Intro to Git for Version Control

Hunter Wade York

06/10/2022

Who am I?

- Rising third-year PhD student in Sociology at Princeton, Office of Population Research Affiliate

Me!

Who am I?

- Rising third-year PhD student in Sociology at Princeton, Office of Population Research Affiliate
- Formerly: MPH at UW and researcher at Institute for Health Metrics and Evaluation, Formerly: AB at Harvard in Human Evolutionary Biology and Music

Me!

Who am I?

- Rising third-year PhD student in Sociology at Princeton, Office of Population Research Affiliate
- Formerly: MPH at UW and researcher at Institute for Health Metrics and Evaluation, Formerly: AB at Harvard in Human Evolutionary Biology and Music
- Interests: Stratification, culture, quant

Me!

Goals

- This presentation will teach you very few hard skills, though you should leave with a basic example of git + github in your toolkit.

Goals

- This presentation will teach you very few hard skills, though you should leave with a basic example of git + github in your toolkit.
- Git is a very flexible program, with many many use cases. This tutorial is catered towards use in academic research settings.

Goals

- This presentation will teach you very few hard skills, though you should leave with a basic example of git + github in your toolkit.
- Git is a very flexible program, with many many use cases. This tutorial is catered towards use in academic research settings.
  - There are a lot of tutorials out there that won't be useful to you because they primarily teach the collaborative part of git as opposed to the version control part of git. This collaborative part is what holds up entire tech platforms: facebook, google, etc. all use a version of git.

Goals

- This presentation will teach you very few hard skills, though you should leave with a basic example of git + github in your toolkit.
- Git is a very flexible program, with many many use cases. This tutorial is catered towards use in academic research settings.
  - There are a lot of tutorials out there that won't be useful to you because they primarily teach the collaborative part of git as opposed to the version control part of git. This collaborative part is what holds up entire tech platforms: facebook, google, etc. all use a version of git.
- If you have questions or want a demonstration during the presentation, STOP ME! No question is too simple, and I can do a few examples on the fly.

Useful Links/Resources

- Git for Novices

Useful Links/Resources

- Git for Novices
- Super Basic Git Workflow for collaboration (fork, edit, push, pull request)

Useful Links/Resources

- Git for Novices
- Super Basic Git Workflow for collaboration (fork, edit, push, pull request)
- Another Git Tutorial

Useful Links/Resources

- Git for Novices
- Super Basic Git Workflow for collaboration (fork, edit, push, pull request)
- Another Git Tutorial
- A helpful introduction to Git and GitHub

Useful Links/Resources

- Git for Novices
- Super Basic Git Workflow for collaboration (fork, edit, push, pull request)
- Another Git Tutorial
- A helpful introduction to Git and GitHub
- Basic Intro to Command Line

Useful Links/Resources

- Git for Novices
- Super Basic Git Workflow for collaboration (fork, edit, push, pull request)
- Another Git Tutorial
- A helpful introduction to Git and GitHub
- Basic Intro to Command Line
- Helpful One-Pager for Command Line

Useful Links/Resources

- Git for Novices
- Super Basic Git Workflow for collaboration (fork, edit, push, pull request)
- Another Git Tutorial
- A helpful introduction to Git and GitHub
- Basic Intro to Command Line
- Helpful One-Pager for Command Line
- GUIs for Macs; Windows; and a tutorial for RStudio's built-in client

Useful Links/Resources

- Git for Novices
- Super Basic Git Workflow for collaboration (fork, edit, push, pull request)
- Another Git Tutorial
- A helpful introduction to Git and GitHub
- Basic Intro to Command Line
- Helpful One-Pager for Command Line
- GUIs for Macs; Windows; and a tutorial for RStudio's built-in client
- This presentation! (Will be on my Github and Xi Song will circulate)

Questions?

Nota Bene

- Historically, the main branch of Git was, by default, termed “Master”

Nota Bene

- Historically, the main branch of Git was, by default, termed "Master"
- This is problematic, and after a bit of pushback, Github changed its defaults in 2020.

Nota Bene

- Historically, the main branch of Git was, by default, termed "Master"
- This is problematic, and after a bit of pushback, Github changed its defaults in 2020.
- I *think* Git has not yet done this, but I could be wrong.

Nota Bene

- Historically, the main branch of Git was, by default, termed “Master”
- This is problematic, and after a bit of pushback, Github changed its defaults in 2020.
- I *think* Git has not yet done this, but I could be wrong.
- In this presentation, there will be some repos that still use the old terminology because they were initiated before the change.

Nota Bene

- Historically, the main branch of Git was, by default, termed "Master"
- This is problematic, and after a bit of pushback, Github changed its defaults in 2020.
- I *think* Git has not yet done this, but I could be wrong.
- In this presentation, there will be some repos that still use the old terminology because they were initiated before the change.
- Some borrowed material may also make the slight.

Nota Bene

- Historically, the main branch of Git was, by default, termed "Master"
- This is problematic, and after a bit of pushback, Github changed its defaults in 2020.
- I *think* Git has not yet done this, but I could be wrong.
- In this presentation, there will be some repos that still use the old terminology because they were initiated before the change.
- Some borrowed material may also make the slight.
- Github gives you instructions for making your default "main" when you create a new repo, and it's easy in Git as well.

Outline

- What is Git?

Outline

- What is Git?
- What is Github?

Outline

- What is Git?
- What is Github?
- Designing a workflow around Git

Outline

- What is Git?
- What is Github?
- Designing a workflow around Git
- Using Git: 3 Use Cases

Outline

- What is Git?
- What is Github?
- Designing a workflow around Git
- Using Git: 3 Use Cases
  - Basic version control (Single User)

Outline

- What is Git?
- What is Github?
- Designing a workflow around Git
- Using Git: 3 Use Cases
    - Basic version control (Single User)
    - Backtracking, branching (Single User)

Outline

- What is Git?
- What is Github?
- Designing a workflow around Git
- Using Git: 3 Use Cases
  - Basic version control (Single User)
  - Backtracking, branching (Single User)
  - Collaboration/tweaking an existing project (Mulitple Users)

What is git?

Git

- Git is a local program on your computer.

Git

Git

- Git is a local program on your computer.
- Many machines come with it preinstalled.

Git

Git

- Git is a local program on your computer.
- Many machines come with it preinstalled.
  - To check open Terminal/Command Line and type “git”

Git

Git

- Git is a local program on your computer.
- Many machines come with it preinstalled.
  - To check open Terminal/Command Line and type "git"
- Software is available here

Git

Git

- Git is a local program on your computer.
- Many machines come with it preinstalled.
  - To check open Terminal/Command Line and type "git"
- Software is available here
- Git can be used in the terminal or with some GUIs (linked earlier). I little/no experience with these.

Git

Why use git?

- Free and open source!

Git

Why use git?

- Free and open source!
- Forces you to think in a structured way about your code.

Git

Why use git?

- Free and open source!
- Forces you to think in a structured way about your code.
    - "What changes have I made since I last saved my code?"

Git

Why use git?

- Free and open source!
- Forces you to think in a structured way about your code.
    - “What changes have I made since I last saved my code?”
- EVERYTHING IS SAVED!

Git

Why use git?

- Free and open source!
- Forces you to think in a structured way about your code.
  - "What changes have I made since I last saved my code?"
- EVERYTHING IS SAVED!
- It is explicit in a way that dropbox/automatic versioning is not.

Git

Why use git?

- Free and open source!
- Forces you to think in a structured way about your code.
  - “What changes have I made since I last saved my code?”
- EVERYTHING IS SAVED!
- It is explicit in a way that dropbox/automatic versioning is not.
  - Alternatives have no potential for branching structure or intentional cooperation.

Git

Why not to use git?

- Steepish learning curve

git remote add origin
git remote origin add
git add remote origin
git add origin remote

Why not to use git?

- Steepish learning curve
  - Many people have never used command line tools, and so that is the bulk of the work.

Why not to use git?

- Steepish learning curve
  - Many people have never used command line tools, and so that is the bulk of the work.
- Hard to troubleshoot if you're not a pro.

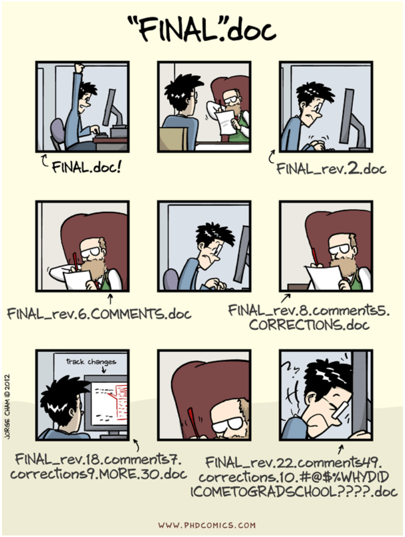
Why not to use git?

- Steepish learning curve
  - Many people have never used command line tools, and so that is the bulk of the work.
- Hard to troubleshoot if you're not a pro.
  - This is restricted mostly to difficult use cases, but you can get yourself in a bind for many reasons.

Why not to use git?

- Steepish learning curve
  - Many people have never used command line tools, and so that is the bulk of the work.
- Hard to troubleshoot if you're not a pro.
  - This is restricted mostly to difficult use cases, but you can get yourself in a bind for many reasons.
  - Stackexchange is your friend!

Why use git?



www.phdcomics.com

What is Github?

Github

- Online “mirror” of git.

Github!

Github

- Online “mirror” of git.
- Allows remote backups.

Github!

Github

- Online “mirror” of git.
- Allows remote backups.
- Allows collaboration.

Github!

Github

- Online "mirror" of git.
- Allows remote backups.
- Allows collaboration.
- Built in GUI (Graphical user interface)

Github!

Github

- Online "mirror" of git.
- Allows remote backups.
- Allows collaboration.
- Built in GUI (Graphical user interface)
- Allows permanent record of code (for publishing papers for example)

GitHub

Github!

Questions?

Before you use git, a note on Projects

Project-based design

- Knowing where your files are allows you to spend more time doing what's important: writing code!

Faizal Rosly
@ijaicool

After watching one episode of Tidying Up with Marie Kondo on Netflix:

Organize!

Project-based design

- Knowing where your files are allows you to spend more time doing what's important: writing code!
- A project is simply, start to finish, everything to get you from idea to finished paper

Faizal Rosly
@ijaicool

After watching one episode of Tidying Up with Marie Kondo on Netflix:

Organize!

Project-based design

- Knowing where your files are allows you to spend more time doing what's important: writing code!
- A project is simply, start to finish, everything to get you from idea to finished paper
- It includes code, data, output files like clean datasets, figures, and possibly a manuscript

Faizal Rosly
@ijaicool

After watching one episode of Tidying Up with Marie Kondo on Netflix:

Organize!

Project-based design

- Knowing where your files are allows you to spend more time doing what's important: writing code!
- A project is simply, start to finish, everything to get you from idea to finished paper
- It includes code, data, output files like clean datasets, figures, and possibly a manuscript
- By having all your files in one directory and using relational paths, you make the entire project replicable!

Faizal Rosly
@ijaicool

After watching one episode of Tidying Up with Marie Kondo on Netflix:

Organize!

Project-based design

- Knowing where your files are allows you to spend more time doing what's important: writing code!
- A project is simply, start to finish, everything to get you from idea to finished paper
- It includes code, data, output files like clean datasets, figures, and possibly a manuscript
- By having all your files in one directory and using relational paths, you make the entire project replicable!
    - Allows you to easily publish code for pubs!

Faizal Rosly
@ijaicool

After watching one episode of Tidying Up with Marie Kondo on Netflix:

Organize!

My Workflow (starting a new project)

- Create a folder on my computer for each project.

My Workflow (starting a new project)

- Create a folder on my computer for each project.
  - "/Users/hyork/Documents/projects/NAME_OF_PROJECT"

My Workflow (starting a new project)

- Create a folder on my computer for each project.
  - "/Users/hyork/Documents/projects/NAME_OF_PROJECT"
- Format subdirectories in a similar manner for all projects

My Workflow (starting a new project)

- Create a folder on my computer for each project.
  - "/Users/hyork/Documents/projects/NAME_OF_PROJECT"
- Format subdirectories in a similar manner for all projects
  - "/code", "/inputs", "/outputs", "/ref"

My Workflow (starting a new project)

- Create a folder on my computer for each project.
  - "/Users/hyork/Documents/projects/NAME_OF_PROJECT"
- Format subdirectories in a similar manner for all projects
  - "/code", "/inputs", "/outputs", "/ref"
- Start a git repository in the main directory or in the code directory and keep up-to-date with GitHub. This is what I'll teach you to do!

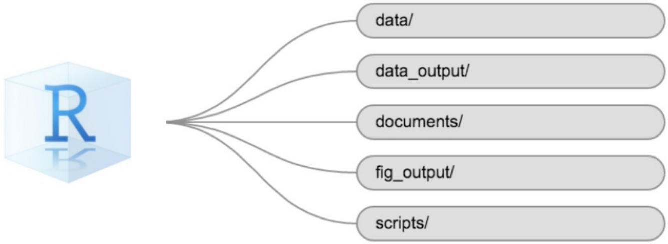
My Workflow (starting a new project)

- Create a folder on my computer for each project.
  - "/Users/hyork/Documents/projects/NAME_OF_PROJECT"
- Format subdirectories in a similar manner for all projects
  - "/code", "/inputs", "/outputs", "/ref"
- Start a git repository in the main directory or in the code directory and keep up-to-date with GitHub. This is what I'll teach you to do!
- A git repository can exist in either just your code directory or in the parent directory. This will depend on if you want to have everything version controlled or just your code. This also depends on file sizes, data privacy concerns, etc.

My Workflow (starting a new project)

- Create a folder on my computer for each project.
  - "/Users/hyork/Documents/projects/NAME_OF_PROJECT"
- Format subdirectories in a similar manner for all projects
  - "/code", "/inputs", "/outputs", "/ref"
- Start a git repository in the main directory or in the code directory and keep up-to-date with GitHub. This is what I'll teach you to do!
- A git repository can exist in either just your code directory or in the parent directory. This will depend on if you want to have everything version controlled or just your code. This also depends on file sizes, data privacy concerns, etc.
- I almost always start it in the parent directory and then only `git add` (I'll teach this in a second) the code subdirectory. This gives me the flexibility to also keep track of "/ref" in case my codebase ends up relying heavily on it.

Workflows



Another Example Directory Setup

Questions?

Git overview - Starting a repo

Git init

- Assume we've started a project, and we have a local folder that we want to keep track of on Git.

Git init

- Assume we've started a project, and we have a local folder that we want to keep track of on Git.
- `git init` is a command which initiates git in a directory.

Git init

- Assume we've started a project, and we have a local folder that we want to keep track of on Git.
- `git init` is a command which initiates git in a directory.
- I navigate to my folder of choice (`/code`), and in it I use this command.

Git init

- Assume we've started a project, and we have a local folder that we want to keep track of on Git.
- `git init` is a command which initiates git in a directory.
- I navigate to my folder of choice (`/code`), and in it I use this command.
  - To navigate within the command line, use `cd PATH_TO_DIRECTORY`. (`cd` stands for change directory.)

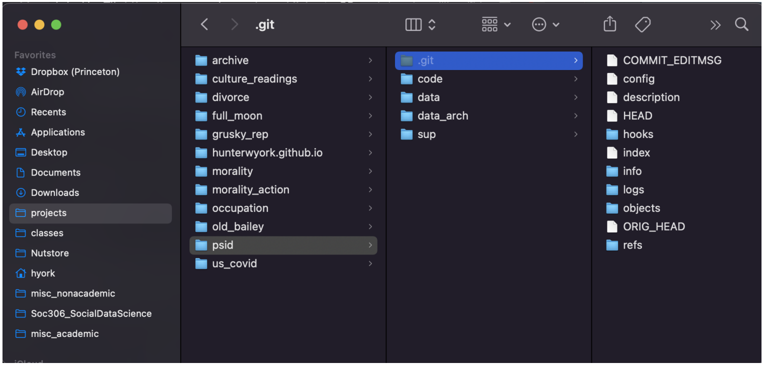
Git init

- Assume we've started a project, and we have a local folder that we want to keep track of on Git.
- `git init` is a command which initiates git in a directory.
- I navigate to my folder of choice (`/code`), and in it I use this command.
  - To navigate within the command line, use `cd PATH_TO_DIRECTORY`. (`cd` stands for change directory.)
- Under the hood, this command creates an invisible subdirectory (`/code/.git`) which git uses to keep track of all my files.

Git init

- Assume we've started a project, and we have a local folder that we want to keep track of on Git.
- `git init` is a command which initiates git in a directory.
- I navigate to my folder of choice (`/code`), and in it I use this command.
  - To navigate within the command line, use `cd PATH_TO_DIRECTORY`. (`cd` stands for change directory.)
- Under the hood, this command creates an invisible subdirectory (`/code/.git`) which git uses to keep track of all my files.
  - Git works by keeping track of insertions and deletions of your code and associating them with commit messages. More on this later.

Git init



archive
culture_readings
divorce
full_moon
grusky_rep
hunterwyork.github.io
morality
morality_action
occupation
old_bailey
psid
us_covid

.git
code
data
data_arch
sup

COMMIT_EDITMSG
config
description
HEAD
hooks
index
info
logs
objects
ORIG_HEAD
refs

File Structure Showing .git folder

git status

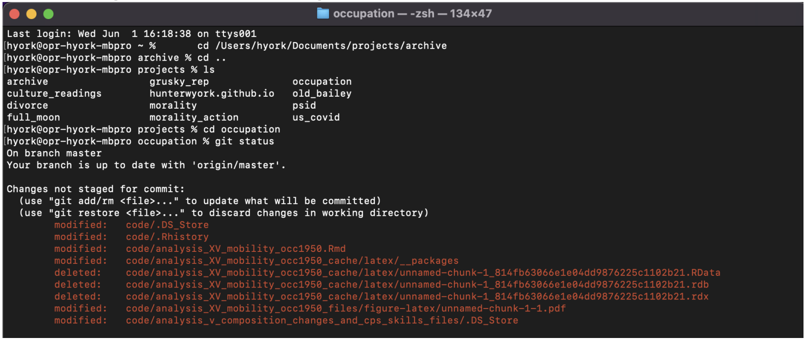
- At any time, you can use `git status` to check what has happened within a folder containing a git repo.

git status

- At any time, you can use `git status` to check what has happened within a folder containing a git repo.
- The below output shows what it might look like.

git status

- At any time, you can use `git status` to check what has happened within a folder containing a git repo.
- The below output shows what it might look like.
- Here I have some unsaved changes in my "/code" subdirectory, including some changes in an RMarkdown file, and some associated file changes from knitting to pdf.



```
Last login: Wed Jun  1 16:18:38 on ttys001
hyork@opr-hyork-mbpro ~ %       cd /Users/hyork/Documents/projects/archive
hyork@opr-hyork-mbpro archive % cd ..
hyork@opr-hyork-mbpro projects % ls
archive                 grusky_rep              occupation
culture_readings        hunterwyork.github.io   old_bailey
divorce                 morality                psid
full_moon               morality_action         us_covid
hyork@opr-hyork-mbpro projects % cd occupation
hyork@opr-hyork-mbpro occupation % git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   code/.DS_Store
        modified:   code/.Rhistory
        modified:   code/analysis_XV_mobility_occ1950.Rmd
        modified:   code/analysis_XV_mobility_occ1950_cache/latex/__packages
        deleted:    code/analysis_XV_mobility_occ1950_cache/latex/unnamed-chunk-1_814fb63066e1e04dd9876225c1102b21.RData
        deleted:    code/analysis_XV_mobility_occ1950_cache/latex/unnamed-chunk-1_814fb63066e1e04dd9876225c1102b21.rdb
        deleted:    code/analysis_XV_mobility_occ1950_cache/latex/unnamed-chunk-1_814fb63066e1e04dd9876225c1102b21.rdx
        modified:   code/analysis_XV_mobility_occ1950_files/figure-latex/unnamed-chunk-1-1.pdf
        modified:   code/analysis_v_composition_changes_and_cps_skills_files/.DS_Store
```

Git Add

- Once you have made changes in a repo, you can `git add` them.

Git Add

- Once you have made changes in a repo, you can `git add` them.
- This "stages" them, preparing a commit that can soon be pushed.

Git Add

- Once you have made changes in a repo, you can `git add` them.
- This "stages" them, preparing a commit that can soon be pushed.
- Adding is as simple as typing `git add filename.R`

Git Add

- Once you have made changes in a repo, you can `git add` them.
- This “stages” them, preparing a commit that can soon be pushed.
- Adding is as simple as typing `git add filename.R`
- To add all files in a directory, type `git add .`

Git Add

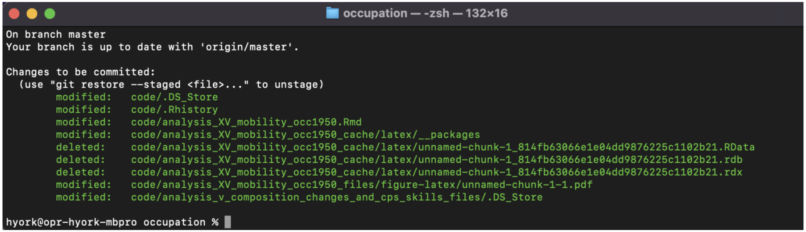
- Once you have made changes in a repo, you can `git add` them.
- This "stages" them, preparing a commit that can soon be pushed.
- Adding is as simple as typing `git add filename.R`
- To add all files in a directory, type `git add .`
- You can also add an entire subdirectory.

Git Add

- Once you have made changes in a repo, you can `git add` them.
- This "stages" them, preparing a commit that can soon be pushed.
- Adding is as simple as typing `git add filename.R`
- To add all files in a directory, type `git add .`
- You can also add an entire subdirectory.
- You can also set files to be ignored every time, so that you can use `git add .` without having it actually pay attention to unimportant files. Just google "gitignore" to learn this trick!

Git Add

- Here we can see that once I've run `git add .`, all the unstaged changes are now staged for a commit.



```
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   code/.DS_Store
        modified:   code/.Rhistory
        modified:   code/analysis_XV_mobility_occ1950.Rmd
        modified:   code/analysis_XV_mobility_occ1950_cache/latex/__packages
        deleted:    code/analysis_XV_mobility_occ1950_cache/latex/unnamed-chunk-1_814fb63066e1e04dd9876225c1102b21.RData
        deleted:    code/analysis_XV_mobility_occ1950_cache/latex/unnamed-chunk-1_814fb63066e1e04dd9876225c1102b21.rdb
        deleted:    code/analysis_XV_mobility_occ1950_cache/latex/unnamed-chunk-1_814fb63066e1e04dd9876225c1102b21.rdx
        modified:   code/analysis_XV_mobility_occ1950_files/figure-latex/unnamed-chunk-1-1.pdf
        modified:   code/analysis_v_composition_changes_and_cps_skills_files/.DS_Store

hyork@opr-hyork-mbpro occupation %
```

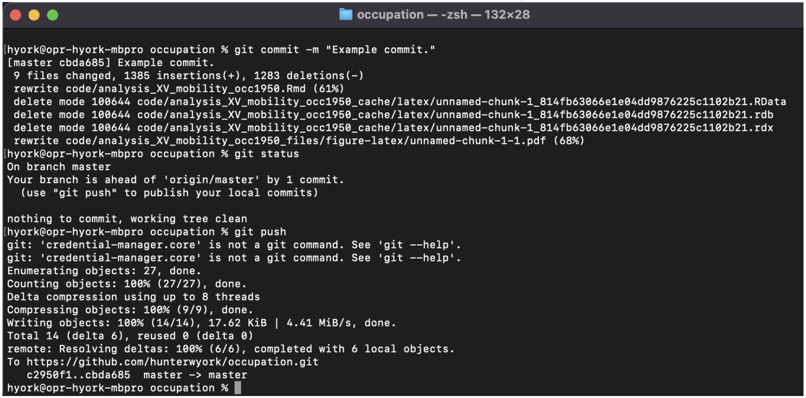
Example output

Git Commit & Push

- Finally, you can use `git commit` to add a commit message to your staged items. This is an informative label that summarizes what you've done since the last commit.

Git Commit & Push

- Finally, you can use `git commit` to add a commit message to your staged items. This is an informative label that summarizes what you've done since the last commit.
- Git push finalizes the commit by pushing local commits.



occupation — -zsh — 132×28

```
hyork@opr-hyork-mbpro occupation % git commit -m "Example commit."
[master cbda685] Example commit.
 9 files changed, 1385 insertions(+), 1283 deletions(-)
 rewrite code/analysis_XV_mobility_occ1950.Rmd (61%)
 delete mode 100644 code/analysis_XV_mobility_occ1950_cache/latex/unnamed-chunk-1_814fb63066e1e04dd9876225c1102b21.RData
 delete mode 100644 code/analysis_XV_mobility_occ1950_cache/latex/unnamed-chunk-1_814fb63066e1e04dd9876225c1102b21.rdb
 delete mode 100644 code/analysis_XV_mobility_occ1950_cache/latex/unnamed-chunk-1_814fb63066e1e04dd9876225c1102b21.rdx
 rewrite code/analysis_XV_mobility_occ1950_files/figure-latex/unnamed-chunk-1-1.pdf (68%)
hyork@opr-hyork-mbpro occupation % git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
hyork@opr-hyork-mbpro occupation % git push
git: 'credential-manager.core' is not a git command. See 'git --help'.
git: 'credential-manager.core' is not a git command. See 'git --help'.
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (14/14), 17.62 KiB | 4.41 MiB/s, done.
Total 14 (delta 6), reused 0 (delta 0)
remote: Resolving deltas: 100% (6/6), completed with 6 local objects.
To https://github.com/hunterwyork/occupation.git
   c2950f1..cbda685  master -> master
hyork@opr-hyork-mbpro occupation %
```

Git Remotes

- Everything done until the `git push` now has **only** used git and not github.

Git Remotes

- Everything done until the `git push` now has **only** used git and not github.
- All changes are saved locally, and I can use git in the terminal to explore previous commits.

Git Remotes

- Everything done until the `git push` now has **only** used git and not github.
- All changes are saved locally, and I can use git in the terminal to explore previous commits.
- However, if I delete the hidden `.git` folder, my computer dies, or something else happens, all previous changes will be lost forever.

Git Remotes

- Everything done until the `git push` now has **only** used git and not github.
- All changes are saved locally, and I can use git in the terminal to explore previous commits.
- However, if I delete the hidden `.git` folder, my computer dies, or something else happens, all previous changes will be lost forever.
- Enter, Github!

Git Remotes

- After or before `git init`, the first thing I like to do is sync git with github.

Git Remotes

- After or before `git init`, the first thing I like to do is sync git with github.
  - This only has to be done once, and then all other git commands automatically duplicate changes locally to the remote when you push.

Git Remotes

- After or before `git init`, the first thing I like to do is sync git with github.
  - This only has to be done once, and then all other git commands automatically duplicate changes locally to the remote when you push.
- On www.github.com, you can follow the simple directions to set up a remote repo. You can then follow the given instructions to sync your local repo with the remote.

Git Remotes

- After or before `git init`, the first thing I like to do is sync git with github.
    - This only has to be done once, and then all other git commands automatically duplicate changes locally to the remote when you push.
- On www.github.com, you can follow the simple directions to set up a remote repo. You can then follow the given instructions to sync your local repo with the remote.
- An example command would be something like `git remote add origin https://github.com/hunterwyork/occupation.git`

Git Remotes

- After or before `git init`, the first thing I like to do is sync git with github.
  - This only has to be done once, and then all other git commands automatically duplicate changes locally to the remote when you push.
- On www.github.com, you can follow the simple directions to set up a remote repo. You can then follow the given instructions to sync your local repo with the remote.
- An example command would be something like `git remote add origin https://github.com/hunterwyork/occupation.git`
- You can set up SSH keys so you don't have to enter you github username and password every time. THIS IS WORTH IT!

Review

- `git init` initializes a repo in a folder.

Review

- `git init` initializes a repo in a folder.
- `git add` stages files for a commit.

Review

- `git init` initializes a repo in a folder.
- `git add` stages files for a commit.
- `git commit` commits the changes locally

Review

- `git init` initializes a repo in a folder.
- `git add` stages files for a commit.
- `git commit` commits the changes locally
- `git push` pushes these changes upstream (usually to remote)

Review

- `git init` initializes a repo in a folder.
- `git add` stages files for a commit.
- `git commit` commits the changes locally
- `git push` pushes these changes upstream (usually to remote)
- `git remote add origin` sets the remote, allowing you to sync with github

Review

- `git init` initializes a repo in a folder.
- `git add` stages files for a commit.
- `git commit` commits the changes locally
- `git push` pushes these changes upstream (usually to remote)
- `git remote add origin` sets the remote, allowing you to sync with github
- `git status` allows you to check what's going on re: unstaged, staged, commited, and up-to-date files

Review

- `git init` initializes a repo in a folder.
- `git add` stages files for a commit.
- `git commit` commits the changes locally
- `git push` pushes these changes upstream (usually to remote)
- `git remote add origin` sets the remote, allowing you to sync with github
- `git status` allows you to check what's going on re: unstaged, staged, commited, and up-to-date files
- By now, you have all that you need to have a basic, single-user version control system for keeping track of a project.

Example

Questions?

Git overview - Backtracking, Branching

Backtracking

- A key utility of Git is that it allows you to go back to any previous commit, restoring the entire repo to the point in time where you saved it.

Backtracking

- A key utility of Git is that it allows you to go back to any previous commit, restoring the entire repo to the point in time where you saved it.
- My commits are often clustered around a big deliverable.

Backtracking

- A key utility of Git is that it allows you to go back to any previous commit, restoring the entire repo to the point in time where you saved it.
- My commits are often clustered around a big deliverable.
  - If I have a meeting with an advisor, I'll name the commit "Meeting with Yu Xie june 6 2022"

Backtracking

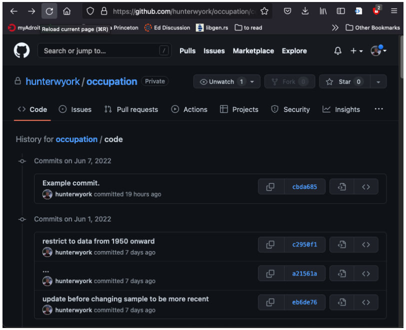
- A key utility of Git is that it allows you to go back to any previous commit, restoring the entire repo to the point in time where you saved it.
- My commits are often clustered around a big deliverable.
    - If I have a meeting with an advisor, I'll name the commit "Meeting with Yu Xie june 6 2022"
    - Later on, I'll have explored some changes that might not work out, and so I decide I want to go back to the stage I was at for the meeting and start over.

Backtracking

- A key utility of Git is that it allows you to go back to any previous commit, restoring the entire repo to the point in time where you saved it.
- My commits are often clustered around a big deliverable.
  - If I have a meeting with an advisor, I'll name the commit "Meeting with Yu Xie june 6 2022"
  - Later on, I'll have explored some changes that might not work out, and so I decide I want to go back to the stage I was at for the meeting and start over.
- This is where `git checkout` and `git branch` enter the picture.

Backtracking

- Each commit is associated with a hash.



History for occupation / code

Commits on Jun 7, 2022

Example commit.
hunterwyork committed 19 hours ago — cbda685

Commits on Jun 1, 2022

restrict to data from 1950 onward
hunterwyork committed 7 days ago — c2950f1

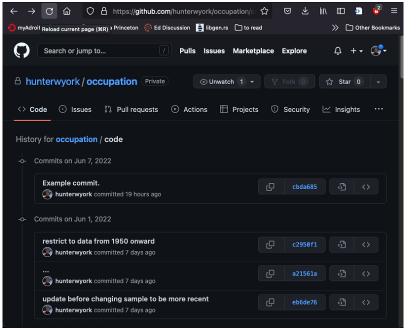
...
hunterwyork committed 7 days ago — a21561a

update before changing sample to be more recent
hunterwyork committed 7 days ago — eb6de76

Example commits

Backtracking

- Each commit is associated with a hash.
- Clicking the copy button copies the hash string onto your clipboard.



History for occupation / code

Commits on Jun 7, 2022

Example commit.
hunterwyork committed 19 hours ago — cbda685

Commits on Jun 1, 2022

restrict to data from 1950 onward
hunterwyork committed 7 days ago — c2950f1

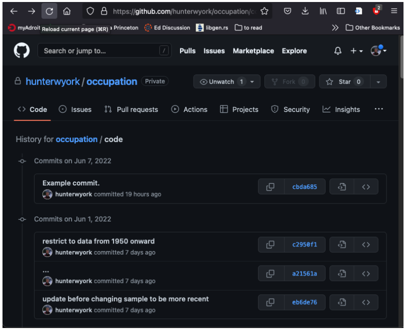
...
hunterwyork committed 7 days ago — a21561a

update before changing sample to be more recent
hunterwyork committed 7 days ago — eb6de76

Example commits

Backtracking

- Each commit is associated with a hash.
- Clicking the copy button copies the hash string onto your clipboard.
- You can also see these manually using `git log`



History for occupation / code

Commits on Jun 7, 2022

Example commit.
hunterwyork committed 19 hours ago — cbda685

Commits on Jun 1, 2022

restrict to data from 1950 onward
hunterwyork committed 7 days ago — c2950f1

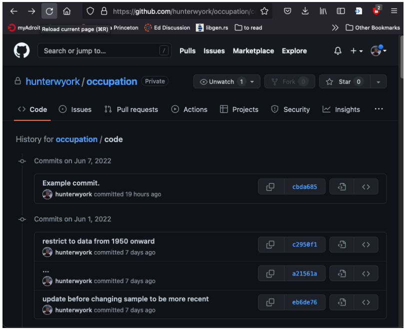
...
hunterwyork committed 7 days ago — a21561a

update before changing sample to be more recent
hunterwyork committed 7 days ago — eb6de76

Example commits

Backtracking

- Each commit is associated with a hash.
- Clicking the copy button copies the hash string onto your clipboard.
- You can also see these manually using `git log`
- `git checkout COPIED_HASH` allows you to rollback the repo to the state it was in when you committed that commit.



History for occupation / code

Commits on Jun 7, 2022

Example commit.
hunterwyork committed 19 hours ago — cbda685

Commits on Jun 1, 2022

restrict to data from 1950 onward
hunterwyork committed 7 days ago — c2950f1

...
hunterwyork committed 7 days ago — a21561a

update before changing sample to be more recent
hunterwyork committed 7 days ago — eb6de76

Example commits

Git overview - Backtracking

- Before you backtrack, you may want to compare two commits

Git overview - Backtracking

- Before you backtrack, you may want to compare two commits
- You can do that using `git diff` in the terminal or by using github within your repo.

Git overview - Backtracking

- Before you backtrack, you may want to compare two commits
- You can do that using `git diff` in the terminal or by using github within your repo.
- This is a major utility of the GUI of github! Visually inspecting changes between commits allow you to see why your results might have totally changed without you remembering.

Git overview - Backtracking

- Before you backtrack, you may want to compare two commits
- You can do that using `git diff` in the terminal or by using github within your repo.
- This is a major utility of the GUI of github! Visually inspecting changes between commits allow you to see why your results might have totally changed without you remembering.
- Online example Link

Git overview - Branching

- If you decide that you want to work off of a previous commit, you can create a branch.

Git overview - Branching

- If you decide that you want to work off of a previous commit, you can create a branch.
- This is exactly what it sounds like.

Git overview - Branching

- If you decide that you want to work off of a previous commit, you can create a branch.
- This is exactly what it sounds like.
- This presentation is actually all done using git. I branched it from a presentation I gave earlier this year on data management in R because there were 3-5 slides of overlap.

Git overview - Branching

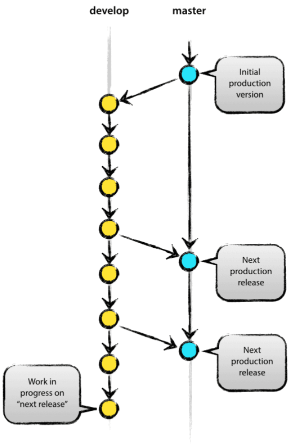
- If you decide that you want to work off of a previous commit, you can create a branch.
- This is exactly what it sounds like.
- This presentation is actually all done using git. I branched it from a presentation I gave earlier this year on data management in R because there were 3-5 slides of overlap.
- This is done by checking out a previous branch. Then you can run `git branch NAME_OF_NEW_BRANCH`. There are a few commands that follow when you eventually push a commit on a new branch, but git in the terminal tells you what to type.

Git overview - Branching

- If you decide that you want to work off of a previous commit, you can create a branch.
- This is exactly what it sounds like.
- This presentation is actually all done using git. I branched it from a presentation I gave earlier this year on data management in R because there were 3-5 slides of overlap.
- This is done by checking out a previous branch. Then you can run `git branch NAME_OF_NEW_BRANCH`. There are a few commands that follow when you eventually push a commit on a new branch, but git in the terminal tells you what to type.
- Example

Git overview - Branching

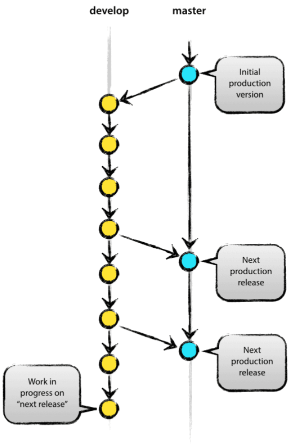
- An alternative workflow that uses branching is constituted by the “dev” and “prod” mentality.


develop
master
Initial production version
Next production release
Next production release
Work in progress on “next release”

https://nvie.com/posts/a-successful-git-branching-model/

Git overview - Branching

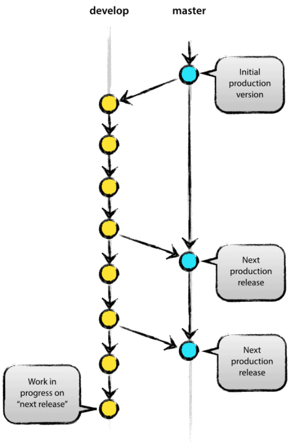
- An alternative workflow that uses branching is constituted by the "dev" and "prod" mentality.
- "Development" branches allow you to test something out before merging it back into "Production" workflows.


develop
master
Initial production version
Next production release
Next production release
Work in progress on "next release"

https://nvie.com/posts/a-successful-git-branching-model/

Git overview - Branching

- An alternative workflow that uses branching is constituted by the “dev” and “prod” mentality.
- “Development” branches allow you to test something out before merging it back into “Production” workflows.
- This is very aligned with how industry uses git.


develop
master
Initial production version
Next production release
Next production release
Work in progress on “next release”

https://nvie.com/posts/a-successful-git-branching-model/

Git overview - Collaboration

Mass Collaboration

- I've worked on several 20+ author papers (and a few 1000+), but usually one one or two people are really in charge of data analysis, or it uses a structure where people work on separate chunks of code.

Mass Collaboration

- I've worked on several 20+ author papers (and a few 1000+), but usually one one or two people are really in charge of data analysis, or it uses a structure where people work on separate chunks of code.
- Git provides a more truly collaborative alternative!

Mass Collaboration

- I’ve worked on several 20+ author papers (and a few 1000+), but usually one one or two people are really in charge of data analysis, or it uses a structure where people work on separate chunks of code.
- Git provides a more truly collaborative alternative!
- Collaboration is hard. Tools like git, trello, and slack help.

Mass Collaboration

- I've worked on several 20+ author papers (and a few 1000+), but usually one one or two people are really in charge of data analysis, or it uses a structure where people work on separate chunks of code.
- Git provides a more truly collaborative alternative!
- Collaboration is hard. Tools like git, trello, and slack help.
- Working in a lab setting? Use a lab handbook: link

Mass Collaboration

- I've worked on several 20+ author papers (and a few 1000+), but usually one one or two people are really in charge of data analysis, or it uses a structure where people work on separate chunks of code.
- Git provides a more truly collaborative alternative!
- Collaboration is hard. Tools like git, trello, and slack help.
- Working in a lab setting? Use a lab handbook: link
- Keys: norms, coordination, and communication.

Cloning an existing repo

- The next part I'm mainly going to talk about conceptually. You can learn the code on your own.

Cloning an existing repo

- The next part I'm mainly going to talk about conceptually. You can learn the code on your own.
- If a collaborator has an ongoing project that you'd like to contribute to, you can fork the repository.

Cloning an existing repo

- The next part I'm mainly going to talk about conceptually. You can learn the code on your own.
- If a collaborator has an ongoing project that you'd like to contribute to, you can fork the repository.
- This makes a copy of the repo on your github. Now, it functions just like any normal repo. You can have it locally on your computer, make changes, stage commits, push them, make branches, etc.

Cloning an existing repo

- The next part I'm mainly going to talk about conceptually. You can learn the code on your own.
- If a collaborator has an ongoing project that you'd like to contribute to, you can fork the repository.
- This makes a copy of the repo on your github. Now, it functions just like any normal repo. You can have it locally on your computer, make changes, stage commits, push them, make branches, etc.
- The remote and the local will not affect the original repo.

Cloning an existing repo

- The next part I'm mainly going to talk about conceptually. You can learn the code on your own.
- If a collaborator has an ongoing project that you'd like to contribute to, you can fork the repository.
- This makes a copy of the repo on your github. Now, it functions just like any normal repo. You can have it locally on your computer, make changes, stage commits, push them, make branches, etc.
- The remote and the local will not affect the original repo.
- If the original repo makes changes, you repo can see them. You can merge in these changes!

Cloning an existing repo

- The next part I'm mainly going to talk about conceptually. You can learn the code on your own.
- If a collaborator has an ongoing project that you'd like to contribute to, you can fork the repository.
- This makes a copy of the repo on your github. Now, it functions just like any normal repo. You can have it locally on your computer, make changes, stage commits, push them, make branches, etc.
- The remote and the local will not affect the original repo.
- If the original repo makes changes, you repo can see them. You can merge in these changes!
  - This use case depends a lot on having the kind of code that can handle multiple users working on it at once!

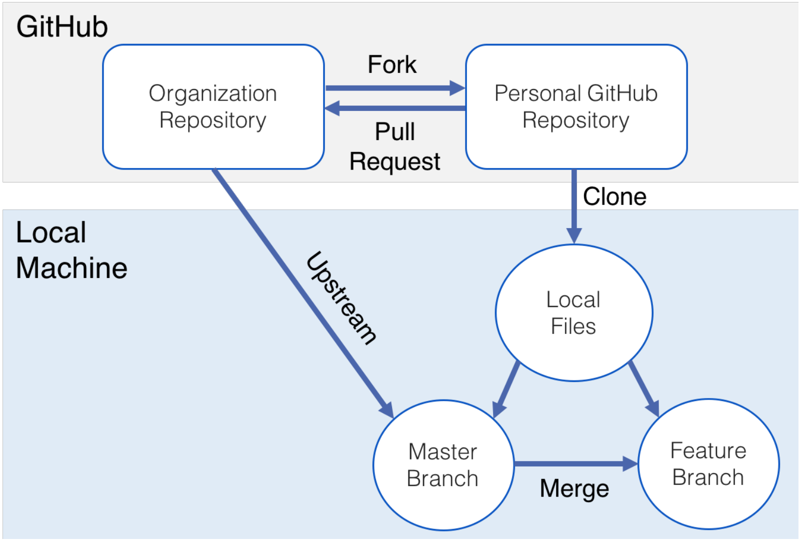
Cloning an existing repo

- The next part I'm mainly going to talk about conceptually. You can learn the code on your own.
- If a collaborator has an ongoing project that you'd like to contribute to, you can fork the repository.
- This makes a copy of the repo on your github. Now, it functions just like any normal repo. You can have it locally on your computer, make changes, stage commits, push them, make branches, etc.
- The remote and the local will not affect the original repo.
- If the original repo makes changes, you repo can see them. You can merge in these changes!
  - This use case depends a lot on having the kind of code that can handle multiple users working on it at once!
  - ("Functional Programming")

Cloning an existing repo

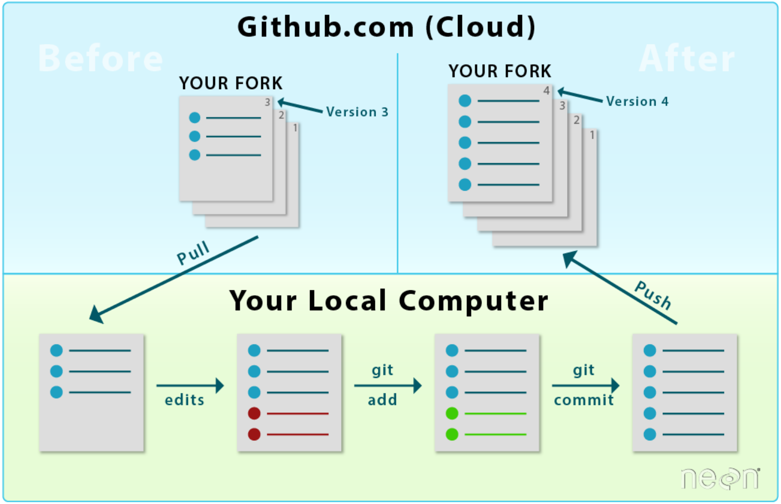
- The next part I'm mainly going to talk about conceptually. You can learn the code on your own.
- If a collaborator has an ongoing project that you'd like to contribute to, you can fork the repository.
- This makes a copy of the repo on your github. Now, it functions just like any normal repo. You can have it locally on your computer, make changes, stage commits, push them, make branches, etc.
- The remote and the local will not affect the original repo.
- If the original repo makes changes, you repo can see them. You can merge in these changes!
  - This use case depends a lot on having the kind of code that can handle multiple users working on it at once!
  - ("Functional Programming")
- Also, if you think a change you've made is wort incorporating into the master branch, you can initiate a pull request!

Cloning an existing repo



https://www.neonscience.org/resources/learning-hub/tutorials/github-git-add

Cloning an existing repo



https://www.neonscience.org/resources/learning-hub/tutorials/github-git-add

Questions?

What I didn't teach you

- `git merge`

What I didn't teach you

- `git merge`
  - When you're working on a branch and want to reincorporate it into the main branch, you will have to merge.

What I didn't teach you

- `git merge`
  - When you're working on a branch and want to reincorporate it into the main branch, you will have to merge.
  - This is the hardest part of Git. If your merge isn't perfect, you'll have conflicts, which you have to sort out by hand.

What I didn't teach you

- `git merge`
  - When you're working on a branch and want to reincorporate it into the main branch, you will have to merge.
  - This is the hardest part of Git. If your merge isn't perfect, you'll have conflicts, which you have to sort out by hand.
- `git fetch`

What I didn't teach you

- `git merge`
  - When you're working on a branch and want to reincorporate it into the main branch, you will have to merge.
  - This is the hardest part of Git. If your merge isn't perfect, you'll have conflicts, which you have to sort out by hand.
- `git fetch`
  - Requests updates from remote repository. Useful for collaboration.

What I didn't teach you

- `git merge`
  - When you're working on a branch and want to reincorporate it into the main branch, you will have to merge.
  - This is the hardest part of Git. If your merge isn't perfect, you'll have conflicts, which you have to sort out by hand.
- `git fetch`
  - Requests updates from remote repository. Useful for collaboration.
- `git clone`

What I didn't teach you

- `git merge`
  - When you're working on a branch and want to reincorporate it into the main branch, you will have to merge.
  - This is the hardest part of Git. If your merge isn't perfect, you'll have conflicts, which you have to sort out by hand.
- `git fetch`
  - Requests updates from remote repository. Useful for collaboration.
- `git clone`
  - For pulling someone elses repo to your local machine from github.

Some final helpful resources

- Git for large files - https://git-lfs.github.com/

Some final helpful resources

- Git for large files - https://git-lfs.github.com/
- Git Cheatsheet - https://wac-cdn.atlassian.com/dam/jcr:e7e22f25-bba2-4ef1-a197-53f46b6df4a5/SWTM-2088_Atlassian-Git-Cheatsheet.pdf?cdnVersion=386

Some final helpful resources

- Git for large files - https://git-lfs.github.com/
- Git Cheatsheet - https://wac-cdn.atlassian.com/dam/jcr:e7e22f25-bba2-4ef1-a197-53f46b6df4a5/SWTM-2088_Atlassian-Git-Cheatsheet.pdf?cdnVersion=386
- Another Cheatsheet - https://elischei.com/wp-content/uploads/2021/06/git-cheat-sheet-large-scaled.jpg

.

https://medium.com/@lulu.ilmaknun.q/kompilasi-meme-git-e2fe49c6e33e

Questions, comments?

Thank you!

- I can be reached at hyork@princeton.edu

Thank you!

- I can be reached at hyork@princeton.edu
- Check out my website hunterwyork.com

Thank you!

- I can be reached at hyork@princeton.edu
- Check out my website hunterwyork.com
- If you found this presentation useful, share it!

Thank you!

- I can be reached at hyork@princeton.edu
- Check out my website hunterwyork.com
- If you found this presentation useful, share it!
- Rmarkdown file and slides are here.