# Computing tools for a Don't Repeat Yourself data analysis workflow and reproducible research

Peter Baker
School of Public Health
< p.baker1@uq.edu.au >

6 Dec 2018

# Section 0

# Table of Contents

# My Background

- Many years as a statistical consultant
  - for NSW Agriculture, CSIRO, UQ Public Health
  - to agricultural, genetics, medical and epidemiological researchers
- Statistical software
  - GENSTAT, Minitab, SAS, SPSS, STATA, S, BUGS, JAGS, . . .
  - R (almost) exclusively since 1998
- Other software for managing data analysis/reporting
  - make & version control (cvs, svn, git)
  - literate programming: sweave, knitr, rmarkdown, . . .

# Real world consulting

Are these scenarios familiar?

- I have a very simple question that will only take 5 minutes. I won't need to see you again

# Real world consulting

Are these scenarios familiar?

- I have a very simple question that will only take 5 minutes. I won't need to see you again
- We have several data points that need deleting. Can you rerun the analysis, insert the new tables and plot into our report by 4pm today?

# Real world consulting

Are these scenarios familiar?

- I have a very simple question that will only take 5 minutes. I won't need to see you again
- We have several data points that need deleting. Can you rerun the analysis, insert the new tables and plot into our report by 4pm today?
- The journal got back to us: Can you rerun the analysis to take account critisicms of our method? Its not the project we did last year but the one in 2014

## Real world consulting

No matter what clients/funders/bosses say, what happens is often very different

**All** these situations need to be well organised and well documented

Standardised systems help

Additionally, good computing tools help this process too

# Section 1

## Data Analysis Workflow

# A *DRY* creek near home

# DRY versus WET workflows

- *DRY:*
  - Don't Repeat Yourself

# DRY versus WET workflows

- *DRY:*
  - Don't Repeat Yourself
- *WET:*
  - Write Everything Twice
  - We Enjoy Typing
  - Waste Everyone's Time
- Copy-cut-and-paste writing/reporting is *WET*

# Workflow of data analysis cycle

1. Plan
2. Document
3. Organise
4. Carry out analysis
5. Communicate results
6. Iterate through steps 1 to 5 and refine process

Long provides a good overview for *Stata* (Long 2009)

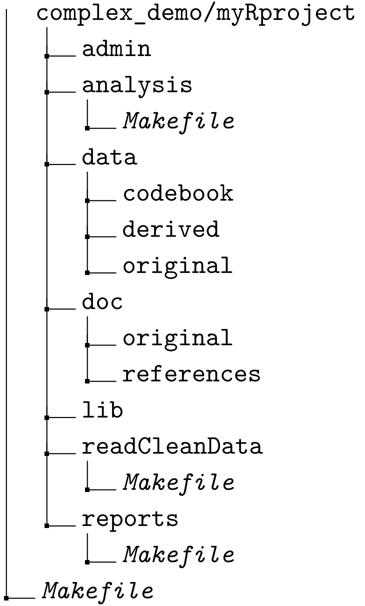
# Workflow of data analysis and reporting

- Efficiency
- Simplicity
- Standardisation
- Automation
- Usability
- Scalability
- Collaboration

# Modularisation

- break large project into smaller manageable chunks
- follow Unix paradigm: each syntax file does one job
- standard directory structure
  - minimal but informative names
  - consistent across projects
- standardised filenames
  - minimal but informative names
  - consistent across projects
- follow a style guide
  - Google R Style Guide
  - Advanced R Style Guide
  - Bioconductor Style Guide
  - many others

# Complex project directory structure



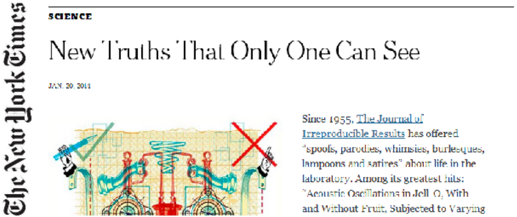
```
complex_demo/myRproject
├── admin
├── analysis
│   └── Makefile
├── data
│   ├── codebook
│   ├── derived
│   └── original
├── doc
│   ├── original
│   └── references
├── lib
├── readCleanData
│   └── Makefile
├── reports
│   └── Makefile
└── Makefile
```

Section 2

# Reproducible Research

# Reproducibility in Popular Press

popular press



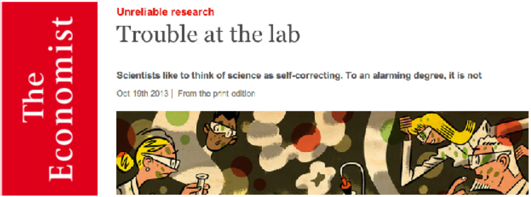
The New York Times

SCIENCE

New Truths That Only One Can See

JAN. 20, 2014

Since 1955, The Journal of Irreproducible Results has offered "spoofs, parodies, whimsies, burlesques, lampoons and satires" about life in the laboratory. Among its greatest hits: "Acoustic Oscillations in Jell-O, With and Without Fruit, Subjected to Varying



The Economist

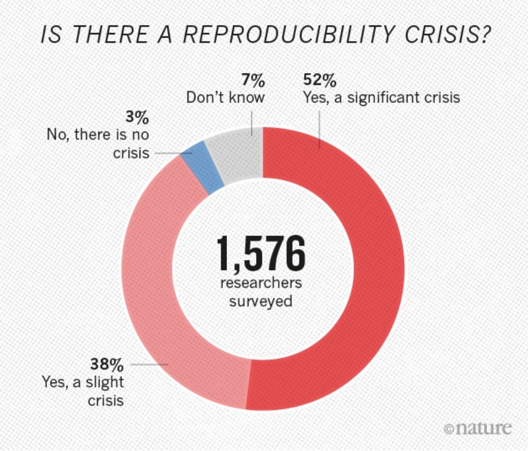
Unreliable research

Trouble at the lab

Scientists like to think of science as self-correcting. To an alarming degree, it is not

Oct 19th 2013 | From the print edition

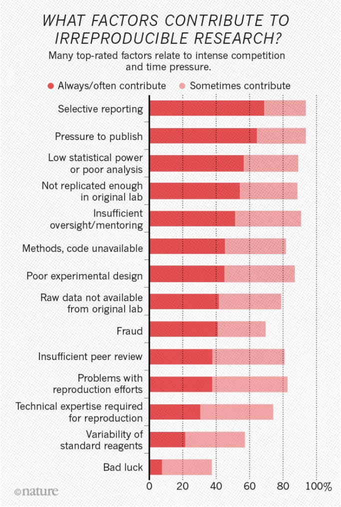
# Reproducibility



1,500 scientists lift
the lid on reproducibility *Nature*

Source: Monya M. Baker (2016b)

# Reproducibility

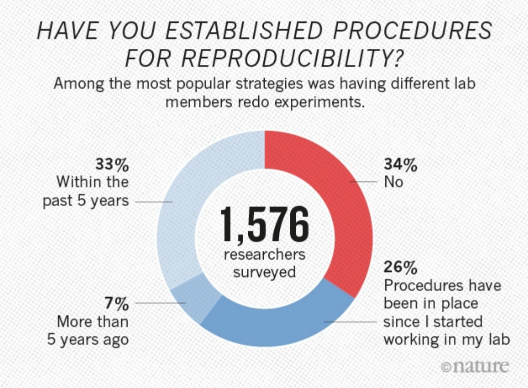


1,500 scientists . . . *Nature*

- John Ioannidis (2005) “Most published scientific findings are false”
- Monya Baker (2016b) highlights contribution to irreproducibility:
  - Methods, code unavailable
  - Raw data not available
  - Problems with reproduction efforts
- Monya Baker (2016a) QA crucial in lab

Source: M. Baker (2016b), Ioannidis (2005), M. Baker (2016a)

# Reproducibility



HAVE YOU ESTABLISHED PROCEDURES FOR REPRODUCIBILITY?

Among the most popular strategies was having different lab members redo experiments.

1,500 scientists lift the lid on reproducibility *Nature*

Source: Monya M. Baker (2016b)

34% in the lab, **higher** in data analysis?

# Reproducibility

Statisticians and data scientists can contribute to:

- Study design and analysis
- Understanding variability
- Reproducible analysis and reporting

# Why move from manual to computer aided approaches?

"A reproducible workflow"
by Ignasi Bartomeus and Francisco Rodríguez-Sánchez

Figure 1: https://youtu.be/s3JldKoA0zw

# Workflow of data analysis and reporting

- Efficiency
- Simplicity
- Standardisation
- Automation
- Usability
- Scalability
- Collaboration

*GNU R*, *GNU Make*, *(GNU) Git*, *GNU Bash*, *Good IDE*, . . .
can help with many of these

# Computing Tools: Projects

- RStudio
- Emacs
- . . .

Organise files in directories/subdirectories

Jump between projects

Start where you left off last time

Other *convenience* features

# Computing Tools: Automation

- write shell scripts, Makefiles, R functions, R packages to automate routine work
- standard directory structure
  - many projects can use same directory structure
  - can create directories using R or shell script
- can also create
  - Makefiles automatically
  - Git repos automatically
  - R syntax automatically or reuse R syntax

# Computing Tools: Rerunning analysis

- manually
  - need to document steps heavily
  - still may forget something
- **GNU Make**
  - automates
  - only rerun steps needed
  - keeps track of the process
    - but need to read *make*

# Computing Tools: Version Control/Collaboration

- **Git**
  - even for one statistician
  - several statisticians
  - clients too

# Computing Tools: literate programming for reports

- **R Markdown**
  - Document/Presentation and syntax in one file
  - Process to run syntax and insert output in document
  - Text, syntax, bibliography, references, images, maths, lists . . .

Section 3

# GNU Make

# Make and reproducible research

> *I would argue that the most important tool for reproducible research is not Sweave or knitr but GNU make.*
> *Karl Broman*
> *Source: https://kbroman.org/minimal_make/*

Many talks I've seen tout R Markdown as being the basis of reproducible research but statisticians don't just write simple reports. . .

I would argue that the three most useful tools we can use to aid the data analysis workflow and facilitate reproducible research are

1. GNU Make
2. Git
3. R Markdown

or alternatives

# GNU Make

- Make was originally developed for compiling large complex programs in C, FORTRAN and assembler
- In software projects only files changed are recompiled and new executable made.
- In 1990s Bob Forrester at CSIRO pointed out we could manage data analysis projects the same way using GENSTAT
- Useful approach even though computers are a lot faster now. Unnecessarily rerunning a huge simulation or analysis is still inefficient
- Works in tandem with `git` to use GNU Make to regenerate only required output and intermediate files for data analysis and reporting projects

# Why GNU Make?

- defacto standard
- can use GNU Make to (re)run anything you can run from command line
- modular operation - break down into smaller tasks so facilitates reproducible research (reporting)
- we specify what depends on what and then *make* only updates necessary files
- also documents workflow
- type make at command line or press button in RStudio/IDE

# Targets and dependencies

Makefiles specify target files and dependency files:

```
target_file: dependency_file_1 dependency_file_2 ...
<TAB> command 1
<TAB> command 2
<TAB> command 3
```

- make compares the times that files were saved
- if dependencies are 'newer' than targets then commands are run

Note that command lines begin with a tab not spaces

**WWW: Be careful if cutting and pasting from webpages: TABS become SPACES**

# Targets and dependencies

Here is a simple `Makefile` that we might use just to read the data:

```
read.Rout: read.R bmi2009.dta
<TAB> R CMD BATCH read.R
```

- make compares the times that files were saved
- if dependencies are ‘newer’ than targets then R BATCH command is run
- `read.Rout` is **target** on LHS :
- `read.R` and `bmi2009.dta` are **dependencies**

# Running make

If either *read.R* or *bmi2009.dta* changes

- target *read.Rout* will be older
- regarded as being **out of date**

Run *make* by typing *make* at the command line or pressing the appropriate button in your *IDE*

If *read.R* newer, *R CMD BATCH read.R* is run

If *read.Rout* is newer, then

```
$ make
make: 'read.Rout is up to date'.
```

# Pattern Rules

GNU Make has pattern rules for many languages
(C, C++, Fortran, Ratfor, Yacc, Lex, Info Texinfo, Tex)

**Problem:** *GNU Make* does not have rules for statistical languages like *R*, *Stata*, *SPSS*, *SAS*, *GENSTAT*, . . .

**Solution:** Define pattern rules, eg

```
%.Rout: %.R
<TAB> R CMD BATCH $<
```

Pattern rules look pretty much like normal rules except

- the wild card symbol % is used before the file extension
- $< is automatic variable: the filename of first dependency

# In practice

- don't need to write pattern rules every time
- *include* rules from a file
- a selection of rules available at github (Baker 2019) https://github.com/petebaker/r-makefile-definitions

Simply **include r-rules.mk** at end of file

*include ~/lib/r-rules.mk*

or similarly on Windows

```
include C:/MyLibrary/r-rules.mk
```

or in system wide directory like `/usr/local/include`

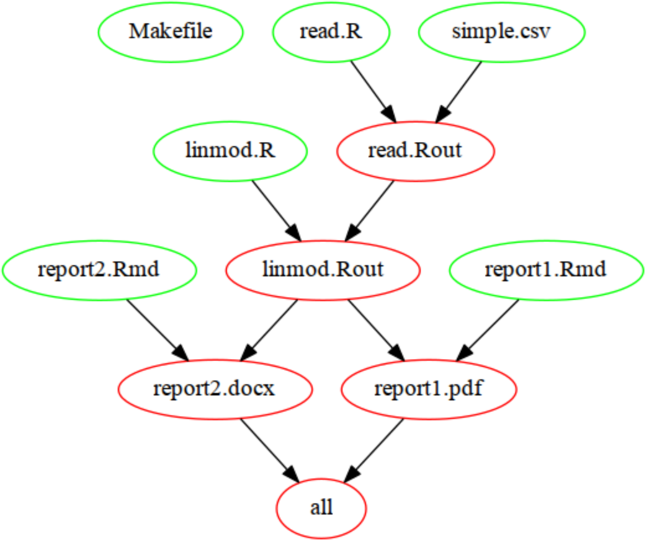
*include r-rules.mk*

Also included in *dryworkflow* package at https://github.com/petebaker/dryworkflow but needs revision

# Simple Makefile

```
## File:    Makefile
## Purpose: Simple Example

.PHONY: all
all: report1.pdf report2.docx

## reports 1&2 depend on results of 'linmod.Rout' & '*.Rmd'
report1.pdf: report1.Rmd linmod.Rout
report2.docx: report2.Rmd linmod.Rout

## data analysis:  dependent on 'linmod.R' and 'read.Rout'
linmod.Rout: linmod.R read.Rout

## read in data: depends on 'read.R' and 'simple.csv'
read.Rout: read.R simple.csv

## include R pattern rule definitions from file
include r-rules.mk
```

# Dependency file graph



# r-rukes.mk rules

Pattern rules provided for

- Statistics packages (and related)
  - R
  - Sweave
  - R Markdown
  - Stata
  - SAS
  - PSPP
- Data science
  - Python
  - Perl

Caveat: *Windows* and *macOS* users may need a better GNU Make

- Win: https://github.com/mbuilov/gnumake-windows
- macOS: install via homebrew https://brew.sh/

Section 4

# Git

# Reflection: How do I work alone and with others?

- Do I keep a track of all my (computer) projects?
  - separate folders/directories for a project?
  - consistent filenames?
  - versioning?
    eg `plot_001.R`, . . ., `plot_final.R`, `plot_final2.R`

# Reflection: How do I work alone and with others?

- Do I keep a track of all my (computer) projects?
  - separate folders/directories for a project?
  - consistent filenames?
  - versioning?
    eg `plot_001.R`, . . ., `plot_final.R`, `plot_final2.R`
- How do I collaborate on
  - data management?
  - data analysis?
  - writing reports and papers?

# Reflection: How do I work alone and with others?

- Do I keep a track of all my (computer) projects?
  - separate folders/directories for a project?
  - consistent filenames?
  - versioning?
    eg `plot_001.R`, ..., `plot_final.R`, `plot_final2.R`
- How do I collaborate on
  - data management?
  - data analysis?
  - writing reports and papers?
- How do I share data, manuscripts, programs?
  - with my team?
  - with others?

# Version Control

Version Control has been used by programmers for many years. Known generically as

- version control system (VCS)
- source code manager (SCM)
- revision control system (RCS)

Used for both keeping track of code and collaborating on programming projects

# Version Control

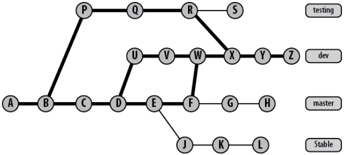
Statisticians only using it recently (some of us since early 90s)

History:

- Early 1980s: Revision Control System (RCS)
- 1986: Concurrent Version System (CVS)
- 1990: CVS greatly improved
- 2001: Subversion (SVN)
- April 2005: Linus Torvalds wrote ‘git’
  (like ‘linux’ he names it after himself $\ddot\smile$ )
- 2013?: RStudio introduced *git* and *svn* support

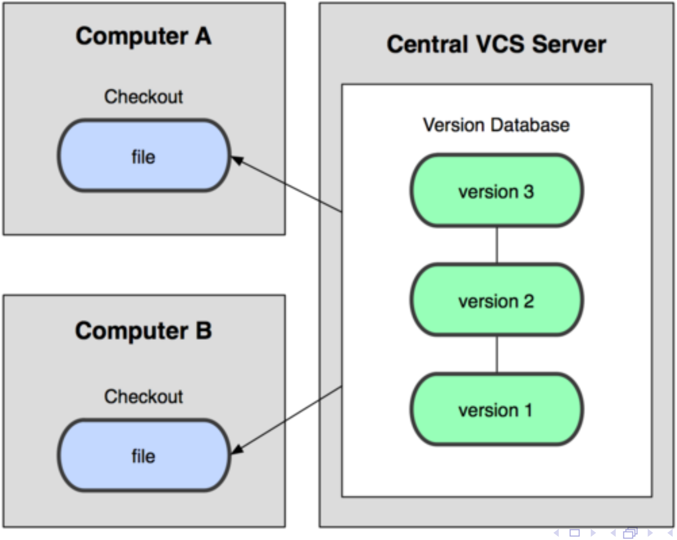
# Version Control

- records changes to a file or set of files over time
- not just programs but *any* file(s)
- revert file(s) back to previous state
- revert entire project back to previous state
- compare changes over time
- see who changed what
- can create experimental branches and only merge back if changes work



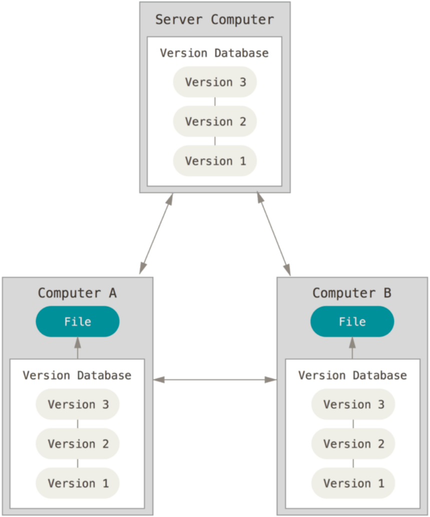
# Central Server Systems

Old style CVS, SVN, R-forge have central server model.

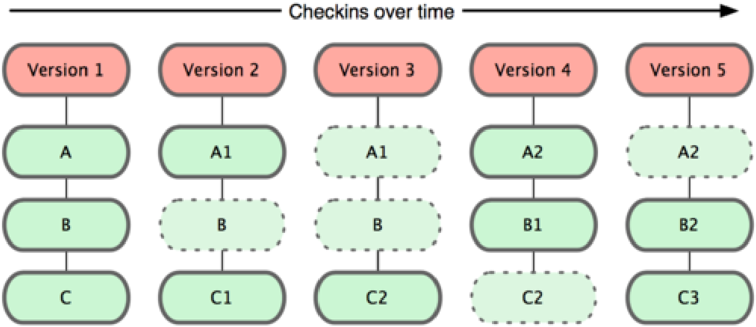


# Distributed Model

*Git* is derived from this peer-to-peer model.



# Snapshots over time



# git documentation

Very good documentation freely available or see Loeliger and McCullough (2012)}

- Pro git book https://git-scm.com/book/en/v2
- RStudio https://support.rstudio.com/hc/en-us/articles/200532077
- Cheatsheets: https://education.github.com/git-cheat-sheet-education.pdf
- githib, bitbucket or TowerGitWorkflow cheatsheets

# Useful git commands

You could do everything you need to with a few basic commands in a terminal

```
git init               # set up initial (local) repository
git add read.R         # add file
git add data/*.csv     # add files
# commit changes
git commit -a -m 'Initial project repository'
## even clone from internet
git clone git://github.com/pretend/grit-pretend.git

## to see which files have been changed or not tracked
git status
```

But just use *RStudio* or *Magit* or *GUI* instead (all easier)

# Remote Repositories

Git can use four distinct protocols to transfer data: Local, HTTP, Secure Shell (SSH) and Git.

You can set up remote repositories for free but please **be aware of any restrictions about storing research data on public servers**

Some public/private but some you need to pay for private repositories.

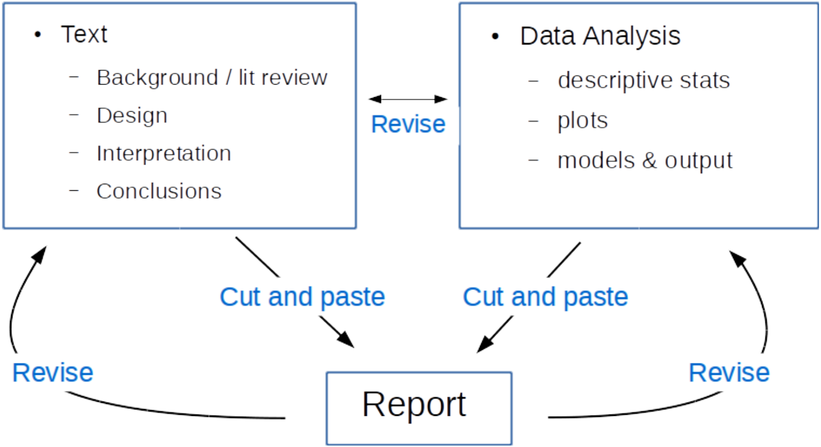
- github: https://github.com
- gitlab: https://gitlab.com
- bitbucket: https://bitbucket.com

Commands: ‘git push’ and ‘git pull’

Section 5

# R Markdown

# Traditional ~~World~~ Word + Menu Driven Stats



- Text
  - Background / lit review
  - Design
  - Interpretation
  - Conclusions

- Data Analysis
  - descriptive stats
  - plots
  - models & output

# Traditional ~~World~~ Word + Menu Driven Stats

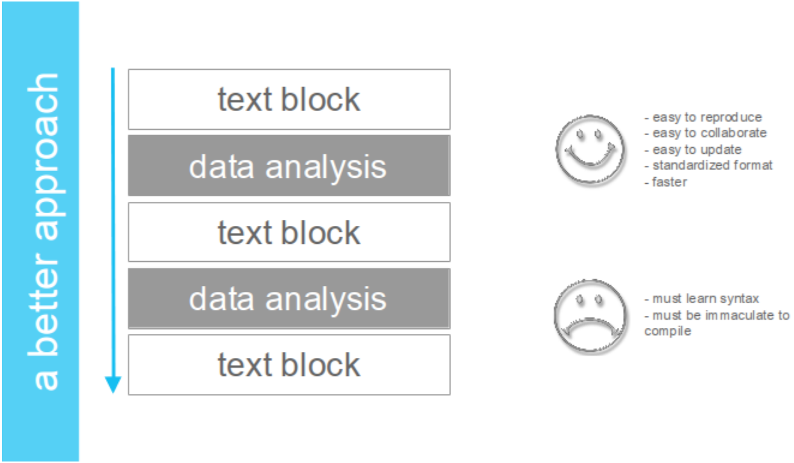
Pros ☺

- familiar format (Word/Powerpoint/SPSS/. . . )

Cons ☹

- impossible to reproduce
- very difficult to update
- very easy for mistakes to creep in
- messy

# A better approach



a better approach

text block

data analysis

text block

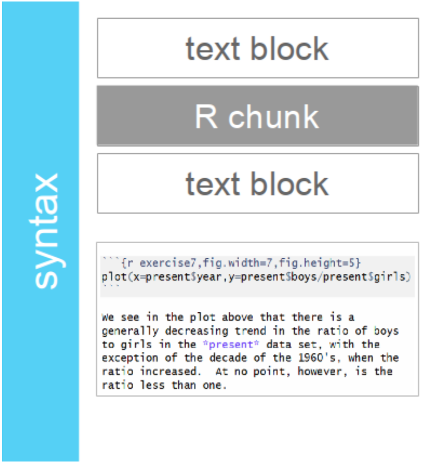
data analysis

text block

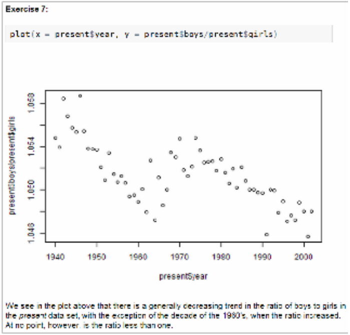
- easy to reproduce
- easy to collaborate
- easy to update
- standardized format
- faster

- must learn syntax
- must be immaculate to compile

# Literate programming (Sweave, R Markdown, Org, . . . )



~~~
```{r exercise7,fig.width=7,fig.height=5}
plot(x=present$year,y=present$boys/present$girls)
```

We see in the plot above that there is a
generally decreasing trend in the ratio of boys
to girls in the *present* data set, with the
exception of the decade of the 1960's, when the
ratio increased.  At no point, however, is the
ratio less than one.
~~~

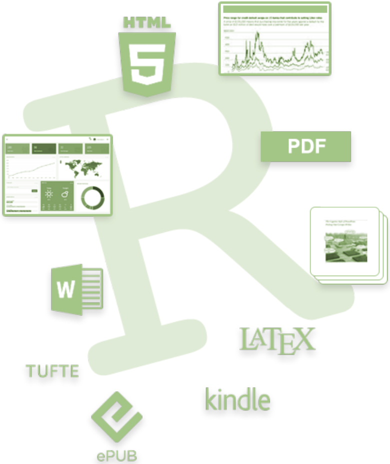


**Exercise 7:**

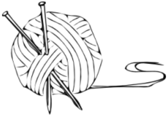
```
plot(x = present$year, y = present$boys/present$girls)
```

We see in the plot above that there is a generally decreasing trend in the ratio of boys to girls in the *present* data set, with the exception of the decade of the 1960's, when the ratio increased. At no point, however, is the ratio less than one.

# Why use R Markdown?



# How do we do it?

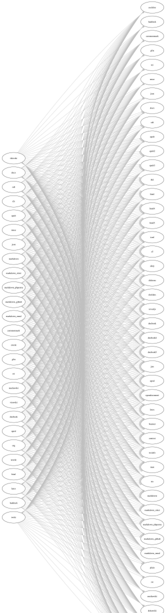
+ + 

or even

+

 +
The TeXbook
TEX

# R Markdown uses Pandoc http://pandoc.org



# R code chunk in R Markdown

Type this into your *.Rmd* R Markdown file

~~~
```{r reg1}
set.seed(12345)                # set RNG seed
x <- 1:30                      # x is 1, 2, 3, ..., 30
y <- 2 + 1.5*x + rnorm(30)     # simulated y
(lm1 <- lm(y ~ x))             # fit regression
```
~~~

Options:

- supress syntax
- don't run syntax
- fonts and sizes
- figure heights, widths
- captions etc etc

**NB: chunks must have a unique name** (here reg1)

# Output from R code chunk

Standard output in HTML, Word Doc, PDF, RTF, ODT, ...

```r
set.seed(12345)            # set RNG seed
x <- 1:30                  # x is 1, 2, 3, ... 30
y <- 2 + 1.5*x + rnorm(30) # simulated y
(lm1 <- lm(y ~ x))         # fit regression
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##       2.030        1.503
```

# Output from R code chunk (Changing options)

Supress R code altogether

- Option: {r, comment="", **echo = FALSE**}

```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
      2.030        1.503
```

NB: May need to supress *warnings*, *message*, *error* (Default: TRUE}

# Reuse code chunk

Can reuse the code from chunk **reg1** with << **reg1** >>

~~~
```{r}
<<reg1>>
```
~~~

```
set.seed(12345)                # set RNG seed
x <- 1:30                      # x is 1, 2, 3, ... 30
y <- 2 + 1.5*x + rnorm(30)     # simulated y
(lm1 <- lm(y ~ x))             # fit regression
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##       2.030        1.503
```

# Display results inline

- Display results inline with `` `r expression` ``, eg

  Slope = `` `r lm1$coefficients['x']` ``

  which displays as

  Slope = 1.5031174

  or

  Slope = `` `r round(lm1$coeff['x'],3)` ``

  which displays as

  Slope = 1.503

# Plots from R chunks

~~~
``{r, fig.height=3.2, fig.width=3.5, fig.cap = 'Simple line
plot(y ~ x)
abline(lm1)
```
~~~

Large number of chunk options:

- eval, echo, results, tidy, etc
- fig.height, fig.width, fig.align, fig.cap

# Plots from R chunks

```r
plot(y ~ x)
abline(lm1)
```

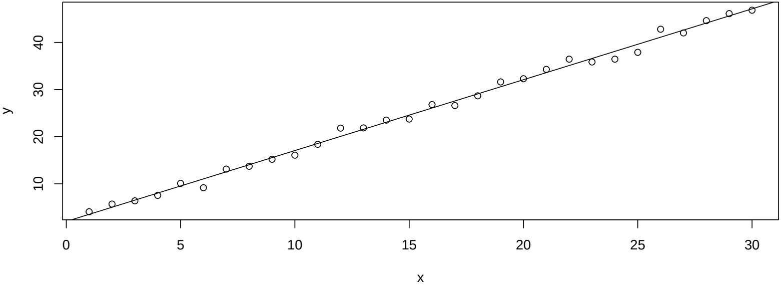


Figure 2: Simple linear regression

# Publication Quality Tables, Data, . . . using kable

```r
knitr::kable(anova(lm1))
```

| | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
|---|---:|---:|---:|---:|---:|
| x | 1 | 5077.91610 | 5077.9160952 | 5574.457 | 0 |
| Residuals | 28 | 25.50592 | 0.9109257 | NA | NA |

```r
options(knitr.kable.NA = '')
kable(anova(lm1), digi = 2, caption = "ANOVA table")
```

Table 2: ANOVA table

| | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
|---|---:|---:|---:|---:|---:|
| x | 1 | 5077.92 | 5077.92 | 5574.46 | 0 |
| Residuals | 28 | 25.51 | 0.91 | | |

# Most standard word processing features

- Works for WORD, HTML, PDF, ODT, . . .
- Text (bold, italics, superscripts, subscripts,. . . )
- Lists
- Headings (`# Header 1`, `## Header 2, ...` )
- Links (URLS, files, . . . )
- insert image files via knitr or pandoc
- citations & referencing (`@smith04` [p. 33] says blah.)
- Equations (inline and equations using LaTeX \$. . . \$) $\sum_{i=1}^{n} X_i$

$$\sum_{i=1}^{n} X_i$$

# Output formats

Some of the output formats that can be produced from R Markdown files

- beamer_presentation (presentation)
- github_document (web page)
- html_document (web page)
- ioslides_presentation (presentation)
- latex_document (markup file)
- md_document (markdown file)
- odt_document (document)
- pdf_document (document)
- powerpoint_presentation (presentation)
- rtf_document (rich text format)
- slidy_presentation (presentation)
- word_document (document)
- Shiny (interactive web apps)
- Dashboards (flexdashboard)

## Language Engines (inputs)

R Markdown can also produce highlighted syntax and output from running other languages

- Python
- Shell scripts (Bash)
- SQL
- Rcpp
- Stan
- JavaScript and CSS
- Julia
- C and Fortran

And (perhaps) more limited:

- SAS
- Stata

## On-line resources

- Can install minimal LaTeX by installing **R** package **tinytex**
- Some options only work for for particular output types
- Well documented online and in cheat sheets

References:

- R Markdown cheatsheet
- R Markdown Reference Guide
- R Markdown: The Definitive Guide: R code options
- https://yihui.name/knitr/options/

# Using R Scipts

- As a starting point: R script usually easier than R Markdown
- Chapter 20 Render an R script http://happygitwithr.com/r-test-drive.html
- in R Markdown: text is top-level and R is in chunks
- in R Script: R is top-level and text is in chunks

Section 6

# Conclusions

# Summary: GNU Make

- GNU Make useful for efficient modular workflow
- Make documents workflow
- Recursive Make may be problematic (Miller 1998)
    - I keep this relatively simple to avoid problems
    - In practice not an issue since GNU Make 4.0
        - can write non-recusive solution
        - recursive solution possible but trickier
- User written and built-in functions available
- Many alternative build systems but few mature or used widely (eg see Drake, Remake Scons)
- Good references
    - GNU Make manual
    - Graham-Cumming (2015)
    - Mecklenburg (2004)

## Summary: Version control

- Git best currently
- useful for solo or group projects
  - local or remote repos
- Git documents changes
  - overall
  - at a file level
- easier to use GUI: RStudio, Emacs Magit, . . .
- only need to use the basics

# Summary: Literate programming using R Markdown

- Document/Presentation and syntax in one file
- Process to run syntax and insert output in document
- Text, syntax, bibliography, references, images, maths, lists . . .

Lots of good online documentation (and books)

- Xie (2016a) https://bookdown.org/
- Xie, Allaire, and Grolemund (2018) html
- Xie (2016b)
- Gandrud (2016)

## Conclusions

I would argue that the three most useful tools we can use to aid the data analysis workflow and facilitate reproducible research are

1. GNU Make
2. Git
3. R Markdown

or alternatives

While there is always a trade off, learning these tools and also specialised tools like writing R functions, R Packages, GENSTAT Procedures, shell scripts, regular expressions, . . . may aid efficiency in the long run

# References

Baker, Monya. 2016a. "How Quality Control Could Save Your Science." *Nature News* 529 (7587): 456. https://doi.org/10.1038/529456a.

———. 2016b. "1,500 Scientists Lift the Lid on Reproducibility." *Nature News* 533 (7604): 452. https://doi.org/10.1038/533452a.

Baker, Peter. 2019. "Using GNU Make to Manage the Workflow of Data Analysis Projects." *Journal of Statistical Software (Accepted Nov 2018)*.

Gandrud, Christopher. 2016. *Reproducible Research with R and R Studio, Second Edition*. 2nd Ed. CRC Press.

Graham-Cumming, John. 2015. *The GNU Make Book*. No Starch Press.

Ioannidis, John P. A. 2005. "Why Most Published Research Findings Are False." *PLOS Medicine* 2 (8): e124. https://doi.org/10.1371/journal.pmed.0020124.

Loeliger, Jon, and Matthew McCullough. 2012. *Version Control with Git: Powerful Tools and Techniques for Collaborative Software Development*. 2nd ed. O'Reilly Media, Inc.

Long, J. Scott. 2009. *The Workflow of Data Analysis Using Stata*. StataCorp LP.

Mecklenburg, Robert. 2004. *Managing Projects with GNU Make*. 3rd ed. O'Reilly Media, Inc.

Miller, Peter. 1998. "Recursive Make Considered Harmful." *AUUGN Journal of AUUG Inc* 19 (1): 14–25. http://www.unix-ag.uni-kl.de/svn/kbibtex/kbibtex/tags/release-0.1/admin/unsermake/doc/auug97.pdf.

Xie, Yihui. 2016a. *Bookdown: Authoring Books and Technical Documents with R Markdown*. Chapman; Hall/CRC.

———. 2016b. *Dynamic Documents with R and Knitr*. 2nd Ed. CRC Press.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. CRC Press.