Before we start today...

- Create a Github account
- If possible (or necessary), install Git in your computer (check `https://happygitwithr.com/install-git.html`)

Git for Social Scientists: Introduction to Version Control with Git

Tomoya Sasaki[1]

Massachusetts Institute of Technology

October 20th, 2023

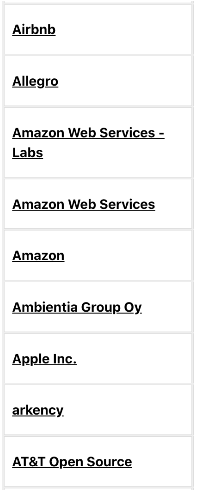
[1]This slide deck is heavily inspired by the workshop materials by Suyeol Yun and Shiro Kuriwaki.

Introduction

Companies using Git

Introduction

Companies using Git



Airbnb

Allegro

Amazon Web Services - Labs

Amazon Web Services

Amazon

Ambientia Group Oy

Apple Inc.

arkency

AT&T Open Source

Companies using Github

Introduction

- Why Git? Why Github? Why version control?

Introduction

- Why Git? Why Github? Why version control?
- These are essential tools for programmers

Introduction

- Why Git? Why Github? Why version control?
- These are essential tools for programmers
- How about social scientists?

Introduction

- Why Git? Why Github? Why version control?
- These are essential tools for programmers
- How about social scientists?
- My opinion: Social scientists also benefit from version control with Git

Introduction

- Why Git? Why Github? Why version control?
- These are essential tools for programmers
- How about social scientists?
- My opinion: Social scientists also benefit from version control with Git
    - Increase in collaborative projects

Introduction

- Why Git? Why Github? Why version control?
- These are essential tools for programmers
- How about social scientists?
- My opinion: Social scientists also benefit from version control with Git
  - Increase in collaborative projects
  - Demand for clean replication materials

Introduction

- Why Git? Why Github? Why version control?
- These are essential tools for programmers
- How about social scientists?
- My opinion: Social scientists also benefit from version control with Git
    - Increase in collaborative projects
    - Demand for clean replication materials
    - Complex data manipulation/preprocessing/analysis

Introduction

- Why Git? Why Github? Why version control?
- These are essential tools for programmers
- How about social scientists?
- My opinion: Social scientists also benefit from version control with Git
  - Increase in collaborative projects
  - Demand for clean replication materials
  - Complex data manipulation/preprocessing/analysis
- "Code and Data for the Social Sciences: A Practitioner's Guide" by Gentzkow and Shapiro has a chapter dedicated for version control

Introduction

- Why Git? Why Github? Why version control?
- These are essential tools for programmers
- How about social scientists?
- My opinion: Social scientists also benefit from version control with Git
    - Increase in collaborative projects
    - Demand for clean replication materials
    - Complex data manipulation/preprocessing/analysis
- "Code and Data for the Social Sciences: A Practitioner's Guide" by Gentzkow and Shapiro has a chapter dedicated for version control
- This workshop

Introduction

- Why Git? Why Github? Why version control?
- These are essential tools for programmers
- How about social scientists?
- My opinion: Social scientists also benefit from version control with Git
  - Increase in collaborative projects
  - Demand for clean replication materials
  - Complex data manipulation/preprocessing/analysis
- "Code and Data for the Social Sciences: A Practitioner's Guide" by Gentzkow and Shapiro has a chapter dedicated for version control
- This workshop
  - Introduction to version control

Introduction

- Why Git? Why Github? Why version control?
- These are essential tools for programmers
- How about social scientists?
- My opinion: Social scientists also benefit from version control with Git
    - Increase in collaborative projects
    - Demand for clean replication materials
    - Complex data manipulation/preprocessing/analysis
- "Code and Data for the Social Sciences: A Practitioner's Guide" by Gentzkow and Shapiro has a chapter dedicated for version control
- This workshop
    - Introduction to version control
    - Pros and cons of Git/Github

Introduction

- Why Git? Why Github? Why version control?
- These are essential tools for programmers
- How about social scientists?
- My opinion: Social scientists also benefit from version control with Git
  - Increase in collaborative projects
  - Demand for clean replication materials
  - Complex data manipulation/preprocessing/analysis
- "Code and Data for the Social Sciences: A Practitioner's Guide" by Gentzkow and Shapiro has a chapter dedicated for version control
- This workshop
  - Introduction to version control
  - Pros and cons of Git/Github
  - Brief introduction to these tools (how Git/Github works and how to use them)

What is version control?

- Version control: tracking and managing changes to file content

What is version control?

- Version control: tracking and managing changes to file content
- Git: (the most popular) software for version control

What is version control?

- Version control: tracking and managing changes to file content
- Git: (the most popular) software for version control
- Github: (the most popular) service to host your git on the Internet (alternatives include GitLab, Bitbucket ...)

What is version control?

- Version control: tracking and managing changes to file content
- Git: (the most popular) software for version control
- Github: (the most popular) service to host your git on the Internet (alternatives include GitLab, Bitbucket ...)
- Repository: unit of a version control project, your project folder with a subfolder named `.git`

What is version control?

- Version control: tracking and managing changes to file content
- Git: (the most popular) software for version control
- Github: (the most popular) service to host your git on the Internet (alternatives include GitLab, Bitbucket ...)
- Repository: unit of a version control project, your project folder with a subfolder named `.git`
  - Often simply called "repo"

What is version control?

- Version control: tracking and managing changes to file content
- Git: (the most popular) software for version control
- Github: (the most popular) service to host your git on the Internet (alternatives include GitLab, Bitbucket ...)
- Repository: unit of a version control project, your project folder with a subfolder named `.git`
    - Often simply called “repo”
    - `.git` folder in a repository tracks and stores every single change you make in the corresponding repository

What is version control?

- Version control: tracking and managing changes to file content
- Git: (the most popular) software for version control
- Github: (the most popular) service to host your git on the Internet (alternatives include GitLab, Bitbucket ...)
- Repository: unit of a version control project, your project folder with a subfolder named `.git`
    - Often simply called "repo"
    - `.git` folder in a repository tracks and stores every single change you make in the corresponding repository
- I focus on Git/Github because they are extremely popular than their alternatives

Benefit of Git/Github: Tracking who/how/when

- You can identify
  - who made changes

Benefit of Git/Github: Tracking who/how/when

- You can identify
    - who made changes
    - how they made the changes

Benefit of Git/Github: Tracking who/how/when

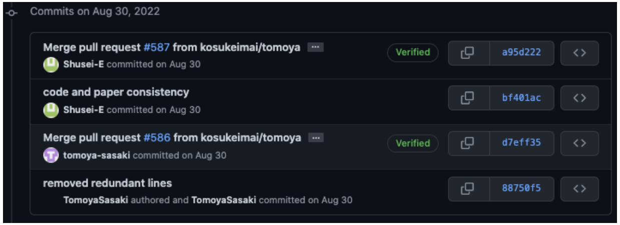
- You can identify
  - who made changes
  - how they made the changes
  - when they made the changes

Benefit of Git/Github: Tracking who/how/when

- You can identify
  - who made changes
  - how they made the changes
  - when they made the changes
- You can check the entire history since you created a repo and move back to previous versions easily (undo changes)

Benefit of Git/Github: Tracking who/how/when

- You can identify
  - who made changes
  - how they made the changes
  - when they made the changes
- You can check the entire history since you created a repo and move back to previous versions easily (undo changes)
- Github can visualize them nicely



Commits on Aug 30, 2022

Merge pull request #587 from kosukeimai/tomoya
Shusei-E committed on Aug 30 — Verified — a95d222

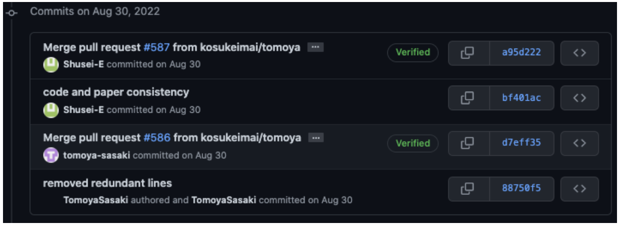
code and paper consistency
Shusei-E committed on Aug 30 — bf401ac

Merge pull request #586 from kosukeimai/tomoya
tomoya-sasaki committed on Aug 30 — Verified — d7eff35

removed redundant lines
TomoyaSasaki authored and TomoyaSasaki committed on Aug 30 — 88750f5

Benefit of Git/Github: Tracking who/how/when

- You can identify
  - who made changes
  - how they made the changes
  - when they made the changes
- You can check the entire history since you created a repo and move back to previous versions easily (undo changes)
- Github can visualize them nicely



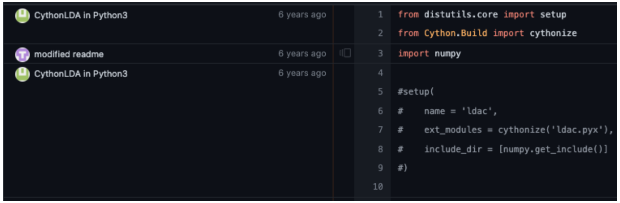
Commits on Aug 30, 2022

Merge pull request #587 from kosukeimai/tomoya
Shusei-E committed on Aug 30 — Verified — a95d222

code and paper consistency
Shusei-E committed on Aug 30 — bf401ac

Merge pull request #586 from kosukeimai/tomoya
tomoya-sasaki committed on Aug 30 — Verified — d7eff35

removed redundant lines
TomoyaSasaki authored and TomoyaSasaki committed on Aug 30 — 88750f5



CythonLDA in Python3 — 6 years ago
modified readme — 6 years ago
CythonLDA in Python3 — 6 years ago

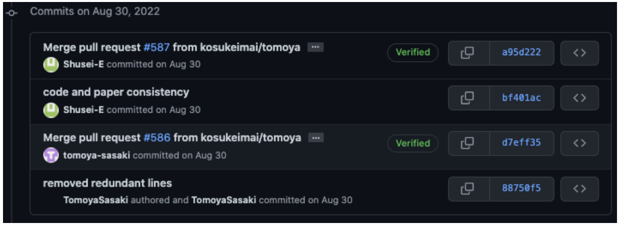
```
from distutils.core import setup
from Cython.Build import cythonize
import numpy

#setup(
#     name = 'ldac',
#     ext_modules = cythonize('ldac.pyx'),
#     include_dir = [numpy.get_include()]
#)

```

Benefit of Git/Github: Tracking who/how/when

- You can identify
  - who made changes
  - how they made the changes
  - when they made the changes
- You can check the entire history since you created a repo and move back to previous versions easily (undo changes)
- Github can visualize them nicely
- Useful when you
  - want to revert your (particular) changes

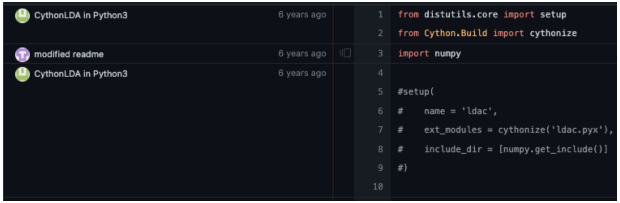

Commits on Aug 30, 2022

Merge pull request #587 from kosukeimai/tomoya
Shusei-E committed on Aug 30 — Verified — a95d222

code and paper consistency
Shusei-E committed on Aug 30 — bf401ac

Merge pull request #586 from kosukeimai/tomoya
tomoya-sasaki committed on Aug 30 — Verified — d7eff35

removed redundant lines
TomoyaSasaki authored and TomoyaSasaki committed on Aug 30 — 88750f5


CythonLDA in Python3 — 6 years ago
modified readme — 6 years ago
CythonLDA in Python3 — 6 years ago

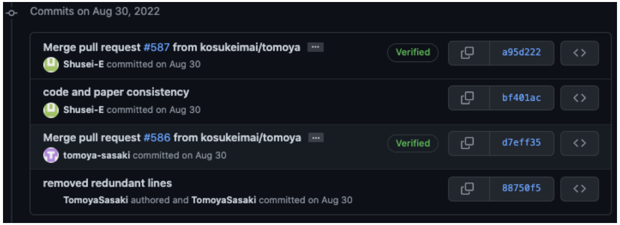
```
from distutils.core import setup
from Cython.Build import cythonize
import numpy

#setup(
#     name = 'ldac',
#     ext_modules = cythonize('ldac.pyx'),
#     include_dir = [numpy.get_include()]
#)

```

Benefit of Git/Github: Tracking who/how/when

- You can identify
  - who made changes
  - how they made the changes
  - when they made the changes
- You can check the entire history since you created a repo and move back to previous versions easily (undo changes)
- Github can visualize them nicely
- Useful when you
  - want to revert your (particular) changes
  - work on a collaborative project



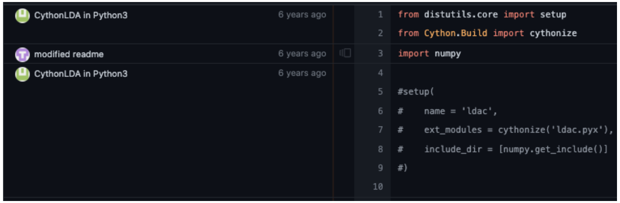
Commits on Aug 30, 2022

Merge pull request #587 from kosukeimai/tomoya ··· Shusei-E committed on Aug 30 — Verified — a95d222

code and paper consistency — Shusei-E committed on Aug 30 — bf401ac

Merge pull request #586 from kosukeimai/tomoya ··· tomoya-sasaki committed on Aug 30 — Verified — d7eff35

removed redundant lines — TomoyaSasaki authored and TomoyaSasaki committed on Aug 30 — 88750f5



CythonLDA in Python3 — 6 years ago

modified readme — 6 years ago

CythonLDA in Python3 — 6 years ago

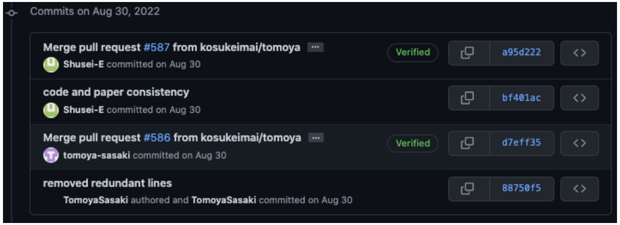
```
from distutils.core import setup
from Cython.Build import cythonize
import numpy

#setup(
#     name = 'ldac',
#     ext_modules = cythonize('ldac.pyx'),
#     include_dir = [numpy.get_include()]
#)

```

Benefit of Git/Github: Tracking who/how/when

- You can identify
  - who made changes
  - how they made the changes
  - when they made the changes
- You can check the entire history since you created a repo and move back to previous versions easily (undo changes)
- Github can visualize them nicely
- Useful when you
  - want to revert your (particular) changes
  - work on a collaborative project
- You don't need to keep
  - different versions of the same file: `clean_data_1104.R`, `clean_data_1020.R`



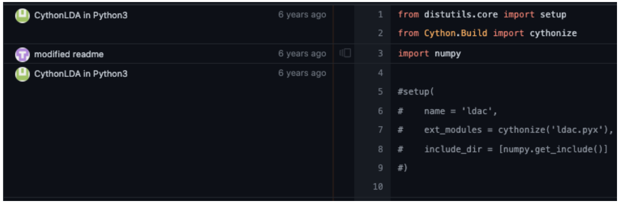
Commits on Aug 30, 2022

Merge pull request #587 from kosukeimai/tomoya
Shusei-E committed on Aug 30 — Verified — a95d222

code and paper consistency
Shusei-E committed on Aug 30 — bf401ac

Merge pull request #586 from kosukeimai/tomoya
tomoya-sasaki committed on Aug 30 — Verified — d7eff35

removed redundant lines
TomoyaSasaki authored and TomoyaSasaki committed on Aug 30 — 88750f5



CythonLDA in Python3 — 6 years ago
modified readme — 6 years ago
CythonLDA in Python3 — 6 years ago

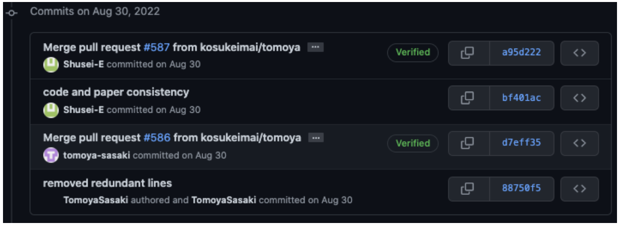
```
from distutils.core import setup
from Cython.Build import cythonize
import numpy

#setup(
#    name = 'ldac',
#    ext_modules = cythonize('ldac.pyx'),
#    include_dir = [numpy.get_include()]
#)

```

Benefit of Git/Github: Tracking who/how/when

- You can identify
  - who made changes
  - how they made the changes
  - when they made the changes
- You can check the entire history since you created a repo and move back to previous versions easily (undo changes)
- Github can visualize them nicely
- Useful when you
  - want to revert your (particular) changes
  - work on a collaborative project
- You don't need to keep
  - different versions of the same file: `clean_data_1104.R`, `clean_data_1020.R`
  - the same file edited by different people: `clean_data_tomoya.R`, `clean_data_adam.R`

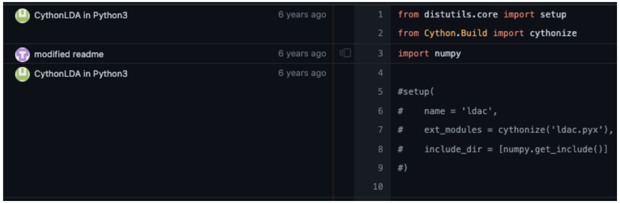

Commits on Aug 30, 2022

Merge pull request #587 from kosukeimai/tomoya
Shusei-E committed on Aug 30 — Verified — a95d222

code and paper consistency
Shusei-E committed on Aug 30 — bf401ac

Merge pull request #586 from kosukeimai/tomoya
tomoya-sasaki committed on Aug 30 — Verified — d7eff35

removed redundant lines
TomoyaSasaki authored and TomoyaSasaki committed on Aug 30 — 88750f5

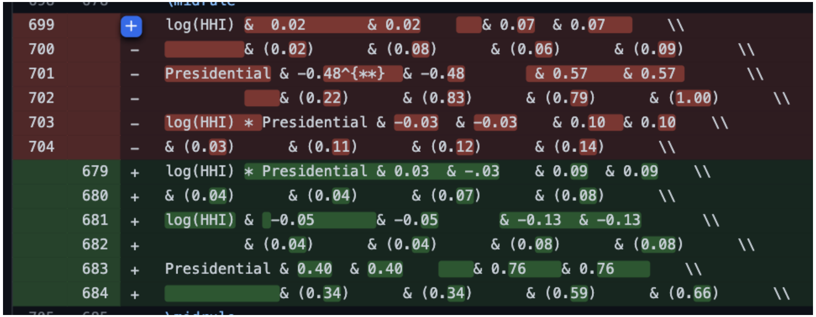

CythonLDA in Python3 — 6 years ago
modified readme — 6 years ago
CythonLDA in Python3 — 6 years ago

```
from distutils.core import setup
from Cython.Build import cythonize
import numpy

#setup(
#    name = 'ldac',
#    ext_modules = cythonize('ldac.pyx'),
#    include_dir = [numpy.get_include()]
#)
```

Benefit of Git/Github: Tracking who/how/when

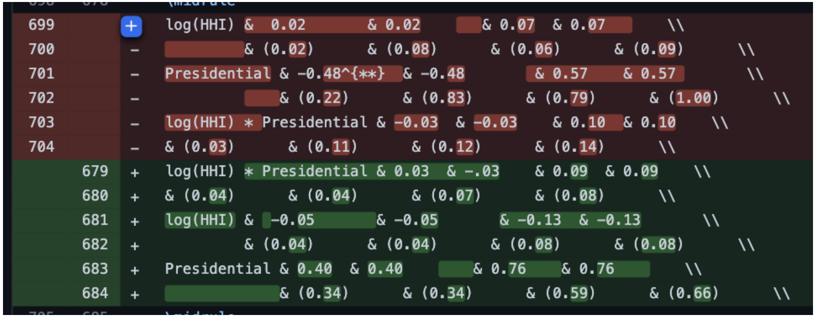
- You can check how the results change when we try different specification



```
- log(HHI) &  0.02          & 0.02       & 0.07  & 0.07        \\
-                    & (0.02)       & (0.08)       & (0.06)       & (0.09)       \\
- Presidential & -0.48^{**}  & -0.48        & 0.57    & 0.57         \\
-                    & (0.22)       & (0.83)       & (0.79)       & (1.00)       \\
- log(HHI) * Presidential & -0.03  & -0.03     & 0.10  & 0.10      \\
- & (0.03)       & (0.11)       & (0.12)       & (0.14)       \\
+ log(HHI) * Presidential & 0.03  & -.03      & 0.09  & 0.09      \\
+ & (0.04)       & (0.04)       & (0.07)       & (0.08)       \\
+ log(HHI) & -0.05          & -0.05        & -0.13  & -0.13         \\
+                    & (0.04)       & (0.04)       & (0.08)       & (0.08)       \\
+ Presidential & 0.40   & 0.40       & 0.76     & 0.76          \\
+                    & (0.34)       & (0.34)       & (0.59)       & (0.66)       \\
```

Benefit of Git/Github: Tracking who/how/when

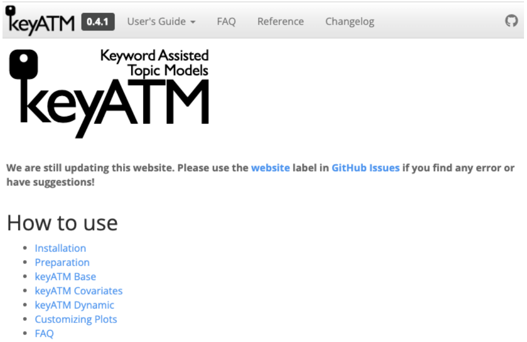
- You can check how the results change when we try different specification
- Easy to track which part of the results changed



```
  log(HHI) &  0.02        & 0.02      & 0.07  & 0.07     \\
-          & (0.02)       & (0.08)      & (0.06)      & (0.09)      \\
- Presidential & -0.48^{**}  & -0.48        & 0.57      & 0.57        \\
-              & (0.22)       & (0.83)      & (0.79)      & (1.00)      \\
- log(HHI) * Presidential & -0.03  & -0.03    & 0.10  & 0.10     \\
- & (0.03)      & (0.11)      & (0.12)      & (0.14)      \\
+ log(HHI) * Presidential & 0.03  & -.03     & 0.09  & 0.09     \\
+ & (0.04)      & (0.04)      & (0.07)      & (0.08)      \\
+ log(HHI) &  -0.05        & -0.05        & -0.13  & -0.13        \\
+          & (0.04)       & (0.04)      & (0.08)      & (0.08)      \\
+ Presidential & 0.40  & 0.40       & 0.76     & 0.76         \\
+              & (0.34)       & (0.34)      & (0.59)      & (0.66)      \\
```

Other side benefits of Git/Github

- Hosting a customizable website (free, no ads, tons of templates)



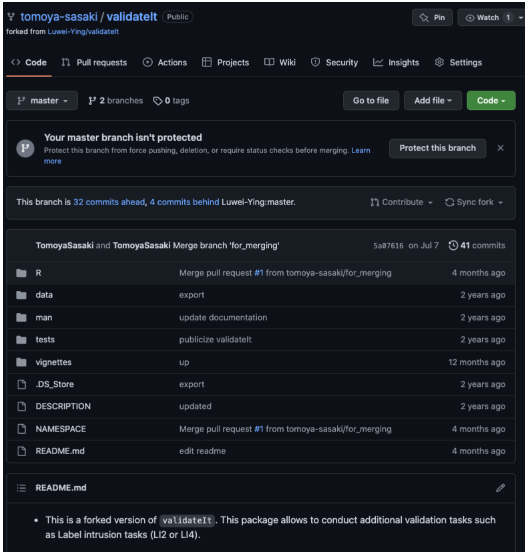
We are still updating this website. Please use the website label in GitHub Issues if you find any error or have suggestions!

How to use

- Installation
- Preparation
- keyATM Base
- keyATM Covariates
- keyATM Dynamic
- Customizing Plots
- FAQ

Other side benefits of Git/Github

- Hosting a customizable website (free, no ads, tons of templates)
- Contribute to software packages hosting on Github


tomoya-sasaki / validateIt Public
forked from Luwei-Ying/validateIt

Your master branch isn't protected
Protect this branch from force pushing, deletion, or require status checks before merging. Learn more

This branch is 32 commits ahead, 4 commits behind Luwei-Ying:master.

TomoyaSasaki and TomoyaSasaki Merge branch 'for_merging' 5a07616 on Jul 7 41 commits

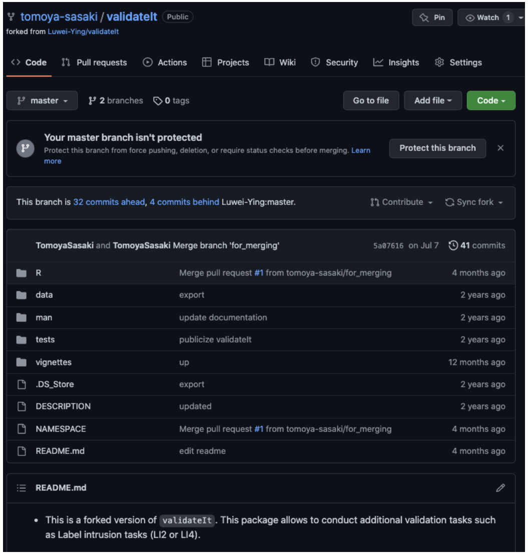
R	Merge pull request #1 from tomoya-sasaki/for_merging	4 months ago
data	export	2 years ago
man	update documentation	2 years ago
tests	publicize validateIt	2 years ago
vignettes	up	12 months ago
.DS_Store	export	2 years ago
DESCRIPTION	updated	2 years ago
NAMESPACE	Merge pull request #1 from tomoya-sasaki/for_merging	4 months ago
README.md	edit readme	4 months ago

README.md

- This is a forked version of `validateIt`. This package allows to conduct additional validation tasks such as Label intrusion tasks (LI2 or LI4).

Other side benefits of Git/Github

- Hosting a customizable website (free, no ads, tons of templates)
- Contribute to software packages hosting on Github
- Tweak a package developed by someone else for your own purposes



Your master branch isn't protected
Protect this branch from force pushing, deletion, or require status checks before merging. Learn more

This branch is 32 commits ahead, 4 commits behind Luwei-Ying:master.

TomoyaSasaki and TomoyaSasaki Merge branch 'for_merging' 5a07616 on Jul 7 41 commits

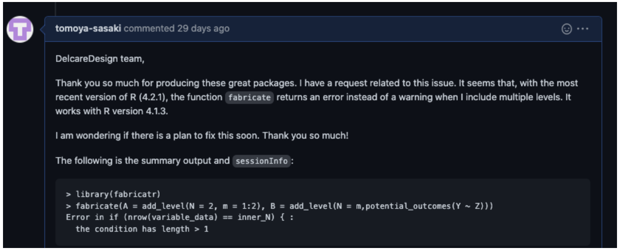
R	Merge pull request #1 from tomoya-sasaki/for_merging	4 months ago
data	export	2 years ago
man	update documentation	2 years ago
tests	publicize validateIt	2 years ago
vignettes	up	12 months ago
.DS_Store	export	2 years ago
DESCRIPTION	updated	2 years ago
NAMESPACE	Merge pull request #1 from tomoya-sasaki/for_merging	4 months ago
README.md	edit readme	4 months ago

README.md

- This is a forked version of `validateIt`. This package allows to conduct additional validation tasks such as Label intrusion tasks (LI2 or LI4).

Other side benefits of Git/Github

- Hosting a customizable website (free, no ads, tons of templates)
- Contribute to software packages hosting on Github
- Tweak a package developed by someone else for your own purposes
- Send a request to package developer (often happens at "Issue")


tomoya-sasaki commented 29 days ago

DelcareDesign team,

Thank you so much for producing these great packages. I have a request related to this issue. It seems that, with the most recent version of R (4.2.1), the function `fabricate` returns an error instead of a warning when I include multiple levels. It works with R version 4.1.3.

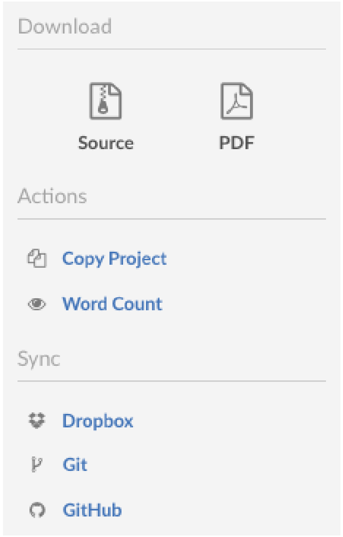
I am wondering if there is a plan to fix this soon. Thank you so much!

The following is the summary output and `sessionInfo`:

```
> library(fabricatr)
> fabricate(A = add_level(N = 2, m = 1:2), B = add_level(N = m,potential_outcomes(Y ~ Z)))
Error in if (nrow(variable_data) == inner_N) { :
  the condition has length > 1
```

Other side benefits of Git/Github

- Hosting a customizable website (free, no ads, tons of templates)
- Contribute to software packages hosting on Github
- Tweak a package developed by someone else for your own purposes
- Send a request to package developer (often happens at “Issue”)
- Nice integration with popular apps/websites such as RStudio and Overleaf



Limitations: not suitable for tracking large files

- Github imposes file size limit:

Limitations: not suitable for tracking large files

- Github imposes file size limit:
  - 25 MB per file limit (you can change this limit up to 100MB by changing setup)

Limitations: not suitable for tracking large files

- Github imposes file size limit:
  - 25 MB per file limit (you can change this limit up to 100MB by changing setup)
  - 1GB per repo limit

Limitations: not suitable for tracking large files

- Github imposes file size limit:
  - 25 MB per file limit (you can change this limit up to 100MB by changing setup)
  - 1GB per repo limit
- Remember that `.git` tracks and stores all the change you make in a repo

Limitations: not suitable for tracking large files

- Github imposes file size limit:
    - 25 MB per file limit (you can change this limit up to 100MB by changing setup)
    - 1GB per repo limit
- Remember that `.git` tracks and stores all the change you make in a repo

  $\rightsquigarrow$ if you store a huge file in the repo and let `.git` tracks its changes, the `.git` folder can grow quite huge

Limitations: not suitable for tracking large files

- Github imposes file size limit:
  - 25 MB per file limit (you can change this limit up to 100MB by changing setup)
  - 1GB per repo limit
- Remember that `.git` tracks and stores all the change you make in a repo

  $\rightsquigarrow$ if you store a huge file in the repo and let `.git` tracks its changes, the `.git` folder can grow quite huge
- Use `.gitignore` to specify files that Git should not track ("ignore")

.gitignore
```
      *.csv # ignore csv files
      /data/ # ignore any file in data folder
```

Limitations: not suitable for tracking large files

- Github imposes file size limit:
  - 25 MB per file limit (you can change this limit up to 100MB by changing setup)
  - 1GB per repo limit
- Remember that `.git` tracks and stores all the change you make in a repo

  $\rightsquigarrow$ if you store a huge file in the repo and let `.git` tracks its changes, the `.git` folder can grow quite huge

- Use `.gitignore` to specify files that Git should not track ("ignore")

`.gitignore`
```
*.csv # ignore csv files
/data/ # ignore any file in data folder
```

- Include huge files as well as sensitive files that contain password, API key etc in `.gitignore`

Limitations: not great if you want to track non-text files

- Git cannot track line by line changes for non-text files such as PDF, Microsoft Word/Excel/Powerpoint, JPG, ...

Limitations: not great if you want to track non-text files

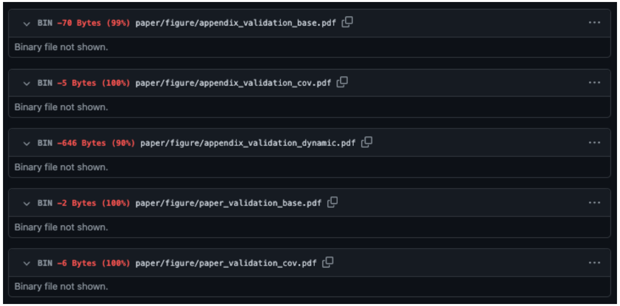
- Git cannot track line by line changes for non-text files such as PDF, Microsoft Word/Excel/Powerpoint, JPG, ...
- Note that Git still tracks changes

Limitations: not great if you want to track non-text files

- Git cannot track line by line changes for non-text files such as PDF, Microsoft Word/Excel/Powerpoint, JPG, ...
- Note that Git still tracks changes
- The value of Git/Github is limited

Limitations: not great if you want to track non-text files

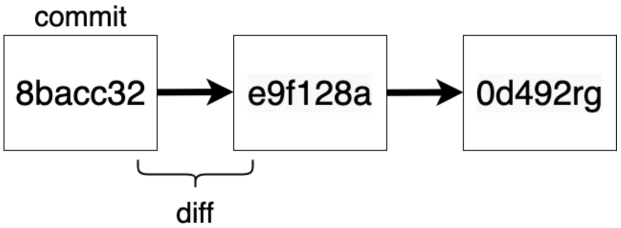
- Git cannot track line by line changes for non-text files such as PDF, Microsoft Word/Excel/Powerpoint, JPG, ...
- Note that Git still tracks changes
- The value of Git/Github is limited
- In the right example, Git/Github recognizes the changes as the changes in file sizes
  $\rightsquigarrow$ even though you update a figure in PDF or PNG format, Git/Github might not recognize it unless the file size changes...



```
BIN -70 Bytes (99%) paper/figure/appendix_validation_base.pdf
Binary file not shown.
BIN -5 Bytes (100%) paper/figure/appendix_validation_cov.pdf
Binary file not shown.
BIN -646 Bytes (90%) paper/figure/appendix_validation_dynamic.pdf
Binary file not shown.
BIN -2 Bytes (100%) paper/figure/paper_validation_base.pdf
Binary file not shown.
BIN -6 Bytes (100%) paper/figure/paper_validation_cov.pdf
Binary file not shown.
```

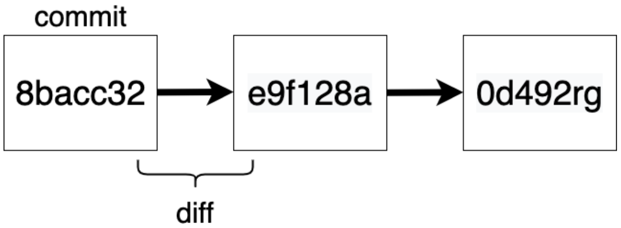
How Git tracks files

- Git tracks changes in files with "**commit**"



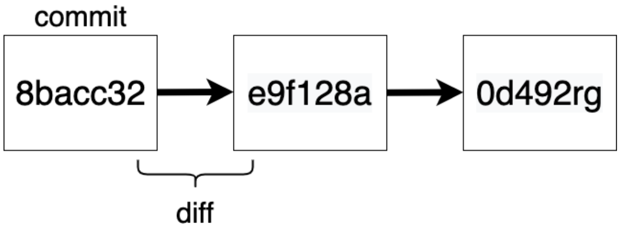
How Git tracks files

- Git tracks changes in files with "**commit**"
- Hash (labeled with a random number) is assigned to each commit



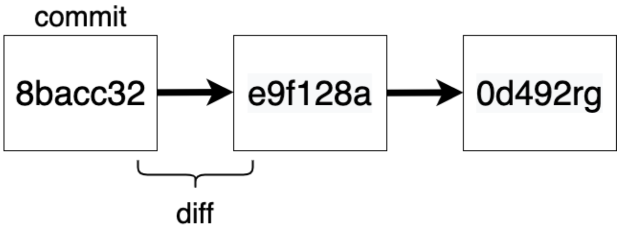
How Git tracks files

- Git tracks changes in files with "**commit**"
- Hash (labeled with a random number) is assigned to each commit
- Changes between commits are called "**diff**"



How Git tracks files

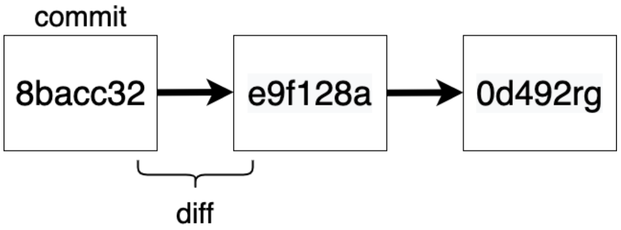
- Git tracks changes in files with "**commit**"
- Hash (labeled with a random number) is assigned to each commit
- Changes between commits are called "**diff**"
- Each commit is a snapshot of your repo at specific times



- 8bacc332: "test commit"
- e9f128a: "edit analysis code"
- 0d492rg: "add new analysis"

How Git tracks files

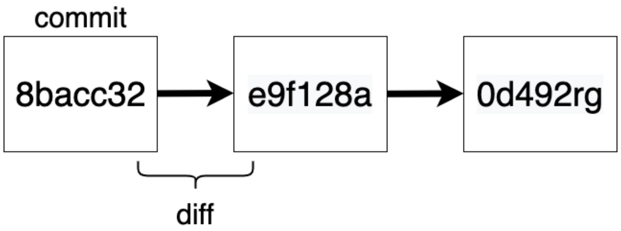
- Git tracks changes in files with "**commit**"
- Hash (labeled with a random number) is assigned to each commit
- Changes between commits are called "**diff**"
- Each commit is a snapshot of your repo at specific times
- Commit has a human-readable message and an (first few characters of) commit ID



- 8bacc332: "test commit"
- e9f128a: "edit analysis code"
- 0d492rg: "add new analysis"

How Git tracks files

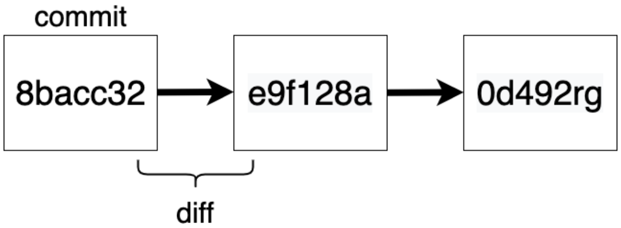
- Git tracks changes in files with "**commit**"
- Hash (labeled with a random number) is assigned to each commit
- Changes between commits are called "**diff**"
- Each commit is a snapshot of your repo at specific times
- Commit has a human-readable message and an (first few characters of) commit ID
- Commit is not automatic and you actively make a "commit" with a message



- 8bacc332: "test commit"
- e9f128a: "edit analysis code"
- 0d492rg: "add new analysis"

How Git tracks files

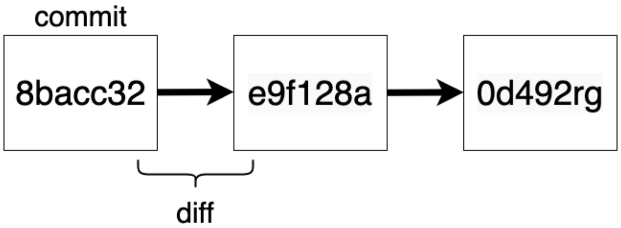
- Git tracks changes in files with "**commit**"
- Hash (labeled with a random number) is assigned to each commit
- Changes between commits are called "**diff**"
- Each commit is a snapshot of your repo at specific times
- Commit has a human-readable message and an (first few characters of) commit ID
- Commit is not automatic and you actively make a "commit" with a message
- Ideally commit should be atomic units of changes that represent a specific idea



- 8bacc332: "test commit"
- e9f128a: "edit analysis code"
- 0d492rg: "add new analysis"

How Git tracks files

- Git tracks changes in files with "**commit**"
- Hash (labeled with a random number) is assigned to each commit
- Changes between commits are called "**diff**"
- Each commit is a snapshot of your repo at specific times
- Commit has a human-readable message and an (first few characters of) commit ID
- Commit is not automatic and you actively make a "commit" with a message
- Ideally commit should be atomic units of changes that represent a specific idea
- You need to "**stage**" the files you want to add to commit (more later)



- 8bacc332: "test commit"
- e9f128a: "edit analysis code"
- 0d492rg: "add new analysis"

Github and more on repos

- If you just work on your local repos (and only on one computer), you don't need Github

Github and more on repos

- If you just work on your local repos (and only on one computer), you don't need Github
- Repos: unit of a version control project, contains a folder with a subfolder named `.git`

Github and more on repos

- If you just work on your local repos (and only on one computer), you don't need Github
- Repos: unit of a version control project, contains a folder with a subfolder named `.git`
    - Local repos: repos (folder) in your own computer, ordinary project folder with subfolder named `.git`

Github and more on repos

- If you just work on your local repos (and only on one computer), you don't need Github
- Repos: unit of a version control project, contains a folder with a subfolder named `.git`
    - Local repos: repos (folder) in your own computer, ordinary project folder with subfolder named `.git`
    - Remote repos: repos (folder) in a web hosting services such as Github

Github and more on repos

- If you just work on your local repos (and only on one computer), you don't need Github
- Repos: unit of a version control project, contains a folder with a subfolder named `.git`
  - Local repos: repos (folder) in your own computer, ordinary project folder with subfolder named `.git`
  - Remote repos: repos (folder) in a web hosting services such as Github
- Remote repos such as Github...

Github and more on repos

- If you just work on your local repos (and only on one computer), you don't need Github
- Repos: unit of a version control project, contains a folder with a subfolder named `.git`
  - Local repos: repos (folder) in your own computer, ordinary project folder with subfolder named `.git`
  - Remote repos: repos (folder) in a web hosting services such as Github
- Remote repos such as Github...
  - come with unique URL (`https://github.com/username/reponame.git`)

Github and more on repos

- If you just work on your local repos (and only on one computer), you don't need Github
- Repos: unit of a version control project, contains a folder with a subfolder named `.git`
    - Local repos: repos (folder) in your own computer, ordinary project folder with subfolder named `.git`
    - Remote repos: repos (folder) in a web hosting services such as Github
- Remote repos such as Github...
    - come with unique URL (`https://github.com/username/reponame.git`)
    - are basically a copy of your local repos

Github and more on repos

- If you just work on your local repos (and only on one computer), you don't need Github
- Repos: unit of a version control project, contains a folder with a subfolder named `.git`
    - Local repos: repos (folder) in your own computer, ordinary project folder with subfolder named `.git`
    - Remote repos: repos (folder) in a web hosting services such as Github
- Remote repos such as Github...
    - come with unique URL (`https://github.com/username/reponame.git`)
    - are basically a copy of your local repos
    - nicely visualize your commits and also serve as a backup of your codes (remember, not for data)

Github and more on repos

- If you just work on your local repos (and only on one computer), you don't need Github
- Repos: unit of a version control project, contains a folder with a subfolder named `.git`
  - Local repos: repos (folder) in your own computer, ordinary project folder with subfolder named `.git`
  - Remote repos: repos (folder) in a web hosting services such as Github
- Remote repos such as Github...
  - come with unique URL (`https://github.com/username/reponame.git`)
  - are basically a copy of your local repos
  - nicely visualize your commits and also serve as a backup of your codes (remember, not for data)
  - useful when you work on the same project with multiple computers

Github and more on repos

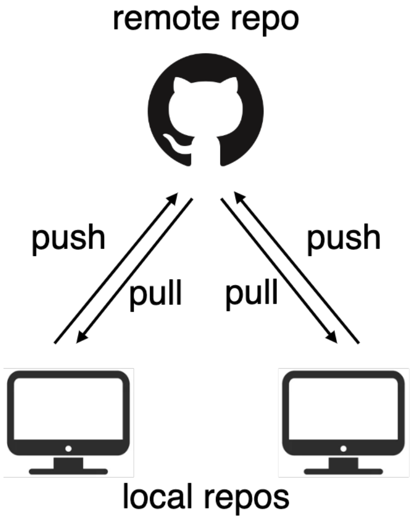
- If you just work on your local repos (and only on one computer), you don't need Github
- Repos: unit of a version control project, contains a folder with a subfolder named `.git`
  - Local repos: repos (folder) in your own computer, ordinary project folder with subfolder named `.git`
  - Remote repos: repos (folder) in a web hosting services such as Github
- Remote repos such as Github...
  - come with unique URL (`https://github.com/username/reponame.git`)
  - are basically a copy of your local repos
  - nicely visualize your commits and also serve as a backup of your codes (remember, not for data)
  - useful when you work on the same project with multiple computers
- You can choose remote repo to be public or private

Github and more on repos

- If you just work on your local repos (and only on one computer), you don't need Github
- Repos: unit of a version control project, contains a folder with a subfolder named `.git`
    - Local repos: repos (folder) in your own computer, ordinary project folder with subfolder named `.git`
    - Remote repos: repos (folder) in a web hosting services such as Github
- Remote repos such as Github...
    - come with unique URL (`https://github.com/username/reponame.git`)
    - are basically a copy of your local repos
    - nicely visualize your commits and also serve as a backup of your codes (remember, not for data)
    - useful when you work on the same project with multiple computers
- You can choose remote repo to be public or private
- Github accounts let you create unlimited number of private repos with upto 3 collaborators unlimited number of public repos with unlimited number of collaborators

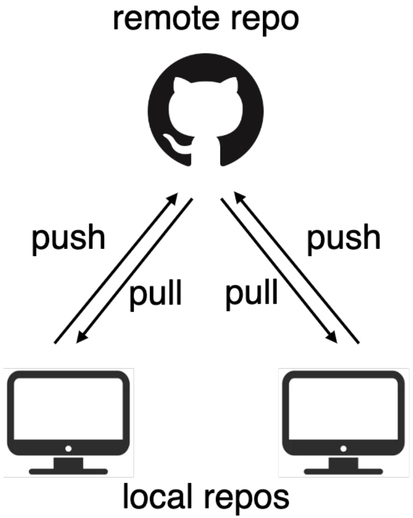
How to interact with remote repos on Github

- Interaction with remote repos: **pull** and **push**



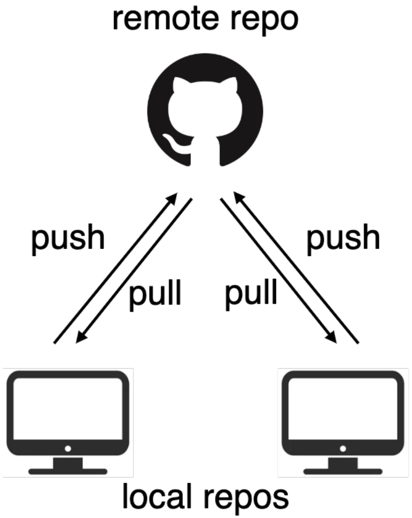
How to interact with remote repos on Github

- Interaction with remote repos: **pull** and **push**
- You **push** the commits you made to remote repos



How to interact with remote repos on Github

- Interaction with remote repos: **pull** and **push**
- You **push** the commits you made to remote repos
- You **pull** new commits on remote repos
- Technically "pull" does **fetch** and **merge**



How to use Git/Github

- So far, we have covered some fundamental concepts around Git/Github

How to use Git/Github

- So far, we have covered some fundamental concepts around Git/Github
- But how do we initialize a repo and commit/pull/push in practice?

How to use Git/Github

- So far, we have covered some fundamental concepts around Git/Github
- But how do we initialize a repo and commit/pull/push in practice?
- Command line, Github desktop app etc

How to use Git/Github

- So far, we have covered some fundamental concepts around Git/Github
- But how do we initialize a repo and commit/pull/push in practice?
- Command line, Github desktop app etc
- This workshop focuses on RStudio

How to use Git/Github

- So far, we have covered some fundamental concepts around Git/Github
- But how do we initialize a repo and commit/pull/push in practice?
- Command line, Github desktop app etc
- This workshop focuses on RStudio
- I personally use command line

General (suggestive) initialization

1. Establish a connection between your computer and Github

General (suggestive) initialization

1. Establish a connection between your computer and Github
2. Create a repo on Github (create a public one for now)

General (suggestive) initialization

1. Establish a connection between your computer and Github
2. Create a repo on Github (create a public one for now)
3. **Clone** your remote repo on Github to your computer (can be any location). This is your local repo.

General (suggestive) initialization

1. Establish a connection between your computer and Github
2. Create a repo on Github (create a public one for now)
3. **Clone** your remote repo on Github to your computer (can be any location). This is your local repo.
4. Edit `.gitignore` to make sure that Git won't track any sensitive or huge file

General (suggestive) initialization

1. Establish a connection between your computer and Github
2. Create a repo on Github (create a public one for now)
3. **Clone** your remote repo on Github to your computer (can be any location). This is your local repo.
4. Edit `.gitignore` to make sure that Git won't track any sensitive or huge file

- If you have any existing project that you want to track with Git, create a remote repo on Github, clone it to your computer (i.e., local repo), and move codes and folders into the local repo

General (suggestive) daily workflow

1. Edit, add, or delete files in your repo

General (suggestive) daily workflow

1. Edit, add, or delete files in your repo
2. Stage files you want to add to commit and make a commit when you make some changes that represent a specific idea or solve particular issues

General (suggestive) daily workflow

1. Edit, add, or delete files in your repo
2. Stage files you want to add to commit and make a commit when you make some changes that represent a specific idea or solve particular issues
    - Add lines to clean data

General (suggestive) daily workflow

1. Edit, add, or delete files in your repo
2. Stage files you want to add to commit and make a commit when you make some changes that represent a specific idea or solve particular issues
    - Add lines to clean data
    - Add lines to visualize regression results

General (suggestive) daily workflow

1. Edit, add, or delete files in your repo
2. Stage files you want to add to commit and make a commit when you make some changes that represent a specific idea or solve particular issues
    - Add lines to clean data
    - Add lines to visualize regression results
    - Delete lines that are irrelevant anymore

General (suggestive) daily workflow

1. Edit, add, or delete files in your repo
2. Stage files you want to add to commit and make a commit when you make some changes that represent a specific idea or solve particular issues
    - Add lines to clean data
    - Add lines to visualize regression results
    - Delete lines that are irrelevant anymore
3. Push when you make a set of commits or when you call it a day

General (suggestive) daily workflow

1. Edit, add, or delete files in your repo
2. Stage files you want to add to commit and make a commit when you make some changes that represent a specific idea or solve particular issues
   - Add lines to clean data
   - Add lines to visualize regression results
   - Delete lines that are irrelevant anymore
3. Push when you make a set of commits or when you call it a day
4. Pull any new commits in the remote repo if there are commits and pushes to the remote repo from other computers

General (suggestive) daily workflow

1. Edit, add, or delete files in your repo
2. Stage files you want to add to commit and make a commit when you make some changes that represent a specific idea or solve particular issues
    - Add lines to clean data
    - Add lines to visualize regression results
    - Delete lines that are irrelevant anymore
3. Push when you make a set of commits or when you call it a day
4. Pull any new commits in the remote repo if there are commits and pushes to the remote repo from other computers
5. Repeat the process above

General (suggestive) daily workflow

1. Edit, add, or delete files in your repo
2. Stage files you want to add to commit and make a commit when you make some changes that represent a specific idea or solve particular issues
   - Add lines to clean data
   - Add lines to visualize regression results
   - Delete lines that are irrelevant anymore
3. Push when you make a set of commits or when you call it a day
4. Pull any new commits in the remote repo if there are commits and pushes to the remote repo from other computers
5. Repeat the process above

- Push at least once a day if you make any edit

General (suggestive) daily workflow

1. Edit, add, or delete files in your repo
2. Stage files you want to add to commit and make a commit when you make some changes that represent a specific idea or solve particular issues
   - Add lines to clean data
   - Add lines to visualize regression results
   - Delete lines that are irrelevant anymore
3. Push when you make a set of commits or when you call it a day
4. Pull any new commits in the remote repo if there are commits and pushes to the remote repo from other computers
5. Repeat the process above

- Push at least once a day if you make any edit
- Let's try out with Github and RStudio

Setup authentication

- When we interact with a remote Git server, such as GitHub, we need to provide credentials

Setup authentication

- When we interact with a remote Git server, such as GitHub, we need to provide credentials
- Github provides two authentication methods, HTTPS and SSH

Setup authentication

- When we interact with a remote Git server, such as GitHub, we need to provide credentials
- Github provides two authentication methods, HTTPS and SSH
- I recommend HTTPS and will use HTTPS in this workshop

Setup authentication

- When we interact with a remote Git server, such as GitHub, we need to provide credentials
- Github provides two authentication methods, HTTPS and SSH
- I recommend HTTPS and will use HTTPS in this workshop
- Procedure

Setup authentication

- When we interact with a remote Git server, such as GitHub, we need to provide credentials
- Github provides two authentication methods, HTTPS and SSH
- I recommend HTTPS and will use HTTPS in this workshop
- Procedure
  1. Go to `https://github.com/settings/tokens` and click "Generate token"

Setup authentication

- When we interact with a remote Git server, such as GitHub, we need to provide credentials
- Github provides two authentication methods, HTTPS and SSH
- I recommend HTTPS and will use HTTPS in this workshop
- Procedure
  1. Go to `https://github.com/settings/tokens` and click "Generate token"
  2. Decide the scope. Choose (at least) "repo", "user", and "workflow".

Setup authentication

- When we interact with a remote Git server, such as GitHub, we need to provide credentials
- Github provides two authentication methods, HTTPS and SSH
- I recommend HTTPS and will use HTTPS in this workshop
- Procedure
  1. Go to `https://github.com/settings/tokens` and click "Generate token"
  2. Decide the scope. Choose (at least) "repo", "user", and "workflow".
  3. Click "Generate token". This is the same as the password to interact with Git. You shouldn't show this to anyone.

Setup authentication

- When we interact with a remote Git server, such as GitHub, we need to provide credentials
- Github provides two authentication methods, HTTPS and SSH
- I recommend HTTPS and will use HTTPS in this workshop
- Procedure
  1. Go to `https://github.com/settings/tokens` and click "Generate token"
  2. Decide the scope. Choose (at least) "repo", "user", and "workflow".
  3. Click "Generate token". This is the same as the password to interact with Git. You shouldn't show this to anyone.
  4. Copy the generated strings (PAT, personal access token)

Setup authentication

- When we interact with a remote Git server, such as GitHub, we need to provide credentials
- Github provides two authentication methods, HTTPS and SSH
- I recommend HTTPS and will use HTTPS in this workshop
- Procedure
  1. Go to `https://github.com/settings/tokens` and click "Generate token"
  2. Decide the scope. Choose (at least) "repo", "user", and "workflow".
  3. Click "Generate token". This is the same as the password to interact with Git. You shouldn't show this to anyone.
  4. Copy the generated strings (PAT, personal access token)
  5. Open RStudio and run `gitcreds::gitcreds_set()` (Install the `gitcreds` package beforehand)

Setup authentication

- When we interact with a remote Git server, such as GitHub, we need to provide credentials
- Github provides two authentication methods, HTTPS and SSH
- I recommend HTTPS and will use HTTPS in this workshop
- Procedure
    1. Go to `https://github.com/settings/tokens` and click "Generate token"
    2. Decide the scope. Choose (at least) "repo", "user", and "workflow".
    3. Click "Generate token". This is the same as the password to interact with Git. You shouldn't show this to anyone.
    4. Copy the generated strings (PAT, personal access token)
    5. Open RStudio and run `gitcreds::gitcreds_set()` (Install the `gitcreds` package beforehand)
    6. Paste your PAT

Setup authentication

- When we interact with a remote Git server, such as GitHub, we need to provide credentials
- Github provides two authentication methods, HTTPS and SSH
- I recommend HTTPS and will use HTTPS in this workshop
- Procedure
  1. Go to `https://github.com/settings/tokens` and click "Generate token"
  2. Decide the scope. Choose (at least) "repo", "user", and "workflow".
  3. Click "Generate token". This is the same as the password to interact with Git. You shouldn't show this to anyone.
  4. Copy the generated strings (PAT, personal access token)
  5. Open RStudio and run `gitcreds::gitcreds_set()` (Install the `gitcreds` package beforehand)
  6. Paste your PAT
  7. RStudio stores and remembers PAT for you

How Git/Github handle collarborative projects

- Easy collaboration is a distinct feature of Git

How Git/Github handle collarborative projects

- Easy collaboration is a distinct feature of Git
- Key concepts: **branch** and **fork**

How Git/Github handle collarborative projects

- Easy collaboration is a distinct feature of Git
- Key concepts: **branch** and **fork**
- Branches are parallel universe of a repo

How Git/Github handle collarborative projects

- Easy collaboration is a distinct feature of Git
- Key concepts: **branch** and **fork**
- Branches are parallel universe of a repo
- Main branch is the default branch and should contain stable version

How Git/Github handle collarborative projects

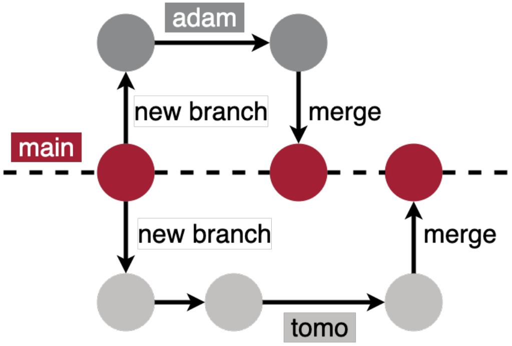
- Easy collaboration is a distinct feature of Git
- Key concepts: **branch** and **fork**
- Branches are parallel universe of a repo
- Main branch is the default branch and should contain stable version
- You contribute to a repo by creating a new branch in isolation from changes that other people are making to the repo

How Git/Github handle collarborative projects

- Easy collaboration is a distinct feature of Git
- Key concepts: **branch** and **fork**
- Branches are parallel universe of a repo
- Main branch is the default branch and should contain stable version
- You contribute to a repo by creating a new branch in isolation from changes that other people are making to the repo
- Whenever you work on a collaborative project, you should always create a new branch

How Git/Github handle collarborative projects

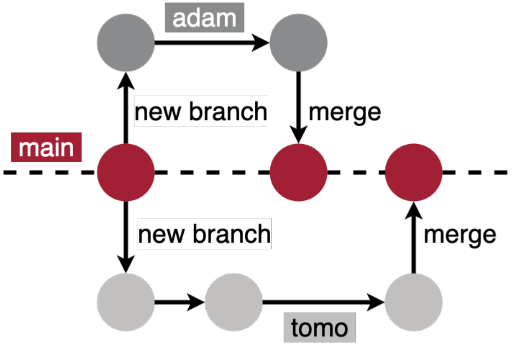
- Easy collaboration is a distinct feature of Git
- Key concepts: **branch** and **fork**
- Branches are parallel universe of a repo
- Main branch is the default branch and should contain stable version
- You contribute to a repo by creating a new branch in isolation from changes that other people are making to the repo
- Whenever you work on a collaborative project, you should always create a new branch



- Each circle represents “commit”

How Git/Github handle collarborative projects

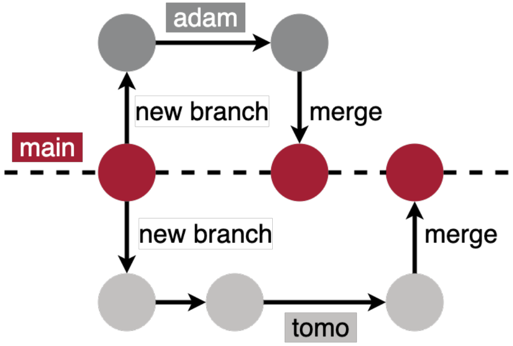
- Easy collaboration is a distinct feature of Git
- Key concepts: **branch** and **fork**
- Branches are parallel universe of a repo
- Main branch is the default branch and should contain stable version
- You contribute to a repo by creating a new branch in isolation from changes that other people are making to the repo
- Whenever you work on a collaborative project, you should always create a new branch



- Each circle represents “commit”
- In this example, Adam and Tomo work on the same project and each of them creates a branch

How Git/Github handle collarborative projects

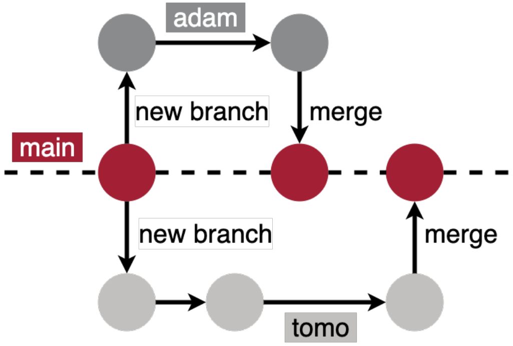
- Once your work is done, open **pull request**, and **merge** your branch into the default branch



- Each circle represents "commit"
- In this example, Adam and Tomo work on the same project and each of them creates a branch

How Git/Github handle collarborative projects

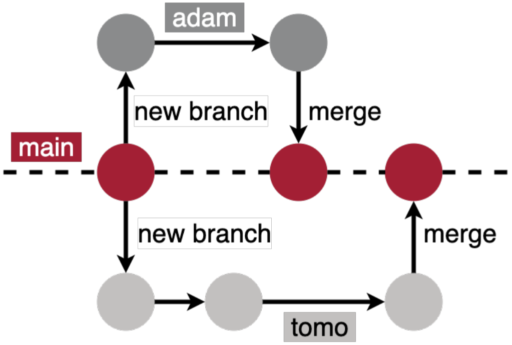
- Once your work is done, open **pull request**, and **merge** your branch into the default branch
- Pull request is an opportunity to discuss or check (**review**) if the changes you make in your branch won't break the default branch



- Each circle represents "commit"
- In this example, Adam and Tomo work on the same project and each of them creates a branch

How Git/Github handle collarborative projects

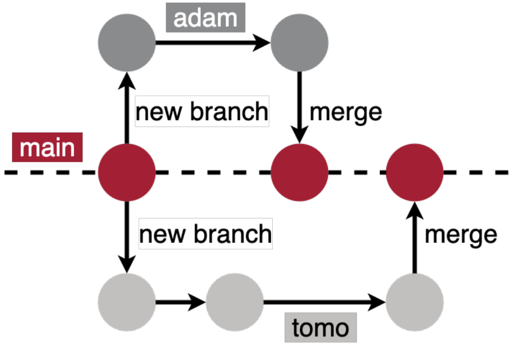
- Once your work is done, open **pull request**, and **merge** your branch into the default branch
- Pull request is an opportunity to discuss or check (**review**) if the changes you make in your branch won't break the default branch
- By merging your branch into the default branch, any edit you make in your branch will be reflected in the default branch



- Each circle represents "commit"
- In this example, Adam and Tomo work on the same project and each of them creates a branch

How Git/Github handle collarborative projects

- Once your work is done, open **pull request**, and **merge** your branch into the default branch
- Pull request is an opportunity to discuss or check (**review**) if the changes you make in your branch won't break the default branch
- By merging your branch into the default branch, any edit you make in your branch will be reflected in the default branch
- When both of them finish their work, they open a pull request and merge their branch into the main branch



- Each circle represents "commit"
- In this example, Adam and Tomo work on the same project and each of them creates a branch

General (suggestive) daily workflow with multiple branches

1. Create a new branch (usually from the default branch)

General (suggestive) daily workflow with multiple branches

1. Create a new branch (usually from the default branch)
2. In your branch...

General (suggestive) daily workflow with multiple branches

1. Create a new branch (usually from the default branch)
2. In your branch...
    1. Edit, add, or delete files in your repo

General (suggestive) daily workflow with multiple branches

1. Create a new branch (usually from the default branch)
2. In your branch...
    1. Edit, add, or delete files in your repo
    2. Stage files you want to add to commit and make a commit when you make some changes that represent a specific idea or solve particular issues

General (suggestive) daily workflow with multiple branches

1. Create a new branch (usually from the default branch)
2. In your branch...
    1. Edit, add, or delete files in your repo
    2. Stage files you want to add to commit and make a commit when you make some changes that represent a specific idea or solve particular issues
    3. Push when you make a set of commits or when you call it a day

General (suggestive) daily workflow with multiple branches

1. Create a new branch (usually from the default branch)
2. In your branch...
    1. Edit, add, or delete files in your repo
    2. Stage files you want to add to commit and make a commit when you make some changes that represent a specific idea or solve particular issues
    3. Push when you make a set of commits or when you call it a day
    4. Repeat the process above

General (suggestive) daily workflow with multiple branches

1. Create a new branch (usually from the default branch)
2. In your branch...
    1. Edit, add, or delete files in your repo
    2. Stage files you want to add to commit and make a commit when you make some changes that represent a specific idea or solve particular issues
    3. Push when you make a set of commits or when you call it a day
    4. Repeat the process above
3. Create a pull request when you finish your project

General (suggestive) daily workflow with multiple branches

1. Create a new branch (usually from the default branch)
2. In your branch...
    1. Edit, add, or delete files in your repo
    2. Stage files you want to add to commit and make a commit when you make some changes that represent a specific idea or solve particular issues
    3. Push when you make a set of commits or when you call it a day
    4. Repeat the process above
3. Create a pull request when you finish your project
4. Merge your branch into the default branch

General (suggestive) daily workflow with multiple branches

1. Create a new branch (usually from the default branch)
2. In your branch...
    1. Edit, add, or delete files in your repo
    2. Stage files you want to add to commit and make a commit when you make some changes that represent a specific idea or solve particular issues
    3. Push when you make a set of commits or when you call it a day
    4. Repeat the process above
3. Create a pull request when you finish your project
4. Merge your branch into the default branch

General (suggestive) daily workflow with multiple branches

1. Create a new branch (usually from the default branch)
2. In your branch...
    1. Edit, add, or delete files in your repo
    2. Stage files you want to add to commit and make a commit when you make some changes that represent a specific idea or solve particular issues
    3. Push when you make a set of commits or when you call it a day
    4. Repeat the process above
3. Create a pull request when you finish your project
4. Merge your branch into the default branch

- Let's try out with Github and RStudio

Working on someone else's repo

- Even if you have a public repo on Github, only those who have permission can directory push and merge to your repo

Working on someone else's repo

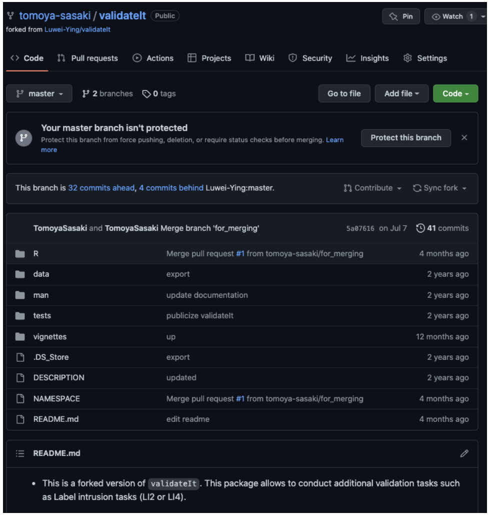
- Even if you have a public repo on Github, only those who have permission can directory push and merge to your repo
- What to do if you want to contribute to someone else's repo or borrow their idea and tailor for your own project?

Working on someone else's repo

- Even if you have a public repo on Github, only those who have permission can directory push and merge to your repo
- What to do if you want to contribute to someone else's repo or borrow their idea and tailor for your own project?
- **Fork** their project to your Github account

Working on someone else's repo

- Even if you have a public repo on Github, only those who have permission can directory push and merge to your repo
- What to do if you want to contribute to someone else's repo or borrow their idea and tailor for your own project?
- **Fork** their project to your Github account
  - Original: `https://github.com/xxx/reponame.git`



Your master branch isn't protected
Protect this branch from force pushing, deletion, or require status checks before merging. Learn more

This branch is 32 commits ahead, 4 commits behind Luwei-Ying:master.

TomoyaSasaki and TomoyaSasaki Merge branch 'for_merging' 5a07616 on Jul 7 41 commits

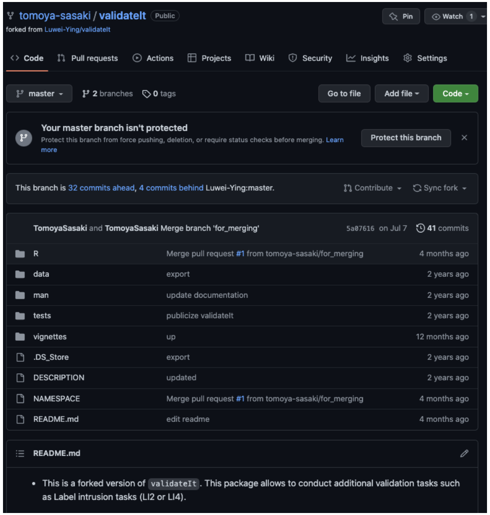
R	Merge pull request #1 from tomoya-sasaki/for_merging	4 months ago
data	export	2 years ago
man	update documentation	2 years ago
tests	publicize validateIt	2 years ago
vignettes	up	12 months ago
.DS_Store	export	2 years ago
DESCRIPTION	updated	2 years ago
NAMESPACE	Merge pull request #1 from tomoya-sasaki/for_merging	4 months ago
README.md	edit readme	4 months ago

README.md

- This is a forked version of `validateIt`. This package allows to conduct additional validation tasks such as Label intrusion tasks (LI2 or LI4).

Working on someone else's repo

- Even if you have a public repo on Github, only those who have permission can directory push and merge to your repo
- What to do if you want to contribute to someone else's repo or borrow their idea and tailor for your own project?
- **Fork** their project to your Github account
    - Original: `https://github.com/xxx/reponame.git`
    - Forked repo: `https://github.com/yourusername/reponame.git`



Your master branch isn't protected
Protect this branch from force pushing, deletion, or require status checks before merging. Learn more

This branch is 32 commits ahead, 4 commits behind Luwei-Ying:master.

TomoyaSasaki and TomoyaSasaki Merge branch 'for_merging' 5a07616 on Jul 7 41 commits

R	Merge pull request #1 from tomoya-sasaki/for_merging	4 months ago
data	export	2 years ago
man	update documentation	2 years ago
tests	publicize validateIt	2 years ago
vignettes	up	12 months ago
.DS_Store	export	2 years ago
DESCRIPTION	updated	2 years ago
NAMESPACE	Merge pull request #1 from tomoya-sasaki/for_merging	4 months ago
README.md	edit readme	4 months ago

README.md

- This is a forked version of `validateIt`. This package allows to conduct additional validation tasks such as Label intrusion tasks (LI2 or LI4).

Comparison and relationship to alternatives (Dropbox, Google Drive)

- Unlike Dropbox or Google Drive, Git/Github does not automatically track changes

Comparison and relationship to alternatives (Dropbox, Google Drive)

- Unlike Dropbox or Google Drive, Git/Github does not automatically track changes

  $\rightsquigarrow$ you need to take (small) action (mostly) when you edit

Comparison and relationship to alternatives (Dropbox, Google Drive)

- Unlike Dropbox or Google Drive, Git/Github does not automatically track changes

  $\rightsquigarrow$ you need to take (small) action (mostly) when you edit
- However, you can make your project folder clean with commits and branches

Comparison and relationship to alternatives (Dropbox, Google Drive)

- Unlike Dropbox or Google Drive, Git/Github does not automatically track changes

  $\rightsquigarrow$ you need to take (small) action (mostly) when you edit
- However, you can make your project folder clean with commits and branches
- You can revert your changes easily and flexibly if you make frequent commits

Comparison and relationship to alternatives (Dropbox, Google Drive)

- Unlike Dropbox or Google Drive, Git/Github does not automatically track changes

  $\rightsquigarrow$ you need to take (small) action (mostly) when you edit
- However, you can make your project folder clean with commits and branches
- You can revert your changes easily and flexibly if you make frequent commits

  $\rightsquigarrow$ Dropbox and Google Drive only preserve once-in-a-day snapshots

Comparison and relationship to alternatives (Dropbox, Google Drive)

- Unlike Dropbox or Google Drive, Git/Github does not automatically track changes

  $\rightsquigarrow$ you need to take (small) action (mostly) when you edit
- However, you can make your project folder clean with commits and branches
- You can revert your changes easily and flexibly if you make frequent commits

  $\rightsquigarrow$ Dropbox and Google Drive only preserve once-in-a-day snapshots

- Putting Git local repos in Dropbox or Google Drive can be very tricky when your Dropbox/Google Drive account is synced to multiple computers

Comparison and relationship to alternatives (Dropbox, Google Drive)

- Unlike Dropbox or Google Drive, Git/Github does not automatically track changes

  $\rightsquigarrow$ you need to take (small) action (mostly) when you edit

- However, you can make your project folder clean with commits and branches
- You can revert your changes easily and flexibly if you make frequent commits

  $\rightsquigarrow$ Dropbox and Google Drive only preserve once-in-a-day snapshots

- Putting Git local repos in Dropbox or Google Drive can be very tricky when your Dropbox/Google Drive account is synced to multiple computers
- Git does not like autosync (autoupdate) by Dropbox/Google Drive

Comparison and relationship to alternatives (Dropbox, Google Drive)

- Unlike Dropbox or Google Drive, Git/Github does not automatically track changes

  $\rightsquigarrow$ you need to take (small) action (mostly) when you edit
- However, you can make your project folder clean with commits and branches
- You can revert your changes easily and flexibly if you make frequent commits

  $\rightsquigarrow$ Dropbox and Google Drive only preserve once-in-a-day snapshots

- Putting Git local repos in Dropbox or Google Drive can be very tricky when your Dropbox/Google Drive account is synced to multiple computers
- Git does not like autosync (autoupdate) by Dropbox/Google Drive
- But you probably want to rely on these other services to store huge data that Git does not track

Comparison and relationship to alternatives (Dropbox, Google Drive)

- Unlike Dropbox or Google Drive, Git/Github does not automatically track changes

  $\leadsto$ you need to take (small) action (mostly) when you edit
- However, you can make your project folder clean with commits and branches
- You can revert your changes easily and flexibly if you make frequent commits

  $\leadsto$ Dropbox and Google Drive only preserve once-in-a-day snapshots

- Putting Git local repos in Dropbox or Google Drive can be very tricky when your Dropbox/Google Drive account is synced to multiple computers
- Git does not like autosync (autoupdate) by Dropbox/Google Drive
- But you probably want to rely on these other services to store huge data that Git does not track
- In my setup, I put Git repos in Dropbox but sync them only with my main computer and the same Git repos in my other computers are located in non-Dropbox folders

What’s next?

- Working with private repos

What's next?

- Working with private repos
- (Gradually) Learning how to use git in command lines

What’s next?

- Working with private repos
- (Gradually) Learning how to use git in command lines
- Probably you mess up once in a while. I still make mistakes sometimes

What's next?

- Working with private repos
- (Gradually) Learning how to use git in command lines
- Probably you mess up once in a while. I still make mistakes sometimes
- You can fix them easily by reverting

Useful resources

- Happy Git and GitHub for the useR: https://happygitwithr.com/index.html
- Git Guide: https://github.com/git-guides
- git-vs-dropbox: https://michaelstepner.com/blog/git-vs-dropbox/