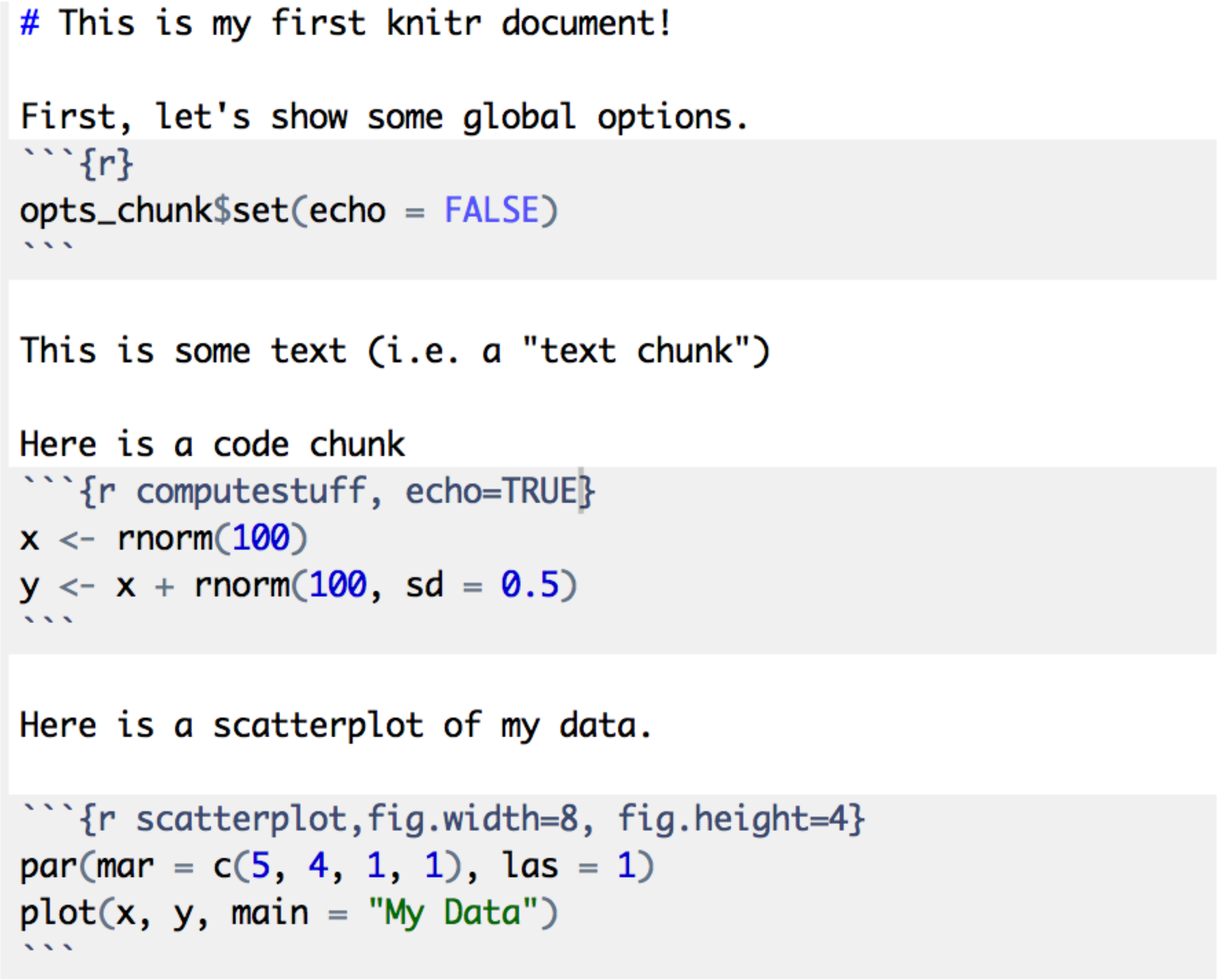
```
par(mar = c(5, 4, 1, 1), las = 1)
plot(x, y, main = "My Data")
```

Graphics: What knitr Produces



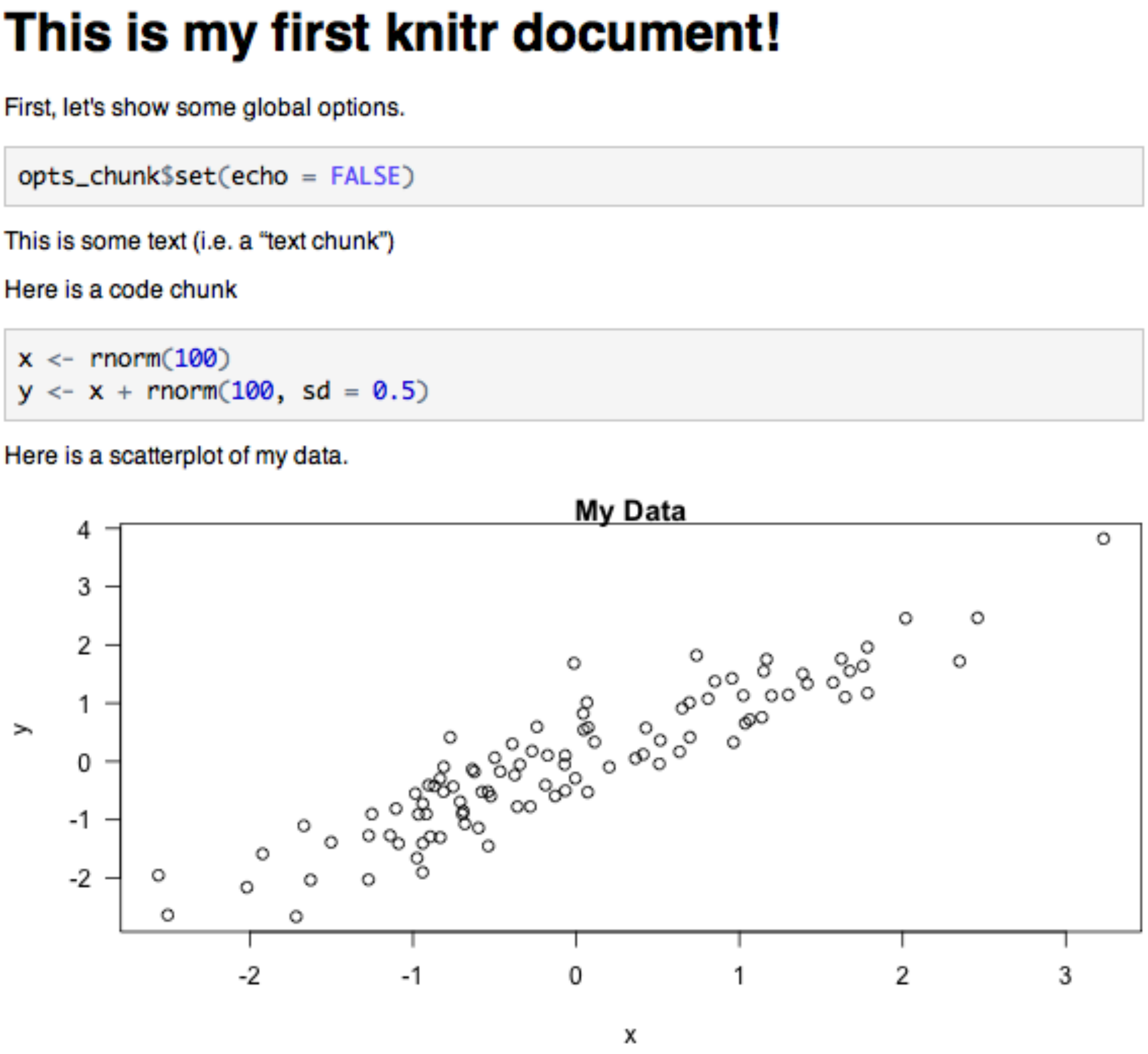
```
<body>
<h1>This is my first knitr document!</h1>

<p>This is some text (i.e. a &ldquo;text chunk&rdquo;)</p>

<p>Here is a code chunk (but it doesn&#39;t print anything!)</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 my data.</p>

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

<p><img src="data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAkAAAAE

</body>

</html>
```

Setting Global Options



~~~
# This is my first knitr document!

First, let's show some global options.
```{r}
opts_chunk$set(echo = FALSE)
```

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

Here is a code chunk
```{r computestuff, echo=TRUE}
x <- rnorm(100)
y <- x + rnorm(100, sd = 0.5)
```

Here is a scatterplot of my data.

```{r scatterplot,fig.width=8, fig.height=4}
par(mar = c(5, 4, 1, 1), las = 1)
plot(x, y, main = "My Data")
```
~~~

Setting Global Options



This is my first knitr document!

First, let's show some global options.

```
opts_chunk$set(echo = FALSE)
```

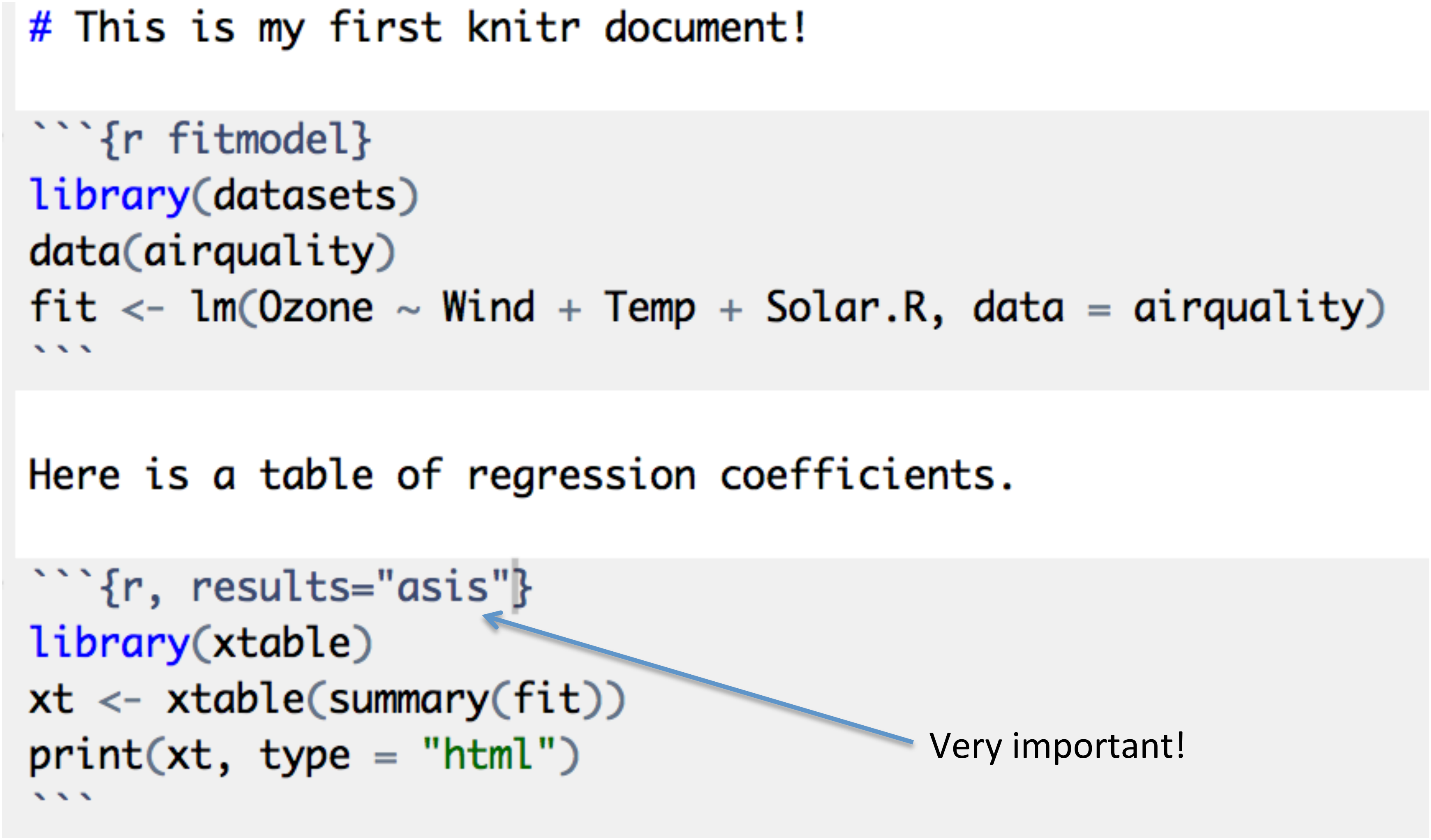
This is some text (i.e. a “text chunk”)

Here is a code chunk

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

Here is a scatterplot of my data.

Making Tables with xtable

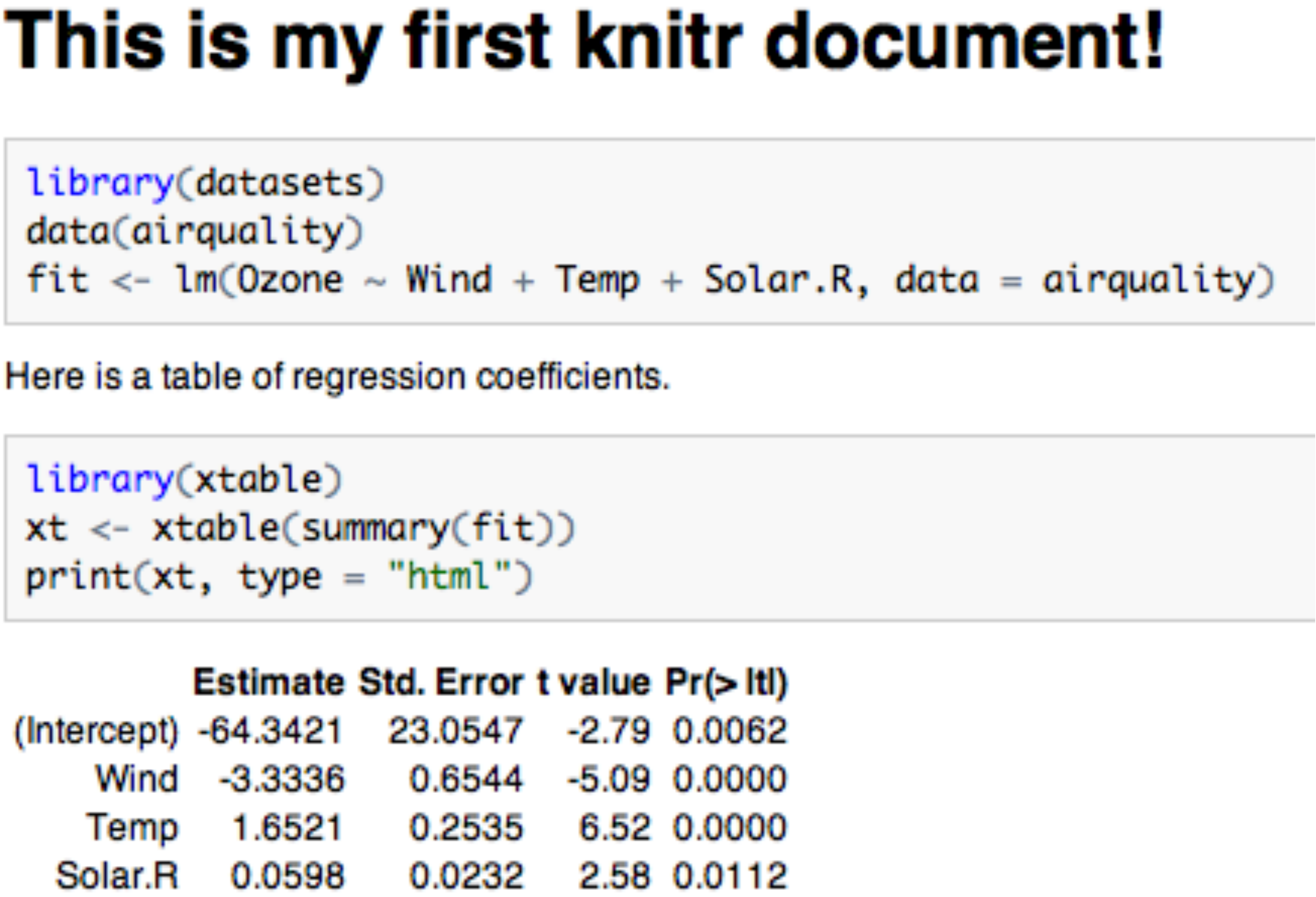


~~~
# This is my first knitr document!

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

Here is a table of regression coefficients.

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

Very important!
```

# Making Tables with xtable



## This is my first knitr document!

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

Here is a table of regression coefficients.

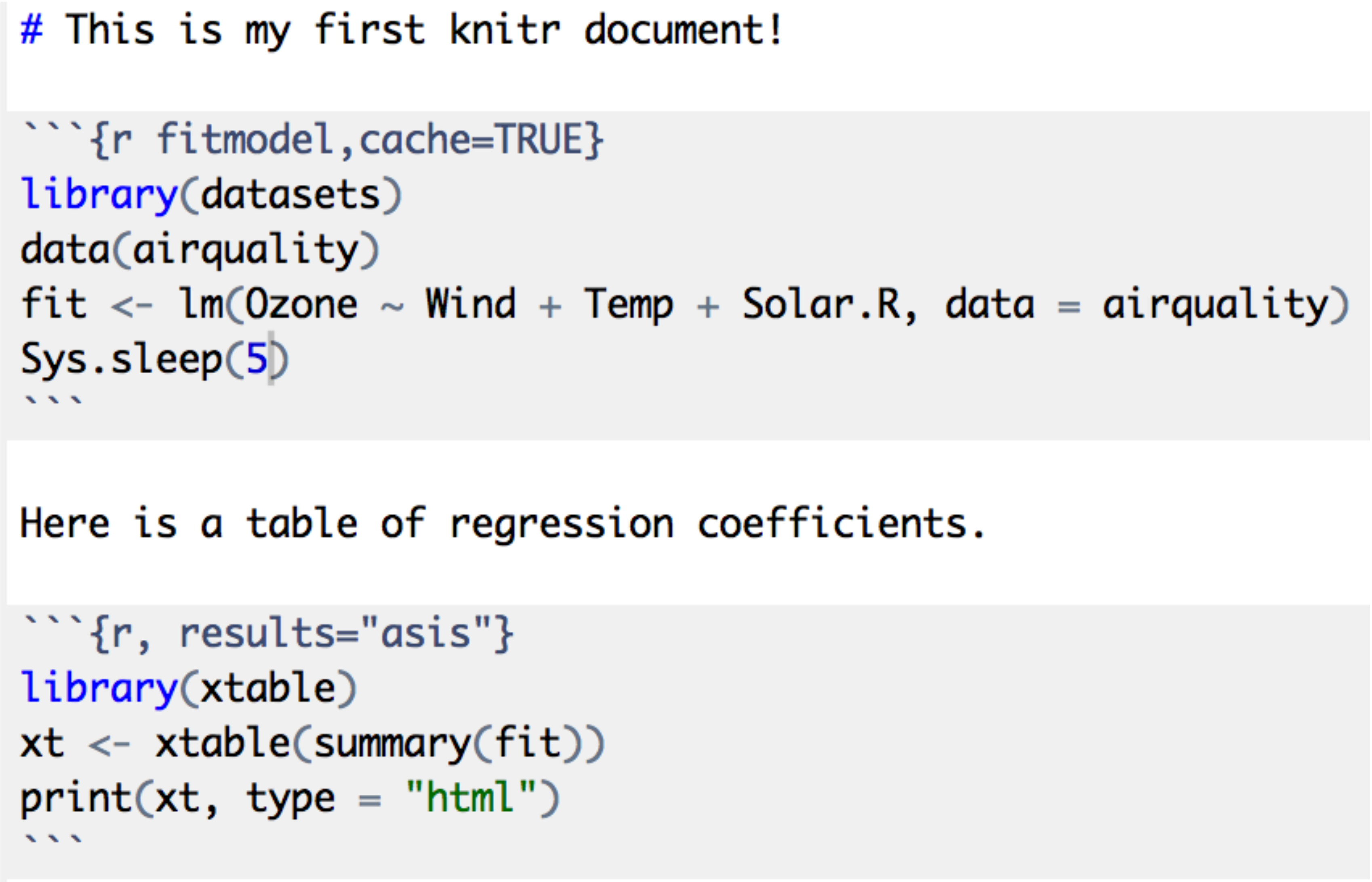
```r
library(xtable)
xt <- xtable(summary(fit))
print(xt, type = "html")
```

	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

Caching Long Computations

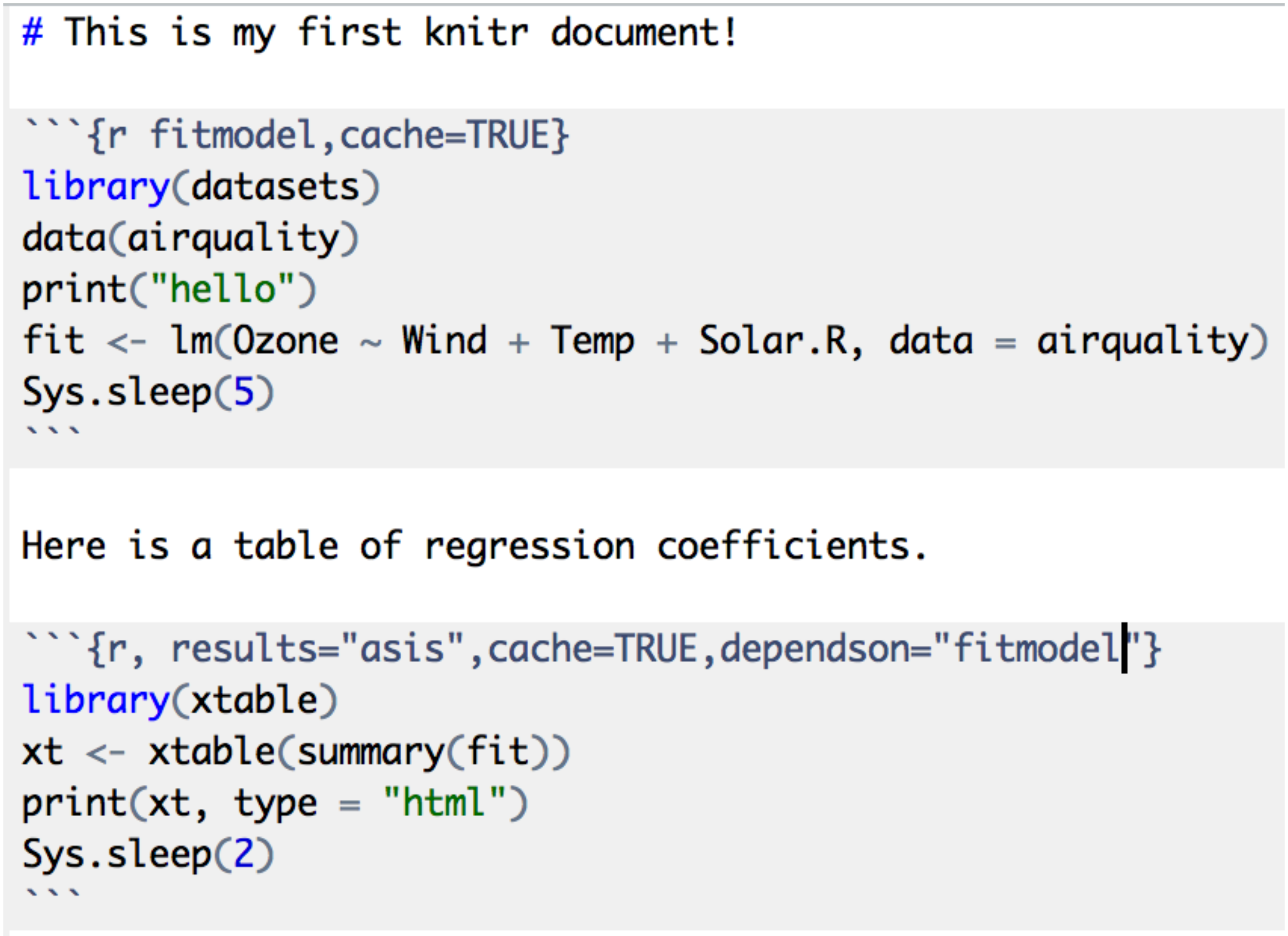
- Important for long documents with complex computations
- Set ‘cache = TRUE’ in chunk option to cache results
- Code has to be run once; results are then stored in a key-value database
- On subsequent runs, results are loaded from database rather than execute code
- If code changes, results are re-run
- Dependencies can be specified via ‘dependson’

Caching Long Computations



~~~
# This is my first knitr document!

```{r fitmodel,cache=TRUE}
library(datasets)
data(airquality)
fit <- lm(Ozone ~ Wind + Temp + Solar.R, data = airquality)
Sys.sleep(5)
```

Here is a table of regression coefficients.

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

Caching Long Computations



~~~
# This is my first knitr document!

```{r fitmodel,cache=TRUE}
library(datasets)
data(airquality)
print("hello")
fit <- lm(Ozone ~ Wind + Temp + Solar.R, data = airquality)
Sys.sleep(5)
```

Here is a table of regression coefficients.

```{r, results="asis",cache=TRUE,dependson="fitmodel"}
library(xtable)
xt <- xtable(summary(fit))
print(xt, type = "html")
Sys.sleep(2)
```
~~~

Summary of Basic Options

- Controlling Output
  - results: “markup”, “asis”, “hide”
  - echo: TRUE, FALSE
  - eval: TRUE, FALSE
- Figures
  - fig.width: width of plot (passed to graphics dev)
  - fig.height: height of plot
  - fig.path: directory to put figures (“figure/”)

Reproducibility: The Bigger Picture

- Reproducibility is not just about using individual tools
- Need a “reproducible workflow” that incorporates a number of other tools/practices
- Version control: git, svn, cvs (!)
- Software versioning, regression/unit testing

Thank you!