

# Version Control with Git

Thomas J. Leeper

Department of Government  
London School of Economics and Political Science

21 September 2017



# RT2 Roadmap

## Motivating Issues

Researchers degrees of freedom  
Scientific misconduct  
Publication bias  
Failure to replicate

## To achieve

Open materials, data, code, & access  
Transparent reporting & disclosure  
Reproducible & replicable results  
Cumulative meta-analyses

Organized Workflow and File Management (OSF, Github)

### Design

Pre-Registration

Pre-Analysis Plans

Power Planning

### Conduct

Data Management

Version Control

Open Notebooks  
(Jupyter/Docker)

### Dissemination

Transparent Reporting  
& Disclosure

Preprints

Open Access

### Archiving

Data Repositories

Dynamic Documents



Thomas Leeper, LSE:  
*I work primarily with survey and  
survey-experimental data in Political  
Science.*



# Version Control as Organization

- Version control helps you stay organized



# Version Control as Organization

- Version control helps you stay organized
  - 1 What's important to keep around?



# Version Control as Organization

- Version control helps you stay organized
  - 1 What's important to keep around?
  - 2 What's not important to keep around?



# Version Control as Organization

- Version control helps you stay organized
  - 1 What's important to keep around?
  - 2 What's not important to keep around?
  - 3 What is all this crap?



# Version Control as Organization

- Version control helps you stay organized
  - 1 What's important to keep around?
  - 2 What's not important to keep around?
  - 3 What is all this crap?
  
- You're probably already version controlling informally!







↪ FINAL.doc!



↪ FINAL\_rev.2.doc

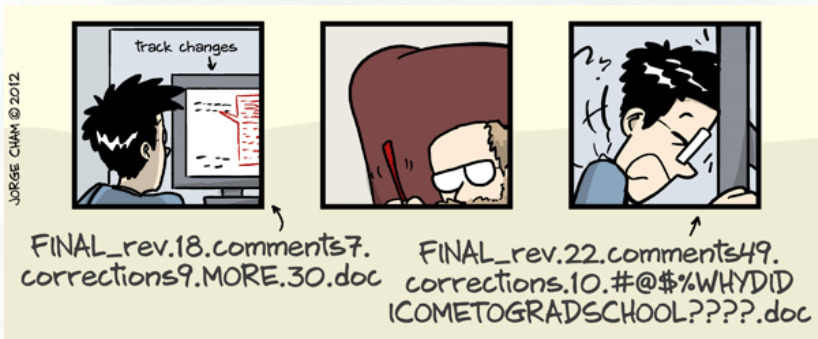




























FINAL\_rev.6.COMMENTS.doc



FINAL\_rev.8.comments5.  
CORRECTIONS.doc





| Name  | Date modified    | Type                | Size   |
|---|------------------|---------------------|--------|
|  Dissertation - Prospectus 0.doc                           | 2015-07-15 09:17 | Microsoft Word 9... | 18 KB  |
|  Dissertation - Prospectus 1.doc                           | 2015-07-15 09:18 | Microsoft Word 9... | 142 KB |
|  Dissertation - Prospectus 2.doc                           | 2015-07-15 09:18 | Microsoft Word 9... | 246 KB |
|  Dissertation - Prospectus 3.doc                           | 2015-07-15 09:18 | Microsoft Word 9... | 250 KB |
|  Dissertation - Prospectus 4.doc                           | 2015-07-15 09:19 | Microsoft Word 9... | 250 KB |
|  Dissertation - Prospectus 5.doc                           | 2015-07-15 09:19 | Microsoft Word 9... | 263 KB |
|  Dissertation - Prospectus 6.doc                           | 2015-07-15 09:19 | Microsoft Word 9... | 287 KB |
|  Dissertation - Prospectus 7.doc                           | 2015-07-15 09:19 | Microsoft Word 9... | 291 KB |
|  Dissertation - Prospectus 8.doc                           | 2015-07-15 09:19 | Microsoft Word 9... | 300 KB |
|  Dissertation - Prospectus 9 (For Jamie).pdf               | 2015-07-15 09:19 | PDF File            | 328 KB |
|  Dissertation - Prospectus 9.doc                           | 2015-07-15 09:19 | Microsoft Word 9... | 340 KB |
|  Dissertation - Prospectus 10 (Big Question).doc           | 2015-07-15 09:18 | Microsoft Word 9... | 19 KB  |
|  Dissertation - Prospectus 10 (New) hml.doc                | 2015-07-15 09:18 | Microsoft Word 9... | 179 KB |
|  Dissertation - Prospectus 10 (New).doc                    | 2015-07-15 09:18 | Microsoft Word 9... | 280 KB |
|  Dissertation - Prospectus 10a (Big Question).doc          | 2015-07-15 09:18 | Microsoft Word 9... | 19 KB  |
|  Dissertation - Prospectus 10b (Big Question).doc          | 2015-07-15 09:18 | Microsoft Word 9... | 32 KB  |
|  Dissertation - Prospectus 10c (Big Question).doc          | 2015-07-15 09:18 | Microsoft Word 9... | 34 KB  |
|  Dissertation - Prospectus 11 (Outline).doc                | 2015-07-15 09:18 | Microsoft Word 9... | 25 KB  |
|  Dissertation - Prospectus 11.doc                          | 2015-07-15 09:18 | Microsoft Word 9... | 287 KB |
|  Dissertation - Prospectus 12.doc                          | 2015-07-15 09:18 | Microsoft Word 9... | 175 KB |
|  Dissertation - Prospectus 12a (Outline).doc               | 2015-07-15 09:18 | Microsoft Word 9... | 40 KB  |
|  Dissertation - Prospectus 12b (Outline).doc               | 2015-07-15 09:18 | Microsoft Word 9... | 50 KB  |
|  Dissertation - Prospectus 13 (Outline).doc                | 2015-07-15 09:18 | Microsoft Word 9... | 159 KB |
|  Dissertation - Prospectus 13a (Outline).doc               | 2015-07-15 09:18 | Microsoft Word 9... | 228 KB |
|  Dissertation - Prospectus 13b (Outline).doc              | 2015-07-15 09:18 | Microsoft Word 9... | 266 KB |
|  Dissertation - Prospectus 13c.doc                       | 2015-07-15 09:18 | Microsoft Word 9... | 320 KB |
|  Dissertation - Prospectus 14 (Methods Draft for Jami... | 2015-07-15 09:18 | Microsoft Word 9... | 44 KB  |



# Version Control as Organization

- Keep what's important!
- Don't keep what's not important!



# Version Control as Organization

- Keep what's important!
- Don't keep what's not important!
- Flexibly play around with both of the above



# Wait, but why do we care?

If we're going to be transparent *in the end* (e.g., at replication or data archiving stage), what do we need to provide?



# Wait, but why do we care?

If we're going to be transparent *in the end* (e.g., at replication or data archiving stage), what do we need to provide?

- A well-organized, reproducible set of project files



# Wait, but why do we care?

If we're going to be transparent *in the end* (e.g., at replication or data archiving stage), what do we need to provide?

- A well-organized, reproducible set of project files

So rather than make that an annoying, post-hoc exercise related to publication, try to get organized and stay organized throughout your project from the very beginning.



**Open Science**

@openscience



Following

"Reproducibility is collaboration with people you don't know, incl. yourself next week." –  
[@philipbstark](#) [#openscience](#)



# Git

- Git is “an open-source distributed version control system”
- Developed in 2005 by Linus Torvalds
- Widely used in software development world



# Why use Git for open science?



# Why use Git for open science?

- Helps you keep and *annotate* snapshots of your project over time
  - Better than renaming your files all the time
  - Better than using within-file VCS
  - Better than single-stream sharing (e.g., Dropbox)

# Why use Git for open science?

- Helps you keep and *annotate* snapshots of your project over time
  - Better than renaming your files all the time
  - Better than using within-file VCS
  - Better than single-stream sharing (e.g., Dropbox)
- Facilitates collaboration, including with your future self



# Why use Git for open science?

- Helps you keep and *annotate* snapshots of your project over time
  - Better than renaming your files all the time
  - Better than using within-file VCS
  - Better than single-stream sharing (e.g., Dropbox)
- Facilitates collaboration, including with your future self
- It's FOSS with lots of clients, tools, and community support



THIS IS GIT. IT TRACKS COLLABORATIVE WORK  
ON PROJECTS THROUGH A BEAUTIFUL  
DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL  
COMMANDS AND TYPE THEM TO SYNC UP.  
IF YOU GET ERRORS, SAVE YOUR WORK  
ELSEWHERE, DELETE THE PROJECT,  
AND DOWNLOAD A FRESH COPY.





# Using Git

- Git create a “local repository” file that you can interact with using a number of tools
  - Command-line git
  - Git Bash
  - Git GUI
  - GitHub Desktop
  - RStudio (via “Projects”)
  - GitHub/Bitbucket/GitLab web interfaces
  - Gitkraken
  - git2r (R package)
  - ...

# Git Essentials

- 1 stage
- 2 commit
- 3 branch
- 4 merge
- 5 push and pull



# Git Essentials

## 1 stage

- **stage**: select files to be recorded in a “snapshot” of the project
- **unstage**: remove files from the snapshot (but not from your computer)

## 2 commit

## 3 branch

## 4 merge

## 5 push and pull



# Git Essentials

- 1 stage
- 2 commit
  - **commit**: record a permanent snapshot of the staged files, labelled with a “commit message”
  - **amend**: modify (typically the most recent) commit with new changes or commit message
- 3 branch
- 4 merge
- 5 push and pull



# Git Essentials

- 1 stage
- 2 commit
- 3 branch
  - produce a complete *local* copy of the project where changes can be made independently of the “master” branch
- 4 merge
- 5 push and pull



# Git Essentials

- 1 stage
- 2 commit
- 3 branch
- 4 merge
  - update a branch with changes from another local branch (or a remote); you can change multiple branches independently.
- 5 push and pull



# Git Essentials

- 1 stage
- 2 commit
- 3 branch
- 4 merge
- 5 push and pull
  - **push**: send the project (any new commits) to a remote server (like GitHub)
  - **pull**: grab new commits from a remote server

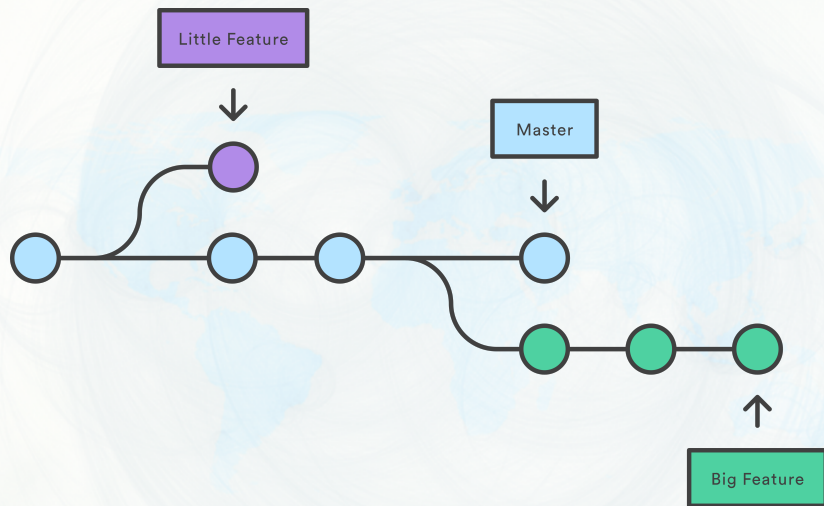


# Git Essentials

- 1 stage
- 2 commit
- 3 branch
- 4 merge
- 5 push and pull







Source: <https://www.atlassian.com/git/tutorials>



# Hands-on practice!



```
git --version
git
git config --global user.name "My Name"
git config --global user.email "me@example.com"
git config --list
```



```
git init
git status
echo Hello world! > README.md
git add README.md
git status
git rm --cached README.md
git status
git add --all
git commit -m "my first commit!"
git status
```

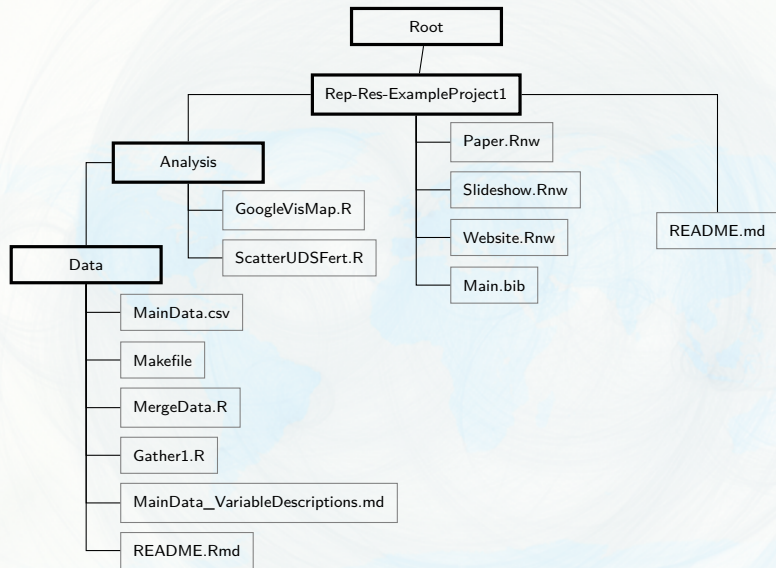
# Initializing a Project Structure

- There's no single best way to organize a project
- But, some words of wisdom:
  - Put like with like
  - Avoid excessive hierarchy
  - Not everything needs to go into git
  - Steal others' structures!



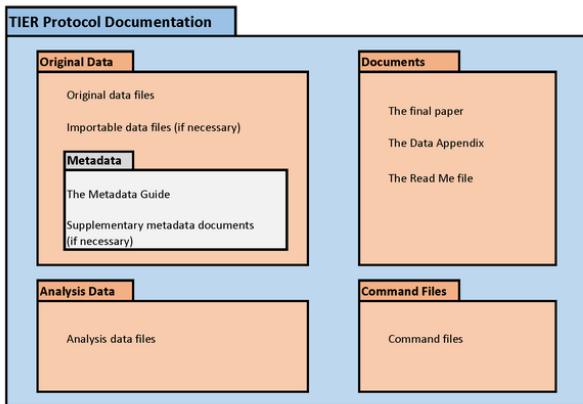
# What makes up the ideal reproducible research product?

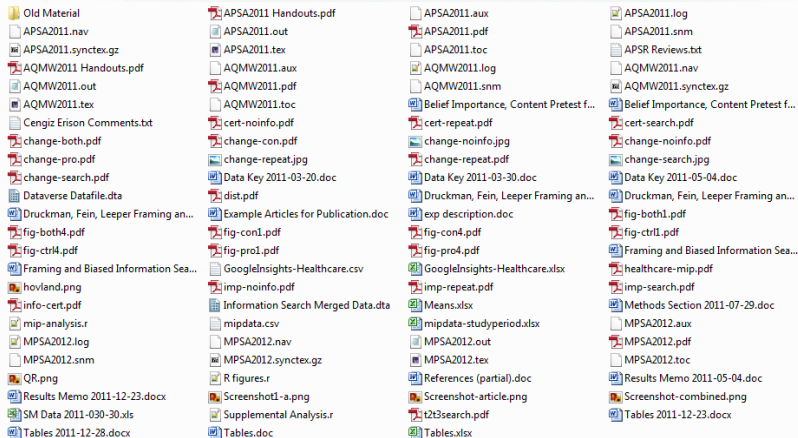
- Gandrud's template
- rOpenSci's "Research Compendium"
- Project TIER





```
project
|- DESCRIPTION      # project metadata and dependencies
|- README.md       # top-level description of content
|
|- data/           # raw data, not changed once created
| +- my_data.csv   # data files in open formats
|
|- analysis/       # any programmatic code
| +- my_scripts.R  # R code used to analyse data
```





```
mkdir code
mkdir data
mkdir figures
git status
```

```
git status
cat README.md
# do something to README.md
git diff
git add README.md
git commit -m "second commit"
git status
git log
git log --oneline
git log --oneline -1
git log --oneline --stat
```



```
git status
git diff README.md
git diff HEAD README.md
git diff HEAD~1 README.md
git diff HEAD~2 README.md
git diff HEAD~3 README.md
git diff HEAD~20 README.md
git diff <commit hash> README.md
git diff <commit hash>
```

# !! DANGER: Amend Commit !!

```
git status
git log --oneline
# maybe add/rm files
git amend
# enter the hell of vim
```

```
git config --global core.editor
"<executable> <options>"
```



# Safe reversion

```
git status
git log --oneline
git revert <commit hash>
# enter the hell of vim
# or something else terrible
git revert --abort
```

# !! DANGER: Unsafe reversion !!

*The StackOverflow Question*



```
git status
echo "bad bad bad" > bad.txt
git status
echo bad.txt > .gitignore
git status
echo bad bad bad > bad1.txt
echo bad bad bad > bad2.txt
echo bad* > .gitignore
git status
git add bad1.txt -f
git status
```

# Navigating History

```
git status
git log
git checkout <commit hash>
git status
ls
cat README.md
git checkout master
```

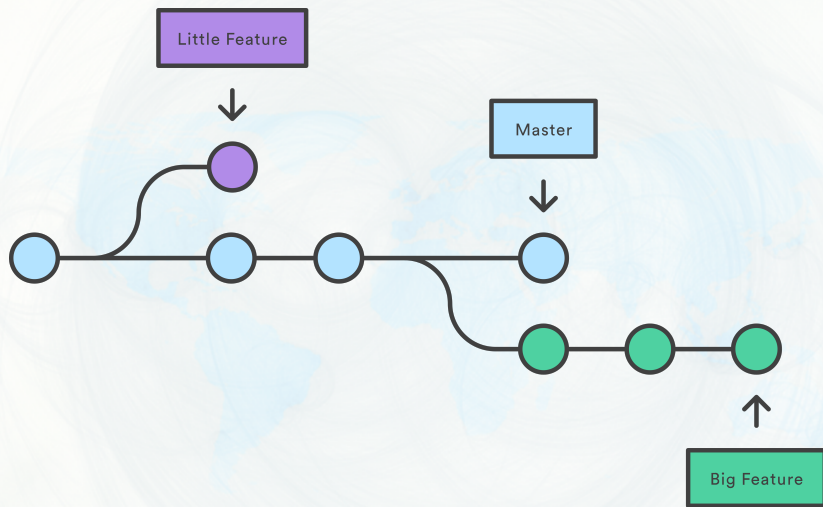


```
git status
git log
git checkout <commit hash>
git status
ls
echo aaaaaah!>manuscript.txt
git checkout master
```



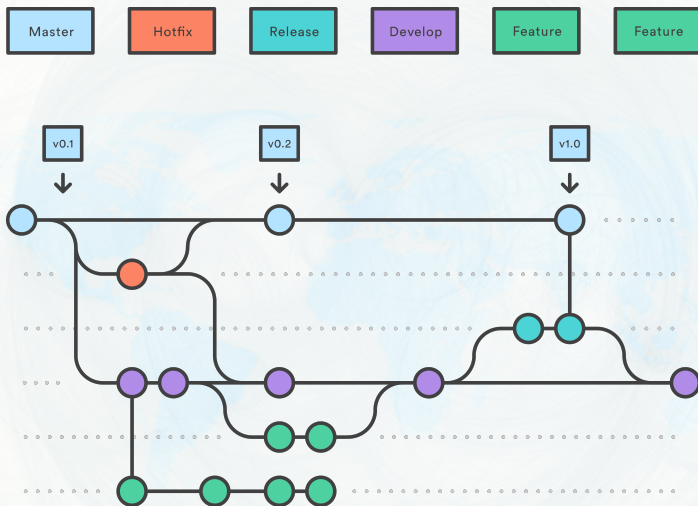
# Branches

- Branches are local, parallel versions of your entire project
- Useful for multiple things:
  - Experimentation
  - Manuscript submissions
  - Collaboration



Source: <https://www.atlassian.com/git/tutorials>





Source: <https://www.atlassian.com/git/tutorials>



# Simple branch and merge

```
git status
git checkout -b thomas
git status
# do something
git add --all
git commit -m "thomas's commit"
git checkout master
git branch
git log --graph --oneline
git merge thomas
```



# GUIs

- You can do everything in Git on the command line
- GUIs can be helpful for:
  - Exploring history
  - Visualizing branches
  - Confirming what you're doing



# Merge conflicts

```
git checkout -b thomas
git status
# do something to README.md
git add --all
git commit -m "change on thomas"
git checkout master
# do something to README.md
git add --all
git commit -m "change on master"
git merge thomas
git log
```





# Remotes

- A server (“cloud”) instance of the Git repository
- Useful for multiple things:
  - Collaboration
  - Transparency
  - Archiving/backups
  - Using web-based Git interfaces



# Remotes

- Three major players in cloud Git
  - GitHub
  - Atlassian Bitbucket
  - GitLab
- Why choose one or the other?
  - Cost
  - Collaborators
  - Private repositories

```
git status
git remote add github
https://github.com/leeper/rt2
git remote
git remote set-url
git remote rename
git remote remove
```



```
git status
git push github master -u
git fetch github
git fetch github master
git checkout -b new-idea
git push github new-idea
git checkout master
git pull github master
git pull
```



```
git status
git tag -a v0.0.1 -m "v0.0.1"
git push --tags

git tag -d v0.0.1
```

# Tags versus Branches

- *Branches* are for working versions of project
  - Collaborator-specific branches
  - Submission-specific branches
  - Experimental or “bug fix” branches
- *Tags* are for marking particular snapshots
  - Significant moments in project history
  - Journal submission or conference version
  - Formal “releases”



# Collaboration

- Technical aspects
  - Give collaborators access on GitHub (or wherever)
  - Work on separate branches
  - Merge agreed changes into **master**
- Human factors aspects
  - Requires agreeing on workflow
  - Communication about what goes in “master”
  - Can feel awkward if moving from a Dropbox- or email-based collaboration style

# Try it with a partner!

- 1 Partner A create a GitHub repo; give Partner B access
- 2 Partner B should `git fetch/git pull` the repo
- 3 Partner B should create a local branch and `git push`
- 4 Partner A should `git fetch` the branch
- 5 Partner A should `git merge` the branch to **master** and `git push`
- 6 Partner B should `git pull` from **master**
- 7 Both use `git log` to compare





# Conclusion

- Once you use Git, you'll never want to go back to your old workflow





# Conclusion

- Once you use Git, you'll never want to go back to your old workflow
- But, collaborators probably don't know or want to use Git!

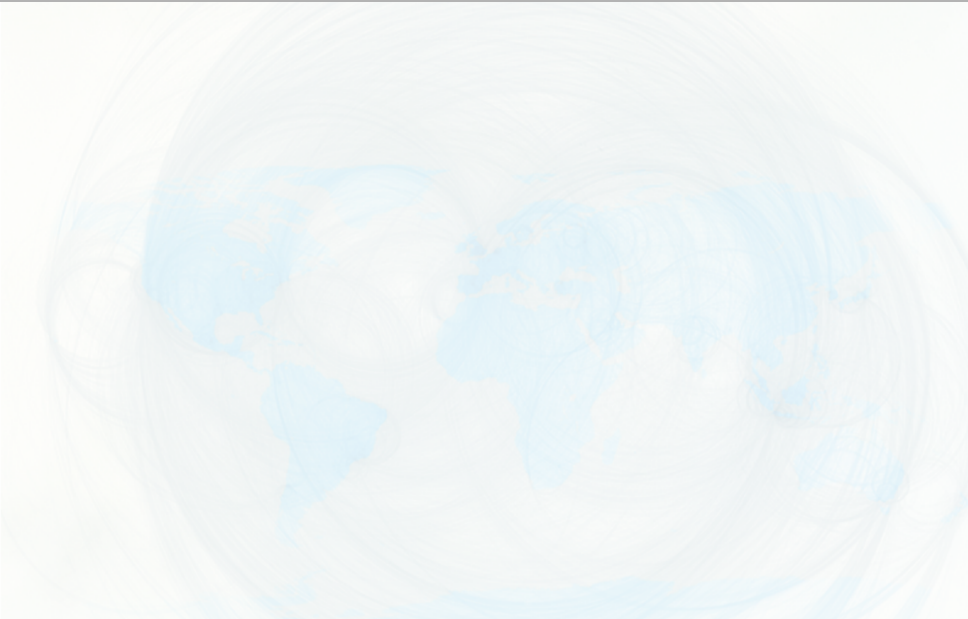


# Conclusion

- Once you use Git, you'll never want to go back to your old workflow
- But, collaborators probably don't know or want to use Git!
- Git is crazy complicated -- StackOverflow is your friend







How do you typically get figures, tables, and other material out of analytic software and into papers or presentations?



# Dynamic Documents in R

- The dynamic documents landscape is evolving very, very rapidly:
  - Early 2000s: Sweave
  - 2010's: knitr
  - Ongoing: Rmarkdown

# Dynamic Documents in R

- The dynamic documents landscape is evolving very, very rapidly:
  - Early 2000s: Sweave
  - 2010's: knitr
  - Ongoing: Rmarkdown
- Embed code (R or otherwise) inside a manuscript that outputs:
  - Word (.docx)
  - HTML
  - LaTeX/PDF
  - HTML or PPT slides

```
# My Manuscript  
Thomas J. Leeper  
This is my manuscript.
```





# Rmarkdown

- 1 YAML metadata header
- 2 Document contents in **markdown**
- 3 Code in “code chunks”:

```
““{r chunk1}  
# R code  
hist(rnorm(1000))  
“““
```

```
---  
- title: My Manuscript  
- author: Thomas J. Leeper  
- date: 2017-09-21  
- output: pdf_document  
---
```

This is my manuscript.

```
“““{r chunk1}  
# R code  
hist(rnorm(1000))  
“““
```

# Markdown Basics

Markdown is a very simple markup language for formatting simple texts:

`*italics*`

*italics*

`*bold*`

**bold**

`'preformatted'`

preformatted

`# Heading`

Heading Level 1

`## Heading`

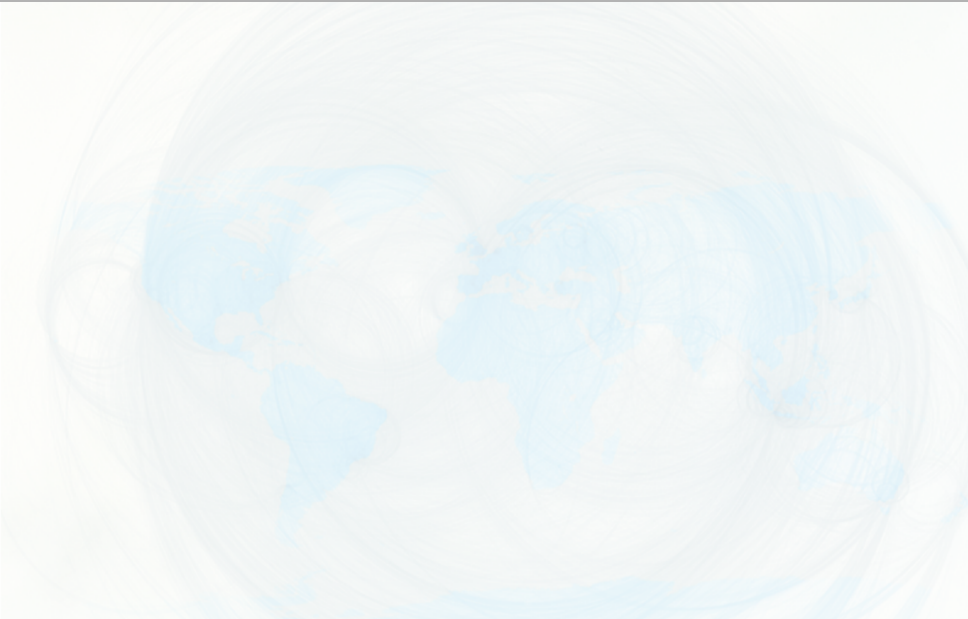
Heading Level 2

`### Heading`

Heading Level 3

`[link] (https://google.com)`

link



# Chunk Options

```
“““{r chunk1, eval=TRUE, echo=TRUE}  
2 + 2  
“““
```

```
“““{r chunk2, eval=TRUE, echo=FALSE}  
2 + 2  
“““
```

```
“““{r chunk3, echo=FALSE, results="hide"}  
2 + 2  
“““
```

# Global Chunk Options

```
“““{r options, eval = TRUE, echo = FALSE}  
library("knitr")  
opts_chunk$set(echo = FALSE,  
               cache = TRUE,  
               message = FALSE)  
“““
```

# Basic Tables

```
““{r table1, results = "asis"}  
xtable::xtable(table(mtcars$cyl, mtcars$gear))  
  
knitr::kable(head(mtcars))  
““
```

# Regression Results Tables

```
““{r table2, results = "asis"}
library("stargazer")
stargazer(
  x1 <- lm(mpg ~ disp + wt,
           data = mtcars),
  x2 <- lm(mpg ~ disp + wt + vs,
           data = mtcars),
  header = FALSE
)
““
```



# Figures

```
“““{r fig1,  
  fig.cap = "Fuel Economy by Weight",  
  fig.height = 4,  
  fig.width = 6}  
library("ggplot2")  
ggplot(mtcars,  
  aes(x = wt,  
      y = mpg,  
      colour = factor(cyl))) +  
  geom_point()  
“““
```