Reproducible Research

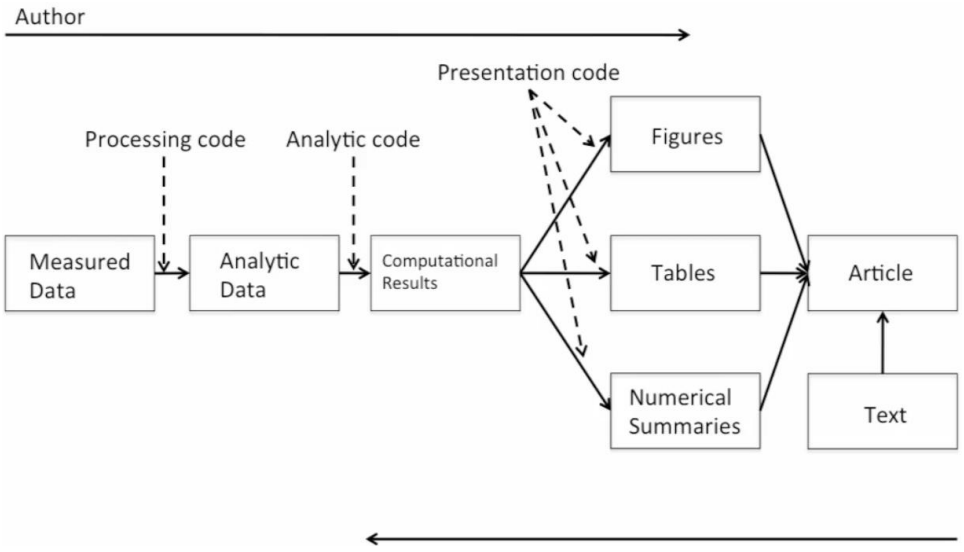
Replication

- 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

- 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

- 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"

Research Pipeline



What do we need?

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."

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

Challenges

- Authors must undertake considerable effort to put data/results on the web (may not have resources like a web server)
- Readers must download data/results individually and piece together which data go with which code sections, etc.
- Readers may not have the same resources as authors
- Few tools to help authors/readers (although toolbox is growing!)

In Reality

- Authors
  - Just put stuff on the web
  - (Infamous) Journal supplementary materials
  - There are some central databases for various fields (e.g. biology, ICPSR)
- Readers
  - Just download the data and (try to) figure it out
  - Piece together the software and run it

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 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 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

Knitr

- 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 (while a graduate student in statistics at Iowa State)
- See http://yihui.name/knitr/

Summary

- Reproducible research is important as a **minimum standard**, particularly for studies that are difficult to replicate
- Infrastructure is needed for **creating** and **distributing** reproducible documents, beyond what is currently available
- There is a growing number of tools for creating reproducible documents

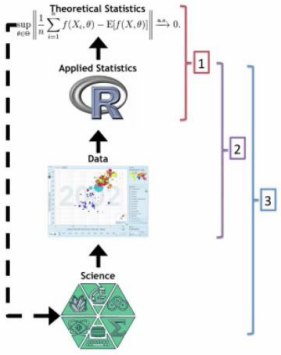
Structure of Data Analysis

- Define the question
- Define the ideal data set
- Determine what data you can access
- Obtain the data
- Clean the data
- Exploratory data analysis
- Statistical prediction/modeling
- Interpret results
- Challenge results
- Synthesize/write up results
- Create reproducible code

Key Challenge

"Ask yourselves, what problem have you solved, ever, that was worth solving, where you knew all of the given information in advance? Where you didn't have a surplus of information and have to filter it out, or you had insufficient information and have to go find some?"

Define a Question



Start with a general question

Can I automatically detect emails that are SPAM that are not?

Make it concrete

Can I use quantitative characteristics of the emails to classify them as SPAM/HAM?

1. Statistical methods development
2. Danger zone!!!
3. Proper data analysis

Define the ideal dataset

- The data set may depend on your goal
  - Descriptive - a whole population
  - Exploratory - a random sample with many variables measured
  - Inferential - the right population, randomly sampled
  - Predictive - a training and test data set from the same population
  - Causal - data from a randomized study
  - Mechanistic - data about all components of the system

Determine what data you can access

- Sometimes you can find data free on the web
- Other times you may need to buy the data
- Be sure to respect the terms of use
- If the data don't exist, you may need to generate it yourself

Obtain the data

- Try to obtain the raw data
- Be sure to reference the source
- Polite emails go a long way
- If you will load the data from an internet source, record the url and time accessed

Clean the data

- Raw data often needs to be processed
- If it is pre-processed, make sure you understand how
- Understand the source of the data (census, sample, convenience sample, etc.)
- May need reformating, subsampling - record these steps
- **Determine if the data are good enough** - if not, quit or change data

Our cleaned data

```
# If it isn't installed, install the kernlab package with install.packages()
library(kernlab)
data(spam)
str(spam[, 1:5])
```

```
'data.frame':   4601 obs. of  5 variables:
 $ make    : num  0 0.21 0.06 0 0 0 0 0 0.15 0.06 ...
 $ address: num  0.64 0.28 0 0 0 0 0 0 0 0.12 ...
 $ all     : num  0.64 0.5 0.71 0 0 0 0 0 0.46 0.77 ...
 $ num3d  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ our     : num  0.32 0.14 1.23 0.63 0.63 1.85 1.92 1.88 0.61 0.19 ...
```

http://search.r-project.org/library/kernlab/html/spam.html

Subsampling our dataset

We need to generate a test and training set (prediction)

```
# If it isn't installed, install the kernlab package
library(kernlab)
data(spam)
# Perform the subsampling
set.seed(3435)
trainIndicator = rbinom(4601, size = 1, prob = 0.5)
table(trainIndicator)
```

```
## trainIndicator
##    0    1
## 2314 2287
```

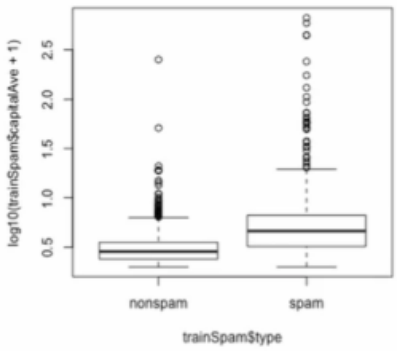
```
trainSpam = spam[trainIndicator == 1, ]
testSpam = spam[trainIndicator == 0, ]
```

Exploratory analysis

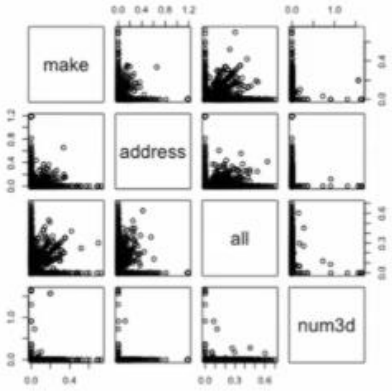
- Look at summaries of the data
- Check for missing data
- Create exploratory plots
- Perform exploratory analyses (e.g. clustering)

Plots

```
plot(log10(trainSpam$capitalAve + 1) ~ trainSpam$type)
```

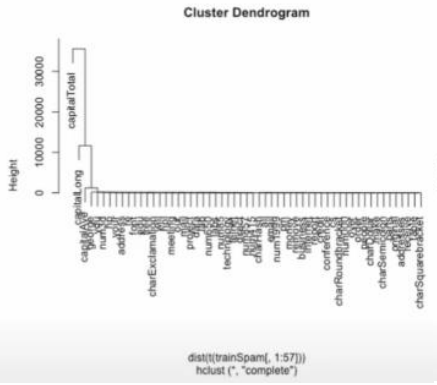



```
plot(log10(trainSpam[, 1:4] + 1))
```

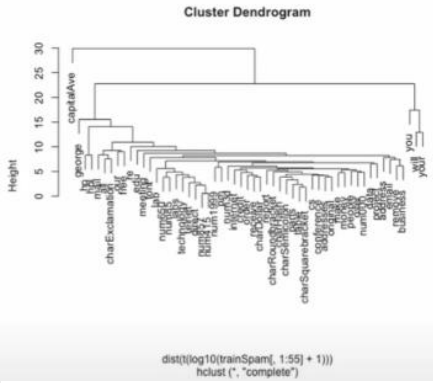


Clustering

```
hCluster = hclust(dist(t(trainSpam[, 1:57])))
plot(hCluster)
```




```
hClusterUpdated = hclust(dist(t(log10(trainSpam[, 1:55] + 1))))
plot(hClusterUpdated)
```



Statistical prediction / modeling

- Should be informed by the results of your exploratory analysis
- Exact methods depend on the question of interest
- Transformations/processing should be accounted for when necessary
- Measures of uncertainty should be reported

```
trainSpam$numType = as.numeric(trainSpam$type) - 1
costFunction = function(x, y) sum(x != (y > 0.5))
cvError = rep(NA, 55)
library(boot)
for (i in 1:55) {
    lmFormula = reformulate(names(trainSpam)[i], response = "numType")
    glmFit = glm(lmFormula, family = "binomial", data = trainSpam)
    cvError[i] = cv.glm(trainSpam, glmFit, costFunction, 2)$delta[2]
}

## Which predictor has minimum cross-validated error?
names(trainSpam)[which.min(cvError)]
```

```
## [1] "charDollar"
```

Get a measure of uncertainty

```
## Use the best model from the group
predictionModel = glm(numType ~ charDollar, family = "binomial", data = trainSpam)

## Get predictions on the test set
predictionTest = predict(predictionModel, testSpam)
predictedSpam = rep("nonspam", dim(testSpam)[1])

## Classify as `spam' for those with prob > 0.5
predictedSpam[predictionModel$fitted > 0.5] = "spam"
```

```
## Classification table
table(predictedSpam, testSpam$type)
```

```
##
## predictedSpam nonspam spam
##       nonspam    1346  458
##       spam         61  449
```

```
## Error rate
(61 + 458)/(1346 + 458 + 61 + 449)
```

```
## [1] 0.2243
```

Interpret Results

- Use the appropriate language
  - describes
  - correlates with/associated with
  - leads to/causes
  - predicts
- Give an explanation
- Interpret coefficients
- Interpret measures of uncertainty

- The fraction of charcters that are dollar signs can be used to predict if an email is Spam
- Anything with more than 6.6% dollar signs is classified as Spam
- More dollar signs always means more Spam under our prediction
- Our test set error rate was 22.4%

Challenge Results

- Challenge all steps:
  - Question
  - Data source
  - Processing
  - Analysis
  - Conclusions
- Challenge measures of uncertainty
- Challenge choices of terms to include in models
- Think of potential alternative analyses

Synthesize / Write-up Results

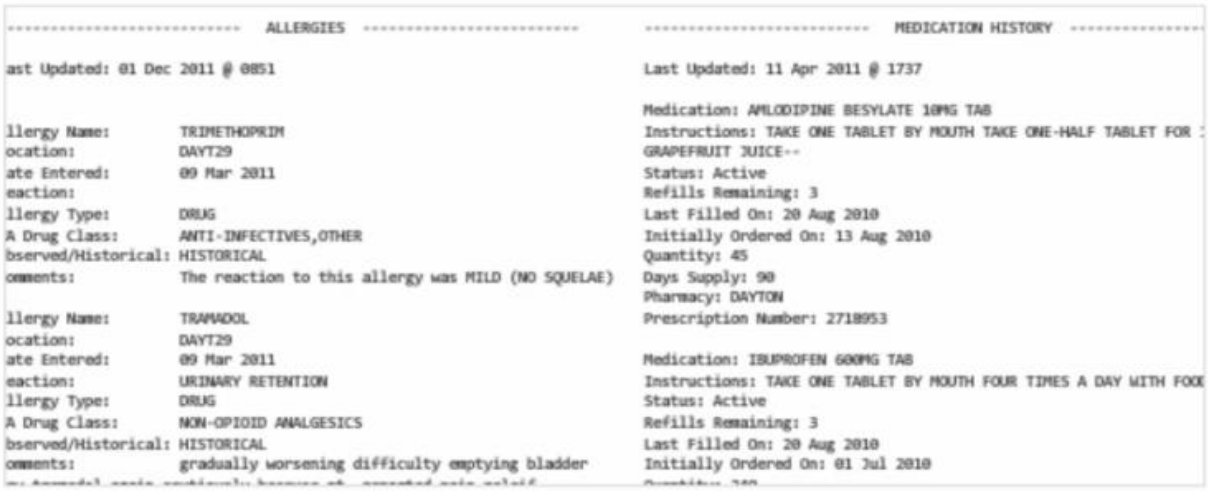
- Lead with the question
- Summarize the analyses into the story
- Don't include every analysis, include it
  - If it is needed for the story
  - If it is needed to address a challenge
- Order analyses according to the story, rather than chronologically
- Include "pretty" figures that contribute to the story

- Lead with the question
  - Can I use quantitative characteristics of the emails to classify them as SPAM/HAM?
- Describe the approach
  - Collected data from UCI -> created training/test sets
  - Explored relationships
  - Choose logistic model on training set by cross validation
  - Applied to test, 78% test set accuracy
- Interpret results
  - Number of dollar signs seems reasonable, e.g. "Make money with Viagra $ $ $ $!"
- Challenge results
  - 78% isn't that great
  - I could use more variables
  - Why logistic regression?

Organize Data Analysis

- Data
  - Raw data
  - Processed data
- Figures
  - Exploratory figures
  - Final figures
- R code
  - Raw / unused scripts
  - Final scripts
  - R Markdown files
- Text
  - README files
  - Text of analysis / report

Raw Data



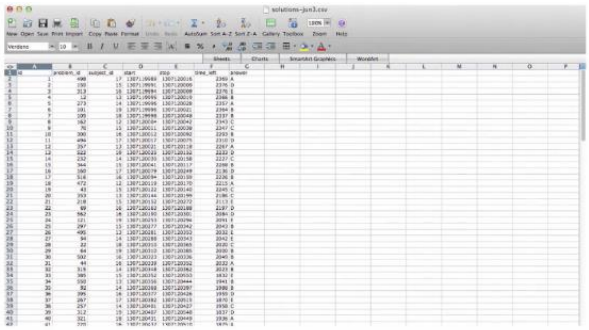
```
--------------------------  ALLERGIES  --------------------------      --------------------------  MEDICATION HISTORY  ----------

ast Updated: 01 Dec 2011 @ 0851                                       Last Updated: 11 Apr 2011 @ 1737

                                                                      Medication: AMLODIPINE BESYLATE 10MG TAB
llergy Name:        TRIMETHOPRIM                                      Instructions: TAKE ONE TABLET BY MOUTH TAKE ONE-HALF TABLET FOR
ocation:            DAYT29                                            GRAPEFRUIT JUICE--
ate Entered:        09 Mar 2011                                       Status: Active
eaction:                                                              Refills Remaining: 3
llergy Type:        DRUG                                              Last Filled On: 20 Aug 2010
A Drug Class:       ANTI-INFECTIVES,OTHER                             Initially Ordered On: 13 Aug 2010
bserved/Historical: HISTORICAL                                        Quantity: 45
omments:            The reaction to this allergy was MILD (NO SQUELAE)  Days Supply: 90
                                                                      Pharmacy: DAYTON
llergy Name:        TRAMADOL                                          Prescription Number: 2718953
ocation:            DAYT29
ate Entered:        09 Mar 2011                                       Medication: IBUPROFEN 600MG TAB
eaction:            URINARY RETENTION                                 Instructions: TAKE ONE TABLET BY MOUTH FOUR TIMES A DAY WITH FOO
llergy Type:        DRUG                                              Status: Active
A Drug Class:       NON-OPIOID ANALGESICS                             Refills Remaining: 3
bserved/Historical: HISTORICAL                                        Last Filled On: 20 Aug 2010
omments:            gradually worsening difficulty emptying bladder   Initially Ordered On: 01 Jul 2010
```

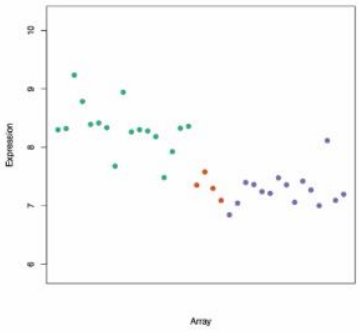
- Should be stored in your analysis folder
- If accessed from the web, include url, description, and date accessed in README

Processed Data



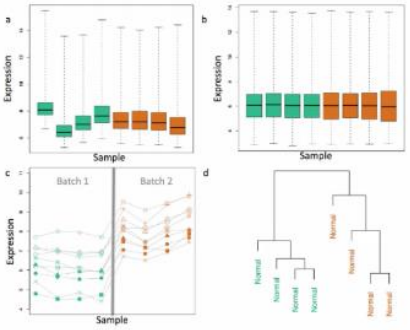
- Processed data should be named so it is easy to see which script generated the data.
- The processing script - processed data mapping should occur in the README
- Processed data should be tidy

Exploratory Figures



- Figures made during the course of your analysis, not necessarily part of your final report.
- They do not need to be "pretty"

Final Figures



- Usually a small subset of the original figures
- Axes/colors set to make the figure clear
- Possibly multiple panels

Raw Scripts

```r
source("regmodel.R")

dp <- ddm[, c("group", "pm25_0", "pm25_1", "symfree0", "symfree1")]
dp$p_id <- row.names(dp)

fitx0 <- lm(pm25_1 ~ pm25_0 + age + no2_0 + pm10_0, data = subset(ddm, group ==
fitx1 <- lm(pm25_1 ~ ns(pm25_0, 2) + age + no2_0 + pm10_0, data = subset(ddm, gr

fity0 <- glm(cbind(symfree1, 14-symfree1) ~ symfree0 + age + factor(gender), dat
fity1 <- glm(cbind(symfree1, 14-symfree1) ~ symfree0 + age + factor(gender), dat

y10 <- predict(fity0, subset(ddm, group == 1), type = "response") * 14
y01 <- predict(fity1, subset(ddm, group == 0), type = "response") * 14
p10 <- predict(fitx0, subset(ddm, group == 1))
p01 <- predict(fitx1, subset(ddm, group == 0))

yy <- data.frame(p_id = as.integer(c(names(y10), names(y01))),
                 symfree00 = c(y10, y01))
pp <- data.frame(p_id = as.integer(c(names(p10), names(p01))),
                 pm25_00 = c(p10, p01))

m <- merge(dp, yy, by = "p_id")
mm <- merge(m, pp, by = "p_id")
```

- May be less commented (but comments help you!)
- May be multiple versions
- May include analyses that are later discarded

Final Scripts

```r
##############################################################
## Main 'pgibbs()' function


pgibbs <- function(gibbsState,
                   maxit = 80000,
                   verbose = TRUE,
                   dbfile = "stategibbs",
                   deleteCache = FALSE,
                   singleAgeCat = TRUE,
                   sigmaE = NULL,
                   delta = NULL) {
        library(MASS)

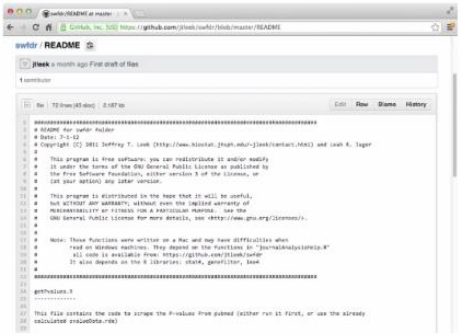
        ## Setup database of results
        if(file.exists(dbfile)) {
                if(deleteCache) {
                        message("removing existing cache file")
                        file.remove(dbfile)
                }
                else
                        stop(sprintf("cache file '%s' already exists", dbfile))
        }
```

- Clearly commented
  - Small comments liberally - what, when, why, how
  - Bigger commented blocks for whole sections
- Include processing details
- Only analyses that appear in the final write-up

R Markdown Files

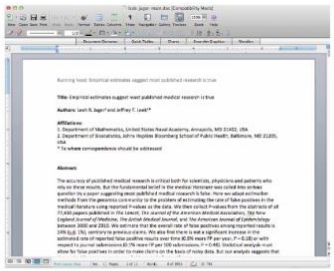
- R markdown files can be used to generate reproducible reports
- Text and R code are integrated
- Very easy to create in Rstudio

Readme Files



- Not necessary if you use R markdown
- Should contain step-by-step instructions for analysis
- Here is an example https://github.com/jtleek/swfdr/blob/master/README.md

Text of the document



- It should include a title, introduction (motivation), methods (statistics you used), results (including measures of uncertainty), and conclusions (including potential problems)
- It should tell a story
- *It should not include every analysis you performed*
- References should be included for statistical methods

Further Resources

- Information about a non-reproducible study that led to cancer patients being mistreated: The Duke Saga Starter Set
- Reproducible research and Biostatistics
- Managing a statistical analysis project guidelines and best practices
- Project template - a pre-organized set of files for data analysis

Coding Standards in 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

- 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

Markdown

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

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

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

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

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

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

```
    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"
```

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

```
First line
Second line
```

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**

- 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

R Markdown Demo

````
My First R Markdown File
========================

This is my first R markdown file.

Here, we're going load some data.

```{r}
library(datasets)
data(airquality)
summary(airquality)
```

Let's first make a pairs plot of the data.

```{r}
pairs(airquality)
```

Here's a regression model of ozone on wind, solar radiation, and temperature.

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


Here's an unordered list:

* First element
* Second element
````

Literal Statistical Programming with knitr

- 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 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

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

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

- 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

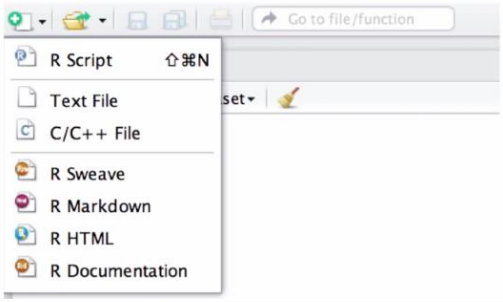
Good For

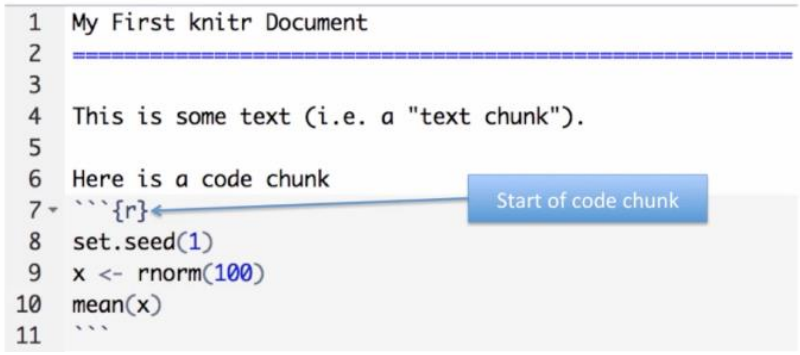
- Manuals
- Short/medium-length technical documents
- Tutorials
- Reports (esp. if generated periodically)
- Data preprocessing documents/summaries

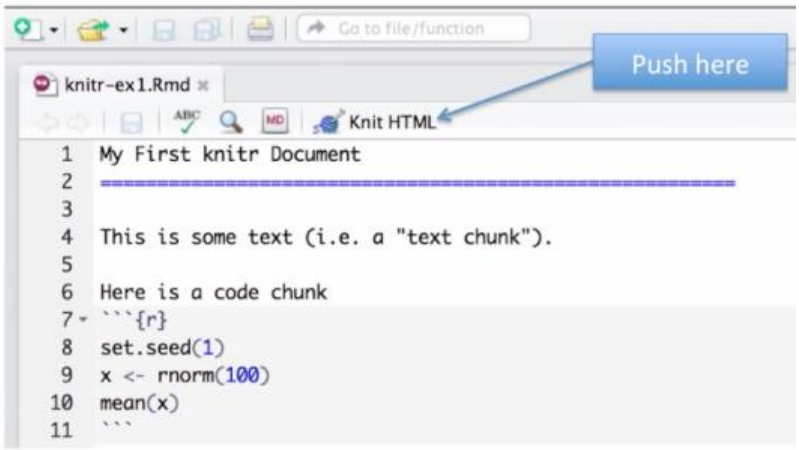
Not Good For

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

Creating Basic knitr document








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

HTML Output

My First knitr Document

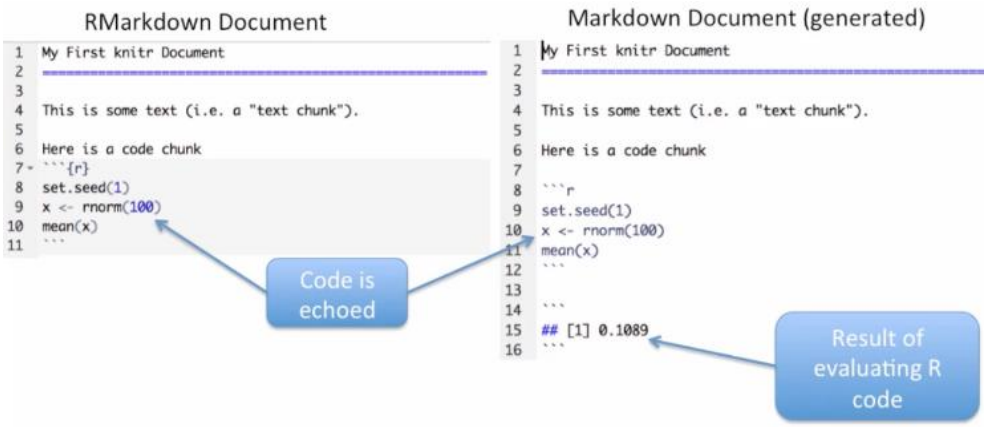
This is some text (i.e. a “text chunk”).

Here is a code chunk

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

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

What knitr produces



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

- 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

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)
```
````

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`.
````

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

```
<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+Gm3BtzzzHPDTNgQCqwKXfZwSeNHHJz1OIT8JjtAq6xWtCLwGPLz1OIT8JjtAq6
```

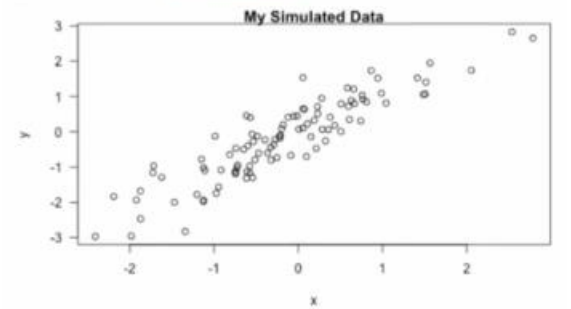


Image is embedded in HTML

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")
```
````

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

````
## 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

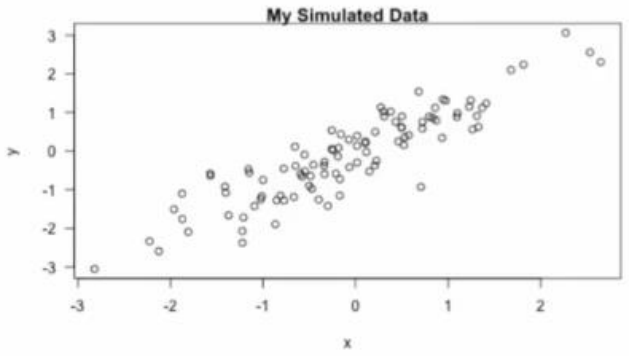
Don't echo code here

Introduction

First simulate data

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

Here is a scatterplot of the data.



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
- 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

Communicating Results

Tl;dr

- People are busy, especially managers and leaders
- Results of data analyses are sometimes presented in oral form, but often the first cut is presented via email
- It is often useful to breakdown the results of an analysis into different levels of granularity / detail
- Getting responses from busy people: http://goo.gl/sJDb9V

Hierarchy of Information

Research Paper

- Title / Author list
- Abstract
- Body / Results
- Supplementary Materials / the gory details
- Code / Data / really gory details

Email Presentation

- Subject line / Sender info
  - At a minimum; include one
  - Can you summarize findings in one sentence?
- Email body
  - A brief description of the problem / context; recall what was proposed and executed; summarize findings / results; 1–2 paragraphs
  - If action needs to be taken as a result of this presentation, suggest some options and make them as concrete as possible.
  - If questions need to be addressed, try to make them yes / no

- Attachment(s)
  - R Markdown file
  - knitr report
  - Stay concise; don't spit out pages of code (because you used knitr we know it's available)
- Links to Supplementary Materials
  - Code / Software / Data
  - GitHub repository / Project web site

RPubs

- Rpubs.com
- Publish documents online to share with public
- Click publish button in R Studio

Reproducible Research Checklist

DO: Start with good science

- Garbage in, garbage out
- Coherent, focused question simplifies many problems
- Working with good collaborators reinforces good practices
- Something that's interesting to you will (hopefully) motivate good habits

DON'T: Do things by hand

- Editing spreadsheets of data to "clean it up"
  - Removing outliers
  - QA / QC
  - Validating
- Editing tables or figures (e.g. rounding, formatting)
- Downloading data from a web site (clicking links in a web browser)
- Moving data around your computer; splitting / reformatting data files
- "We're just going to do this once...."

Things done by hand need to be precisely documented (this is harder than it sounds)

DON'T: Point and click

- Many data processing / statistical analysis packages have graphical user interfaces (GUIs)
- GUIs are convenient / intuitive but the actions you take with a GUI can be difficult for others to reproduce
- Some GUIs produce a log file or script which includes equivalent commands; these can be saved for later examination
- In general, be careful with data analysis software that is highly *interactive*; ease of use can sometimes lead to non-reproducible analyses
- Other interactive software, such as text editors, are usually fine

DO: Teach a computer

- If something needs to be done as part of your analysis / investigation, try to teach your computer to do it (even if you only need to do it once)
- In order to give your computer instructions, you need to write down exactly what you mean to do and how it should be done
- Teaching a computer almost guarantees reproducibilty

For example, by hand, you can

1. Go to the UCI Machine Learning Repository at http://archive.ics.uci.edu/ml/
2. Download the Bike Sharing Dataset by clicking on the link to the Data Folder, then clicking on the link to the zip file of dataset, and choosing "Save Linked File As..." and then saving it to a folder on your computer

Or You can teach your computer to do the same thing using R:

```
download.file("http://archive.ics.uci.edu/ml/machine-learning-databases/00275/
              Bike-Sharing-Dataset.zip", "ProjectData/Bike-Sharing-Dataset.zip")
```

Notice here that

- The full URL to the dataset file is specified (no clicking through a series of links)
- The name of the file saved to your local computer is specified
- The directory in which the file was saved is specified ("ProjectData")
- Code can always be executed in R (as long as link is available)

DO: Use some version control

- Slow things down
- Add changes in small chunks (don't just do one massive commit)
- Track / tag snapshots; revert to old versions
- Software like GitHub / BitBucket / SourceForge make it easy to publish results

DO: Keep track of your software environ.

- If you work on a complex project involving many tools / datasets, the software and computing environment can be critical for reproducing your analysis
- **Computer architecture**: CPU (Intel, AMD, ARM), GPUs,
- **Operating system**: Windows, Mac OS, Linux / Unix
- **Software toolchain**: Compilers, interpreters, command shell, programming languages (C, Perl, Python, etc.), database backends, data analysis software
- **Supporting software / infrastructure**: Libraries, R packages, dependencies
- **External dependencies**: Web sites, data repositories, remote databases, software repositories
- **Version numbers**: Ideally, for everything (if available)

```
sessionInfo()
```

```
## R version 3.0.2 Patched (2014-01-20 r64849)
## Platform: x86_64-apple-darwin13.0.0 (64-bit)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  base
##
## other attached packages:
## [1] slidify_0.3.3
##
## loaded via a namespace (and not attached):
## [1] evaluate_0.5.1 formatR_0.10   knitr_1.5      markdown_0.6.3
## [5] stringr_0.6.2  tools_3.0.2    whisker_0.3-2  yaml_2.1.8
```

DON'T: Save Output

- Avoid saving data analysis output (tables, figures, summaries, processed data, etc.), except perhaps temporarily for efficiency purposes.
- If a stray output file cannot be easily connected with the means by which it was created, then it is not reproducible.
- Save the data + code that generated the output, rather than the output itself
- Intermediate files are okay as long as there is clear documentation of how they were created

DO: Set your seed

- Random number generators generate pseudo-random numbers based on an initial seed (usually a number or set of numbers)
    - In R you can use the `set.seed()` function to set the seed and to specify the random number generator to use
- Setting the seed allows for the stream of random numbers to be exactly reproducible
- Whenever you generate random numbers for a non-trivial purpose, **always set the seed**

DO: Think about the entire pipeline

- Data analysis is a lengthy process; it is not just tables / figures / reports
- Raw data → processed data → analysis → report
- How you got the end is just as important as the end itself
- The more of the data analysis pipeline you can make reproducible, the better for everyone

Summary: Checklist

- Are we doing good science?
- Was any part of this analysis done by hand?
  - If so, are those parts *precisely* document?
  - Does the documentation match reality?
- Have we taught a computer to do as much as possible (i.e. coded)?
- Are we using a version control system?
- Have we documented our software environment?
- Have we saved any output that we cannot reconstruct from original data + code?
- How far back in the analysis pipeline can we go before our results are no longer (automatically) reproducible?

Reproducible Research with Evidence based Data Analysis

Replication and Reproducibility

Reproducibility

- Focuses on the validity of the data analysis
- "Can we trust this analysis?"
- Arguably a minimum standard for any scientific study
- New investigators, same data, same methods
- Important when replication is impossible

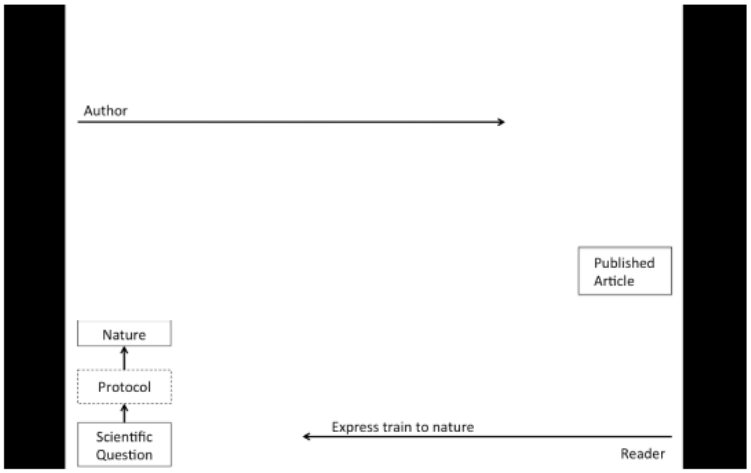
Background and underlying trends

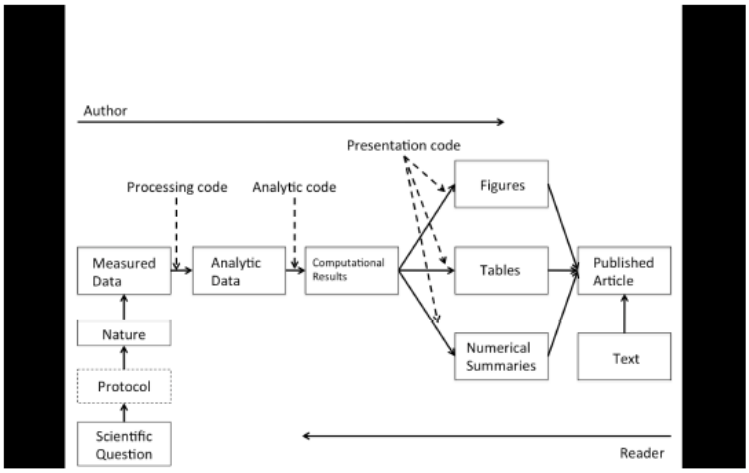
- Some studies cannot be replicated: No time, No money, Unique/opportunistic
- Technology is increasing data collection throughput; data are more complex and high-dimensional
- Existing databases can be merged to become bigger databases (but data are used off-label)
- Computing power allows more sophisticated analyses, even on "small" data
- For every field "X" there is a "Computational X"

The Result?

- Even basic analyses are difficult to describe
- Heavy computational requirements are thrust upon people without adequate training in statistics and computing
- Errors are more easily introduced into long analysis pipelines
- Knowledge transfer is inhibited
- Results are difficult to replicate or reproduce
- Complicated analyses cannot be trusted

What is Reproducible Research?





What Problem Does Reproducibility Solve?

What we get

- Transparency
- Data Availability
- Software / Methods Availability
- Improved Transfer of Knowledge

What we do NOT get

- Validity / Correctness of the analysis

An analysis can be reproducible and still be wrong

We want to know “can we trust this analysis?”

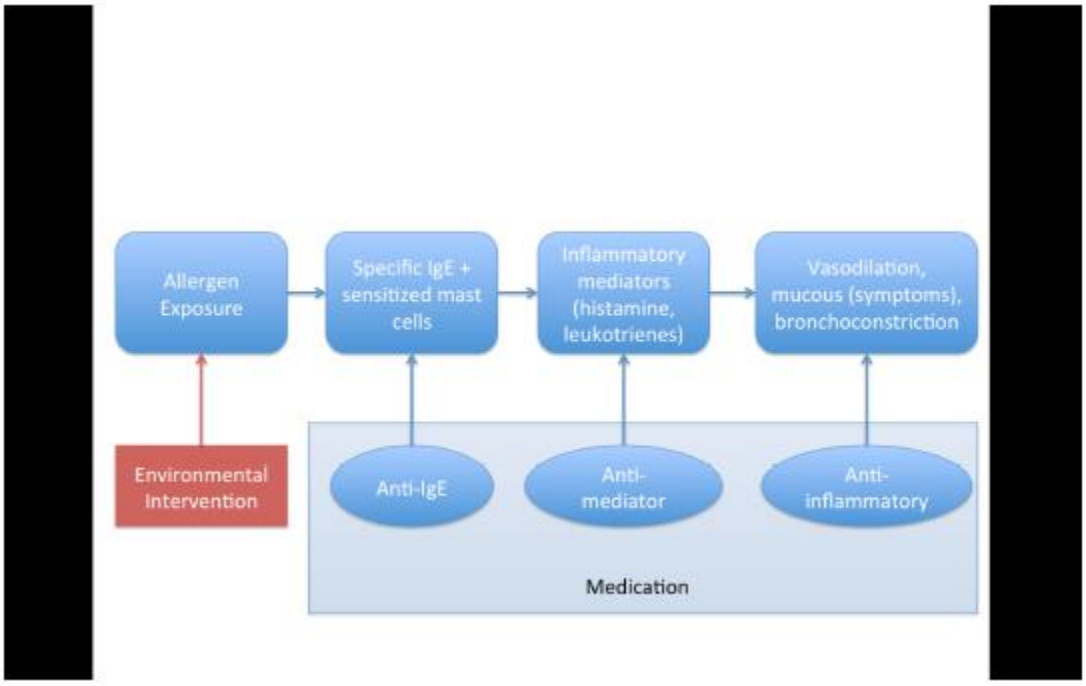
Does requiring reproducibility deter bad analysis?

Problems with Reproducibility

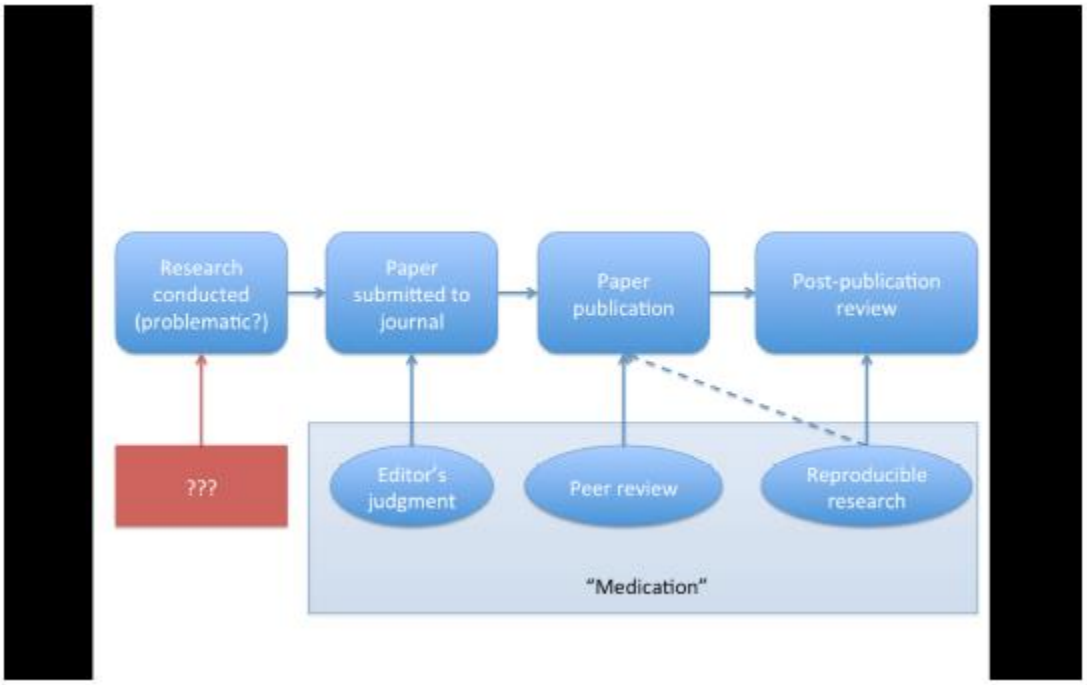
The premise of reproducible research is that with data/code available, people can check each other and the whole system is self-correcting

- Addresses the most “downstream” aspect of the research process – post-publication
- Assumes everyone plays by the same rules and wants to achieve the same goals (i.e. scientific discovery)

An Analogy from Asthma

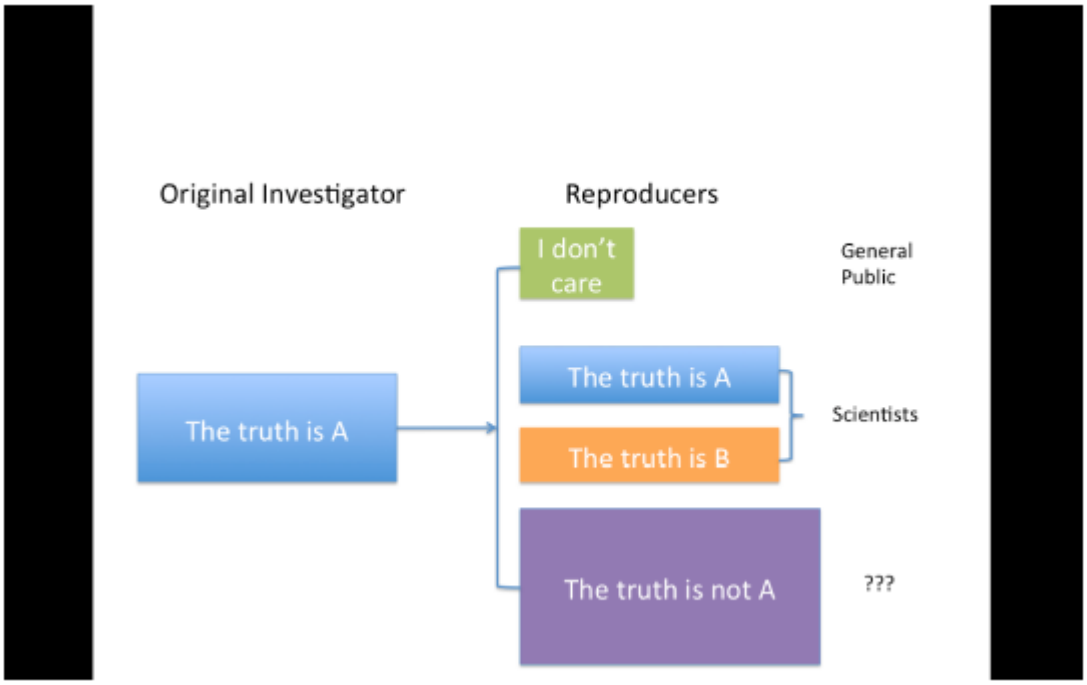


Scientific Dissemination Process



Who Reproduces Research?

- For reproducibility to be effective as a means to check validity, someone needs to do something
  - Re-run the analysis; check results match
  - Check the code for bugs/errors
  - Try alternate approaches; check sensitivity
- The need for someone to do something is inherited from traditional notion of replication
- Who is "someone" and what are their goals?

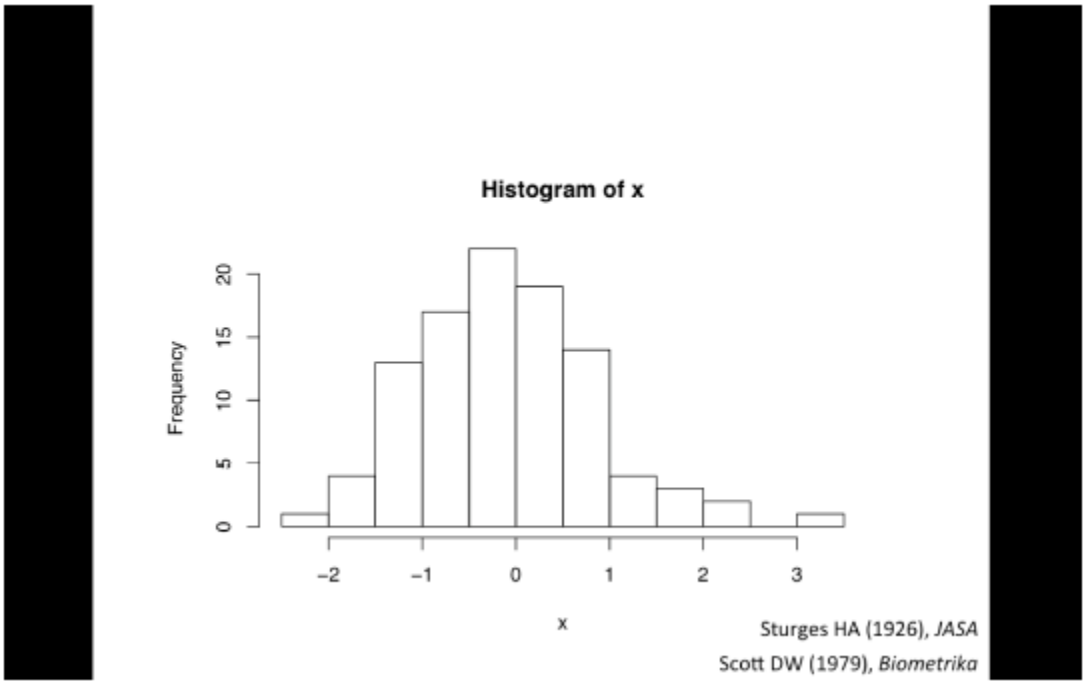


The Story So Far

- Reproducibility brings transparency (wrt code+data) and increased transfer of knowledge
- A lot of discussion about how to get people to share data
- Key question of "can we trust this analysis?" is not addressed by reproducibility
- Reproducibility addresses potential problems long after they've occurred ("downstream")
- Secondary analyses are inevitably coloured by the interests/motivations of others

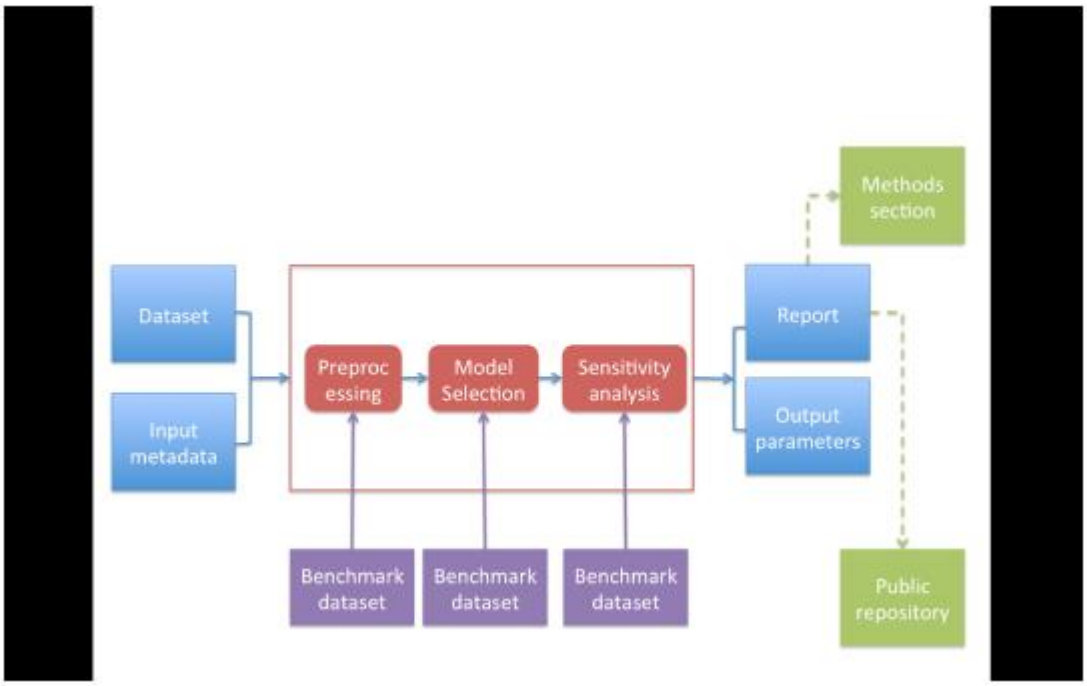
Evidence-based Data Analysis

- Most data analyses involve stringing together many different tools and methods
- Some methods may be standard for a given field, but others are often applied ad hoc
- We should apply thoroughly studied (via statistical research), mutually agreed upon methods to analyze data whenever possible
- There should be evidence to justify the application of a given method



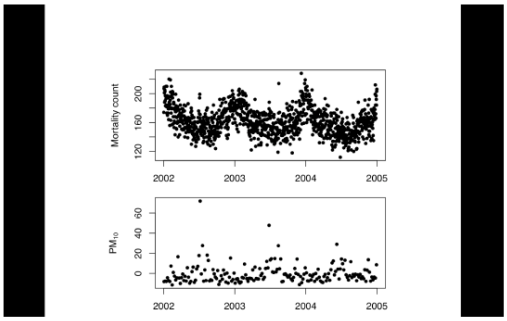
- Create analytic pipelines from evidence-based components – standardize it
- A Deterministic Statistical Machine http://goo.gl/Qvlhuv
- Once an evidence-based analytic pipeline is established, we shouldn't mess with it
- Analysis with a "transparent box"
- Reduce the "researcher degrees of freedom"
- Analogous to a pre-specified clinical trial protocol

Deterministic Statistical Machine



Case Study: Estimating Acute Effects of Ambient Air Pollution Exposure

- Acute/short-term effects typically estimated via panel studies or time series studies
- Work originated in late 1970s early 1980s
- Key question: "Are short-term changes in pollution associated with short-term changes in a population health outcome?"
- Studies usually conducted at community level
- Long history of statistical research investigating proper methods of analysis



- Can we encode everything that we have found in statistical/epidemiological research into a single package?
- Time series studies do not have a huge range of variation; typically involves similar types of data and similar questions
- We can create a deterministic statistical machine for this area?

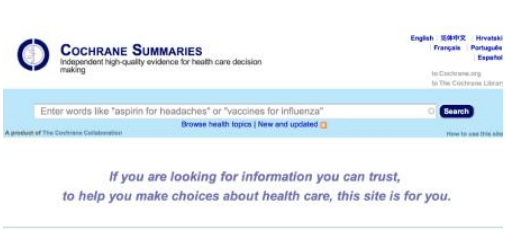
DSM Modules for Time Series Studies

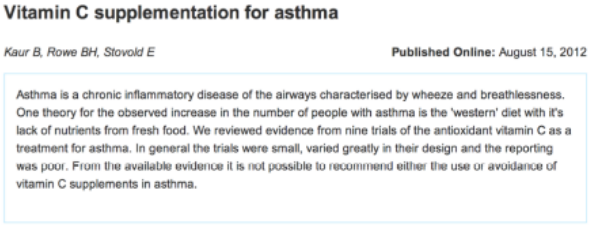
1. Check for outliers, high leverage, overdispersion
2. Fill in missing data? NO!
3. Model selection: Estimate degrees of freedom to adjust for unmeasured confounders
   - Other aspects of model not as critical
4. Multiple lag analysis
5. Sensitivity analysis wrt
   - Unmeasured confounder adjustment
   - Influential points

Where to Go From Here?

- One DSM is not enough, we need many!
- Different problems warrant different approaches and expertise
- A curated library of machines providing state-of-the art analysis pipelines
- A CRAN/CPAN/CTAN/… for data analysis
- Or a “Cochrane Collaboration” for data analysis

A Model: Cochrane Collaboration





A Curated Library of Data Analysis

- Provide packages that encode data analysis pipelines for given problems, technologies, questions
- Curated by experts knowledgeable in the field
- Documentation/references given supporting each module in the pipeline
- Changes introduced after passing relevant benchmarks/unit tests

Summary

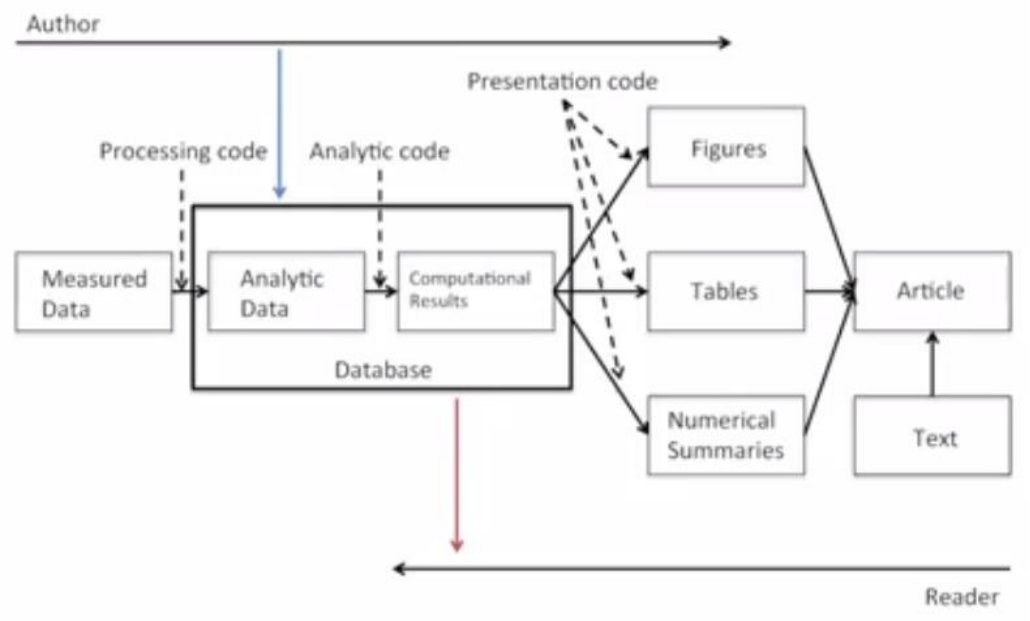
- Reproducible research is important, but does not necessarily solve the critical question of whether a data analysis is trustworthy
- Reproducible research focuses on the most "downstream" aspect of research dissemination
- Evidence-based data analysis would provide standardized, best practices for given scientific areas and questions
- Gives reviewers an important tool without dramatically increasing the burden on them
- More effort should be put into improving the quality of "upstream" aspects of scientific research

Literate (Statistical) Programming

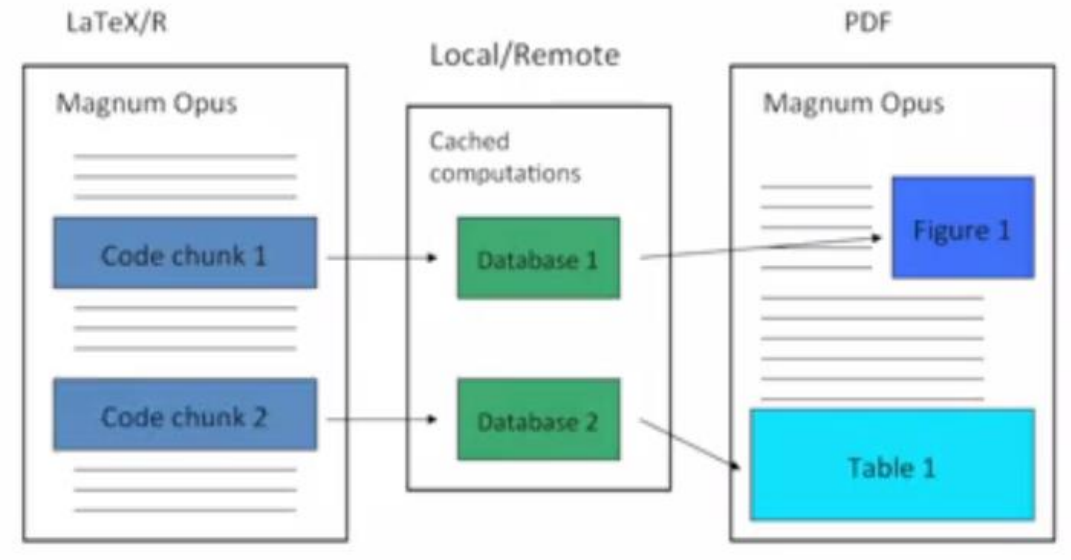
- An article is a stream of **text** and **code**
- Analysis code is divided into text and code "chunks"
- Each code chunk loads data and computes results
- Presentation code formats results (tables, figures, etc.)
- Article text explains what is going on
- Literate programs can be **weaved** to produce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

- Literate 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`
- Alternatives to LATEX/R exist, suchas HTML/R (package R2HTML) and ODF/R (package odfWeave).

Research Pipeline



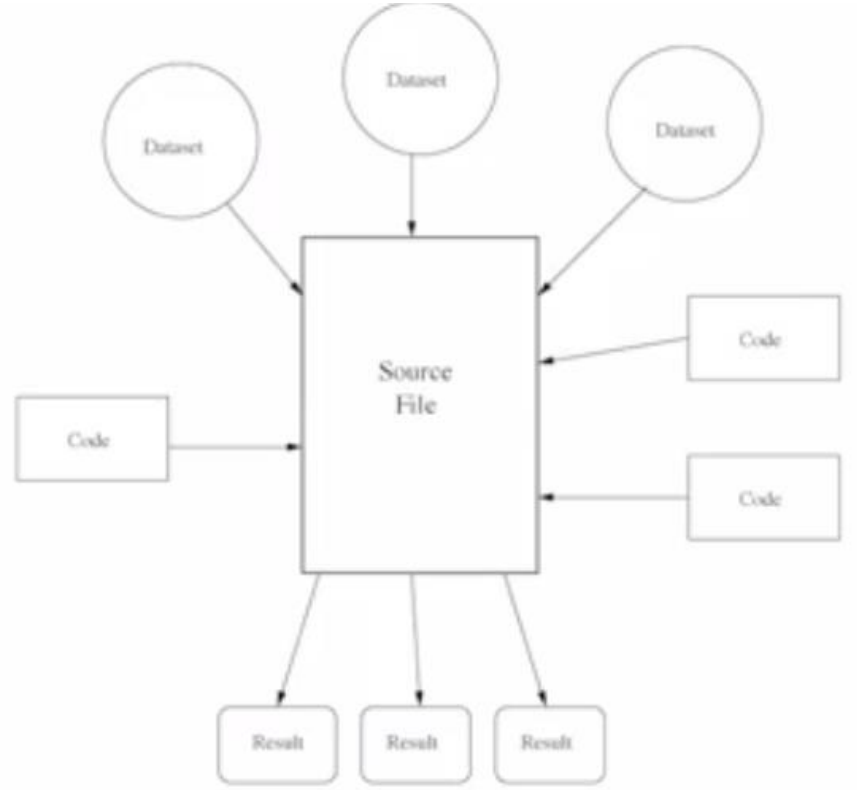
Caching Computations



Cacher package for R

- Add-on package for R
- Evaluates code written in files and stores intermediate results in a key-value database
- R expressions are given SHA-1 hash values so that changes can be tracked and code reevaluated if necessary
- “Cacher packages” can be built for distribution
- Others can “clone” an analysis and evaluate subsets of code or inspect data objects

Conceptual Model

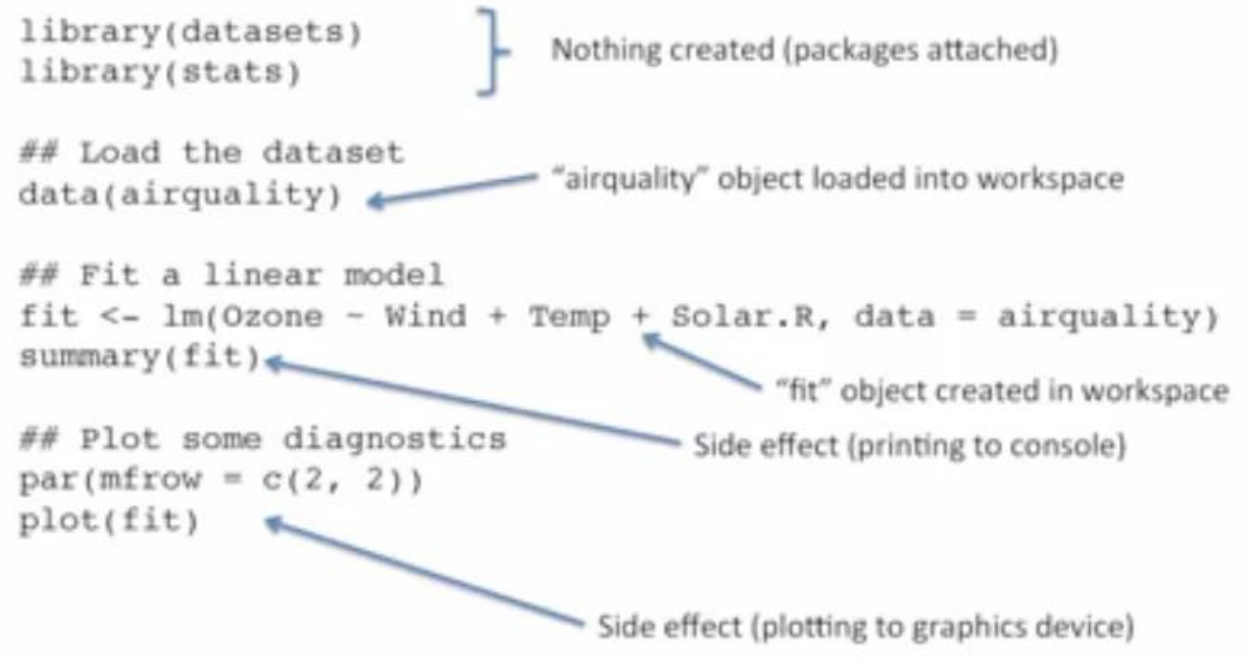


User Cacher as an Author

1. Parse the R source file; Create the necessary cache directories and subdirectories
2. Cycle through each expression in the source file:
   - If an expression has never been evaluated, evaluate it and store any resulting R objects in the cache database,
   - If a cached result exists, lazy-load the results from the cache database and move to the next expression,
   - If an expression does not create any R objects (i.e., there is nothing to cache), add the expression to the list of expressions where evaluation needs to be forced
   - Write out metadata for this expression to the metadata file.

- The `cachepackage` function creates a `cacher` package storing
  - Source file
  - Cached data objects
  - Metadata
- Package file is zipped and can be distributed
- Readers can unzip the file and immediately investigate its contents via `cacher` package

Example: Simple Analysis



Using Cacher as a Reader

A journal article says...

> "...the code and data for this analysis can be found in the cacher package 092dcc7dda4b93e42f23e038a60e1d44dbec7b3f."

```
> library(cacher)
> clonecache(id = "092dcc7dda4b93e42f23e038a60e1d44dbec7b3f")
> clonecache(id = "092d")  ## Same as above
created cache directory '.cache'

> showfiles()
[1] "top20.R"
> sourcefile("top20.R")
```

Cloning an Analysis

- Local directories created
- Source code files and metadata are downloaded
- Data objects are *not* downloaded by default
- References to data objects are loaded and corresponding data can be lazy-loaded on demand

Running Code

- The `runcode` function executes code in the source file
- By default, expressions that results in an object being created are *not* run and the resulting objects is lazy-loaded into the workspace
- Expressions not resulting in objects are evaluated

Check Code and Objects

- The `checkcode` function evaluates all expressions from scratch (no lazy-loading)
- Results of evaluation are checked against stored results to see if the results are the same as what the author calculated
  - Setting RNG seeds is critical for this to work
- The integrity of data objects can be verified with the `checkobjects` function to check for possible corruption of data (i.e. in transit)

Inspecting Data Objects

```
> loadcache()

> ls()
[1] "cities"    "classes"   "data"      "effect"
[5] "estimates" "stderr"    "vars"

> cities
/ transferring cache db file b8fd490bcf1d48cd06...
 [1] "la"   "ny"   "chic" "dlft" "hous" "phoe"
 [7] "staa" "sand" "miam" "det"  "seat" "sanb"
[13] "sanj" "minn" "rive" "phil" "atla" "oakl"
[19] "denv" "clev"
```

```
> effect
/ transferring cache db file 584115c69e5e2a4ae5...
[1] 0.0002313219

> stderr
/ transferring cache db file 81b6dc23736f3d72c6...
[1] 0.000052457
```

A 10 unit increase in PM_{10} is associated with a 0.23% increase in daily mortality

Cacher Summary

- The `cacher` package can be used by authors to create cache packages from data analyses for distribution
- Readers can use the `cacher` package to inspect others' data analyses by examining cached computations
- `cacher` is mindful of readers' resources and efficiently loads only those data objects that are needed

Reproducible Research Case Study
What Causes PM to be Toxic

- PM is composed of many different chemical elements
- Some components of PM may be more harmful than others
- Some sources of PM may be more dangerous than others
- Identifying harmful chemical constituents may lead us to strategies for controlling sources of PM

NMMAPS

- The National Morbidity, Mortality, and Air Pollution Study (NMMAPS) was a national study of the short-term health effects of ambient air pollution
- Focused primarily on particulate matter (PM_{10}) and ozone (O_3)
- Health outcomes included mortality from all causes and hospitalizations for cardiovascular and respiratory diseases
- Key publications
  - http://www.ncbi.nlm.nih.gov/pubmed/11098531
  - http://www.ncbi.nlm.nih.gov/pubmed/11354823
- Funded by the Health Effects Institute
  - Roger Peng currently serves on the Health Effects Institute Health Review Committee

NMMAPS and Reproducibility

- Data made available at the Internet-based Health and Air Pollution Surveillance System (http://www.ihapss.jhsph.edu)
- Research results and software also available at iHAPSS
- Many studies (over 67 published) have been conducted based on the public data http://www.ncbi.nlm.nih.gov/pubmed/22475833
- Has served as an important test bed for methodological development

What Causes Particulate Matter to be Toxic

Research

Cardiovascular Effects of Nickel in Ambient Air

Morton Lippmann,[1] Kazuhiko Ito,[1] Jing-Shiang Hwang,[2] Polina Maciejczyk,[1] and Lung-Chi Chen[1*]*

[1]New York University School of Medicine, Nelson Institute of Environmental Medicine, Tuxedo, New York, USA; [2]Insti Science, Academia Sinica, Taipei, Taiwan

http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1665439/

- Lippmann *et al.* found strong evidence that Ni modified the short-term effect of PM_{10} across 60 US communities
- No other PM chemical constituent seemed to have the same modifying effect
- To simple to be true?

Reanalysis of the Lippmann *et al.* Study

Research

Does the Effect of PM_{10} on Mortality Depend on PM Nickel and Vanadium Content? A Reanalysis of the NMMAPS Data

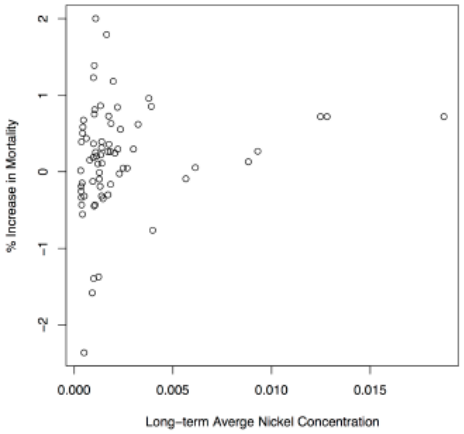
Francesca Dominici,[1] Roger D. Peng,[1] Keita Ebisu,[2] Scott L. Zeger,[1] Jonathan M. Samet,[3] and Michelle L. Bell[2]

[1]Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health, Baltimore, Maryland, USA; [2]School of Forestry and Environmental Studies, Yale University, New Haven, Connecticut, USA; [3]Department of Epidemiology, Johns Hopkins Bloomberg School of Public Health, Baltimore, Maryland, USA

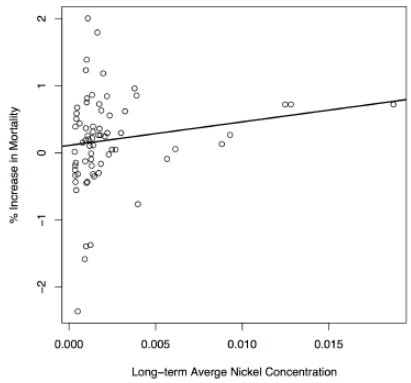
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2137127/

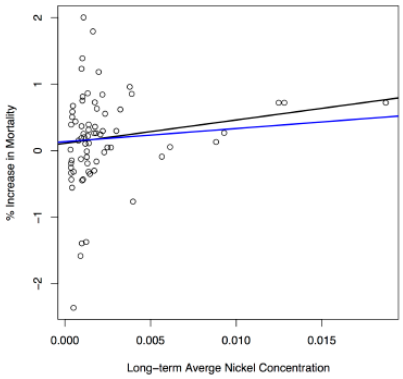
- Reexamine the data from NMMAPS and link with PM chemical constituent data
- Are the findings sensitive to levels of Nickel in New York City?

Does Nickel Make PM Toxic?

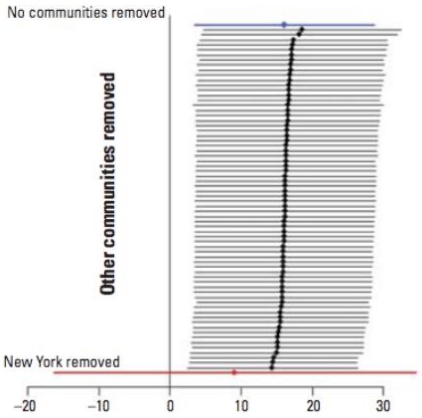


- Long-term average nickel concentrations appear correlated with PM risk
- There appear to be some outliers on the right-hand side (New York City)





- Regression line statistically significant ($p < 0.01$)
- Adjusted regression line (blue) no longer statistically significant ($p < 0.31$)



What Have We Learned?

- New York does have very high levels of nickel and vanadium, much higher than any other US community
- There is evidence of a positive relationship between Ni concentrations and PM_{10} risk
- The strength of this relationship is highly sensitive to the observations from New York City
- Most of the information in the data is derived from just 3 observations
- Reproducibility of NMMAPS allowed for a secondary analysis (and linking with PM chemical constituent data) investigating a novel hypothesis (Lippmann *et al.*)
- Reproducibility also allowed for a critique of that new analysis and some additional new analysis (Dominici *et al.*)
- Original hypothesis not necessarily invalidated, but evidence not as strong as originally suggested (more work should be done)
- Reproducibility allows for the scientific discussion to occur in a timely and informed manner
- This is how science works