Exploratory Data Analysis with R

Git and Github

Xuemao Zhang
East Stroudsburg University

September 26, 2022

Outline

- Relative and absolute paths
- Git: a version control system
- Github

Reproducible Research

Your closest collaborator is you six months ago, but you don't reply to emails.

`- Karl Broman`

http://kbroman.org/Tools4RR/assets/lectures/06_org_eda.pdf

Components of Reproducible Research

1. Data provenance
2. Code + documentation: RMarkdown
3. Versions of software: Git

Relative and absolute paths

- **Working Directories**
- R “looks” for files on your computer relative to the **working** directory
- Many people recommend not setting a directory in the scripts
  - assume you’re in the directory the script is in
  - If you open an R file with a new RStudio session, it does this for you.
- If you do set a working directory, do it at the beginning of your script.
- Example of getting and setting the working directory:

```
getwd(); #obtain the current working directory
setwd("../Lecture09");  #set (change) the working directory
```

Relative and absolute paths

Setting a Working Directory

- Setting the directory can sometimes be finicky (locally or on cloud)
  - **Windows**: Default directory structure involves single backslashes (\), but R interprets these as "escape" characters. So you must replace the backslash with forward slashes ("/") or two backslashes ("\")
  - **Mac/Linux**: Default is forward slashes, so you are okay
- Typical directory structure syntax applies
  - ".." - goes up one level
  - "./" - is the current directory
  - "∼" - is your "home" directory

Relative and absolute paths

Note that the `dir()` function interfaces with your operating system and can show you which files are in your current working directory.

You can try some directory navigation:

```
dir("./") # shows directory contents
```

```
##  [1] "git.png"               "git_github.png"        "git_jenny.png
##  [4] "github.png"            "github_ggplot2.png"    "github_issues
##  [7] "githubLab1.png"        "githubLab2.png"        "githubLab3_1.
## [10] "githubLab3_2.png"      "githubLab3_3.png"      "Lecture09_git
## [13] "Lecture09_github.pdf" "Lecture09_github.Rmd" "newpj.png"
## [16] "path.png"              "path_absolute.png"     "path_absolute
## [19] "path_here1.png"        "path_here2.png"        "path_here3.pn
## [22] "path_relative.png"     "path_relative2.png"    "path_relative
```

```
dir("..")
```

```
##  [1] "88x31.png"   "data"        "Lecture01"   "Lecture02"   "Lec
##  [6] "Lecture04"   "Lecture05"   "Lecture06"   "Lecture07"   "Lec
## [11] "Lecture09"   "Lecture10"   "Lecture11"   "Lecture12"   "Lec
## [16] "Lecture14"   "Lecture15"   "Lecture16"   "Lecture17"   "Lec
```

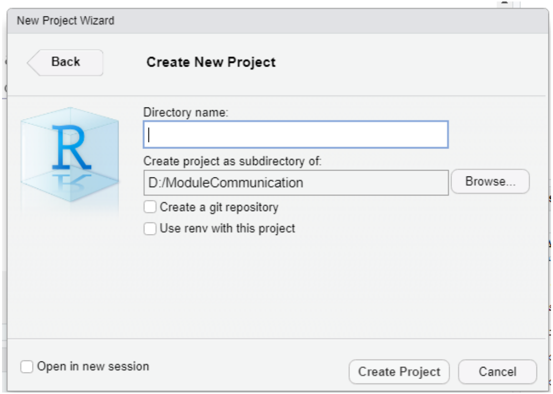
Relative and absolute paths

- Relative vs. absolute paths (from Wiki): *An **absolute or full path** points to the same location in a file system, regardless of the current working directory. To do that, it must include the **root directory**.*
  - This means if I try your code, and you use absolute paths, it won't work unless we have the exact same folder structure where R is looking (bad).
- *By contrast, a **relative path starts from some given working directory**, avoiding the need to provide the full absolute path. A filename can be considered as a relative path based at the current working directory.*
- In RStudio, go to `Session --> Set Working Directory --> To Source File Location`
  - RStudio should put code in the Console, similar to this:

```
setwd("~/LectureNotes/Lecture09")
```

Relative and absolute paths

- We explain relative and absolute paths with Rstudio.cloud.
  - We are using local R and Rstudio, so we just put a project in a folder named `project` shown in the next slide.
  - To create a project, click `File->New Project... -> New Directory -> New Project`



Directory name:

Create project as subdirectory of: D:/ModuleCommunication

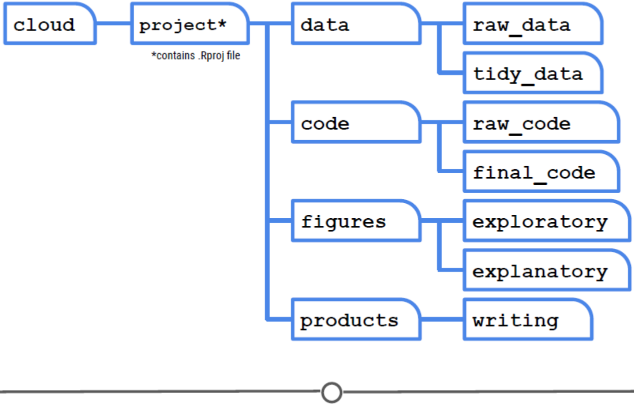
□ Create a git repository

□ Use renv with this project

□ Open in new session

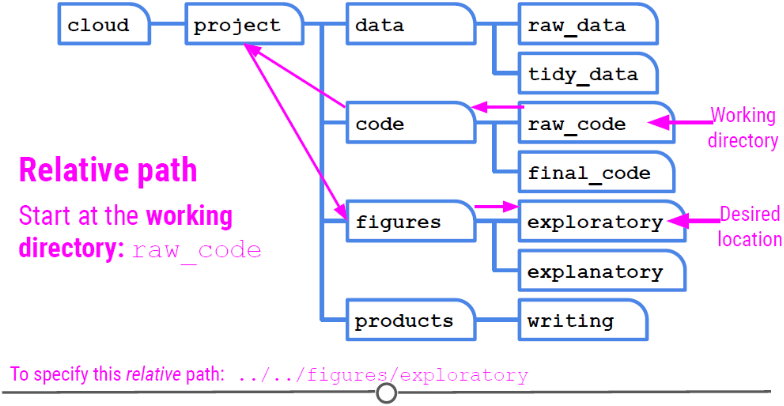
Relative and absolute paths

- Suppose we have a project called `project`



Relative and absolute paths

- Suppose `raw_data` is our working directory,

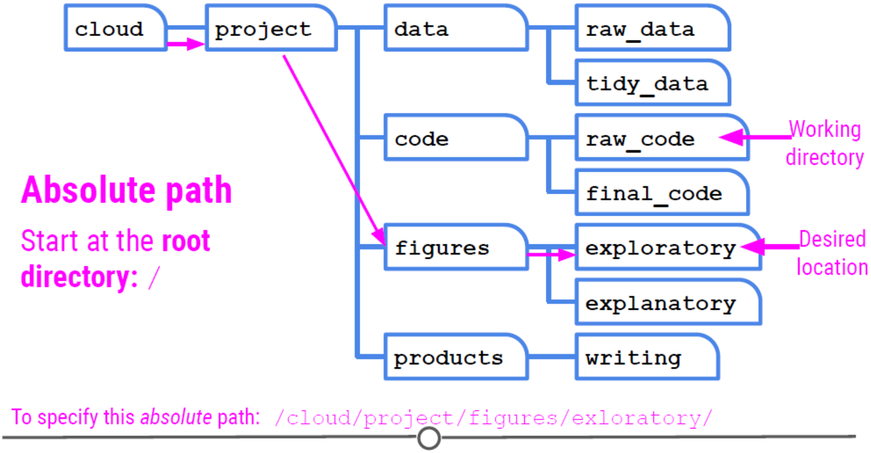


Relative path

Start at the **working directory:** `raw_code`

To specify this *relative* path: `../../figures/exploratory`

Relative and absolute paths



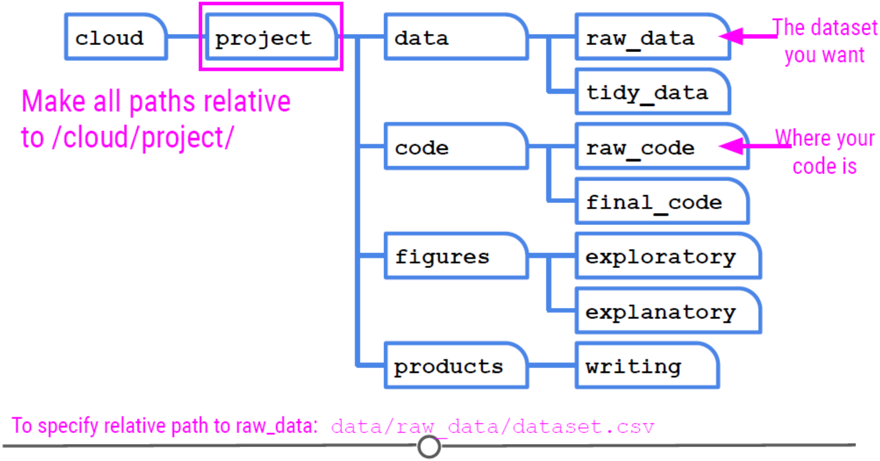
Absolute path

Start at the **root directory:** /

To specify this *absolute* path: /cloud/project/figures/exloratory/

Relative and absolute paths

- Suppose your data `dataset.csv` is in the folder `raw_data`



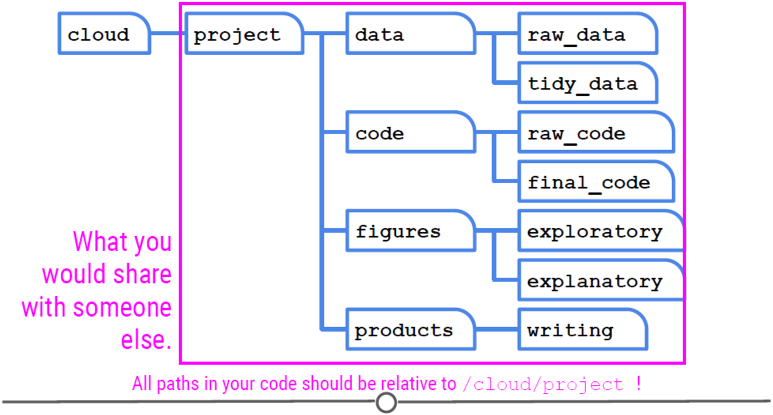
The dataset you want

Where your code is

Make all paths relative to /cloud/project/

To specify relative path to raw_data: `data/raw_data/dataset.csv`

Relative and absolute paths

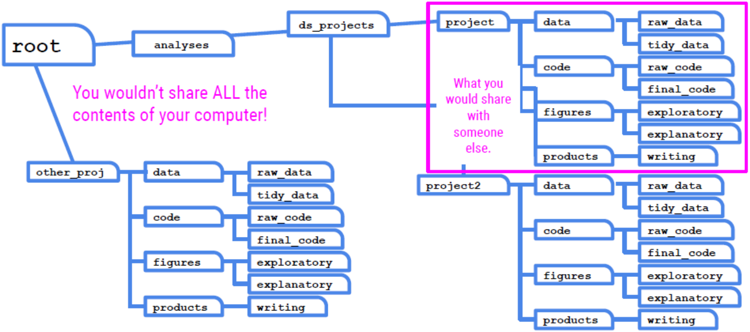


What you would share with someone else.

All paths in your code should be relative to `/cloud/project` !

Relative and absolute paths

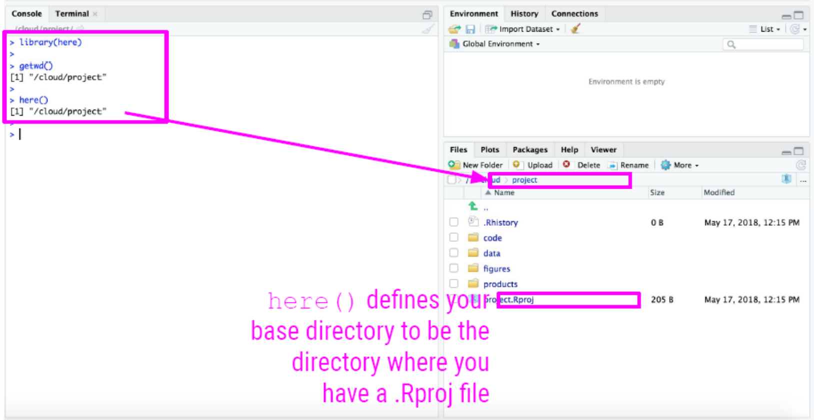
- Absolute paths do not make sense on someoneelse's computer



Absolute paths like `/root/analyses/ds_projects/project/`
wouldn't make sense on someone else's computer !

Relative and absolute paths

- The `here` package allows you to set the top level of your project folder as "here" and to specify where things live relative to that location.

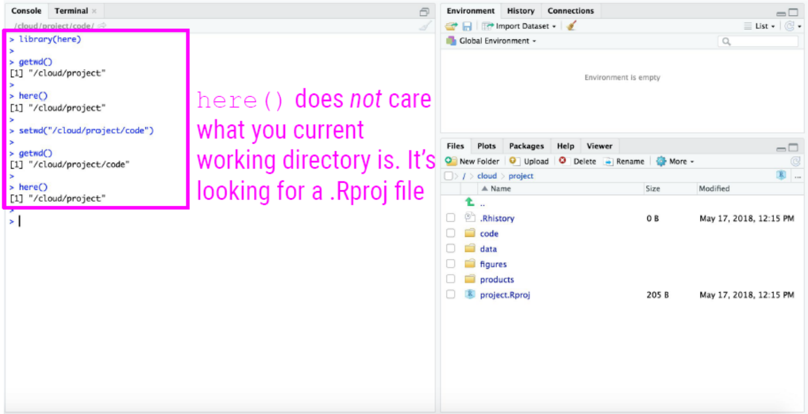


```
> library(here)
>
> getwd()
[1] "/cloud/project"
>
> here()
[1] "/cloud/project"
>
> |
```

Name	Size	Modified
..		
.Rhistory	0 B	May 17, 2018, 12:15 PM
code		
data		
figures		
products		
project.Rproj	205 B	May 17, 2018, 12:15 PM

here() defines your base directory to be the directory where you have a .Rproj file

Relative and absolute paths



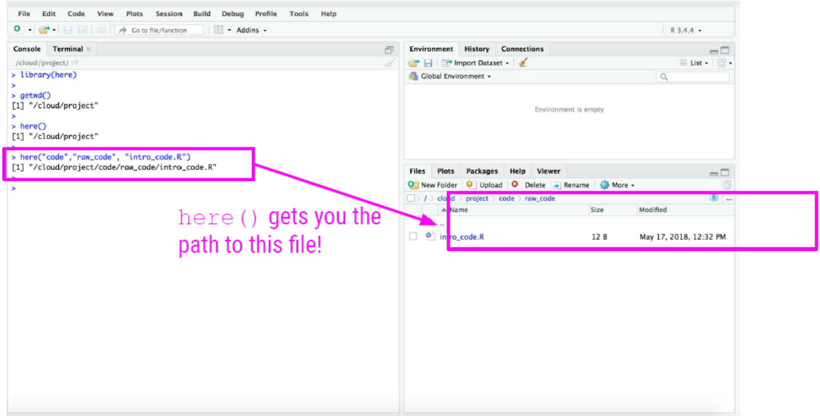
```
> library(here)
>
> getwd()
[1] "/cloud/project"
>
> here()
[1] "/cloud/project"
>
> setwd("/cloud/project/code")
>
> getwd()
[1] "/cloud/project/code"
>
> here()
[1] "/cloud/project"
>
> |
```

Name	Size	Modified
..		
.Rhistory	0 B	May 17, 2018, 12:15 PM
code		
data		
figures		
products		
project.Rproj	205 B	May 17, 2018, 12:15 PM

`here()` does *not* care what you current working directory is. It's looking for a .Rproj file

Relative and absolute paths

- `here()` simply builds a path to the top level of your project file everytime you use it.
    - ▸ If your data/code lives in the depths of several folders, you just string the folder references together.



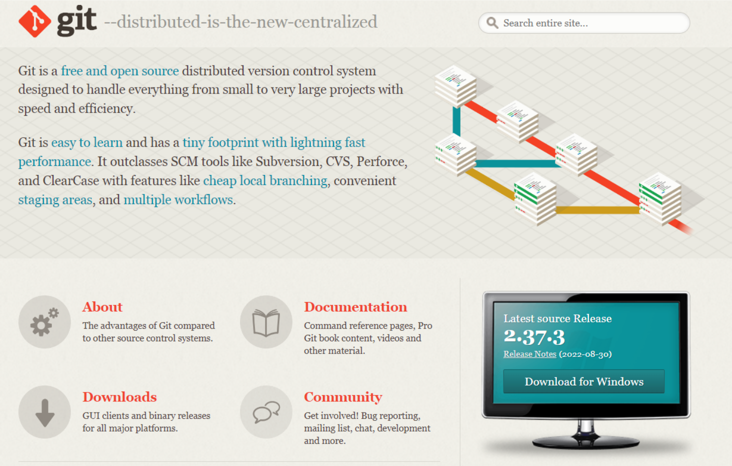
```
/cloud/project/
> library(here)
>
> getwd()
[1] "/cloud/project"
>
> here()
[1] "/cloud/project"
>
> here("code","raw_code", "intro_code.R")
[1] "/cloud/project/code/raw_code/intro_code.R"
>
>
```

/ > cloud > project > code > raw_code

Name	Size	Modified
..		
intro_code.R	12 B	May 17, 2018, 12:32 PM

`here()` gets you the path to this file!

Git: a version control system



git --distributed-is-the-new-centralized

Search entire site...

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release
2.37.3
Release Notes (2022-08-30)

Git: a version control system

- When developers create something (an app, for example), they make constant changes to the code, releasing new versions up to and after the first official (non-beta) release.
- VCS (Version control systems) keep these revisions straight, storing the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute.
- People who have nothing to do with the development of a project can still download the files and use them.

Git: a version control system

- Git is the most widely used VCS that was started by Linus Trovalds — the same person who created Linux.
- Git **repository** is a bunch of (local) files you want to manage in a sane way.
- GitHub is a Git repository hosting service. It is a place to hold Git repositories on the web.
- You want to publish/share the changes of a project to a remote repository (GitHub) so that others may see them/incorporate them into their own work.

Git: a version control system

- Git **repository** is a bunch of (local) files you want to manage in a sane way.
- repo = repository
- You can set up repo . . . then start your work;
- Or you can make a set of existing files and make them into a repo

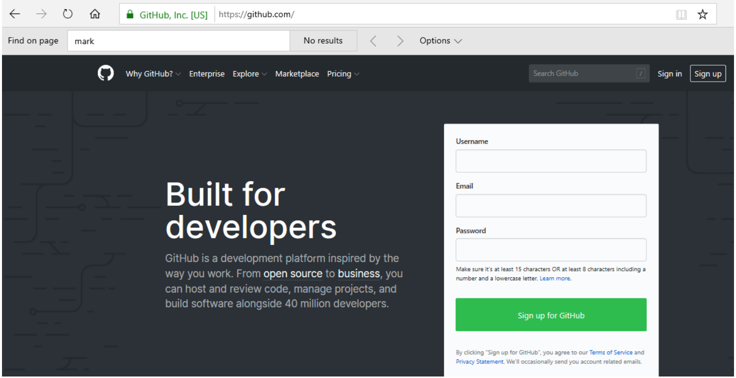
Git: a version control system

Basice use of Git:

- Change some files
- See what you have changed
  - `git status`
  - `git diff`
  - `git log`
- Indicate what changes to save
  - `git add`
- Commit to those changes
  - `git commit`
- Push changes to Github
  - `git push`
- Pull changes from your collaborator
  - `git pull`

Github

- We'll skip the version control part. We use Git for Github only in this course.



Find on page: mark

Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 40 million developers.

Username

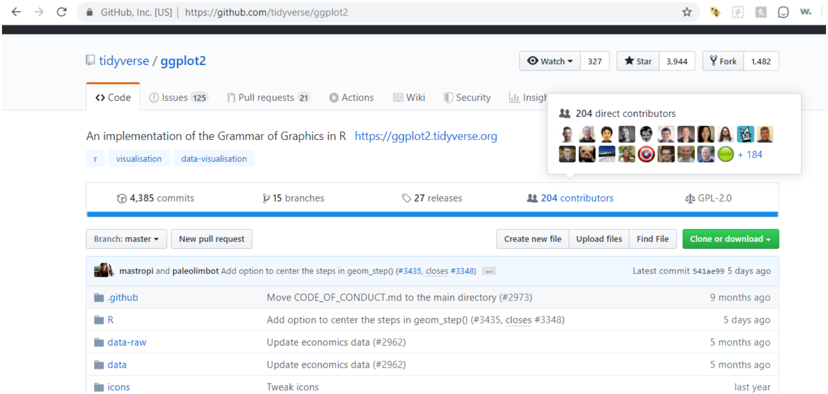
Email

Password

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. Learn more.

By clicking "Sign up for GitHub", you agree to our Terms of Service and Privacy Statement. We'll occasionally send you account related emails.

Github = a place to host Git repositories on the web; Git $\neq$ Github



GitHub, Inc. [US] | https://github.com/tidyverse/ggplot2

tidyverse / ggplot2

Watch 327 | Star 3,944 | Fork 1,482

Code | Issues 125 | Pull requests 21 | Actions | Wiki | Security | Insigh

204 direct contributors

+ 184

An implementation of the Grammar of Graphics in R https://ggplot2.tidyverse.org

r | visualisation | data-visualisation

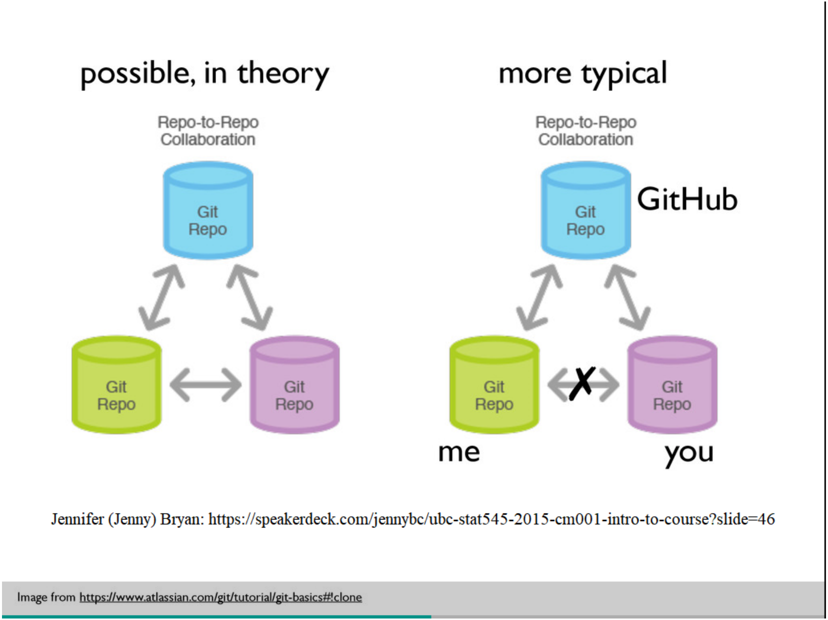
4,385 commits | 15 branches | 27 releases | 204 contributors | GPL-2.0

Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

mastropi and paleolimbot Add option to center the steps in geom_step() (#3435, closes #3348) ... Latest commit 541ae99 5 days ago

.github	Move CODE_OF_CONDUCT.md to the main directory (#2973)	9 months ago
R	Add option to center the steps in geom_step() (#3435, closes #3348)	5 days ago
data-raw	Update economics data (#2962)	5 months ago
data	Update economics data (#2962)	5 months ago
icons	Tweak icons	last year

Github



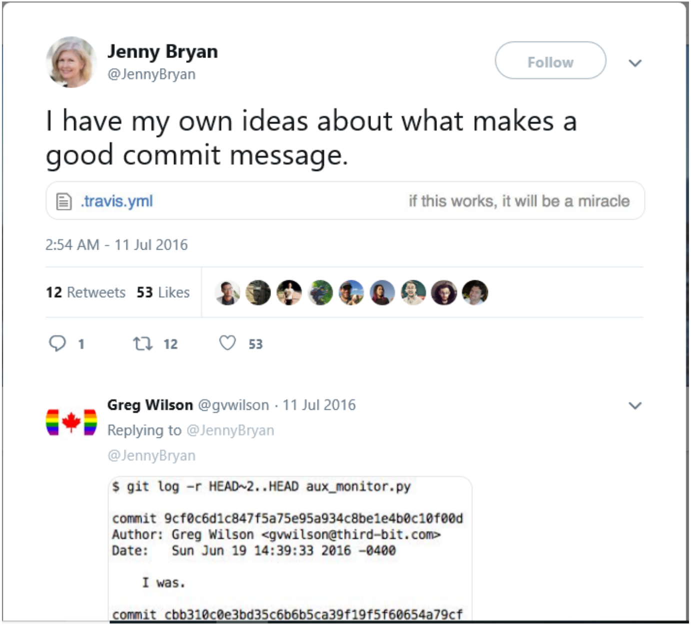
Jennifer (Jenny) Bryan: https://speakerdeck.com/jennybc/ubc-stat545-2015-cm001-intro-to-course?slide=46

Image from https://www.atlassian.com/git/tutorial/git-basics#!clone

Github

- Many R packages are developed in the open on Github.
- You can see exactly how files changed, when and by whom. If commit message is good, you'll see why.
  - Commit = a formal `checkpoint` or snapshot of the state of the repository.
- Github renders comma(.csv) and tab (.tsv) delimited files nicely.
- Github renders Markdown files nicely.
- You can see the raw Markdown too!

Github



Jenny Bryan
@JennyBryan

I have my own ideas about what makes a good commit message.

.travis.yml if this works, it will be a miracle

2:54 AM - 11 Jul 2016

12 Retweets 53 Likes

Greg Wilson @gvwilson · 11 Jul 2016
Replying to @JennyBryan
@JennyBryan

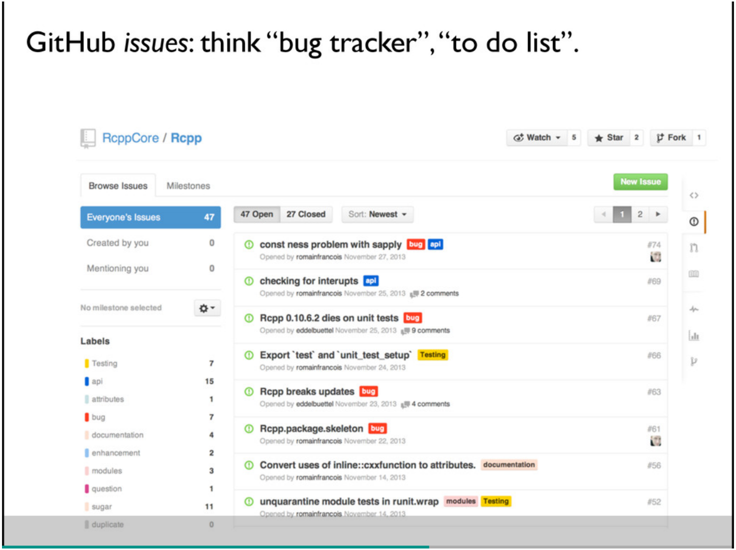
```
$ git log -r HEAD~2..HEAD aux_monitor.py

commit 9cf0c6d1c847f5a75e95a934c8be1e4b0c10f00d
Author: Greg Wilson <gvwilson@third-bit.com>
Date:   Sun Jun 19 14:39:33 2016 -0400

    I was.

commit cbb310c0e3bd35c6b6b5ca39f19f5f60654a79cf
```

Github



GitHub *issues*: think “bug tracker”, “to do list”.

RcppCore / **Rcpp**

Everyone's Issues	47
Created by you	0
Mentioning you	0

No milestone selected

Labels

Testing	7
api	15
attributes	1
bug	7
documentation	4
enhancement	2
modules	3
question	1
sugar	11
duplicate	0

- **const ness problem with sapply** bug api #74
  Opened by romainfrancois November 27, 2013
- **checking for interupts** api #69
  Opened by romainfrancois November 25, 2013 2 comments
- **Rcpp 0.10.6.2 dies on unit tests** bug #67
  Opened by eddelbuettel November 25, 2013 9 comments
- **Export `test` and `unit_test_setup`** Testing #66
  Opened by romainfrancois November 24, 2013
- **Rcpp breaks updates** bug #63
  Opened by eddelbuettel November 23, 2013 4 comments
- **Rcpp.package.skeleton** bug #61
  Opened by romainfrancois November 22, 2013
- **Convert uses of inline::cxxfunction to attributes.** documentation #56
  Opened by romainfrancois November 14, 2013
- **unquarantine module tests in runit.wrap** modules Testing #52
  Opened by romainfrancois November 14, 2013

For more information, see
https://github.com/jennybc/happy-git-and-github-for-the-user

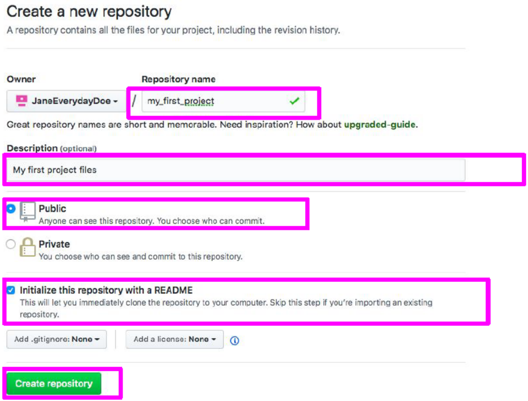
Install Git Locally

http://happygitwithr.com/install-git.html

- The lecture slides illustrate how to `push` your project in Rstudio.cloud to Github.
  - We do this in a similar way locally after we install `Git`.
  - So, let's install `Git` https://git-scm.com/downloads first.

Github Lab

1. Create a Github account at https://github.com/ (using the email address for signing up rstudio.cloud if you are using rstudio.cloud).
2. Create a repo and call it "my_first_project".



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: JaneEverydayDoe / Repository name: my_first_project ✓

Great repository names are short and memorable. Need inspiration? How about upgraded-guide.

Description (optional): My first project files

● Public
Anyone can see this repository. You choose who can commit.

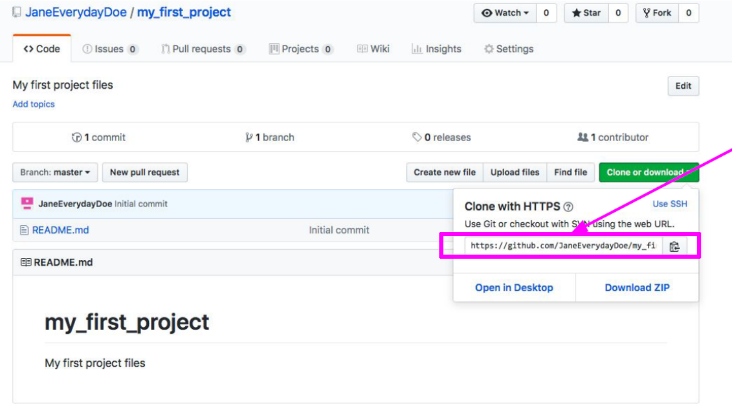
○ Private
You choose who can see and commit to this repository.

☑ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

Github Lab

3. Create a new project from this Github repo:
   - (a) First get the URL using the Clone/Download button on Github



JaneEverydayDoe / my_first_project

My first project files

JaneEverydayDoe Initial commit

README.md Initial commit

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

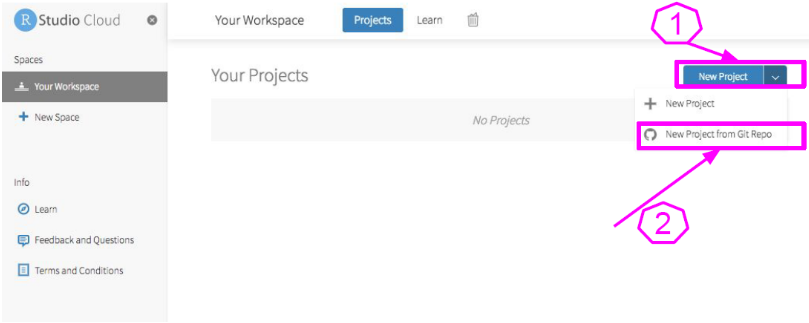
https://github.com/JaneEverydayDoe/my_fi

my_first_project

My first project files

Github Lab

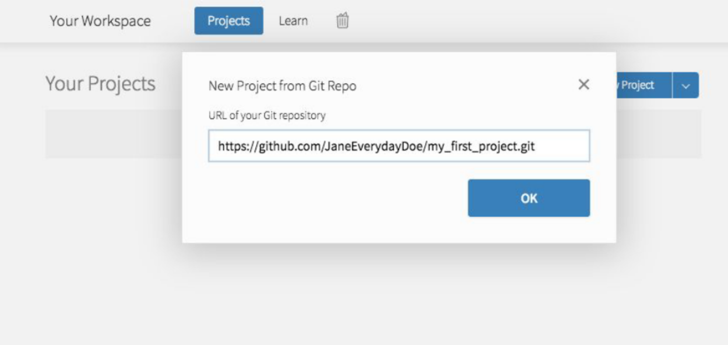
3. Create a new project from this Github repo:
   - (b) Then create a new project from Git repo on rstudio.cloud (or locally)



Github Lab

3. Create a new project from this Github repo:

- (b) Then create a new project from Git repo on rstudio.cloud (or locally)



New Project from Git Repo

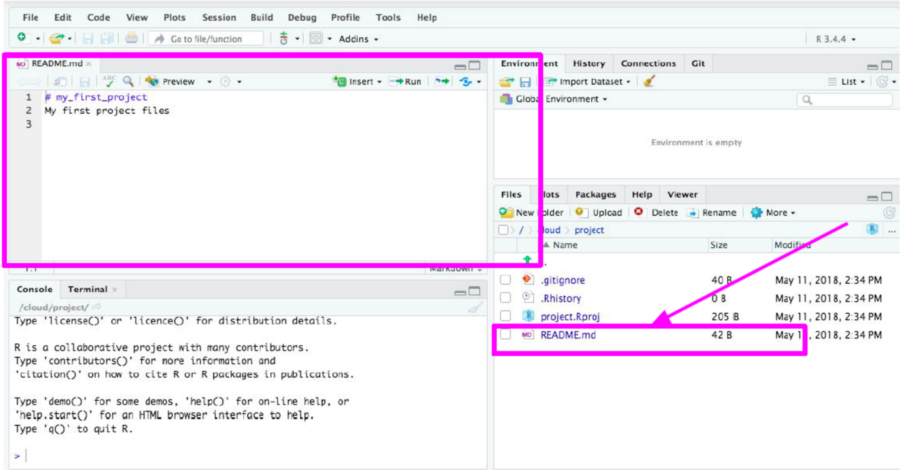
URL of your Git repository

https://github.com/JaneEverydayDoe/my_first_project.git

- You may be prompted for your username and password.

Github Lab

4. Update file README.md in this new repo by adding "Here, I am going to describle what is going on in my project."



```
# my_first_project
My first project files
```

	Name	Size	Modified
□	..		
□	.gitignore	40 B	May 11, 2018, 2:34 PM
□	.Rhistory	0 B	May 11, 2018, 2:34 PM
□	project.Rproj	205 B	May 11, 2018, 2:34 PM
□	README.md	42 B	May 11, 2018, 2:34 PM

```
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Github Lab

5. Add, commit, and push this file using Rstudio `terminal` or `Git Bash` as described below (or use the `Git` tab in Rstudio).

```
git add .
git commit -m "changed readme file"
git push
```

6. Look at the url https://github.com/your_username/my_first_project to confirm the updated file is there.

Github Lab

7. Create a set of directories in the folder on rstudio.cloud (for example, data/code/writing)
8. Commit, add and push this folder structure. What do you see? (You have to add a new file)
9. On https://github.com/yourusername/your_new_project click on the place where it says something like “3 commits”. You can see each of the changes you have made.
10. Find a partner and do this exercise (for later): https://github.com/kbroman/Tools4RR/blob/master/05_Git_Lab/git_lab.md

See **https://happygitwithr.com/new-github-first.html** for more information.

License