Tutorial: Methods for Reproducible Research

Roger D. Peng

Department Biostatistics
Johns Hopkins Bloomberg School of Public Health

ENAR 2009

Replication

The ultimate standard for strengthening scientific evidence is **replication** of findings and studies with independent

- multiple investigators
- data
- analytical methods
- laboratories
- instruments

Replication is particularly important in studies that can impact broad policy or regulatory decisions.

Reproducible Research

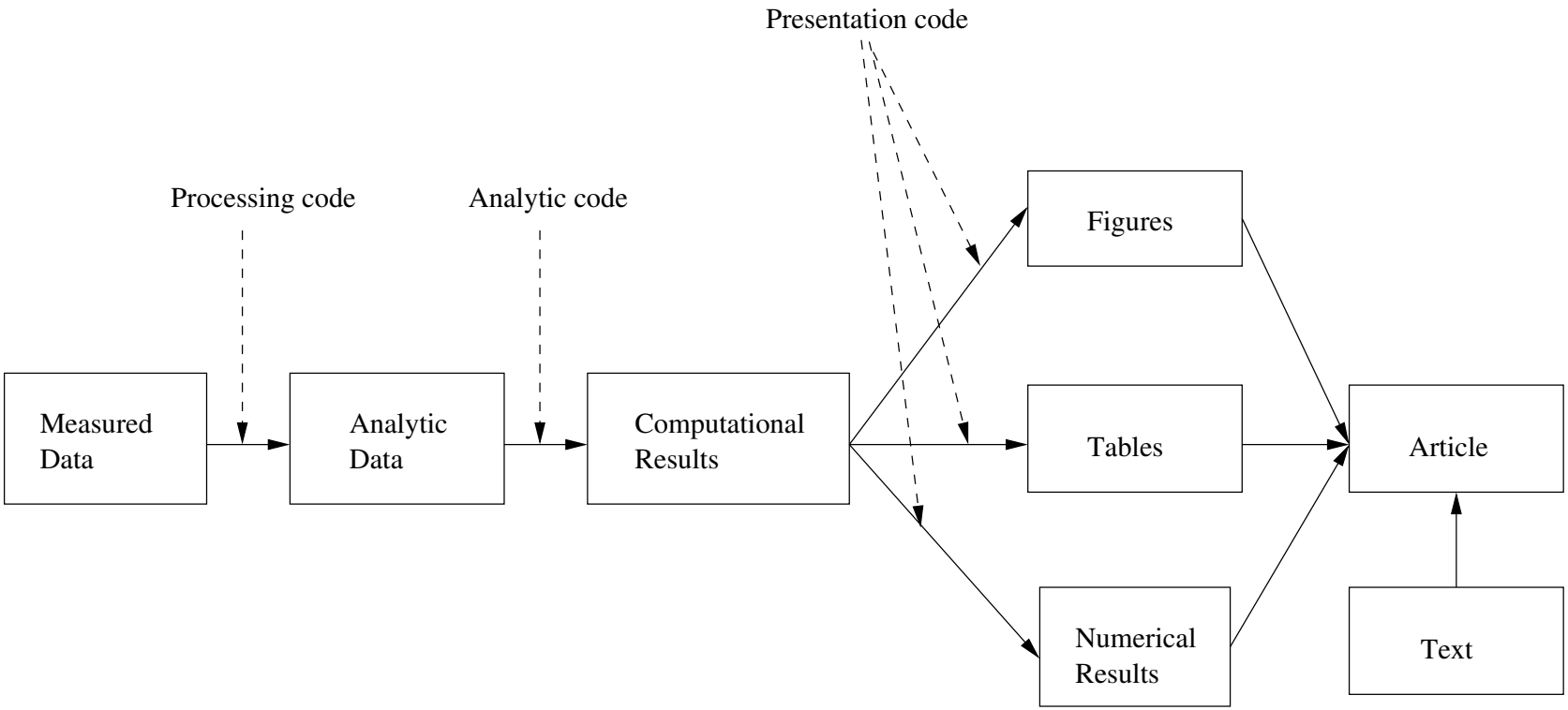
Why do we need reproducible research?

- Many studies cannot be replicated
  - No time
  - No money
  - Unique
- 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" (de Leeuw's Law)

Reproducible Research

Today, scientific papers published in journals represent the **advertising** of the research (Claerbout)

Research Pipeline: Model for Reproducible Research



Reproducible Research

What is this reproducible research?

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

Who are the Players?

Authors

- Want to make their research reproducible
- Want tools for RR to make their lives easier (or at least not much harder)

Readers

- Want to reproduce (and perhaps expand upon) interesting findings
- Want tools for RR to make their lives easier

Theory...

American Journal of Epidemiology

DOI: 10.1093/aje/

Commentary

Reproducible Epidemiologic Research

Roger D. Peng, Francesca Dominici, and Scott L. Zeger

From the Biostatistics Department, Johns Hopkins Bloomberg School of Public Health, Baltimore, MD.

Received for publication November 4, 2005; accepted for publication January 10, 2006.

The replication of important findings by multiple independent investigators is fundamental to the accumulation of scientific evidence. Researchers in the biologic and physical sciences expect results to be replicated by independent data, analytical methods, laboratories, and instruments. Epidemiologic studies are commonly used to quantify small health effects of important, but subtle, risk factors, and replication is of critical importance where results can inform substantial policy decisions. However, because of the time, expense, and opportunism of many current epidemiologic studies, it is often impossible to fully replicate their findings. An attainable minimum standard is "reproducibility," which calls for data sets and software to be made available for verifying published findings and conducting alternative analyses. The authors outline a standard for reproducibility and evaluate the reproducibility of current epidemiologic research. They also propose methods for reproducible research and implement them by use of a case study in air pollution and health.

...Methods?

Authors

- Just put stuff on the web
- Journal supplementary materials
- There are some central databases for various fields (e.g. biology, ICPSR)

Readers

- Just download the data and figure it out
- Get the software and run it

Problems

Even in the best of cases

- Authors must undertake considerable effort to put data/results on the web (may not have resource like a webserver)
- 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 Programming

The idea of a literate program comes from Don Knuth:

- 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 Programming

Literate programming is a general concept. We need

1. A documentation language (human readable)
2. A programming language (machine readable)

We will be using LaTeX and R as our documentation and programming languages.

- The system implementing the necessary machinery is called **Sweave**, developed by Friedrich Leisch (member of the R Core)
- Main web site: http://www.statistik.lmu.de/~leisch/Sweave/

Alternatives to LaTeX/R exist, suchas HTML/R (package **R2HTML**) and ODF/R (package **odfWeave**).

Example of Literate Programming

I want to calculate the current time in R.

```
> time <- format(Sys.time(), "%a %b %d %X %Y")
```

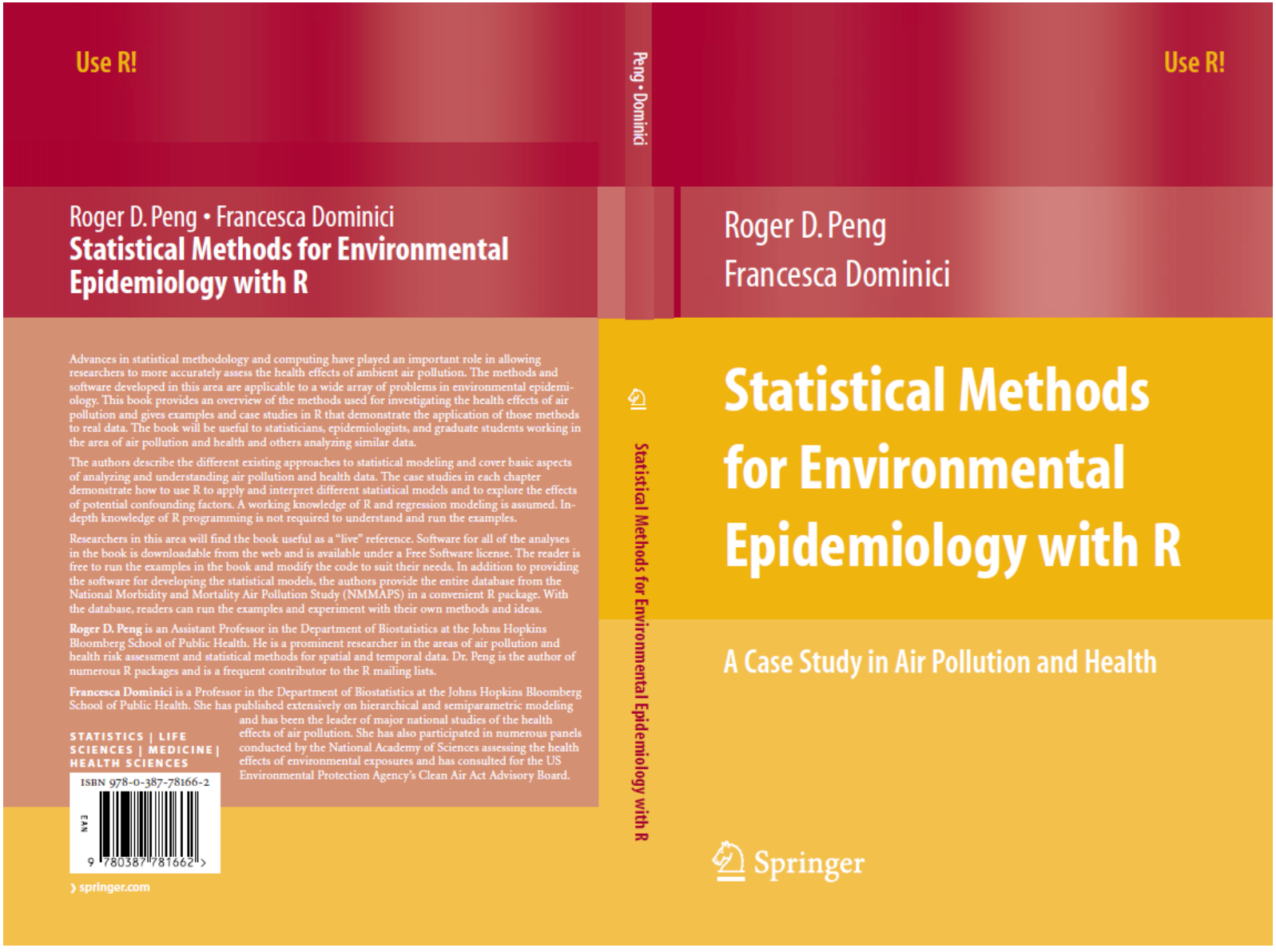
The current time is Sun Mar 15 23:37:49 2009. The text and R code are interwoven:

```
The time is Sun Mar 15 23:37:49 2009
```

Papers, dissertations, and presentations can be written using literate programming.

Example of Literate Programming

Even books can be written!



Literate Programming: Pros and Cons

Advantages of switching to literate programming

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

Some disadvantages

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

The **make** tool can be of great help but we will not discuss that here.

Sweave

What is Sweave?

- Sweave is a function and also a command-line script that comes with R (it is part of the **utils** package)
- The function can be invoked as `Sweave()`
- The command-line script is in the form `R CMD Sweave`

There is also Stangle

- `Stangle()`
- `R CMD Stangle`

But one thing at a time....

Basic Sweave Document: example.Rnw

```
\documentclass[11pt]{article}
\title{My First Sweave Document}
\begin{document}
\maketitle

This is some text (i.e. a ``text chunk'').

Here is a code chunk
<<>>=
set.seed(1)
x <- rnorm(100)
mean(x)
@
\end{document}
```

Processing a Sweave Document

```
## create 'example.tex'
## In R
library(utils)
Sweave("example.Rnw")

## On the command line
R CMD Sweave example.Rnw

## Usual LaTeX processing
## One of the following will work
texi2dvi example.tex  ## Create DVI file
latex example.tex
texi2dvi --pdf example.tex  ## Create PDF file
pdflatex example.tex
```

What R CMD Sweave Produces: example.tex

```
\documentclass[11pt]{article}
\title{My First Sweave Document}
\usepackage{Sweave}
\begin{document}
\maketitle
This is some text (i.e. a ``text chunk'').
Here is a code chunk
\begin{Schunk}
\begin{Sinput}
> set.seed(1)
> x <- rnorm(100)
> mean(x)
\end{Sinput}
\begin{Soutput}
[1] 0.1088874
\end{Soutput}
\end{Schunk}
\end{document}
```

The Resulting PDF Document

My First Sweave Document

March 14, 2009

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

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

[1] 0.1088874
```

A Few Good Notes

Code chunks begin with

```
<<>>=
```

and end with

```
@
```

All R code goes in between.

Code chunks can have **names**, which is useful when we start making graphics (more later).

```
<<loaddata>>=
## R code goes here
@
```

By default, the code in a code chunk will be **echo**ed, as will the results of the computation (if there is something to print).

Note on Processing Sweave Documents

It's important to remember that the order is

1. example.Rnw
2. example.tex
3. example.pdf

The `.tex` file is not something that we care about and **should not edit** (always edit the `.Rnw` file). It is merely an intermediary between the Sweave document and the PDF.

Basic Sweave Document: example2.Rnw

```
\documentclass[11pt]{article}
\title{My First Sweave Document}
\author{Roger D. Peng}
\begin{document}
\maketitle
\section{Introduction}
This is some text (i.e. a ``text chunk'').
Here is a code chunk
<<simulation,echo=false>>=
set.seed(1)
x <- rnorm(100)
mean(x)
@
\end{document}
```

Result

My First Sweave Document

Roger D. Peng

March 14, 2009

1 Introduction

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

```
[1] 0.1088874
```

Basic Sweave Document: example3.Rnw

```
\documentclass[11pt]{article}
\title{My First Sweave Document}
\begin{document}
\maketitle

\section{Introduction}
This is some text (i.e. a ``text chunk'').
Here is a code chunk but it doesn't print anything!
<<simulation,echo=false,results=hide>>=
x <- rnorm(100); y <- x + rnorm(100, sd = 0.5)
mean(x)
@
\end{document}
```

Result

My First Sweave Document

March 14, 2009

1 Introduction

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

Inline Text: example4.Rnw

```
\documentclass[11pt]{article}
\begin{document}
\section{Introduction}

<<computetime,echo=false>>=
time <- format(Sys.time(), "%a %b %d %X %Y")
rand <- rnorm(1)
@
The current time is \Sexpr{time}. My favorite random
number is \Sexpr{rand}.
\end{document}
```

Inline Text

1 Introduction

The current time is Sat Mar 14 15:03:01 2009. My favorite random number is 1.02265944276426.

Graphics: example5.Rnw

```
\documentclass[11pt]{article}
\begin{document}
\section{Introduction}
Let's first simulate some data.
<<computetime,echo=true>>=
x <- rnorm(100); y <- x + rnorm(100, sd = 0.5)
@
Here is a scatterplot of the data.
<<scatterplot,fig=true,width=8,height=4>>=
par(mar = c(5, 4, 1, 1), las = 1)
plot(x, y, main = "My Data")
@
\end{document}
```

What Sweave Produces

```
\documentclass[11pt]{article}
\usepackage{Sweave}

\begin{document}

\section{Introduction}
Let's first simulate some data.
\begin{Schunk}
\begin{Sinput}
> x <- rnorm(100)
> y <- x + rnorm(100, sd = 0.5)
\end{Sinput}
\end{Schunk}
```

What Sweave Produces (cont'd)

```
Here is a scatterplot of the data.
\begin{Schunk}
\begin{Sinput}
> par(mar = c(5, 4, 1, 1), las = 1)
> plot(x, y, main = "My Data")
\end{Sinput}
\end{Schunk}

\includegraphics{example5-scatterplot}

\end{document}
```

Graphics

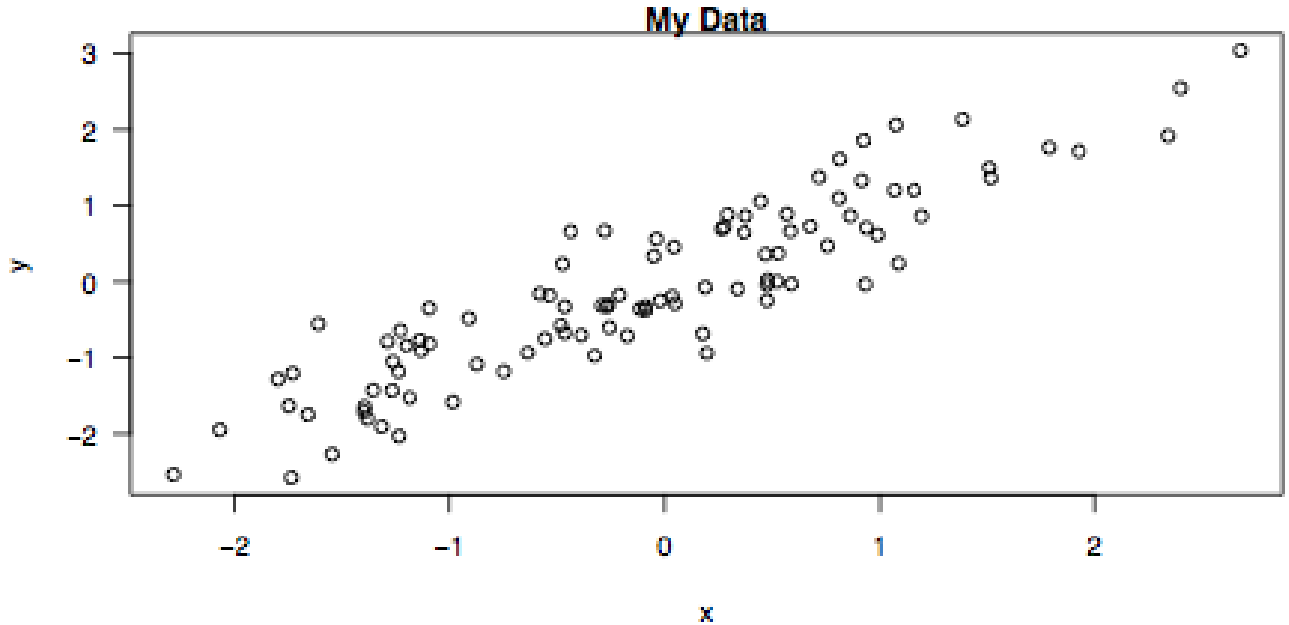
1 Introduction

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 Data")
```



Figures

```
\documentclass[11pt]{article}

\begin{document}
\section{Introduction}

Let's first simulate some data.

<<simulation,echo=true>>=
x <- rnorm(100); y <- x + rnorm(100, sd = 0.5)
@
```

Figures (cont'd)

```
Figure~\ref{plot} shows a scatterplot of the data.

\begin{figure}
<<scatterplot,fig=true,width=8,height=4>>=
par(mar = c(5, 4, 1, 1), las = 1)
plot(x, y, main = "My Data")
@
\caption{Scatterplot}
\label{plot}
\end{figure}

\end{document}
```

Getting the Code Out

Sometimes it is easier to have all the R code in a separate file by itself, without all of the LaTeX markup. We can use `Stangle` to do that.

```
## In R
> Stangle("example5.Rnw")
Writing to file example5.R

## On the command line
amelia:> R CMD Stangle example5.Rnw
Writing to file example5.R
```

Then we can call `source("example5.R")` to run all the code in the file.

Tangled Output

```
##############################################################
### chunk number 1: computetime
##############################################################
x <- rnorm(100); y <- x + rnorm(100, sd = 0.5)


##############################################################
### chunk number 2: scatterplot
##############################################################
par(mar = c(5, 4, 1, 1), las = 1)
plot(x, y, main = "My Data")
```

Setting Global Options: `example6.Rnw`

Sometimes, we want to set options for **every** code chunk that are non-default values. We can use `\SweaveOpts` to do that.

```
\SweaveOpts{option1=value1,option2=value2,...}
```

For example, we may want to suppress all code echoing and results output

```
\SweaveOpts{echo=false,results=hide}
```

The call to `\SweaveOpts` goes in the preamble.

Setting Global Options: example6.Rnw

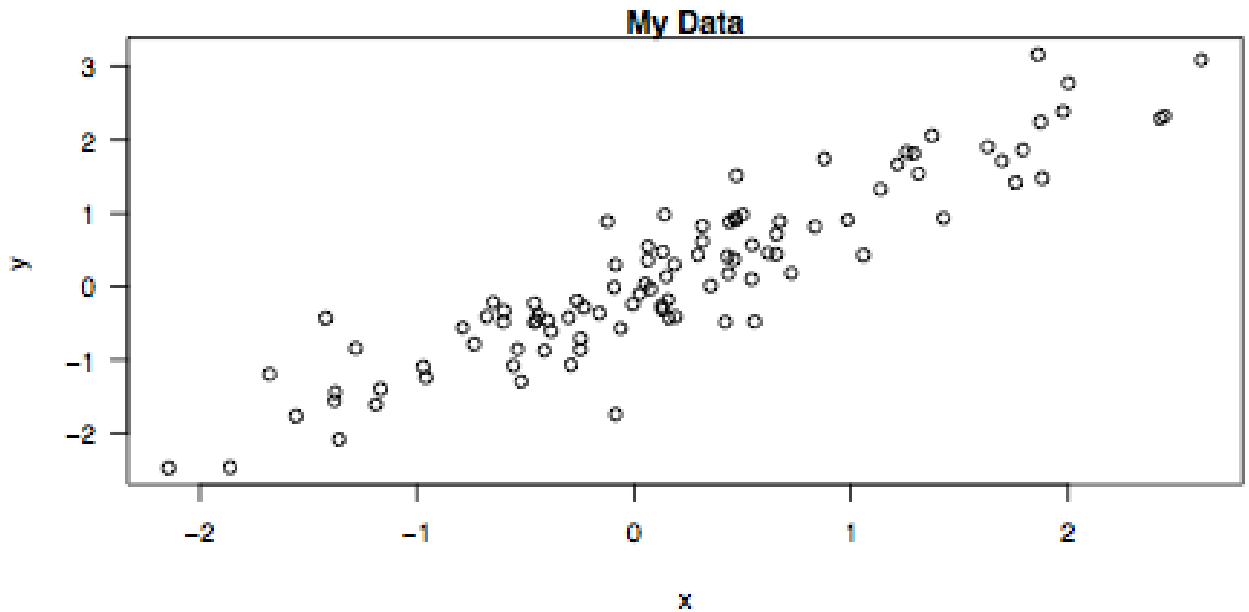
```
\documentclass[11pt]{article}
\SweaveOpts{echo=false}

\begin{document}
\section{Introduction}
<<computetime,echo=true>>=
x <- rnorm(100); y <- x + rnorm(100, sd = 0.5)
@

Here is a scatterplot of some simulated data.\\

<<scatterplot,fig=true,width=8,height=4>>=
par(mar = c(5, 4, 1, 1), las = 1)
plot(x, y, main = "My Data")
@
\end{document}
```

Setting Global Options

1 Introduction

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

Here is a scatterplot of some simulated data.



Making Tables with **xtable**: `example7.Rnw`

```
\documentclass[11pt]{article}
\begin{document}
\section{Introduction}
<<fitmodel>>=
library(datasets)
data(airquality)
fit <- lm(Ozone ~ Wind + Temp + Solar.R, data = airquality)
@

Here is a table of regression coefficients.\\

<<xtable,results=tex>>=
library(xtable)
xt <- xtable(summary(fit))
print(xt)
@
\end{document}
```

Tables

1 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)
```

	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

Summary of Options

Output

- results: verbatim (default), tex, hide
- echo: true (default), false
- eval: true (default), false

Figures

- fig: true, false (default)
- width: width of plot (passed to plot device)
- height: height of plot (passed to plot device)

Package vignettes

- A Sweave style vignette is a .Rnw file that contains chunks of code that are evaluated by R at 'R CMD build' time or on demand by the user with the Sweave command.
- The code contained in those chunks should show a typical workflow i.e. the commands (+ output) issued by a user during a typical interactive session with the package.
- The vignette should preferably demonstrates how to use the package to accomplish a non-trivial task. Why is this package important?
- Vignettes are just like standard Sweave documents but also include

  `\VignetteIndexEntry{Name of Vignette}`

  in the preamble

See also the writing R extensions manual.

Package Directory Structure

Vignettes go in the `inst/doc` directory of the package

```
amelia:> ls
./              .git/          NAMESPACE    inst/          src/
../             DESCRIPTION    R/           man/           tests/
amelia:> ls inst/doc
./              Sweave.sty     combined.bib  filehash.pdf
../             asa.bst        filehash.Rnw
```

`R CMD build` will automatically try to build the vignette for you.

Finding Vignettes in R

```
> vignette()

Vignettes in package 'Matrix':

Comparisons                Comparisons of Least Squares calcul
                           (source, pdf)
Design-issues              Design Issues in Matrix package Dev
                           (source, pdf)
Intro2Matrix               2nd Introduction to the Matrix Pack
                           pdf)
Introduction               Introduction to the Matrix Package
                           pdf)
sparseModels               Sparse Model Matrices (source, pdf)
```

Viewing Vignettes in R

```
## Launch vignette in (default) PDF viewer
vignette("filehash")

## Look at code in default text editor
v <- vignette("filehash")
edit(v)
```

Caching Computations

The **cacheSweave** package (on CRAN) can be used to cache long-running computations when developing a Sweave document

```
<<longcomputation,cache=true>>==
## Run MCMC sampler
result <- runmcmc(N = 10000)
@

<<traceplot,fig=true>>=
## Make trace plot of the parameter values
plot(result)
@
```

Processing Documents with **cacheSweave**

```
## In R
library(cacheSweave)

## Set cache directory (default is ".")
setCacheDir("cache")

## Process document
Sweave("mydocument.Rnw", driver = cacheSweaveDriver)
```

cacheSweave Caveats

Some caveats when using **cacheSweave**

- If the data/code changes, you will need to re-run cached code chunks
- Dependencies aren't checked, so if code in a cached chunk depends on computations in previous chunk that have changed, this inconsistency won't be detected (the **weaver** package tries to do this)
- Chunks that have **side effects** generally cannot be cached (e.g. plotting)

Reproducible Research Pipeline (Modified)