

# Best Practices for the Political Scientist

J. Alexander Branham

Fall 2015

## Why best practices?

- ▶ Traditionally, scholars have separated data analysis and the documentation describing the results and findings

## Why best practices?

- ▶ Traditionally, scholars have separated data analysis and the documentation describing the results and findings
- ▶ This makes updating work inefficient and reproduction difficult or impossible

## Why best practices?

- ▶ Traditionally, scholars have separated data analysis and the documentation describing the results and findings
- ▶ This makes updating work inefficient and reproduction difficult or impossible
- ▶ Why best practices?

# Why best practices?

- ▶ Traditionally, scholars have separated data analysis and the documentation describing the results and findings
- ▶ This makes updating work inefficient and reproduction difficult or impossible
- ▶ Why best practices?
- ▶ To solve many recurring issues with research

# Why best practices?

- ▶ Traditionally, scholars have separated data analysis and the documentation describing the results and findings
- ▶ This makes updating work inefficient and reproduction difficult or impossible
- ▶ Why best practices?
- ▶ To solve many recurring issues with research
  - ▶ Difficulty recreating a particular method

# Why best practices?

- ▶ Traditionally, scholars have separated data analysis and the documentation describing the results and findings
- ▶ This makes updating work inefficient and reproduction difficult or impossible
- ▶ Why best practices?
- ▶ To solve many recurring issues with research
  - ▶ Difficulty recreating a particular method
  - ▶ Difficulty recreating a figure

# Why best practices?

- ▶ Traditionally, scholars have separated data analysis and the documentation describing the results and findings
- ▶ This makes updating work inefficient and reproduction difficult or impossible
- ▶ Why best practices?
- ▶ To solve many recurring issues with research
  - ▶ Difficulty recreating a particular method
  - ▶ Difficulty recreating a figure
  - ▶ Difficulty recreating a table



# Why best practices?

- ▶ Traditionally, scholars have separated data analysis and the documentation describing the results and findings
- ▶ This makes updating work inefficient and reproduction difficult or impossible
- ▶ Why best practices?
- ▶ To solve many recurring issues with research
  - ▶ Difficulty recreating a particular method
  - ▶ Difficulty recreating a figure
  - ▶ Difficulty recreating a table
  - ▶ Steps taken in code not reported in paper

# Why best practices?

- ▶ Traditionally, scholars have separated data analysis and the documentation describing the results and findings
- ▶ This makes updating work inefficient and reproduction difficult or impossible
- ▶ Why best practices?
- ▶ To solve many recurring issues with research
  - ▶ Difficulty recreating a particular method
  - ▶ Difficulty recreating a figure
  - ▶ Difficulty recreating a table
  - ▶ Steps taken in code not reported in paper
    - ▶ Example: dropping oversamples

# Why best practices?

- ▶ Traditionally, scholars have separated data analysis and the documentation describing the results and findings
- ▶ This makes updating work inefficient and reproduction difficult or impossible
- ▶ Why best practices?
- ▶ To solve many recurring issues with research
  - ▶ Difficulty recreating a particular method
  - ▶ Difficulty recreating a figure
  - ▶ Difficulty recreating a table
  - ▶ Steps taken in code not reported in paper
    - ▶ Example: dropping oversamples
- ▶ Solution: **Reproducible research**

# Why best practices?

- ▶ Traditionally, scholars have separated data analysis and the documentation describing the results and findings
- ▶ This makes updating work inefficient and reproduction difficult or impossible
- ▶ Why best practices?
- ▶ To solve many recurring issues with research
  - ▶ Difficulty recreating a particular method
  - ▶ Difficulty recreating a figure
  - ▶ Difficulty recreating a table
  - ▶ Steps taken in code not reported in paper
    - ▶ Example: dropping oversamples
- ▶ Solution: **Reproducible research**
  - ▶ Helps future people trying to extend your work

# Why best practices?

- ▶ Traditionally, scholars have separated data analysis and the documentation describing the results and findings
- ▶ This makes updating work inefficient and reproduction difficult or impossible
- ▶ Why best practices?
- ▶ To solve many recurring issues with research
  - ▶ Difficulty recreating a particular method
  - ▶ Difficulty recreating a figure
  - ▶ Difficulty recreating a table
  - ▶ Steps taken in code not reported in paper
    - ▶ Example: dropping oversamples
- ▶ Solution: **Reproducible research**
  - ▶ Helps future people trying to extend your work
  - ▶ Helps coauthors see what you're doing

# Why best practices?

- ▶ Traditionally, scholars have separated data analysis and the documentation describing the results and findings
- ▶ This makes updating work inefficient and reproduction difficult or impossible
- ▶ Why best practices?
- ▶ To solve many recurring issues with research
  - ▶ Difficulty recreating a particular method
  - ▶ Difficulty recreating a figure
  - ▶ Difficulty recreating a table
  - ▶ Steps taken in code not reported in paper
    - ▶ Example: dropping oversamples
- ▶ Solution: **Reproducible research**
  - ▶ Helps future people trying to extend your work
  - ▶ Helps coauthors see what you're doing
  - ▶ Helps you in six months (or tomorrow)

## A Quick Overview...

1. Write using plain text

## A Quick Overview...

1. Write using plain text
2. Code well



## A Quick Overview...

1. Write using plain text
2. Code well
3. Use a version control system

# Write using plain text!

- ▶ What if we write an article now that gets famous?

## Write using plain text!

- ▶ What if we write an article now that gets famous?
- ▶ 20 years later, some grad student wants to extend our work

## Write using plain text!

- ▶ What if we write an article now that gets famous?
- ▶ 20 years later, some grad student wants to extend our work
- ▶ How did we make Figure 1?

## Write using plain text!

- ▶ What if we write an article now that gets famous?
- ▶ 20 years later, some grad student wants to extend our work
- ▶ How did we make Figure 1?
- ▶ Non-plain text files may be unusable 20 years from now

## Write using plain text!

- ▶ What if we write an article now that gets famous?
- ▶ 20 years later, some grad student wants to extend our work
- ▶ How did we make Figure 1?
- ▶ Non-plain text files may be unusable 20 years from now
- ▶ Word processors (like MS Word) are stupid and inefficient

## Write using plain text!

- ▶ What if we write an article now that gets famous?
- ▶ 20 years later, some grad student wants to extend our work
- ▶ How did we make Figure 1?
- ▶ Non-plain text files may be unusable 20 years from now
- ▶ Word processors (like MS Word) are stupid and inefficient
- ▶ Bonus: plain text files are usually *much* smaller than their Word/pdf counterparts

# Code well - DO NOT CLICK A MOUSE

- ▶ All results should be replicable from a source file



# Code well - DO NOT CLICK A MOUSE

- ▶ All results should be replicable from a source file
  - ▶ And these should be named in a manner that tells you what they are

# Code well - DO NOT CLICK A MOUSE

- ▶ All results should be replicable from a source file
  - ▶ And these should be named in a manner that tells you what they are
  - ▶ e.g. `fig1.R` and not `messing-around-with-figs.R`

# Code well - DO NOT CLICK A MOUSE

- ▶ All results should be replicable from a source file
  - ▶ And these should be named in a manner that tells you what they are
  - ▶ e.g. `fig1.R` and not `messing-around-with-figs.R`
- ▶ These source files (for example, `.R` for R scripts or `.do` for Stat do-files) should be liberally commented

# Code well - DO NOT CLICK A MOUSE

- ▶ All results should be replicable from a source file
  - ▶ And these should be named in a manner that tells you what they are
  - ▶ e.g. `fig1.R` and not `messing-around-with-figs.R`
- ▶ These source files (for example, `.R` for R scripts or `.do` for Stat do-files) should be liberally commented
- ▶ Comments explain what you are doing to your future self, collaborators, and others

## Comment example

```
# This code creates Fig 1
# I use the mtcars dataset (included with R)
library(ggplot2)
ggplot(mtcars, aes(mpg, wt)) +
  geom_point() +
  geom_smooth(method="lm") # Adds OLS line with SEs
ggsave("fig/fig1.pdf")
```

- Save this code snippet as fig1.R (or similar)

# Literate programming and reproducible research

- ▶ Previous example - have two files for one plot in a paper

# Literate programming and reproducible research

- ▶ Previous example - have two files for one plot in a paper
  - ▶ The paper itself (`document.tex` or similar)

# Literate programming and reproducible research

- ▶ Previous example - have two files for one plot in a paper
  - ▶ The paper itself (`document.tex` or similar)
  - ▶ the script to create the figure (`fig1.R` or similar)



# Literate programming and reproducible research

- ▶ Previous example - have two files for one plot in a paper
  - ▶ The paper itself (`document.tex` or similar)
  - ▶ the script to create the figure (`fig1.R` or similar)
- ▶ What if we could combine these to have everything in one easy-to-read file?

# Literate programming and reproducible research

- ▶ Previous example - have two files for one plot in a paper
  - ▶ The paper itself (`document.tex` or similar)
  - ▶ the script to create the figure (`fig1.R` or similar)
- ▶ What if we could combine these to have everything in one easy-to-read file?
  - ▶ This is what literate programming is all about!

## Literate programming example (using knitr and $\text{\LaTeX}$ )

```
\section{Introduction to Literate Programming}
```

This is an example paragraph, written in  $\text{\LaTeX}$ .

Using knitr, we can include R code in the following manner.

I can reference the figure number by calling ref:

Figure  $\text{\ref{fig:mpg-and-weight}}$ .

% NOTE: updating that figure with  $x^2$  doesn't change it

```
\begin{figure}
```

```
\centering
```

```
<<fig1plot, echo=FALSE>>= # Starts R code, labels it `fig1`
```

```
# I use the mtcars dataset (included with R)
```

```
library(ggplot2)
```

```
ggplot(mtcars, aes(mpg, wt)) +
```

```
  geom_point() +
```

```
  geom_smooth(method="lm") # Adds OLS line with SEs
```

```
@ % closes R code
```

```
\caption{Miles per gallon and weight}
```

```
\label{fig:mpg-and-weight}
```

```
\end{figure}
```

# Version Control

- ▶ **YOU NEED TO USE VERSION CONTROL!!!!**

## Version Control

- ▶ **YOU NEED TO USE VERSION CONTROL!!!!**
- ▶ This can be as simple as putting things in Dropbox, which enables you to recover previous file versions automatically

# Version Control

- ▶ **YOU NEED TO USE VERSION CONTROL!!!!**
- ▶ This can be as simple as putting things in Dropbox, which enables you to recover previous file versions automatically
  - ▶ Though I've had issues with this, so definitely wouldn't suggest it

# Version Control

- ▶ **YOU NEED TO USE VERSION CONTROL!!!!**
- ▶ This can be as simple as putting things in Dropbox, which enables you to recover previous file versions automatically
  - ▶ Though I've had issues with this, so definitely wouldn't suggest it
- ▶ Word's "track changes" feature. . .

# Version Control

- ▶ **YOU NEED TO USE VERSION CONTROL!!!!**
- ▶ This can be as simple as putting things in Dropbox, which enables you to recover previous file versions automatically
  - ▶ Though I've had issues with this, so definitely wouldn't suggest it
- ▶ Word's "track changes" feature...
  - ▶ This is usually used poorly



# Version Control

- ▶ **YOU NEED TO USE VERSION CONTROL!!!!**
- ▶ This can be as simple as putting things in Dropbox, which enables you to recover previous file versions automatically
  - ▶ Though I've had issues with this, so definitely wouldn't suggest it
- ▶ Word's "track changes" feature...
  - ▶ This is usually used poorly
  - ▶ Requires saving a new file each time you edit

## Pathologies of the Dropbox/Word approach:

- ▶ Assumes they're both going to be around (in the same form(ish)) in the future

## Pathologies of the Dropbox/Word approach:

- ▶ Assumes they're both going to be around (in the same form(ish)) in the future
  - ▶ Remember computers 20 years ago???

## Pathologies of the Dropbox/Word approach:

- ▶ Assumes they're both going to be around (in the same form(ish)) in the future
  - ▶ Remember computers 20 years ago???
- ▶ Word is expensive; not everyone has it

## Pathologies of the Dropbox/Word approach:

- ▶ Assumes they're both going to be around (in the same form(ish)) in the future
  - ▶ Remember computers 20 years ago???
- ▶ Word is expensive; not everyone has it
- ▶ You can “clobber” work (save over good work)

## Pathologies of the Dropbox/Word approach:

- ▶ Assumes they're both going to be around (in the same form(ish)) in the future
  - ▶ Remember computers 20 years ago???
- ▶ Word is expensive; not everyone has it
- ▶ You can “clobber” work (save over good work)
- ▶ Must tell coauthors not to touch files while you're working on them

## Pathologies of the Dropbox/Word approach:

- ▶ Assumes they're both going to be around (in the same form(ish)) in the future
  - ▶ Remember computers 20 years ago???
- ▶ Word is expensive; not everyone has it
- ▶ You can “clobber” work (save over good work)
- ▶ Must tell coauthors not to touch files while you're working on them
  - ▶ Though this may be changing. . .

## Pathologies of the Dropbox/Word approach:

- ▶ Assumes they're both going to be around (in the same form(ish)) in the future
  - ▶ Remember computers 20 years ago???
- ▶ Word is expensive; not everyone has it
- ▶ You can “clobber” work (save over good work)
- ▶ Must tell coauthors not to touch files while you're working on them
  - ▶ Though this may be changing. . .
- ▶ Word conflates two parts of creating a paper:



## Pathologies of the Dropbox/Word approach:

- ▶ Assumes they're both going to be around (in the same form(ish)) in the future
  - ▶ Remember computers 20 years ago???
- ▶ Word is expensive; not everyone has it
- ▶ You can “clobber” work (save over good work)
- ▶ Must tell coauthors not to touch files while you're working on them
  - ▶ Though this may be changing. . .
- ▶ Word conflates two parts of creating a paper:
  - ▶ Writing the actual document

## Pathologies of the Dropbox/Word approach:

- ▶ Assumes they're both going to be around (in the same form(ish)) in the future
  - ▶ Remember computers 20 years ago???
- ▶ Word is expensive; not everyone has it
- ▶ You can “clobber” work (save over good work)
- ▶ Must tell coauthors not to touch files while you're working on them
  - ▶ Though this may be changing. . .
- ▶ Word conflates two parts of creating a paper:
  - ▶ Writing the actual document
  - ▶ Typesetting

## Pathologies of the Dropbox/Word approach:

- ▶ Assumes they're both going to be around (in the same form(ish)) in the future
  - ▶ Remember computers 20 years ago???
- ▶ Word is expensive; not everyone has it
- ▶ You can “clobber” work (save over good work)
- ▶ Must tell coauthors not to touch files while you're working on them
  - ▶ Though this may be changing. . .
- ▶ Word conflates two parts of creating a paper:
  - ▶ Writing the actual document
  - ▶ Typesetting
  - ▶ We want to only focus on the former (especially in the early stages!)

## Pathologies of the Dropbox/Word approach:

- ▶ Assumes they're both going to be around (in the same form(ish)) in the future
  - ▶ Remember computers 20 years ago???
- ▶ Word is expensive; not everyone has it
- ▶ You can “clobber” work (save over good work)
- ▶ Must tell coauthors not to touch files while you're working on them
  - ▶ Though this may be changing. . .
- ▶ Word conflates two parts of creating a paper:
  - ▶ Writing the actual document
  - ▶ Typesetting
  - ▶ We want to only focus on the former (especially in the early stages!)
- ▶ You also tend to end up with many versions of the same project

## Pathologies of the Dropbox/Word approach:

- ▶ Assumes they're both going to be around (in the same form(ish)) in the future
  - ▶ Remember computers 20 years ago???
- ▶ Word is expensive; not everyone has it
- ▶ You can “clobber” work (save over good work)
- ▶ Must tell coauthors not to touch files while you're working on them
  - ▶ Though this may be changing. . .
- ▶ Word conflates two parts of creating a paper:
  - ▶ Writing the actual document
  - ▶ Typesetting
  - ▶ We want to only focus on the former (especially in the early stages!)
- ▶ You also tend to end up with many versions of the same project
  - ▶ ((show example paper))

# Git

- ▶ Git is a formal *distributed version control system*

# Git

- ▶ Git is a formal *distributed version control system*
  - ▶ There are others (Subversion and Mercurial are the big two), but we're going to focus on git

# Git

- ▶ Git is a formal *distributed version control system*
  - ▶ There are others (Subversion and Mercurial are the big two), but we're going to focus on git
- ▶ It is an easy way to keep track of all the revisions you have saved



# Git

- ▶ Git is a formal *distributed version control system*
  - ▶ There are others (Subversion and Mercurial are the big two), but we're going to focus on git
- ▶ It is an easy way to keep track of all the revisions you have saved
- ▶ It enables easy collaborating between many different people, without the need to email files back and forth or tell them when you're working on something

# Git

- ▶ Git is a formal *distributed version control system*
  - ▶ There are others (Subversion and Mercurial are the big two), but we're going to focus on git
- ▶ It is an easy way to keep track of all the revisions you have saved
- ▶ It enables easy collaborating between many different people, without the need to email files back and forth or tell them when you're working on something
- ▶ You only have *one* version of a file at any one time

# Git

- ▶ Git is a formal *distributed version control system*
  - ▶ There are others (Subversion and Mercurial are the big two), but we're going to focus on git
- ▶ It is an easy way to keep track of all the revisions you have saved
- ▶ It enables easy collaborating between many different people, without the need to email files back and forth or tell them when you're working on something
- ▶ You only have *one* version of a file at any one time
- ▶ You can see the entire history of a file easily

# Git

- ▶ Git is a formal *distributed version control system*
  - ▶ There are others (Subversion and Mercurial are the big two), but we're going to focus on git
- ▶ It is an easy way to keep track of all the revisions you have saved
- ▶ It enables easy collaborating between many different people, without the need to email files back and forth or tell them when you're working on something
- ▶ You only have *one* version of a file at any one time
- ▶ You can see the entire history of a file easily
- ▶ You can see exactly what changed in each new commit

# Git

- ▶ Git is a formal *distributed version control system*
  - ▶ There are others (Subversion and Mercurial are the big two), but we're going to focus on git
- ▶ It is an easy way to keep track of all the revisions you have saved
- ▶ It enables easy collaborating between many different people, without the need to email files back and forth or tell them when you're working on something
- ▶ You only have *one* version of a file at any one time
- ▶ You can see the entire history of a file easily
- ▶ You can see exactly what changed in each new commit
- ▶ There are *many* different GUIs so you don't have to deal with the command line

# Git

- ▶ Git is a formal *distributed version control system*
  - ▶ There are others (Subversion and Mercurial are the big two), but we're going to focus on git
- ▶ It is an easy way to keep track of all the revisions you have saved
- ▶ It enables easy collaborating between many different people, without the need to email files back and forth or tell them when you're working on something
- ▶ You only have *one* version of a file at any one time
- ▶ You can see the entire history of a file easily
- ▶ You can see exactly what changed in each new commit
- ▶ There are *many* different GUIs so you don't have to deal with the command line
  - ▶ I like Sourcetree (for Windows) and SmartGit (for Linux)

# Git

- ▶ Git is a formal *distributed version control system*
  - ▶ There are others (Subversion and Mercurial are the big two), but we're going to focus on git
- ▶ It is an easy way to keep track of all the revisions you have saved
- ▶ It enables easy collaborating between many different people, without the need to email files back and forth or tell them when you're working on something
- ▶ You only have *one* version of a file at any one time
- ▶ You can see the entire history of a file easily
- ▶ You can see exactly what changed in each new commit
- ▶ There are *many* different GUIs so you don't have to deal with the command line
  - ▶ I like Sourcetree (for Windows) and SmartGit (for Linux)
  - ▶ Rstudio can also do most things

# Git

- ▶ Git is a formal *distributed version control system*
  - ▶ There are others (Subversion and Mercurial are the big two), but we're going to focus on git
- ▶ It is an easy way to keep track of all the revisions you have saved
- ▶ It enables easy collaborating between many different people, without the need to email files back and forth or tell them when you're working on something
- ▶ You only have *one* version of a file at any one time
- ▶ You can see the entire history of a file easily
- ▶ You can see exactly what changed in each new commit
- ▶ There are *many* different GUIs so you don't have to deal with the command line
  - ▶ I like Sourcetree (for Windows) and SmartGit (for Linux)
  - ▶ Rstudio can also do most things
- ▶ ((show example git history))



# Github

- ▶ Github is the most popular online git service

# Github

- ▶ Github is the most popular online git service
  - ▶ There are many others, such as BitBucket

# Github

- ▶ Github is the most popular online git service
  - ▶ There are many others, such as BitBucket
- ▶ Each project gets a repository (“repo”)

# Github

- ▶ Github is the most popular online git service
  - ▶ There are many others, such as BitBucket
- ▶ Each project gets a repository (“repo”)
- ▶ Each repo is version-controlled (using git)

# Github

- ▶ Github is the most popular online git service
  - ▶ There are many others, such as BitBucket
- ▶ Each project gets a repository (“repo”)
- ▶ Each repo is version-controlled (using git)
- ▶ Default is open-source (public)

# Github

- ▶ Github is the most popular online git service
  - ▶ There are many others, such as BitBucket
- ▶ Each project gets a repository (“repo”)
- ▶ Each repo is version-controlled (using git)
- ▶ Default is open-source (public)
- ▶ You can make repos private (for a fee - students get 5 for free)

# Github

- ▶ Github is the most popular online git service
  - ▶ There are many others, such as BitBucket
- ▶ Each project gets a repository (“repo”)
- ▶ Each repo is version-controlled (using git)
- ▶ Default is open-source (public)
- ▶ You can make repos private (for a fee - students get 5 for free)
- ▶ This file is a part of my “math-camp” repo here

# Github

- ▶ Github is the most popular online git service
  - ▶ There are many others, such as BitBucket
- ▶ Each project gets a repository (“repo”)
- ▶ Each repo is version-controlled (using git)
- ▶ Default is open-source (public)
- ▶ You can make repos private (for a fee - students get 5 for free)
- ▶ This file is a part of my “math-camp” repo here
  - ▶ Feel free to fork-edit-pull request any changes!



# Github

- ▶ Github is the most popular online git service
  - ▶ There are many others, such as BitBucket
- ▶ Each project gets a repository (“repo”)
- ▶ Each repo is version-controlled (using git)
- ▶ Default is open-source (public)
- ▶ You can make repos private (for a fee - students get 5 for free)
- ▶ This file is a part of my “math-camp” repo here
  - ▶ Feel free to fork-edit-pull request any changes!
  - ▶ I included the PDFs, which is unusual since they aren't plain text

## Bonus best practice - Test your code!

- ▶ If you write your own function, it is important to test it to make sure it does what you want it to do!

```
my_mean <- function(dat){  
  the_sum <- numeric()  
  N <- length(dat)  
  for (i in 1:N){  
    if(i==1){the_sum <- dat[1]}  
    else{  
      the_sum <- the_sum * dat[i]  
    }  
  }  
  my_mean <- the_sum / N  
  my_mean  
}  
mean_of_zero <- c(-2, -1, 1, 2)  
my_mean(mean_of_zero)
```

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

## For further reading - Literate data analysis

- ▶ For R:

## For further reading - Literate data analysis

- ▶ For R:
  - ▶ CRAN has a webpage dedicated to just this

## For further reading - Literate data analysis

- ▶ For R:
  - ▶ CRAN has a webpage dedicated to just this
  - ▶ Markdown: `rmarkdown`

## For further reading - Literate data analysis

- ▶ For R:
  - ▶ CRAN has a webpage dedicated to just this
  - ▶ Markdown: `rmarkdown`
  - ▶ LaTeX: `knitr`

## For further reading - Literate data analysis

- ▶ For R:
  - ▶ CRAN has a webpage dedicated to just this
  - ▶ Markdown: rmarkdown
  - ▶ LaTeX: knitr
  - ▶ WYSIWYM: Lyx + knitr

## For further reading - Literate data analysis

- ▶ For R:
  - ▶ CRAN has a webpage dedicated to just this
  - ▶ Markdown: `rmarkdown`
  - ▶ LaTeX: `knitr`
  - ▶ WYSIWYM: `Lyx` + `knitr`
  - ▶ Office: `odfWeave`



## For further reading - Literate data analysis

- ▶ For R:
  - ▶ CRAN has a webpage dedicated to just this
  - ▶ Markdown: rmarkdown
  - ▶ LaTeX: knitr
  - ▶ WYSIWYM: Lyx + knitr
  - ▶ Office: odfWeave
- ▶ For Stata:

## For further reading - Literate data analysis

- ▶ For R:
  - ▶ CRAN has a webpage dedicated to just this
  - ▶ Markdown: rmarkdown
  - ▶ LaTeX: knitr
  - ▶ WYSIWYM: Lyx + knitr
  - ▶ Office: odfWeave
- ▶ For Stata:
  - ▶ StatWeave

# For further reading - Literate data analysis

- ▶ For R:
  - ▶ CRAN has a webpage dedicated to just this
  - ▶ Markdown: rmarkdown
  - ▶ LaTeX: knitr
  - ▶ WYSIWYM: Lyx + knitr
  - ▶ Office: odfWeave
- ▶ For Stata:
  - ▶ StatWeave
  - ▶ Google “Sweave for Stata” or “reproducible research and Stata”

## For further reading - version control systems

- ▶ Git

## For further reading - version control systems

- ▶ Git
  - ▶ Nice tutorial here

## For further reading - version control systems

- ▶ Git
  - ▶ Nice tutorial [here](#)
  - ▶ Here's more than you need to know about git

## For further reading - version control systems

- ▶ Git
  - ▶ Nice tutorial [here](#)
  - ▶ Here's more than you need to know about git
  - ▶ Using GitFlow