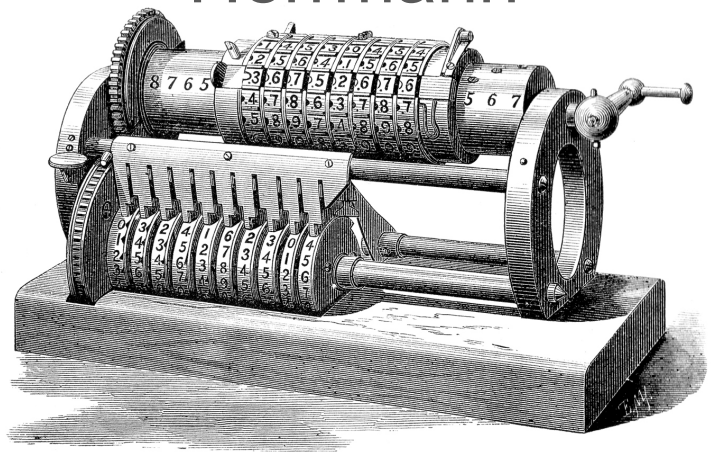
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Programming in R

Binder
Herrmann



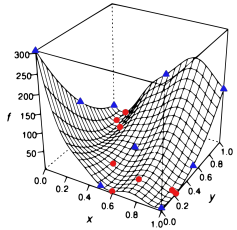
2020-04-21

Street Cred: Martin Binder

- mb706 on GitHub

- Three packets on CRAN, combined ≈ 3k downloads / month
  - mosmafs
  - mlrCPO
  - mlr3pipelines

- 2nd year PhD, Compstat Group (Bischl)

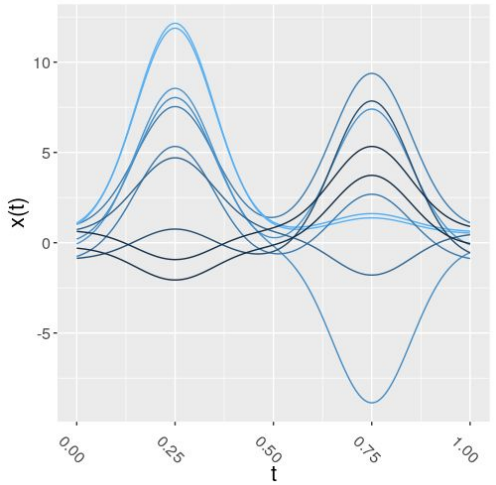


- Manages Mattermost Chat Server

- If you like what we do here and you impress me and you can get a "Hiwi" position

Street Cred: Moritz Hermann

- 2nd year PhD
- Functional data analysis Group (Scheipl)



- HerrMo auf GitHub
- moritz.herrmann@stat.uni-muenchen.de

- Working on bringing some didactics into teaching and the reaccreditation

Why are we here?

Programming in R

Shouldn't need explanation

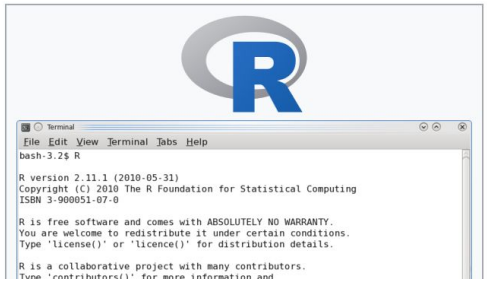
R (programming language)

From Wikipedia, the free encyclopedia

R is a programming language and free software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing.[6] The R language is widely used among statisticians and data miners for developing statistical software[7] and data analysis.[8] Polls, data mining surveys, and studies of scholarly literature databases show substantial increases in popularity;[9] as of February 2020, R ranks 13th in the TIOBE index, a measure of popularity of programming languages.[10]

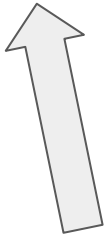
A GNU package,[11] the official R software environment is written primarily in C, Fortran, and R itself[12] (thus, it is partially self-hosting) and is freely available under the GNU General Public License. Pre-compiled executables are provided for various operating

R



... etc.

Programming in R



We will spend some time on this!

Programming in R: Programming is Useful

- Programming is about letting something else do the work for you
- Most of (scientific / industry) work will have *some* degree of programming
- Your course work *will* get easier if you know programming

Programming in R: Programming is Valuable

Amazon Data Scientist Salary

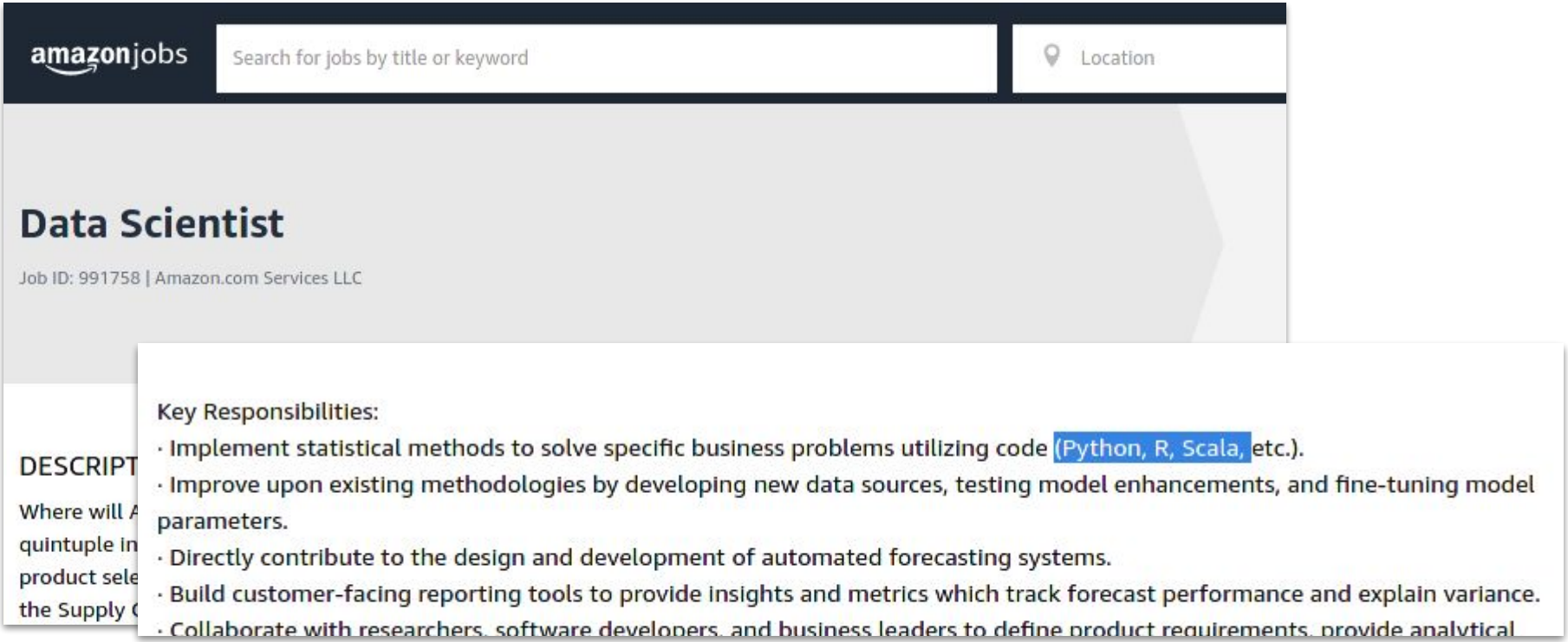
The average Amazon Data Scientist earns $164,114 annually, which includes a base salary of $125,296 with a $38,818 bonus. This total compensation is $35,131 more than the US average for a Data Scientist. Data Scientist salaries at Amazon can range from $110,000 - $217,000 with equity ranging from 0-150K+.

The Engineering Department at Amazon earns $14,890 more on average than the HR Department. Comparably data has a total of 21 salary records from Amazon Data Scientists.

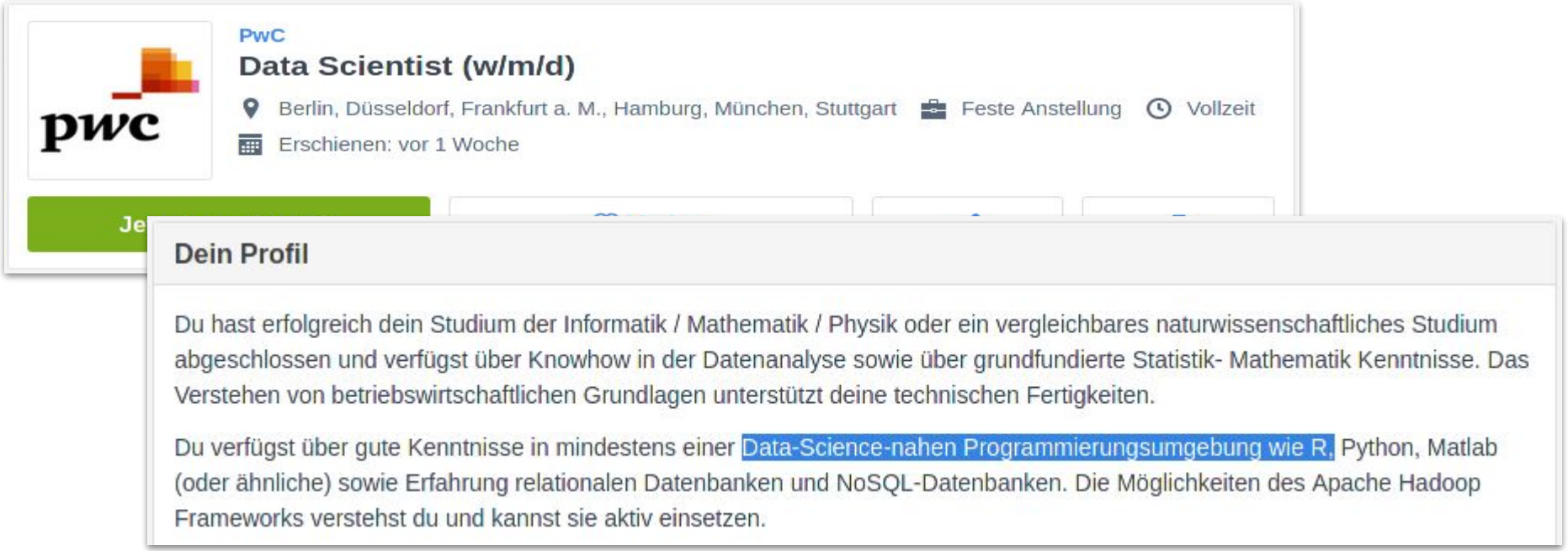
Last updated 3 days ago.

(To be fair, it is harder to earn this kind of money in Germany)

Programming in R: Programming is Valuable

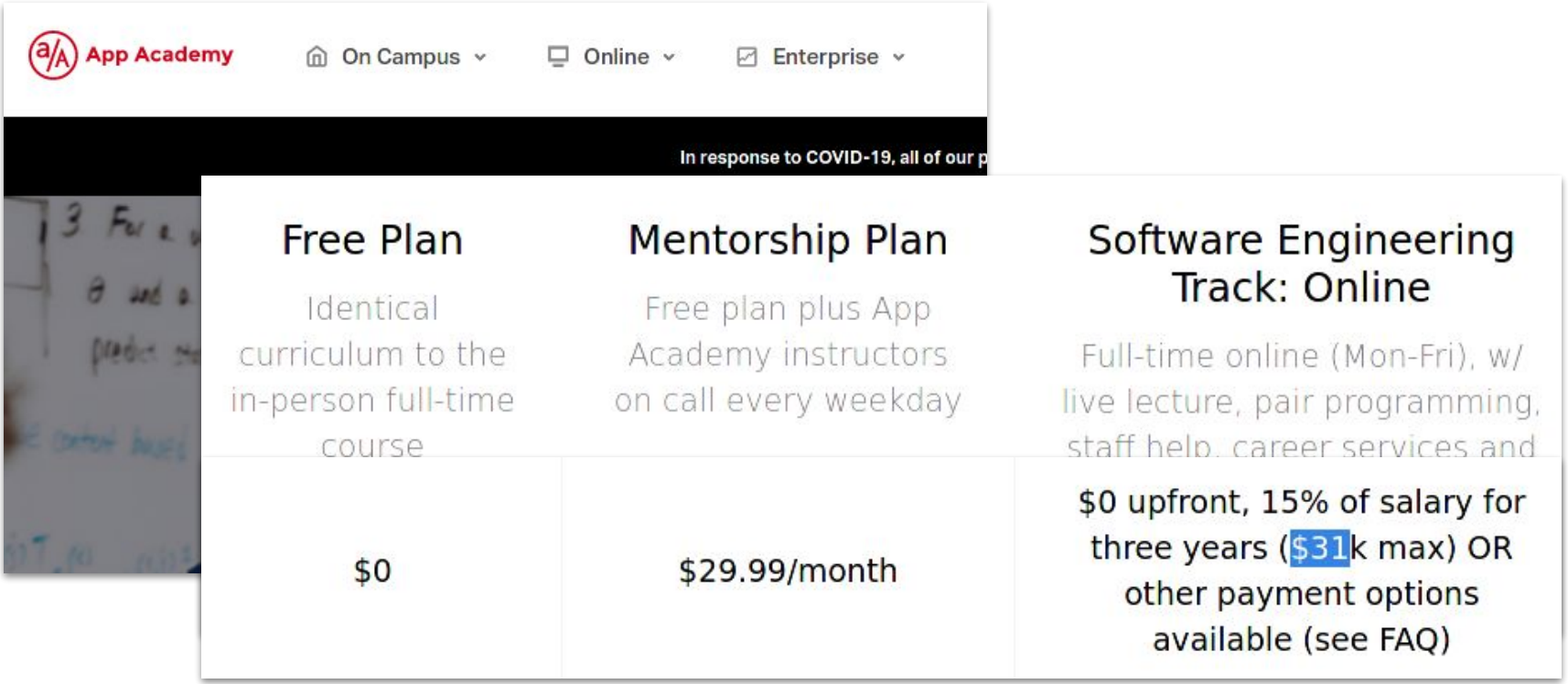


Programming in R: Programming is Valuable

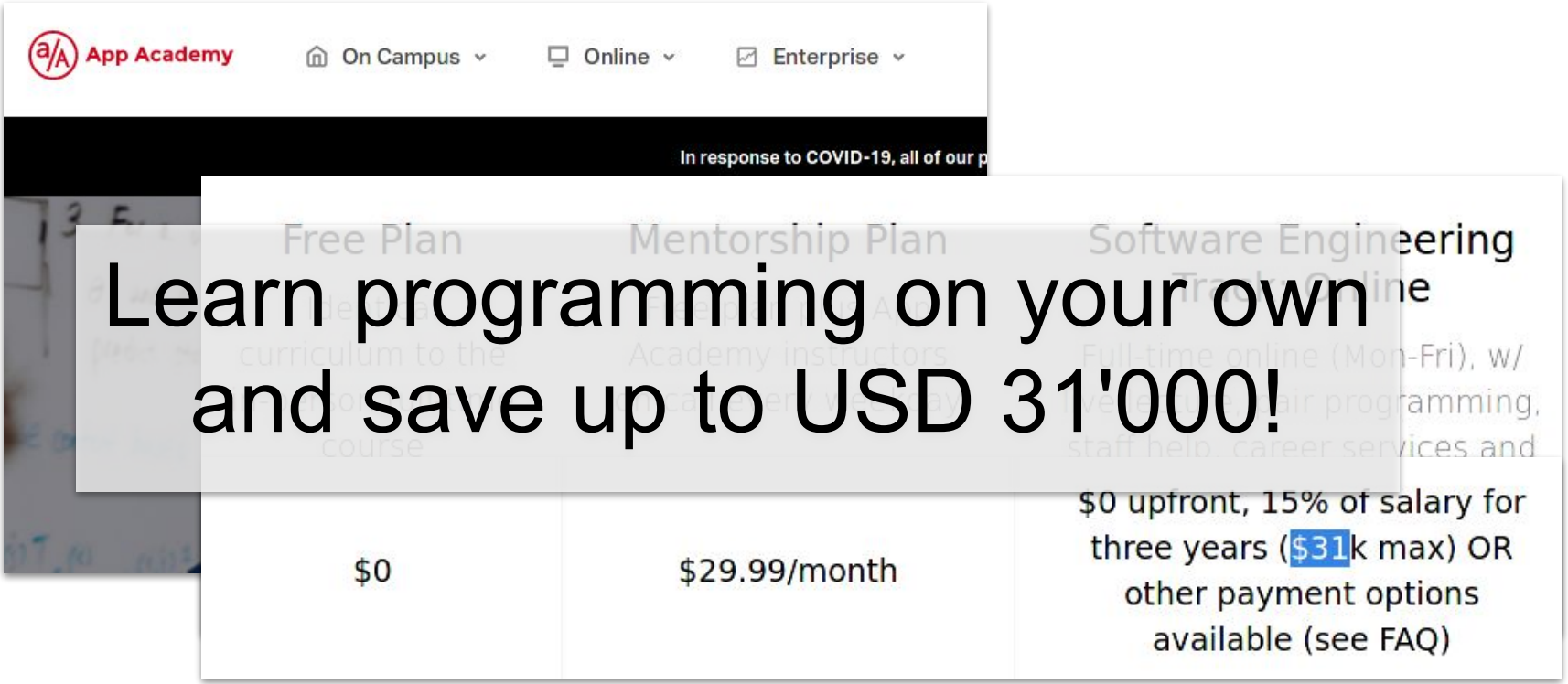


Programming in R: Programming Courses are Valuable

Programming in R: Programming Courses are Valuable



Programming in R: Programming Courses are Valuable



Learn programming on your own and save up to USD 31'000!

Programming in R: Programming Courses are Valuable

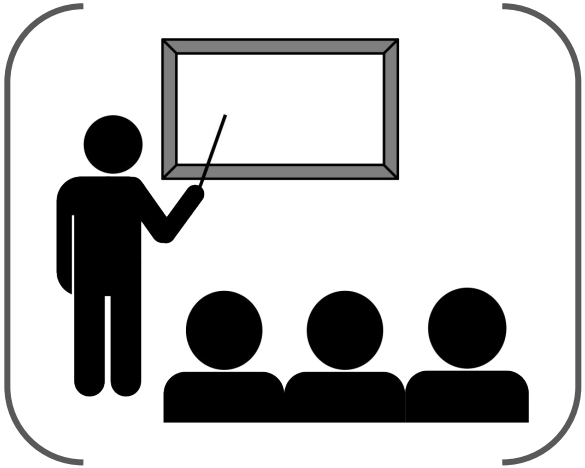
Of course that is not a fair comparison, but....

- There are lots of free courses online, we try to give you more than that
- Make use of this opportunity now or regret it later
- You should not be coming here for the credit, you are coming here to learn programming

Structure of this Course

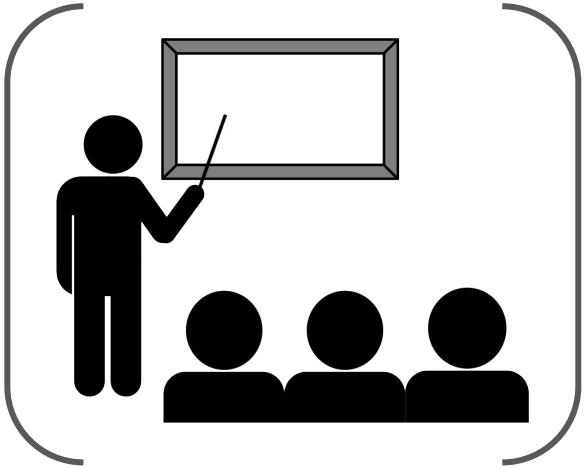
Structure of this Course

- This is an online course by necessity.
- ➔ "Inverted Classroom"



Structure of this Course

- This is an online course by necessity.
- ➔ "Inverted Classroom"

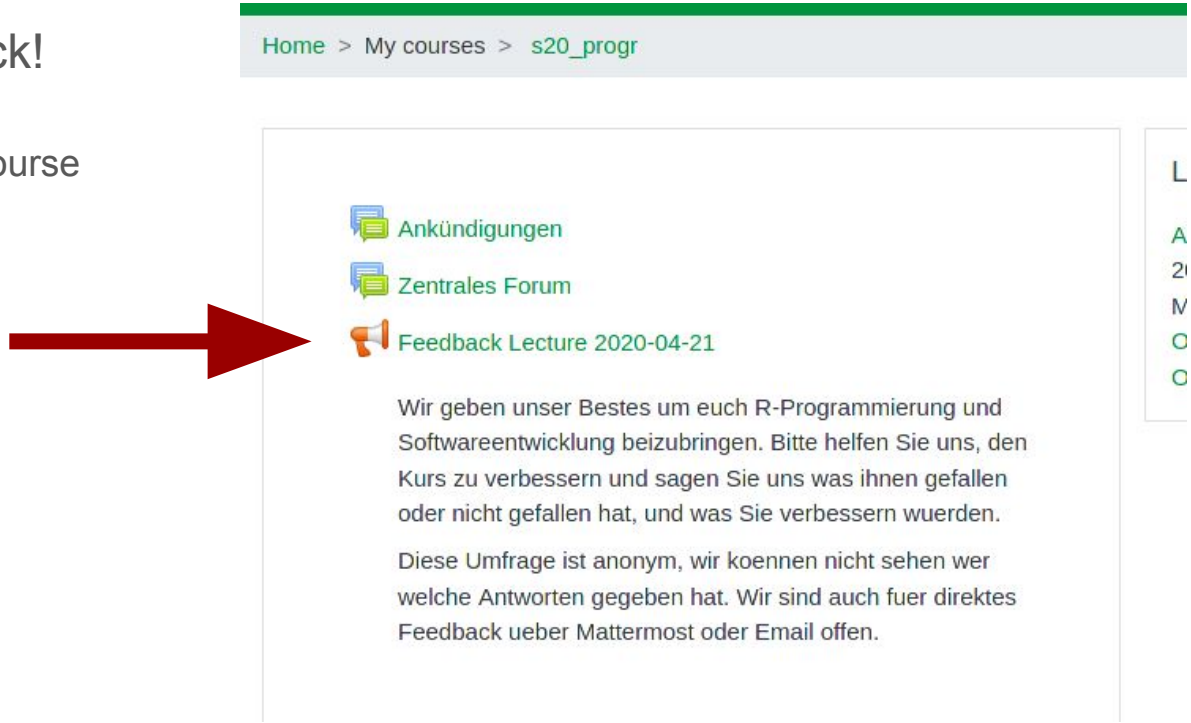


-1

- We stay in contact with you!
  - Moodle Forum
  - Mattermost Chat
- Please fill in the weekly anonymous feedback form!

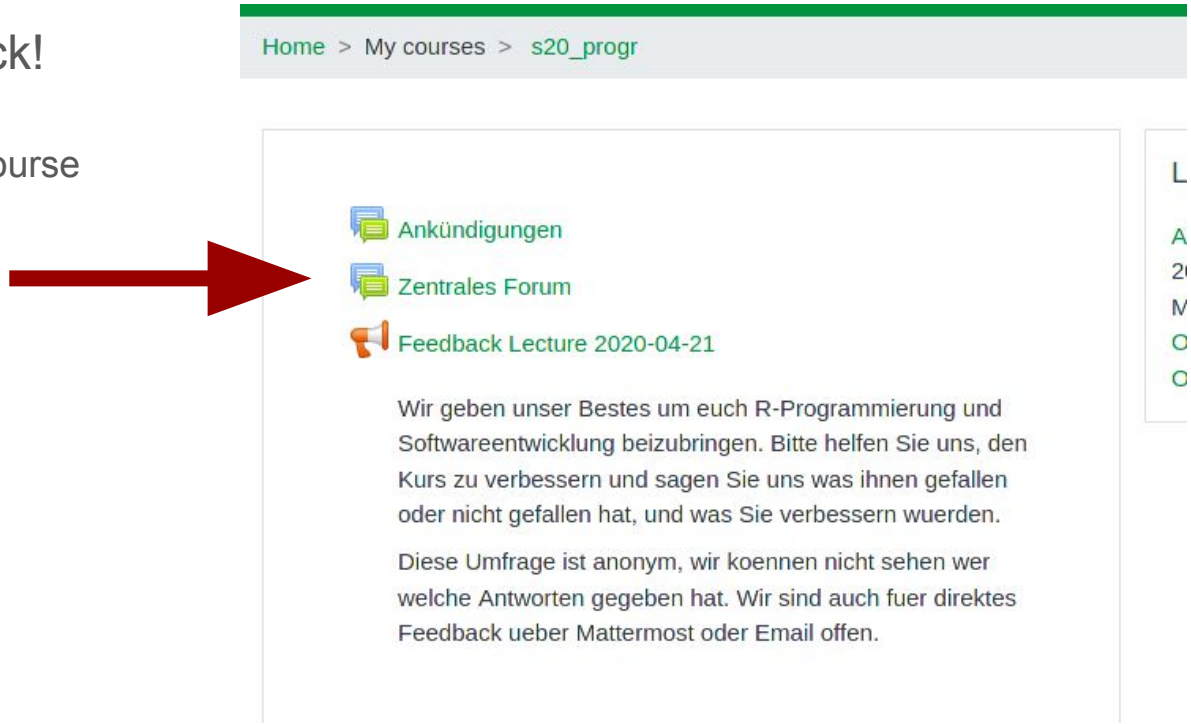
Structure of this Course

- Please give us feedback!
  - Anonymous
  - Helps us improve the course



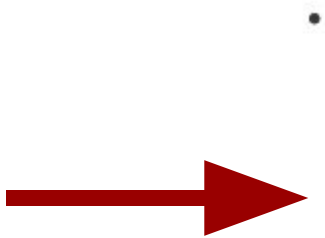
Structure of this Course

- Please give us feedback!
  - Anonymous
  - Helps us improve the course
- Please talk to us!
  - Forum

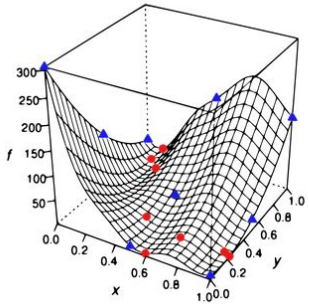


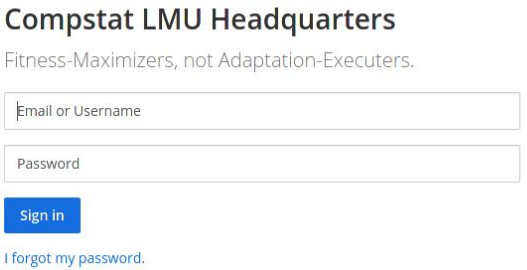
Structure of this Course

- Please give us feedback!
  - Anonymous
  - Helps us improve the course
- Please talk to us!
  - Forum
  - Mattermost



- **Mattermost**: Wir haben einer
  Kommunikation der Teilnehm
  untereinander ermoeglicht. Bi
  Sollten Sie noch keinen Zugri
  diesem Invite-Link. Sie muessen



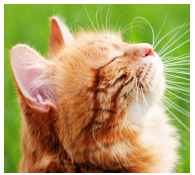


Structure of this Course

- Experience shows that students
  - do not read book chapters and
  - do not watch lecture videos
- But apparently they read slides...
- And they do homework if we incentivise it

Structure of this Course

- We publish a **Homework Task** on **Friday before a lecture timeslot**
- You may try to solve the Task
- You meet us on **Tuesday during the lecture timeslot**
  - Ask questions
  - Work in teams to solve the Task and exchange ideas
  - Come even if you don't understand anything, we will try to help
  - Come even if you understand everything, teaching others is one of the best ways to learn more yourself ;-)
- Hand in your solution no later than **end of Thursday after the lecture timeslot**
- Automatic checks pass --> Extra exam points



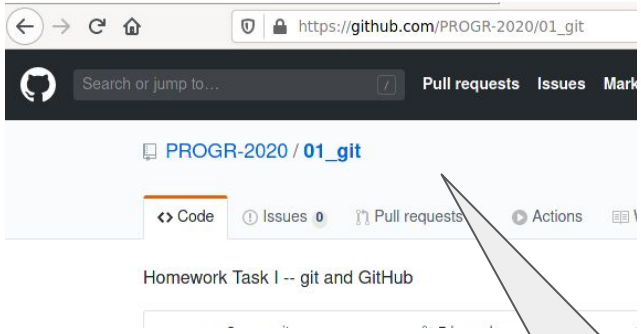
Lecture	Date
1	2020-04-21
2	2020-04-28
3	2020-05-05
4	2020-05-12
5	2020-05-19
6	2020-05-26
7	2020-06-09
8	2020-06-16
9	2020-06-23
10	2020-06-30
11	2020-07-07
12	2020-07-15
13	2020-07-21

Structure of this Course

"Automatic checks pass --> Extra exam points"

- This is even easier than it sounds, because you get immediate feedback. You will always know if your solution is correct.
- Homework tasks mimic exam conditions. The exam will essentially be a large homework task.
- Team work: Honour system
  - We *encourage* you to work together to make sure everyone understands what is going on. The Tuesday lecture slots are exactly for that.
  - We *discourage* you from blindly copying each other's work. You *will* fail the exam if you can not solve the homework tasks.

Anatomy of a Homework Task



Github

Link to the Slides

Further Material

Task Description

README.md	define the tasks	19 minutes ago
sier.png	define the tasks	19 minutes ago
sierpinski.Rmd	framework setup	3 hours ago

README.md

Programming in R

Homework Task I -- git and GitHub

This is your first homework task. Since it will not be graded there is no deadline. Nevertheless, we strongly encourage you to try out this task, both to get to know the homework task format and to get to know git. All the following tasks will assume that you know how to use `git` and *GitHub*.

Information Material

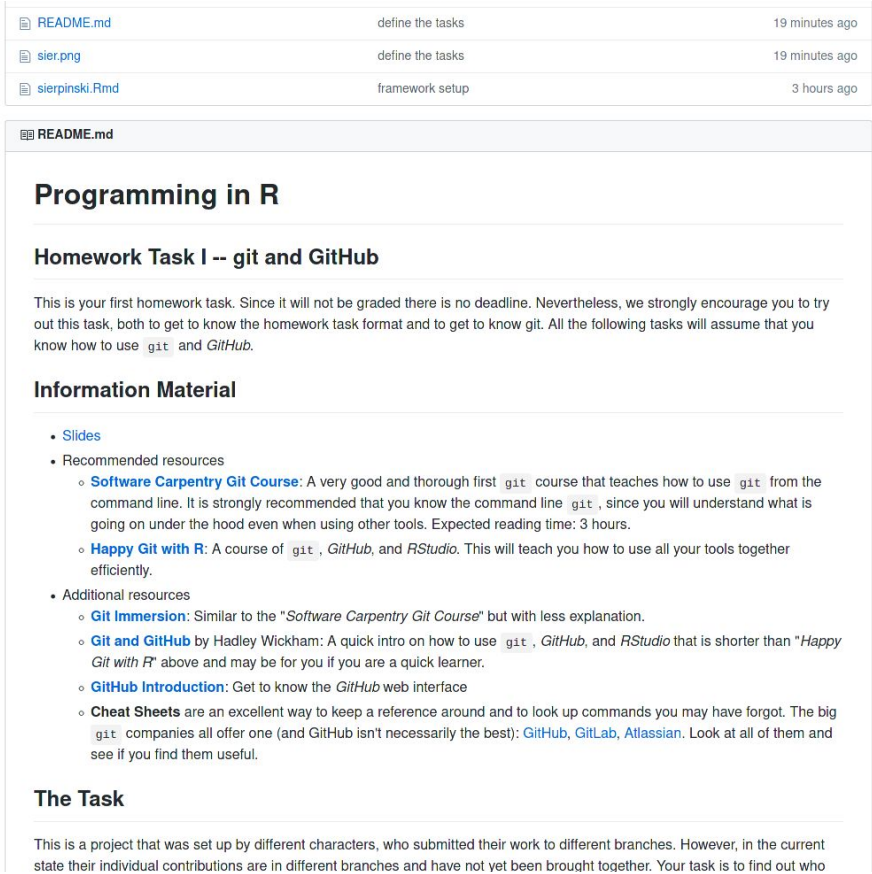
- Slides
- Recommended resources
  - **Software Carpentry Git Course**: A very good and thorough first `git` course that teaches how to use `git` from the command line. It is strongly recommended that you know the command line `git`, since you will understand what is going on under the hood even when using other tools. Expected reading time: 3 hours.
  - **Happy Git with R**: A course of `git`, *GitHub*, and *RStudio*. This will teach you how to use all your tools together efficiently.
- Additional resources
  - **Git Immersion**: Similar to the "*Software Carpentry Git Course*" but with less explanation.
  - **Git and GitHub** by Hadley Wickham: A quick intro on how to use `git`, *GitHub*, and *RStudio* that is shorter than "*Happy Git with R*" above and may be for you if you are a quick learner.
  - **GitHub Introduction**: Get to know the *GitHub* web interface
  - **Cheat Sheets** are an excellent way to keep a reference around and to look up commands you may have forgot. The big `git` companies all offer one (and GitHub isn't necessarily the best): GitHub, GitLab, Atlassian. Look at all of them and see if you find them useful.

The Task

This is a project that was set up by different characters, who submitted their work to different branches. However, in the current state their individual contributions are in different branches and have not yet been brought together. Your task is to find out who

Anatomy of a Homework Task

- Slides for you to read
- Lots of links to read further
- Task description of what you are supposed to do
- Code repository that you build up on
- Correctness is checked & graded automatically
  (not for first Task, sorry...)



README.md	define the tasks	19 minutes ago
sier.png	define the tasks	19 minutes ago
sierpinski.Rmd	framework setup	3 hours ago

README.md

Programming in R

Homework Task I -- git and GitHub

This is your first homework task. Since it will not be graded there is no deadline. Nevertheless, we strongly encourage you to try out this task, both to get to know the homework task format and to get to know git. All the following tasks will assume that you know how to use `git` and *GitHub*.

Information Material

- Slides
- Recommended resources
  - **Software Carpentry Git Course**: A very good and thorough first `git` course that teaches how to use `git` from the command line. It is strongly recommended that you know the command line `git`, since you will understand what is going on under the hood even when using other tools. Expected reading time: 3 hours.
  - **Happy Git with R**: A course of `git`, *GitHub*, and *RStudio*. This will teach you how to use all your tools together efficiently.
- Additional resources
  - **Git Immersion**: Similar to the "*Software Carpentry Git Course*" but with less explanation.
  - **Git and GitHub** by Hadley Wickham: A quick intro on how to use `git`, *GitHub*, and *RStudio* that is shorter than "*Happy Git with R*" above and may be for you if you are a quick learner.
  - **GitHub Introduction**: Get to know the *GitHub* web interface
  - **Cheat Sheets** are an excellent way to keep a reference around and to look up commands you may have forgot. The big `git` companies all offer one (and GitHub isn't necessarily the best): GitHub, GitLab, Atlassian. Look at all of them and see if you find them useful.

The Task

This is a project that was set up by different characters, who submitted their work to different branches. However, in the current state their individual contributions are in different branches and have not yet been brought together. Your task is to find out who

DataCamp

- We have a DataCamp Classroom over the time of this semester

  → you have full access to all DataCamp courses

  → Invite-Link on moodle

- You are not obliged to do anything on DataCamp to pass the course

  → it is voluntary

  → intended as an additional benefit of the course

We don’t have unlimited accounts though, so please only subscribe if you really intend to use it - if you are not sure, use your free test account first!

Scope of this Course

Scope of this Course

- This is a **programming** course
  - Focus on **creating programs**
  - Not on data science / statistics (you won't learn about all the `mean()`s and `rnorm()`s here)
  - Not a survey of packages for specific tasks (you already had tidy + ggplot last year)
- Three "tracks"
  - Learn things about the R language: "R"
  - Get to know nice tools to use: "Tools"
  - Learn things about software development in general: "Dev"

Lecture	Date	Topic
1	2020-04-21	Introduction
2	2020-04-28	Structured Programming in R
3	2020-05-05	
4	2020-05-12	Tabular Data
5	2020-05-19	Debugging
6	2020-05-26	Object Oriented Programming
7	2020-06-09	
8	2020-06-16	Parallelization
9	2020-06-23	Reproducibility
10	2020-06-30	Software Engineering
11	2020-07-07	
12	2020-07-15	
13	2020-07-21	Omnibus

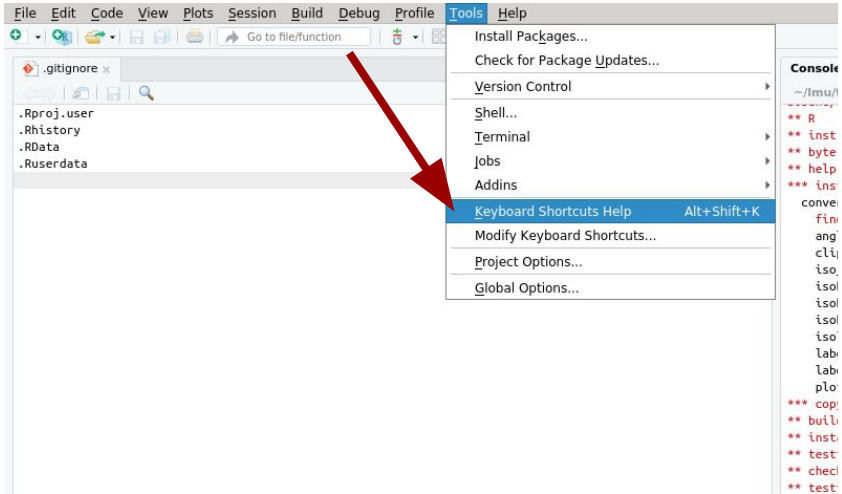
Your Setup

R / RStudio

- You should have R and RStudio installed
- If not already, you will necessarily get more familiar with RStudio as the course progresses
- A few things to keep in mind...

R / RStudio

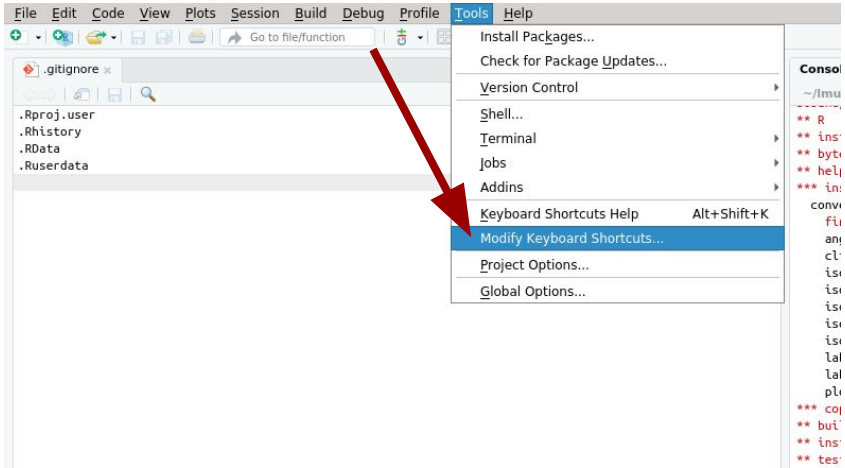
Learn the hotkeys!



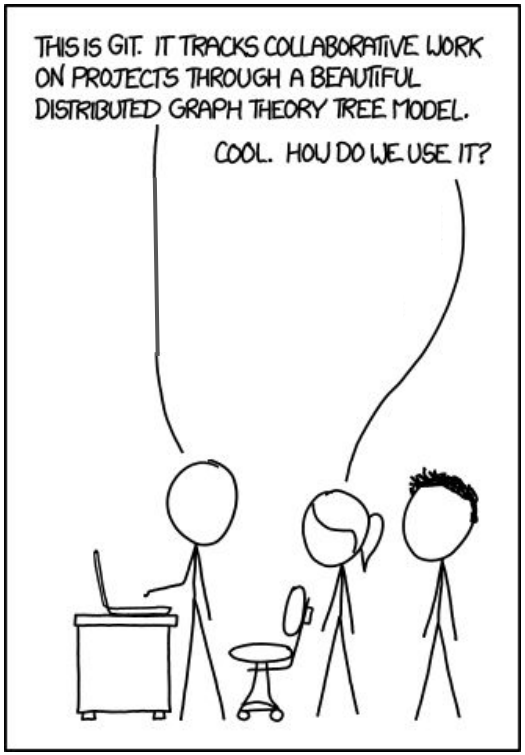
R / RStudio

You can adapt the hotkeys

(but this may make things hard when you switch work places)



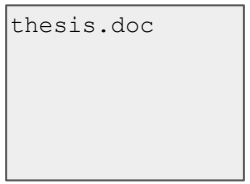
Git



https://xkcd.com/1597/

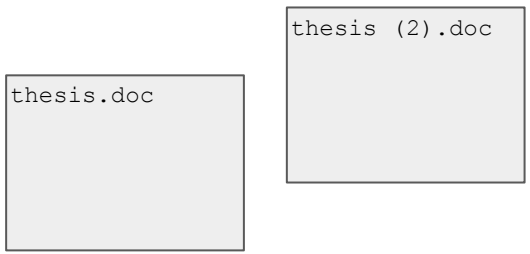
Git -- "Version Control System" (VCS)

You probably know this:



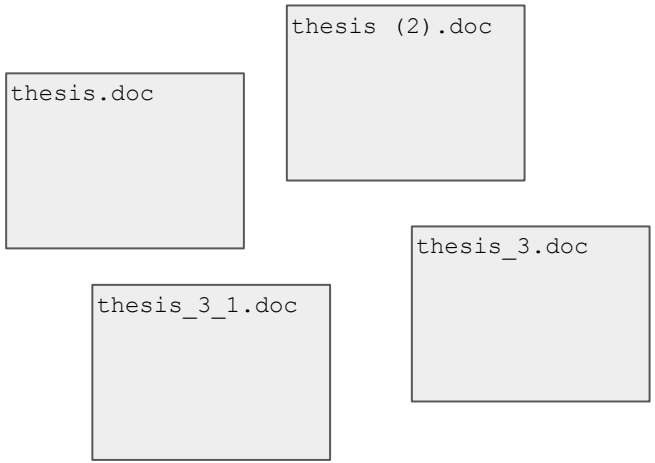
Git -- "Version Control System" (VCS)

You probably know this:



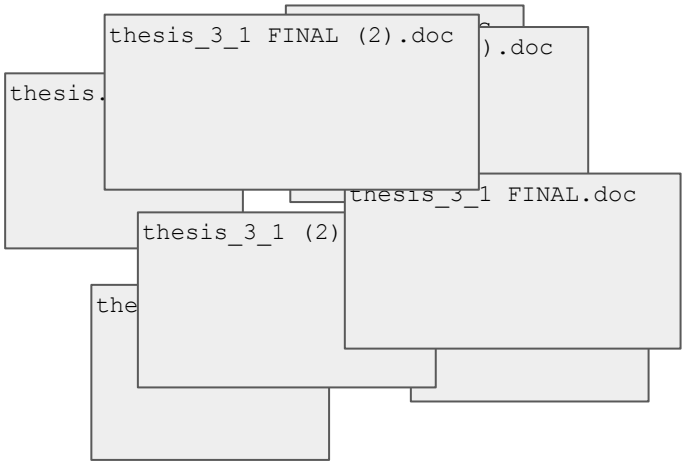
Git -- "Version Control System" (VCS)

You probably know this:



Git -- "Version Control System" (VCS)

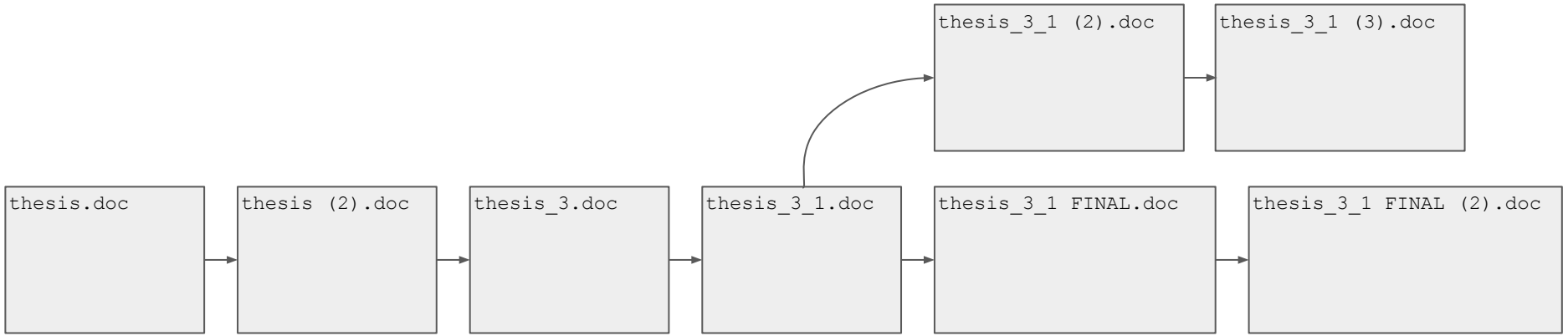
You probably know this:



Git -- "Version Control System" (VCS)

How about we organize things:

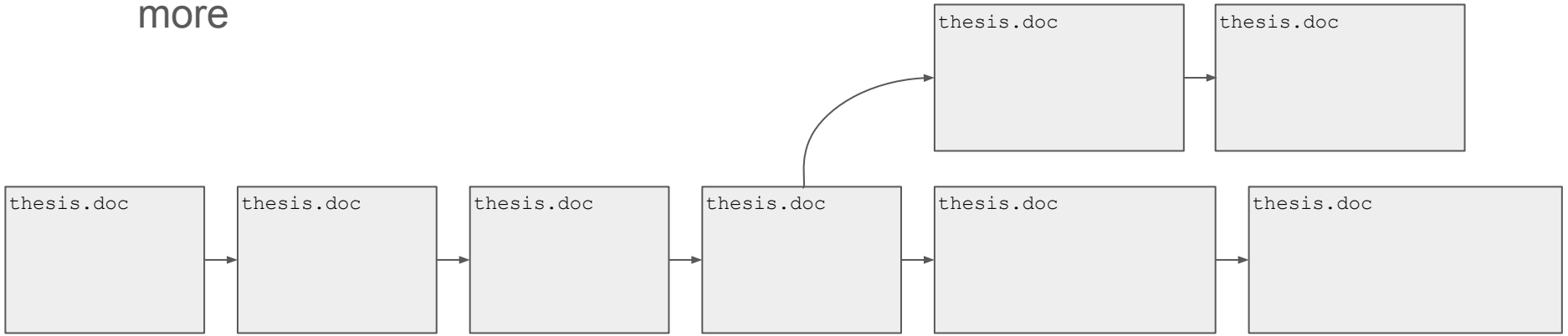
- Tree-like structure



Git -- "Version Control System" (VCS)

How about we organize things:

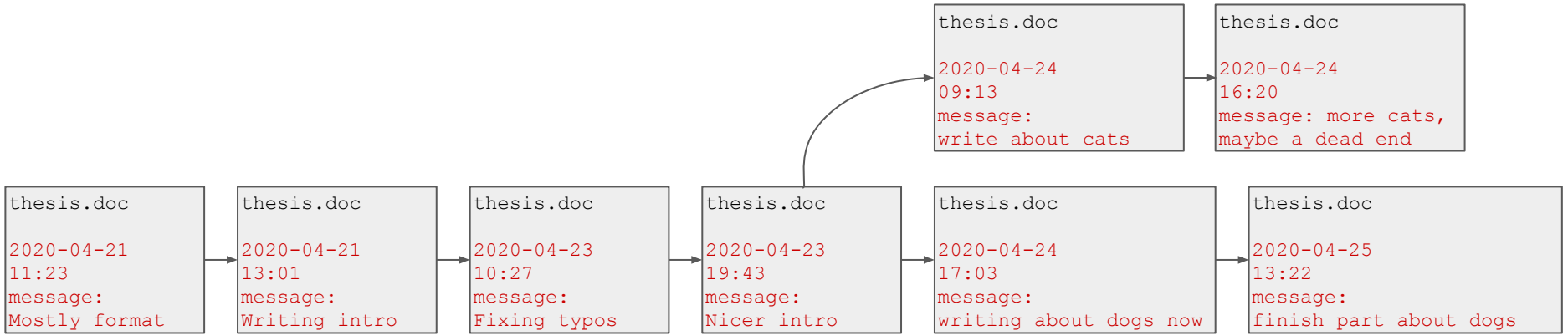
- We don't need different file names any more



Git -- "Version Control System" (VCS)

How about we organize things:

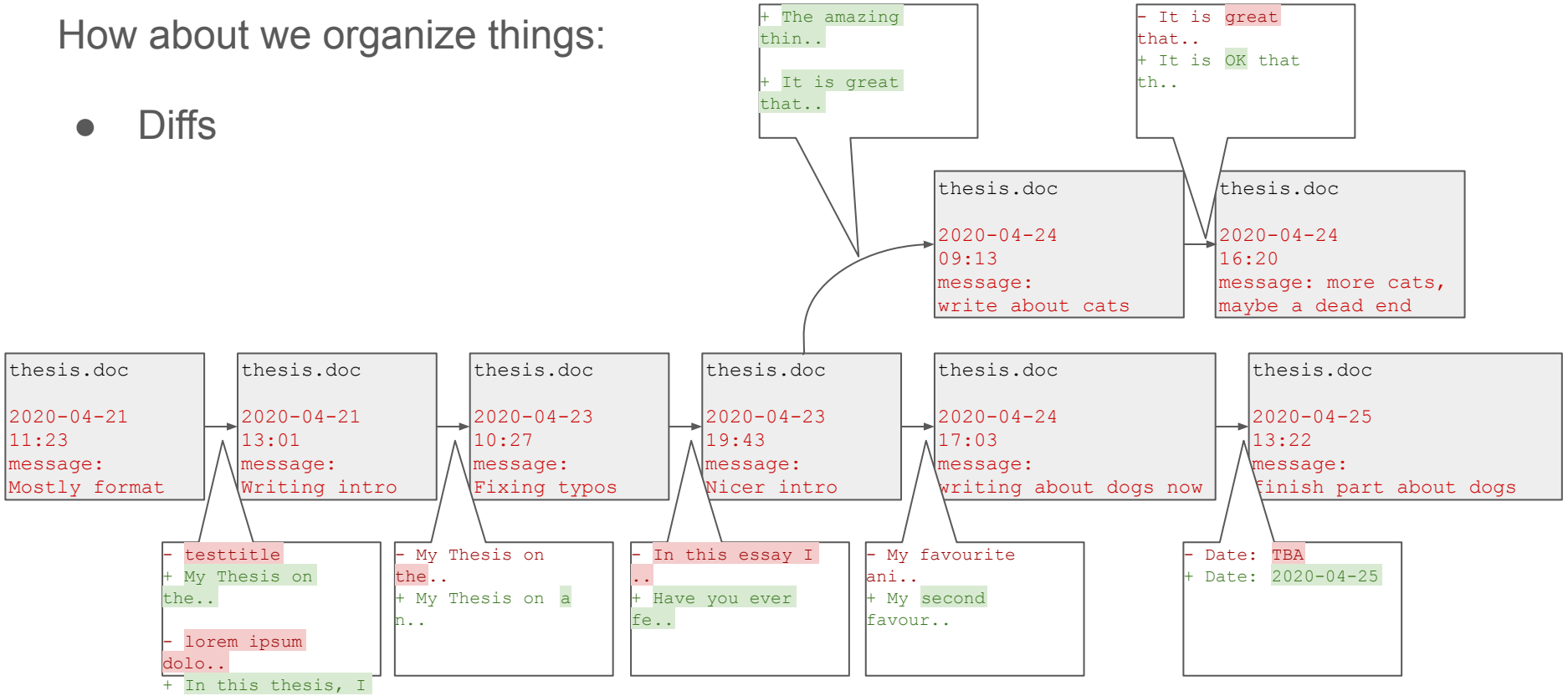
- Metadata



Git -- "Version Control System" (VCS)

How about we organize things:

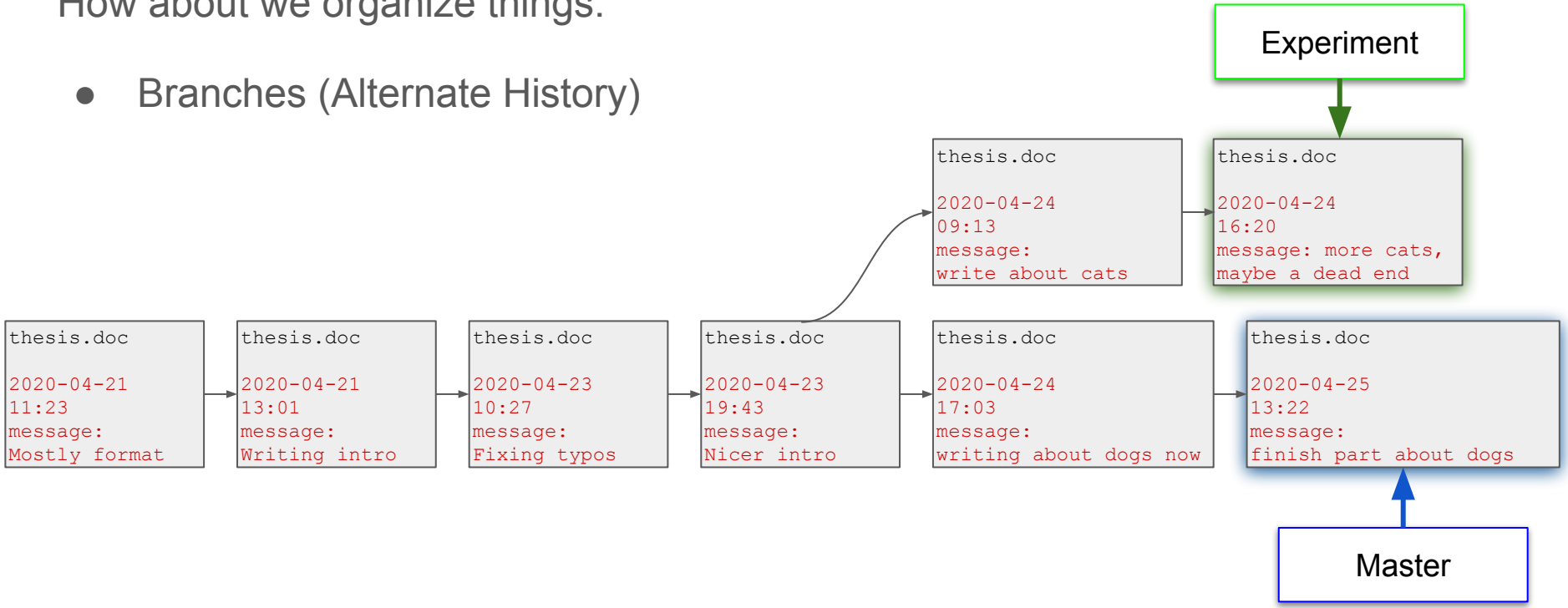
- Diffs



Git -- "Version Control System" (VCS)

How about we organize things:

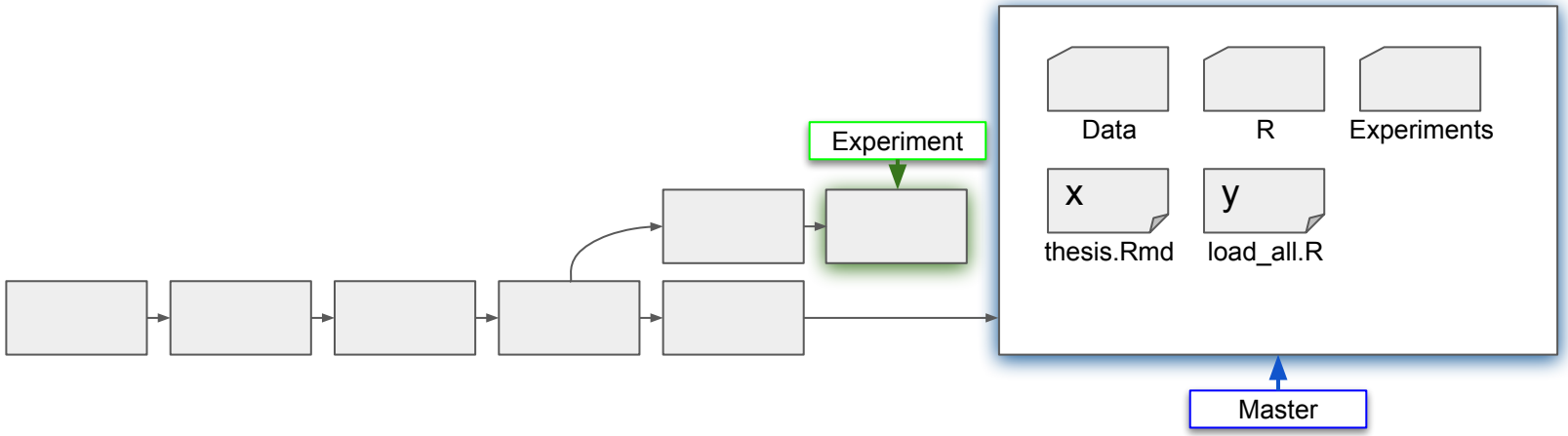
- Branches (Alternate History)



Git -- "Version Control System" (VCS)

How about we organize things:

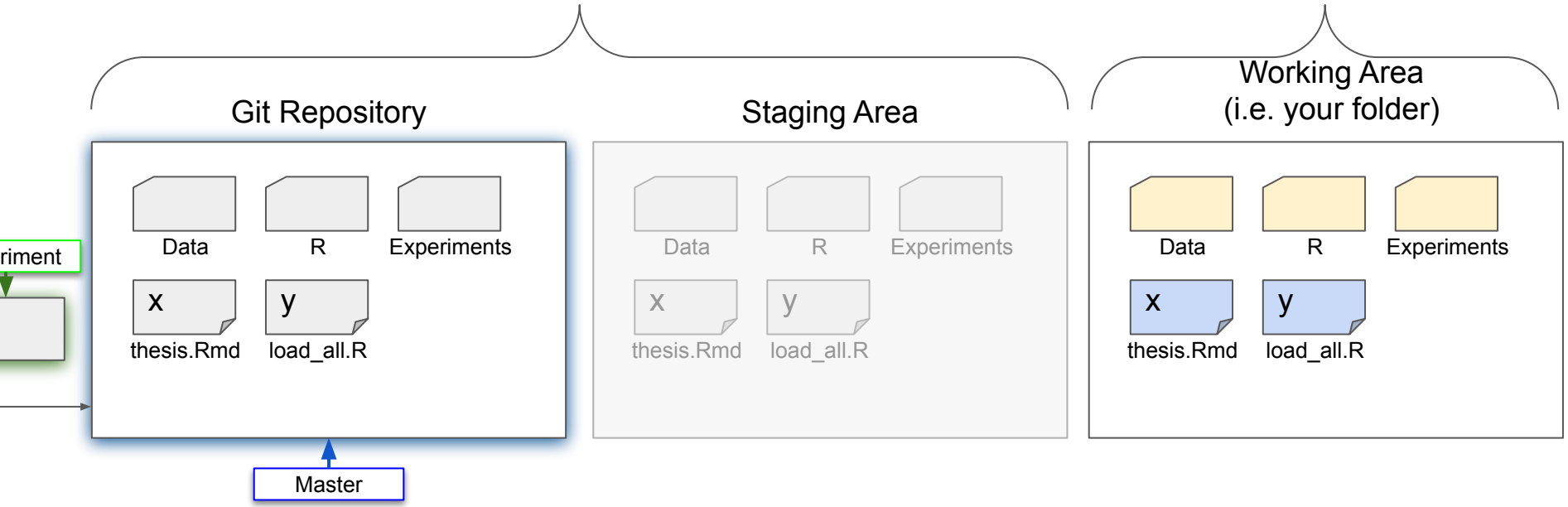
- More than one File



Git -- "Version Control System" (VCS)

How about we organize things:

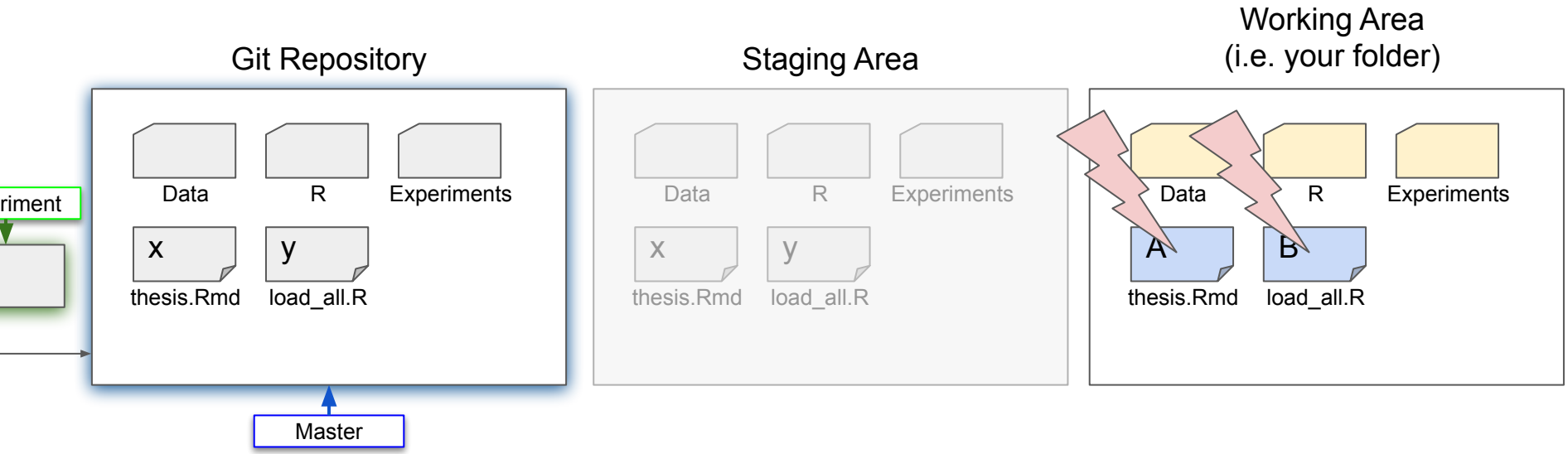
- Abstractions



Git -- "Version Control System" (VCS)

How about we organize things:

- Changing Files

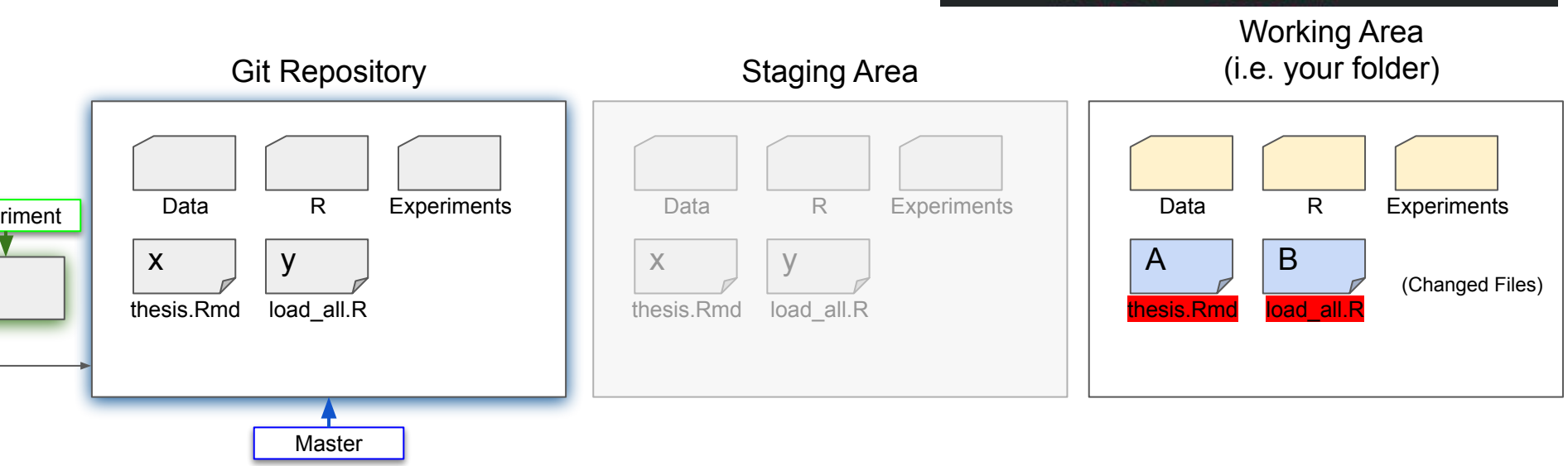


Git -- "Version Control System" (VCS)

How about we organize things:

- Changing Files: `> git status`

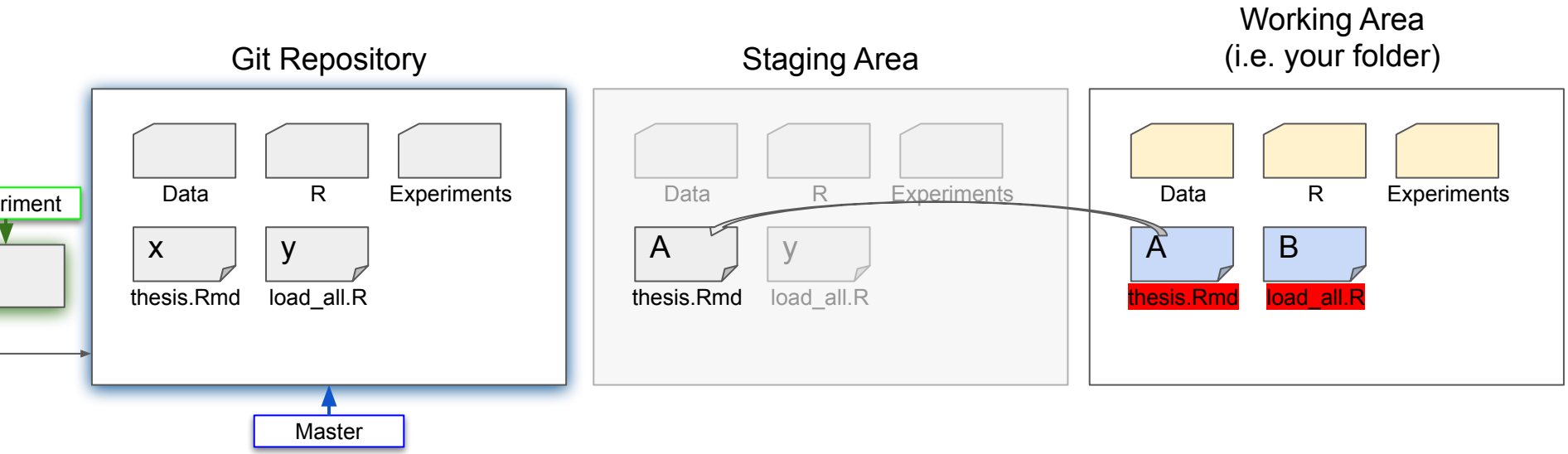
```
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will
  (use "git checkout -- <file>..." to discard
        modified:   load_all.R
        modified:   thesis.Rmd
```



Git -- "Version Control System" (VCS)

How about we organize things:

- "Staging" Files: `> git add <filename>`

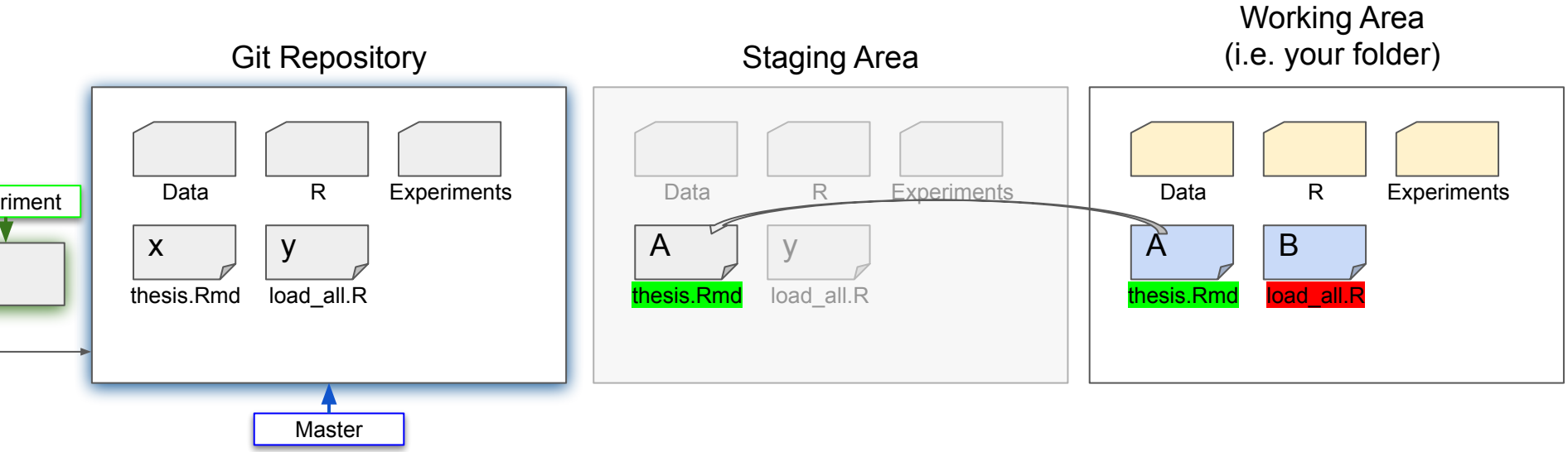


Git -- "Version Control System" (

How about we organize things:

- "Staging" Files: `> git status`

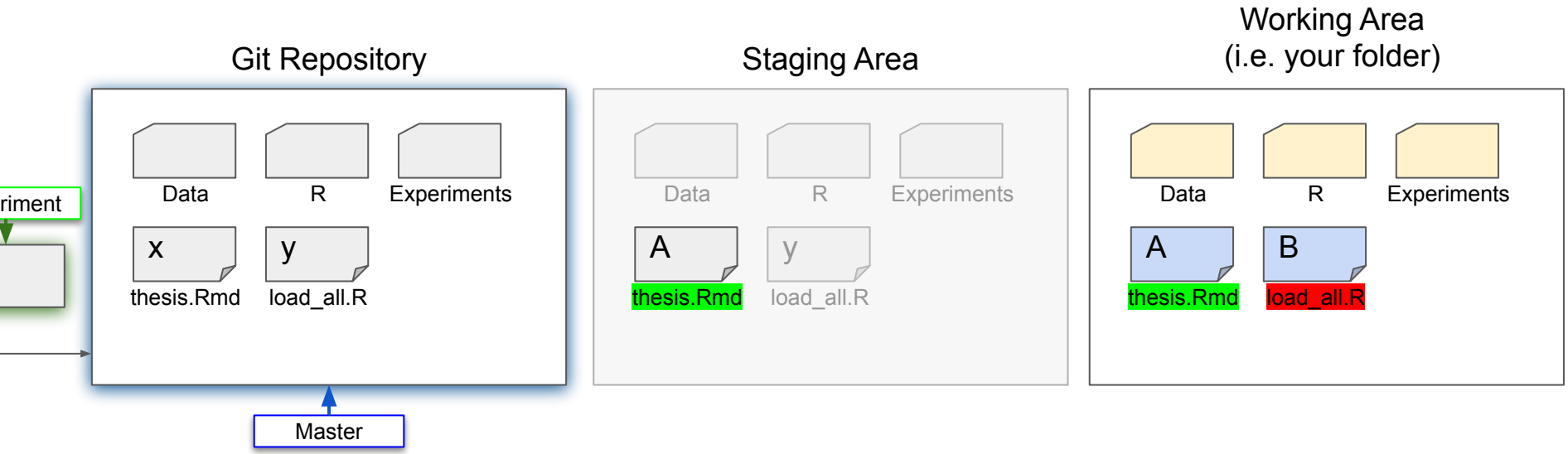
```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   thesis.Rmd

Changes not staged for commit:
  (use "git add <file>..." to update what will
  (use "git checkout -- <file>..." to discard c

        modified:   load_all.R
```



Git -- "Version Control System" (VCS)

How about we organize things:

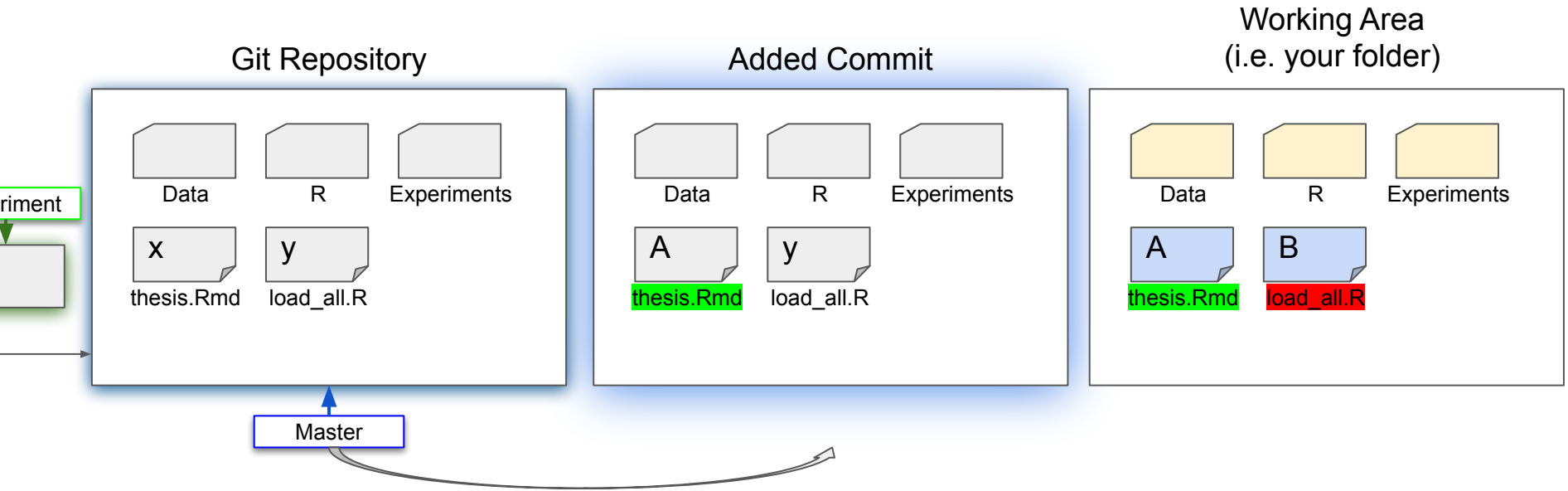
- "Commit" Files: `> git commit`



Git -- "Version Control System" (VCS)

How about we organize things:

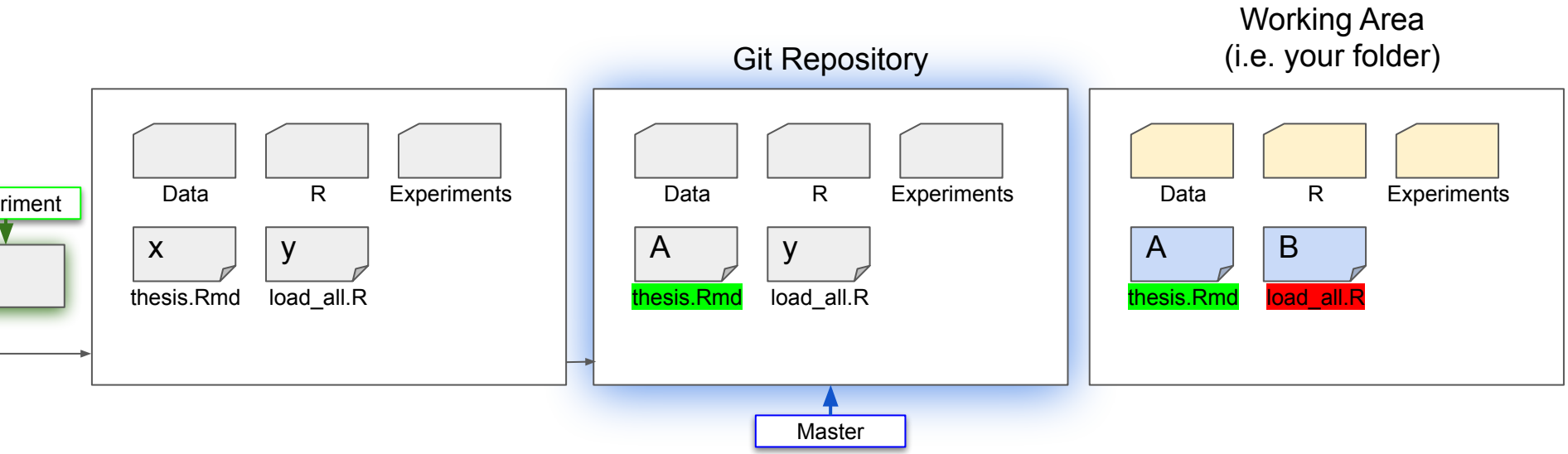
- "Commit" Files: `> git commit`



Git -- "Version Control System" (VCS)

How about we organize things:

- "Commit" Files: `> git commit`

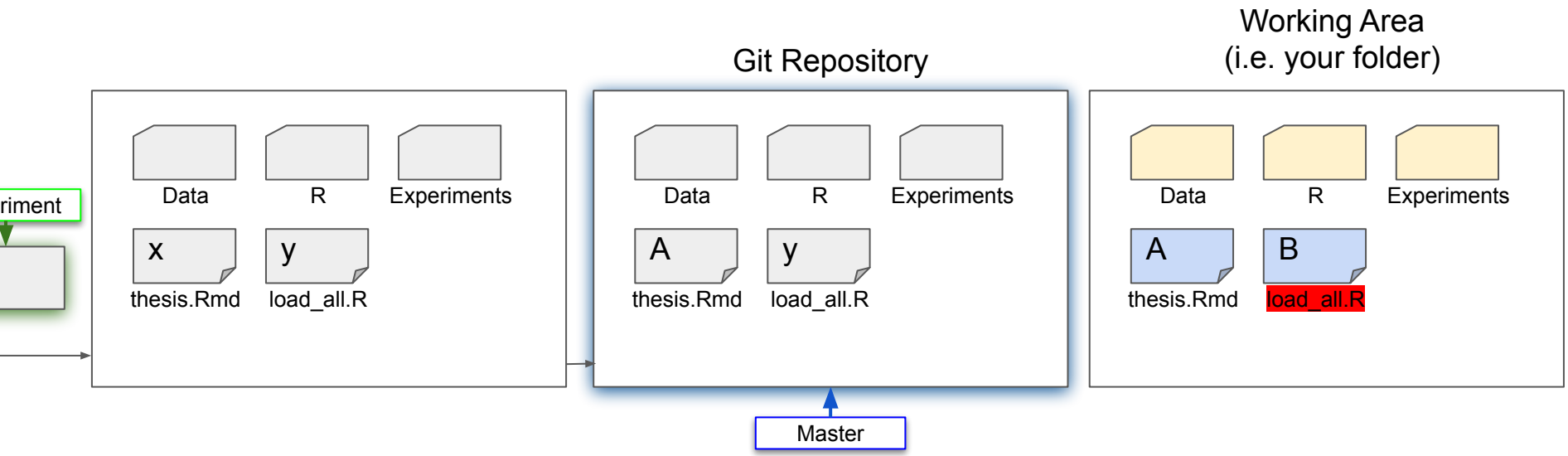


Git -- "Version Control System" (VCS)

How about we organize things:

- Status now: `> git status`

```
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will
  (use "git checkout -- <file>..." to discard
        modified:   load_all.R
```

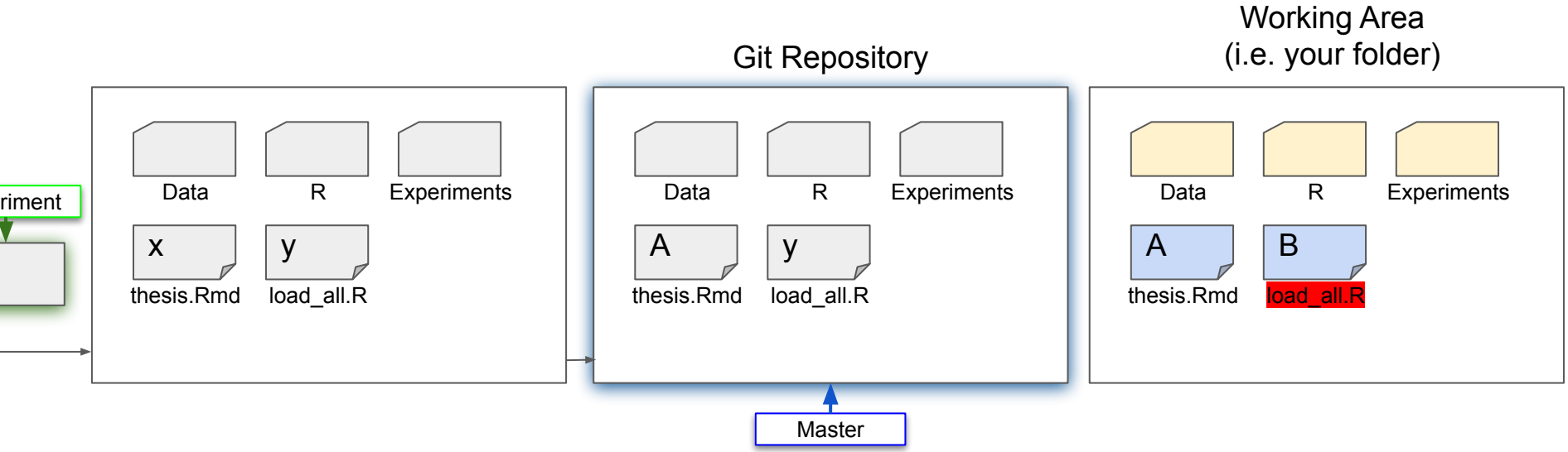


Git -- "Version Control System" (VCS)

How about we organize things:

- Changed lines: `> git diff`

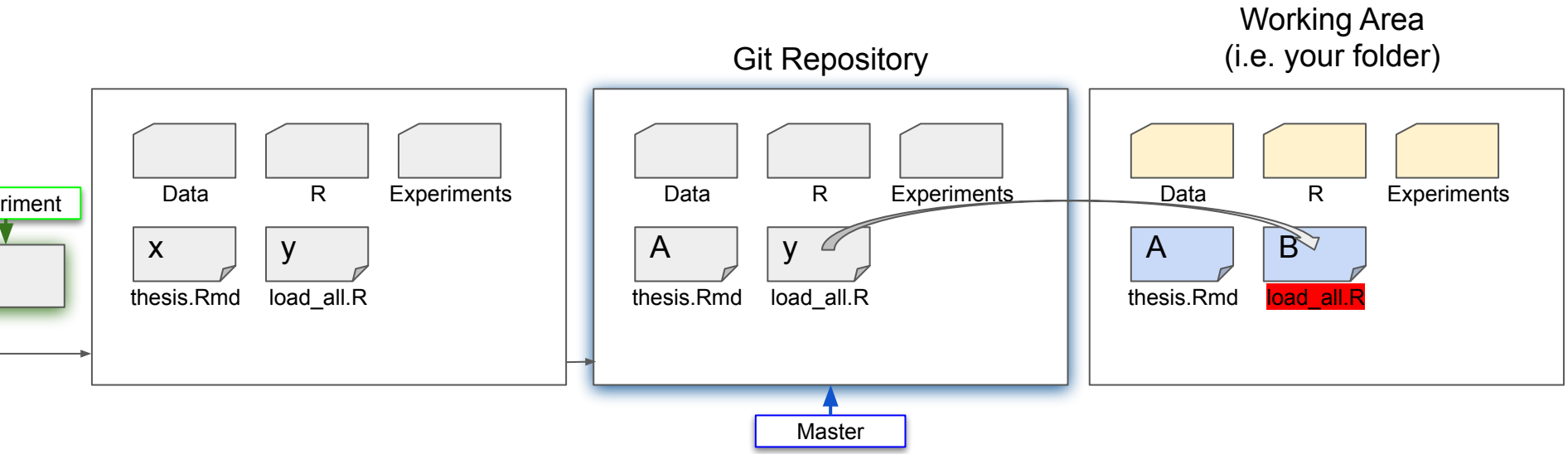
```
diff --git a/load_all.R b/load_all.R
index 975fbec..223b783 100644
--- a/load_all.R
+++ b/load_all.R
@@ -1 +1 @@
-y
+B
```



Git -- "Version Control System" (VCS)

How about we organize things:

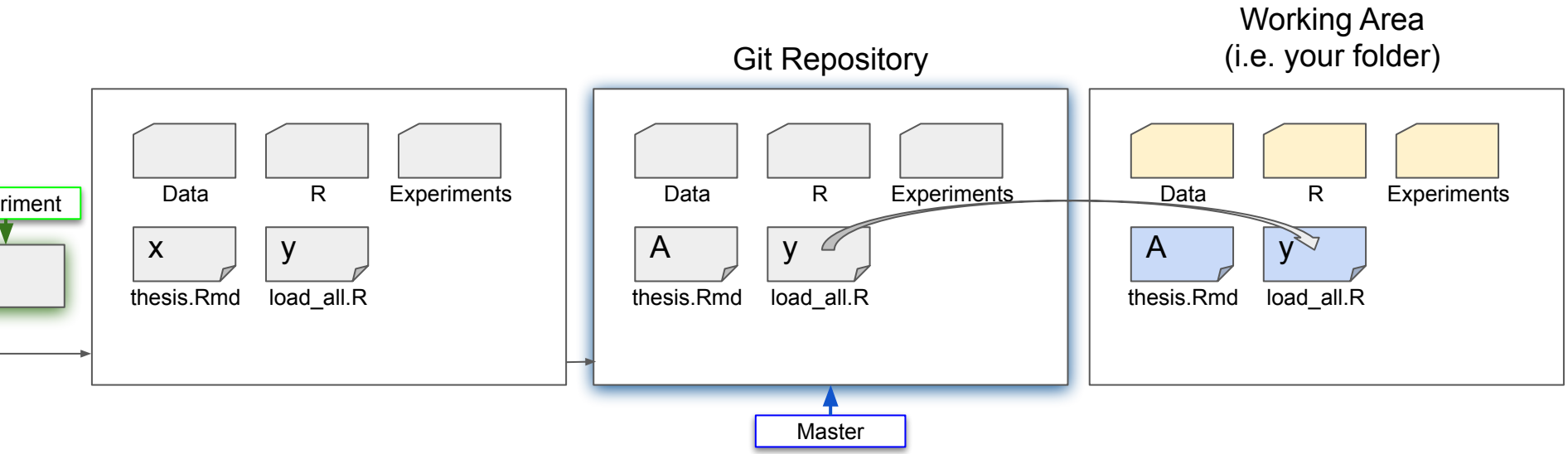
- "check out" Files: `> git checkout load_all.R`



Git -- "Version Control System" (VCS)

How about we organize things:

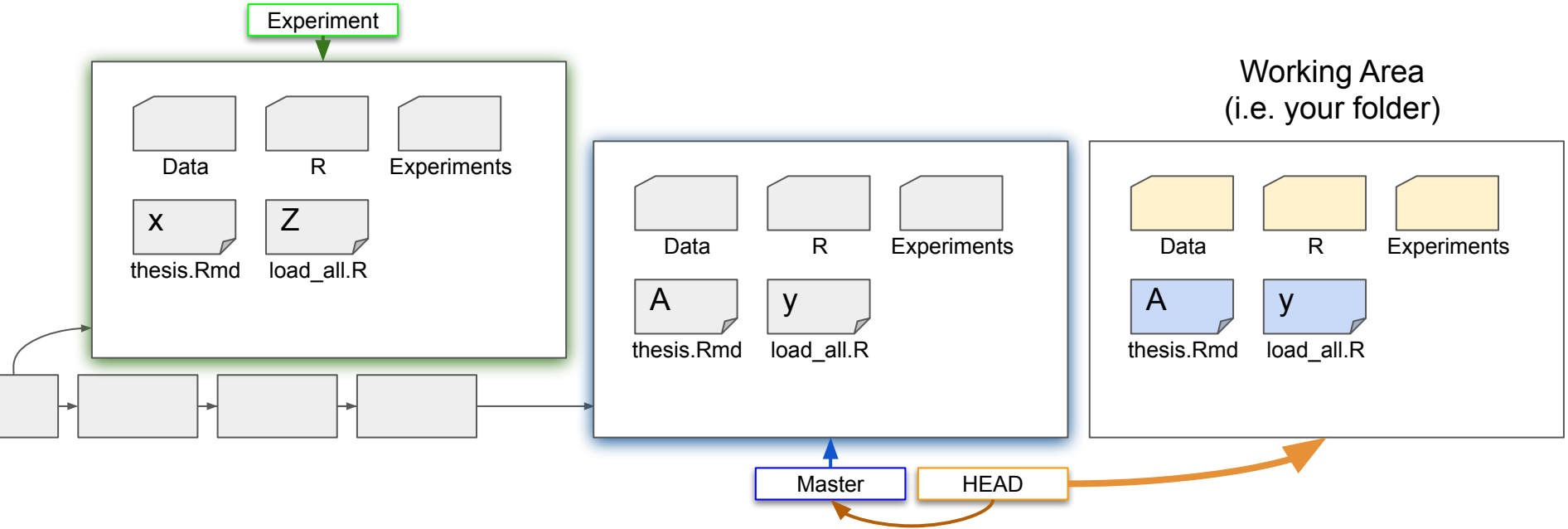
- "check out" Files: `> git checkout load_all.R`



Git -- "Version Control System" (VCS)

How about we organize things:

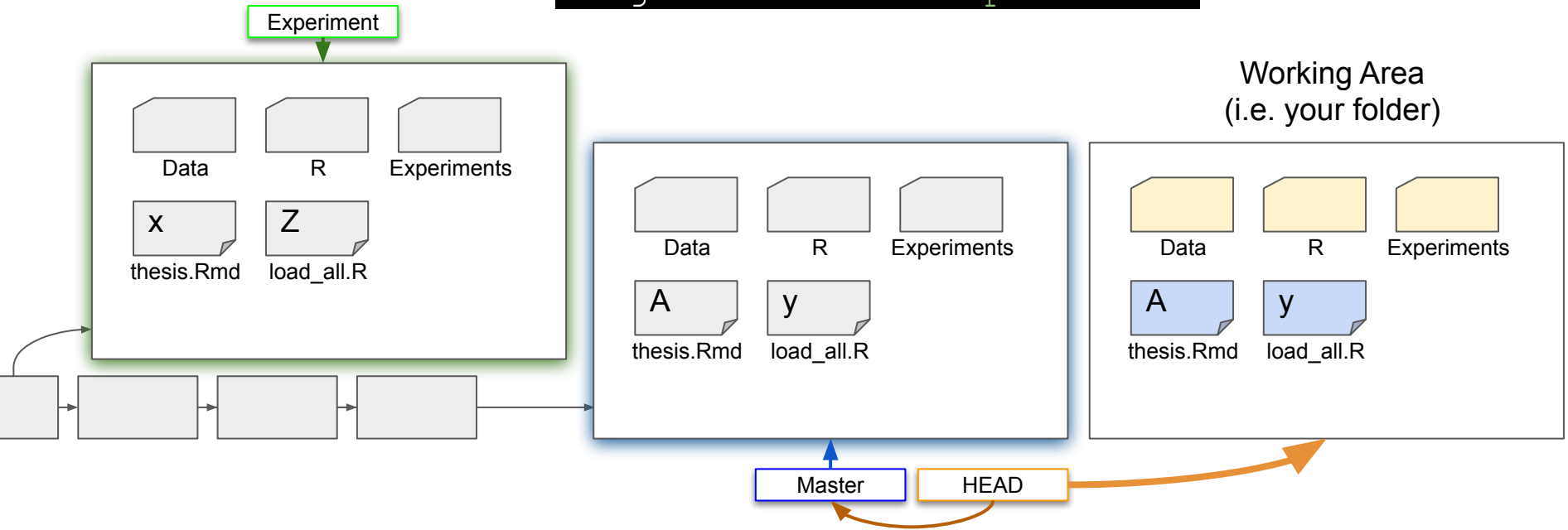
- different Branches



Git -- "Version Control System" (VCS)

How about we organize things:

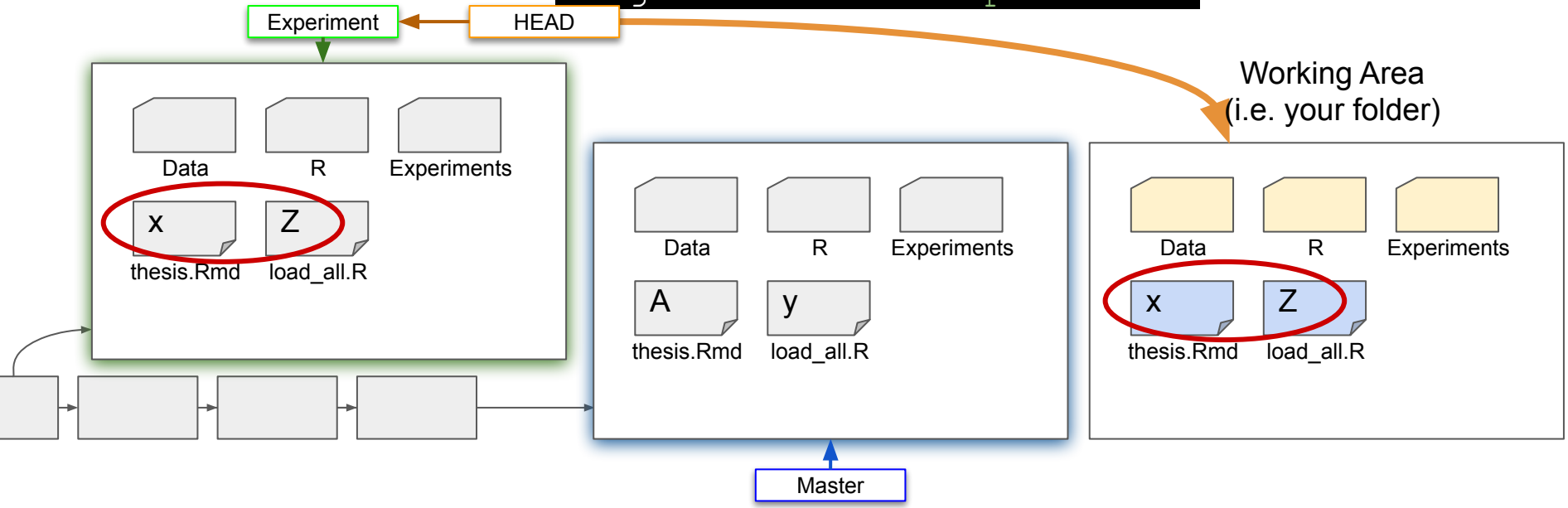
- "check out" *branches*: `> git checkout Experiment`



Git -- "Version Control System" (VCS)

How about we organize things:

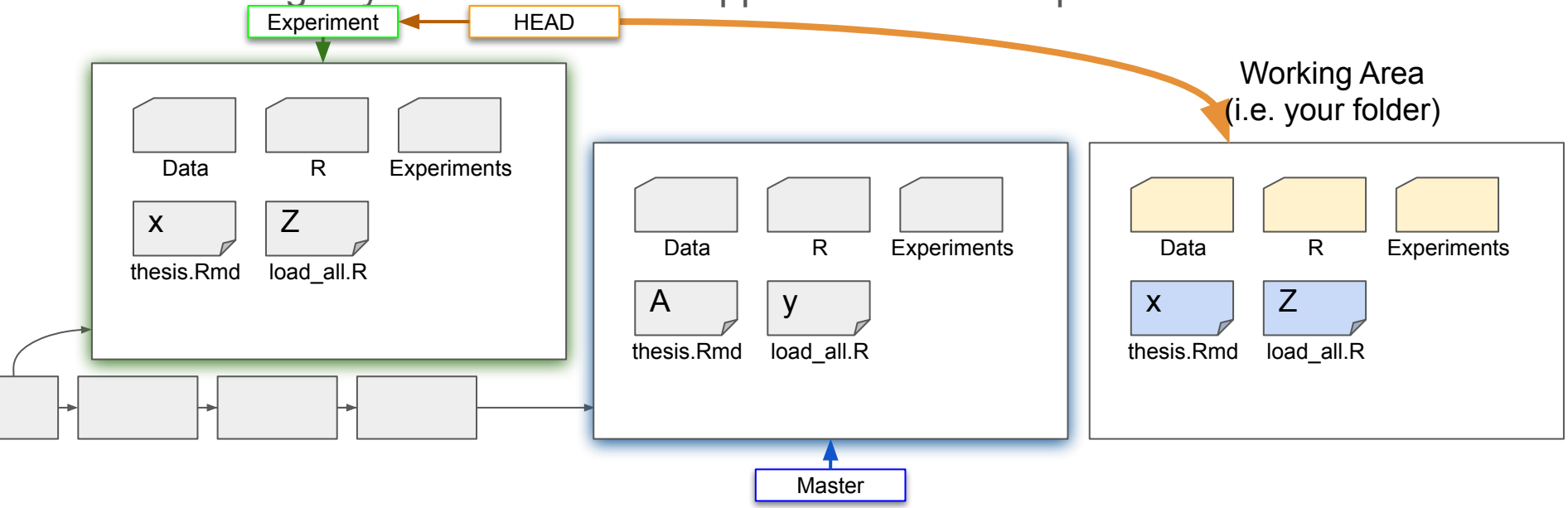
- "check out" *branches*: `> git checkout Experiment`



Git -- "Version Control System" (VCS)

How about we organize things:

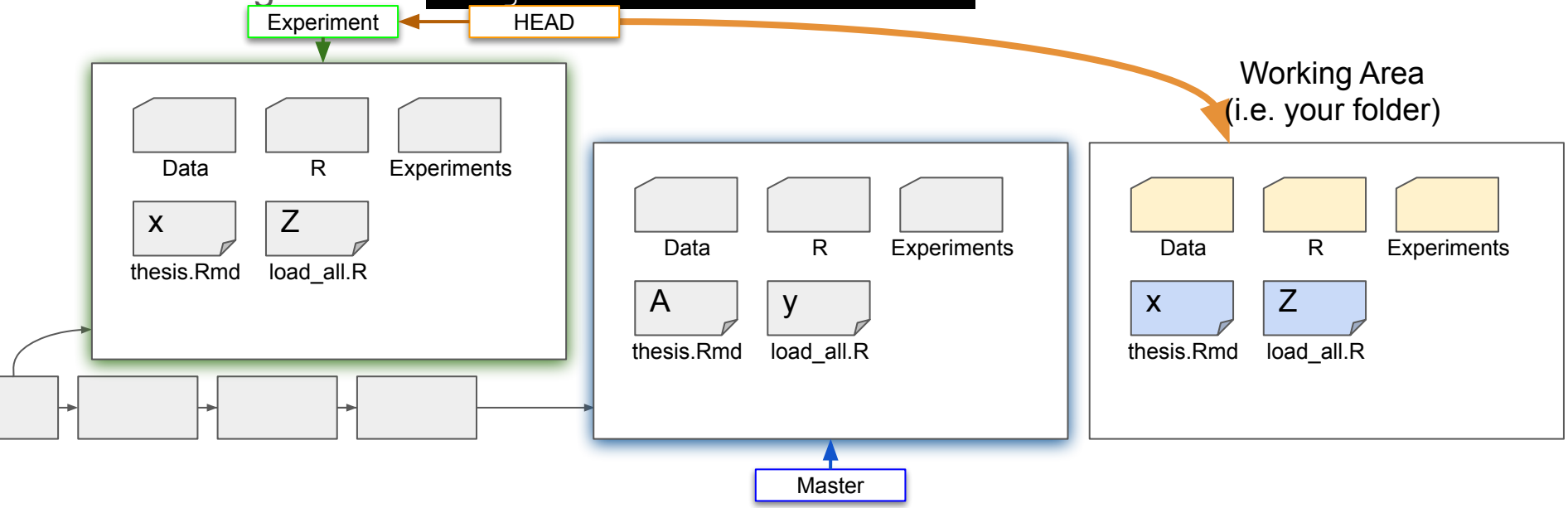
- all changes you make are now appended to the "Experiment" branch!



Git -- "Version Control System" (VCS)

How about we organize things:

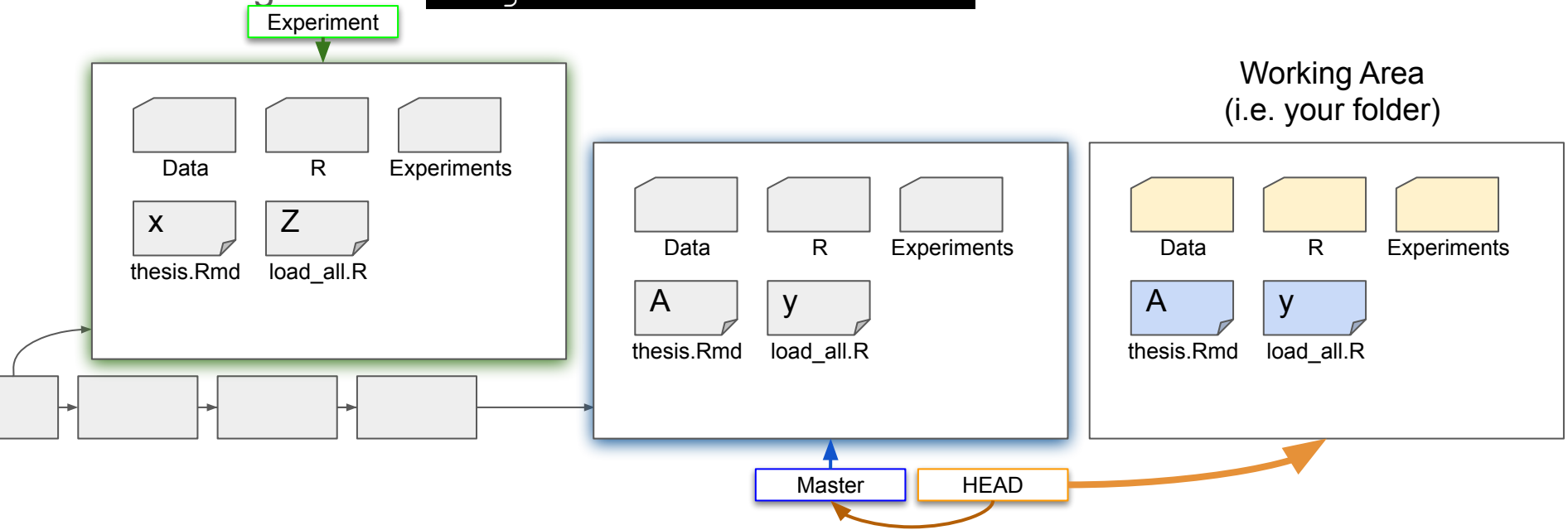
- Let's go back: `> git checkout master`



Git -- "Version Control System" (VCS)

How about we organize things:

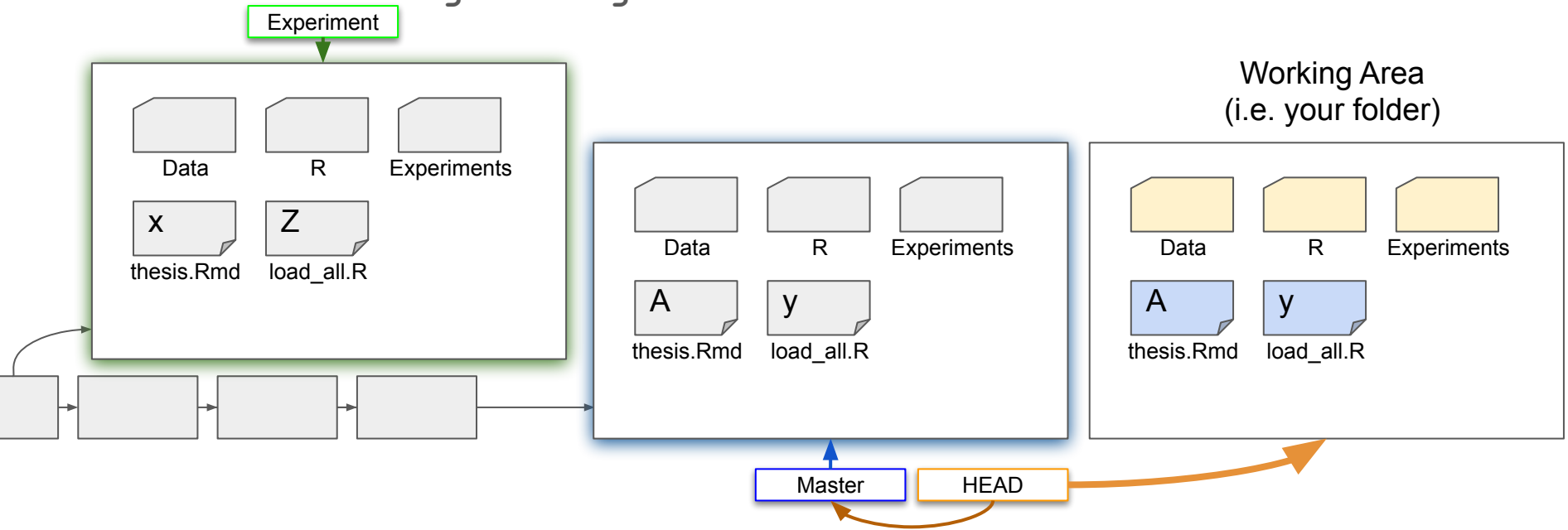
- Let's go back: `> git checkout master`



Git -- "Version Control System" (VCS)

How about we organize things:

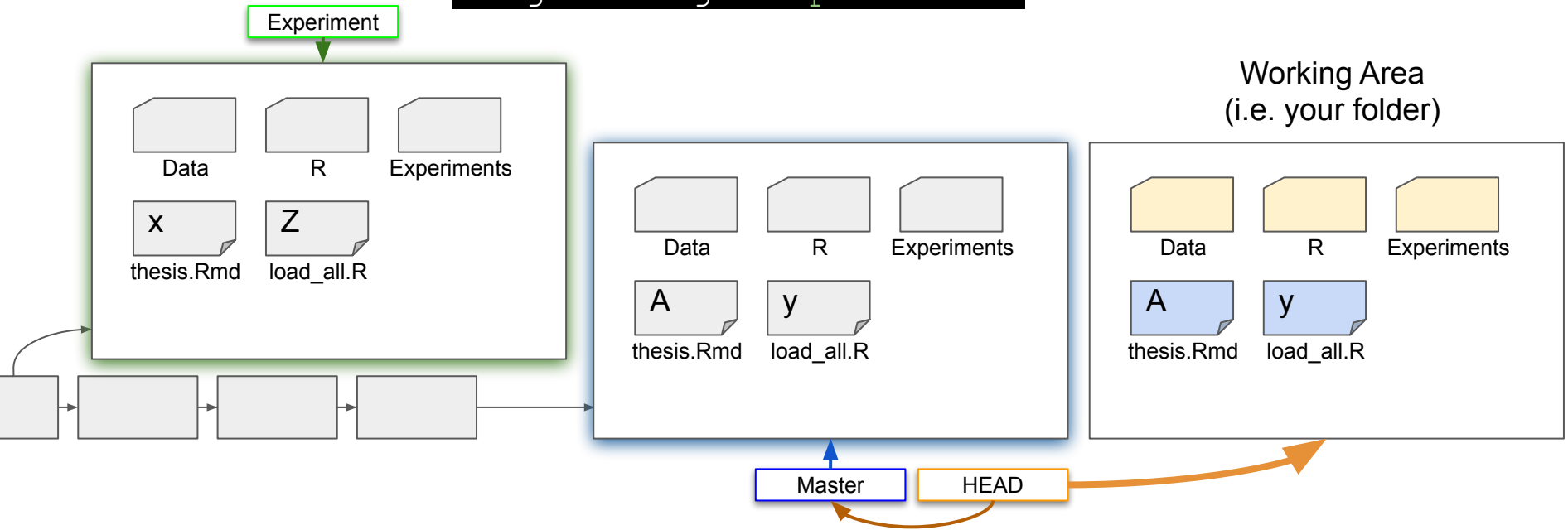
- Unite Histories: `git merge`



Git -- "Version Control System" (VCS)

How about we organize things:

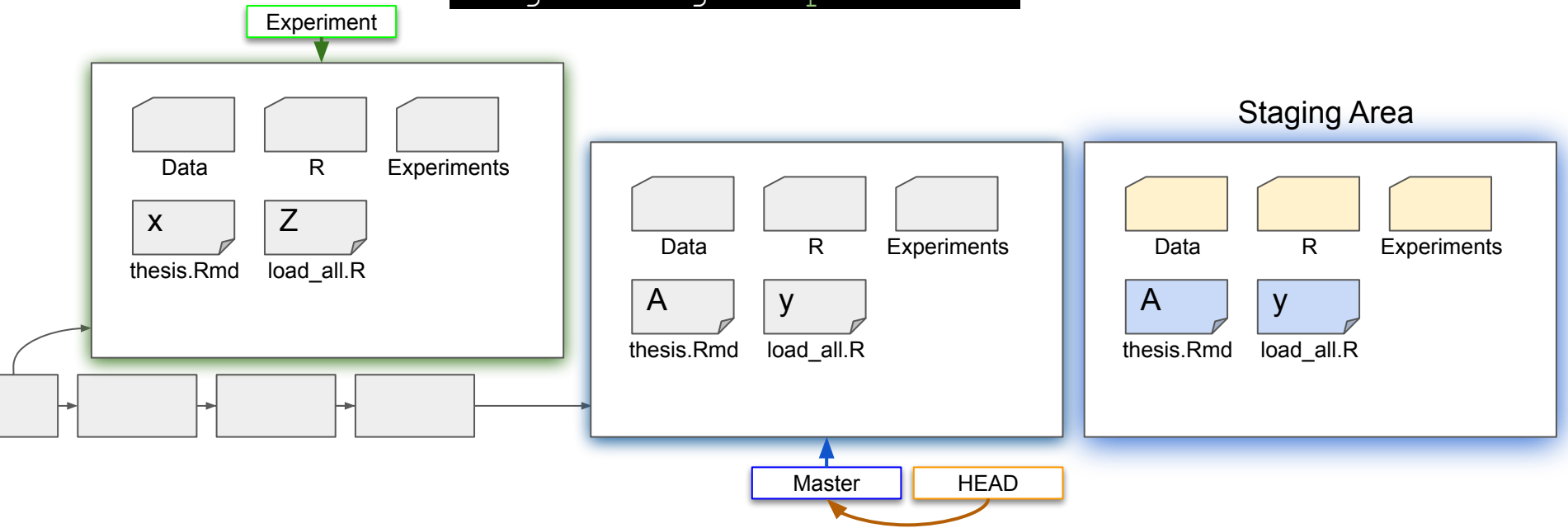
- Unite Histories: `> git merge Experiment`



Git -- "Version Control System" (VCS)

How about we organize things:

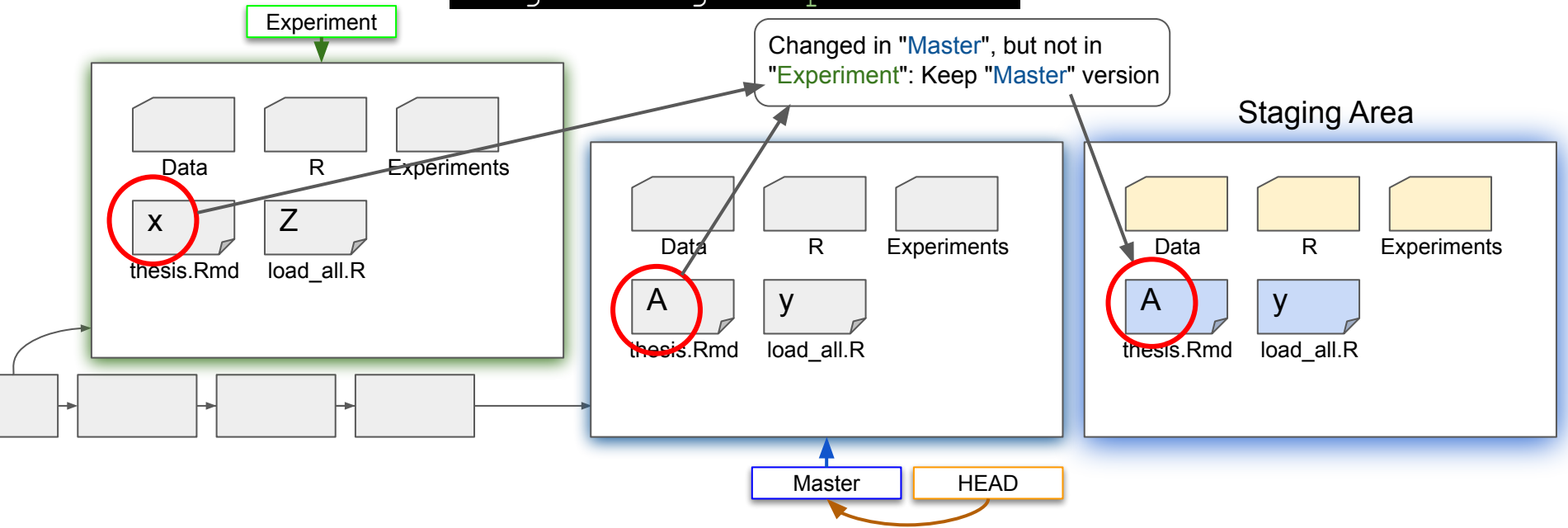
- Unite Histories: `> git merge Experiment`



Git -- "Version Control System" (VCS)

How about we organize things:

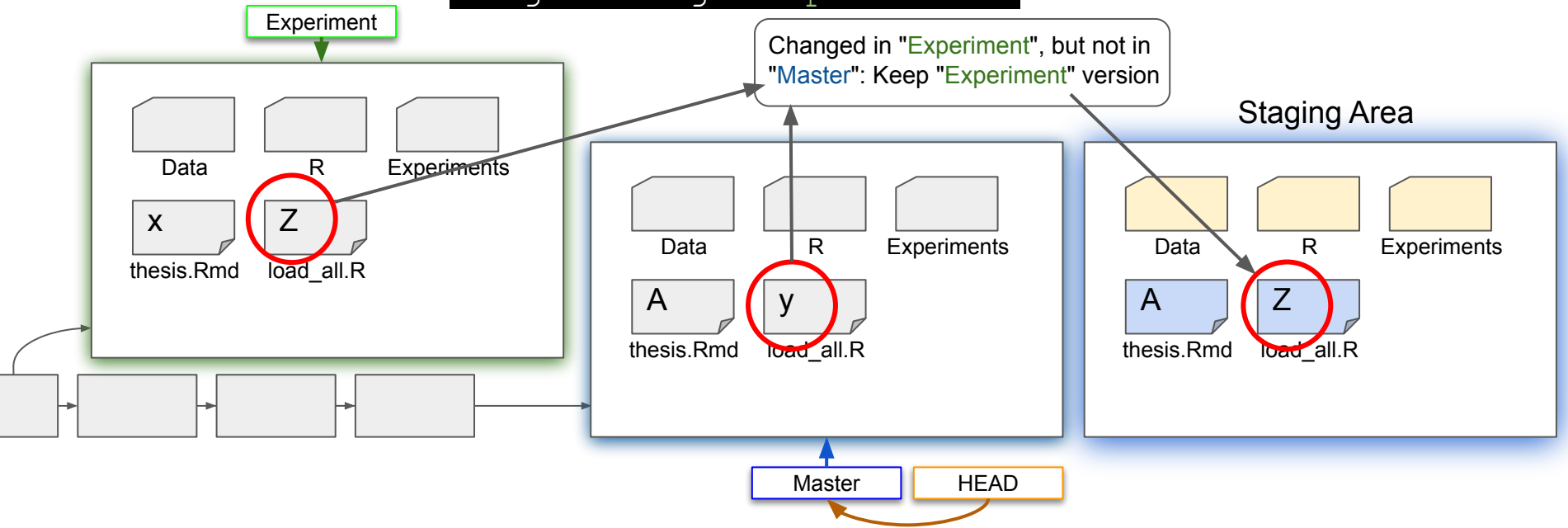
- Unite Histories: `> git merge Experiment`



Git -- "Version Control System" (VCS)

How about we organize things:

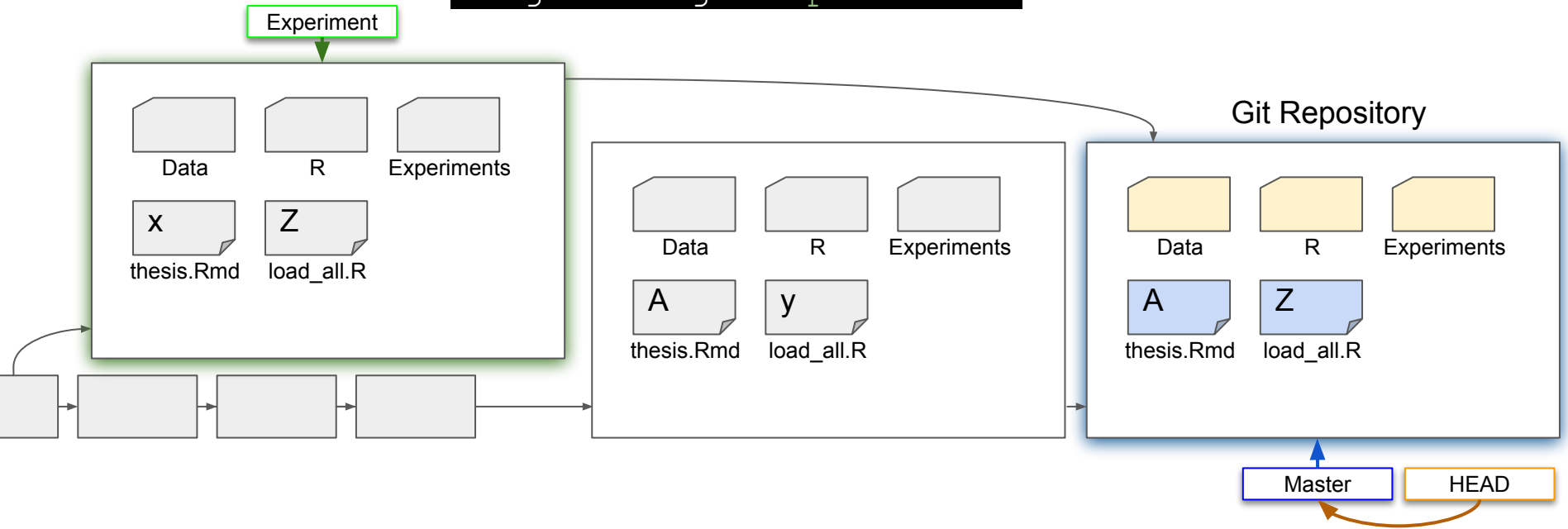
- Unite Histories: `> git merge Experiment`



Git -- "Version Control System" (VCS)

How about we organize things:

- Unite Histories: `> git merge Experiment`

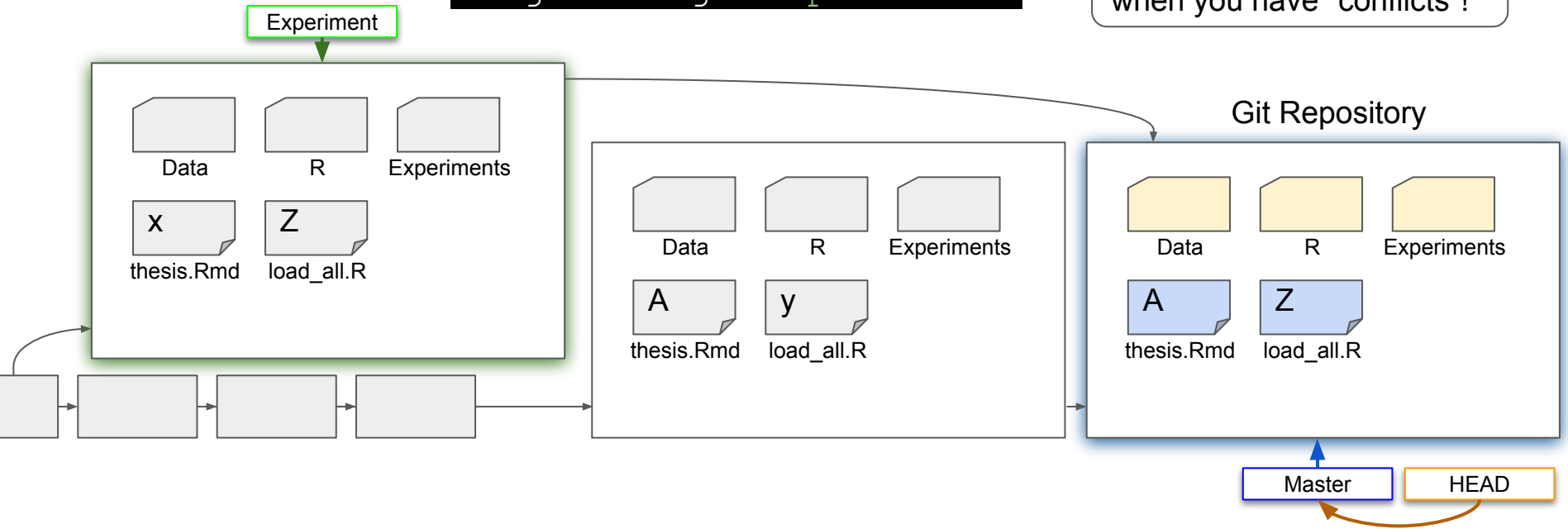


Git -- "Version Control System" (VCS)

How about we organize things:

- Unite Histories: `> git merge Experiment`

Not always this smooth when you have "conflicts"!



Git -- "Version Control System" (VCS)

- You should now see that **git** is very useful
- Learning git is notoriously difficult, but you certainly get what you pay for
- Features of git we expect you to learn:
  - simple version control -- keep track of changed files
  - branching for alternate history of files
  - "merging" of branches to combine edits done in these branches
  - Not shown here: Using git for collaborative work
    - "`fetch`" / "`pull`" from *remote repositories*
    - "`push`" to remote repositories
    - You will learn this in your homework!

What We Expect You to Know

git commands to know:

- `git init` -- create a new (empty) repository
- `git clone` -- create a repository linked to another (online) repo
- `git status` -- see if files have changed
- `git checkout` -- check out files of branches
  - `git checkout -b "new_branch_name"` -- create new branch
- `git branch` -- find out on what branch you are
- `git merge` -- take changes from another branch into the current one
  - know how to handle merge conflicts!
- `git diff` -- see what has changed (default: since last commit)
- `git commit` -- commit changes
- `git fetch` -- get changes from remote repo and DON'T merge
- `git pull` -- get changes from remote repo and DO merge
- `git push` -- push changes to remote repo
- The `.gitignore` file -- let git ignore certain files / file types

What We Expect You to Know

git commands to know:

- `git init` -- create a new (empty) repository
- `git clone` -- create a repository linked to another (online) repo
- `git status` -- see if files have changed
- `git checkout` -- check out files or branches
  - `git checkout -b "new_branch_name"` -- create new branch
- `git branch` -- find out on what branch you are
- `git merge` -- take changes from another branch into the current one
  - know how to handle merge conflicts!
- `git diff` -- see what has changed (default: since last commit)
- `git commit` -- commit changes
- `git fetch` -- get changes from remote repo and DON'T merge
- `git pull` -- get changes from remote repo and DO merge
- `git push` -- push changes to remote repo
- The `.gitignore` file -- let git ignore certain files / file types

Git

https://xkcd.com/1597/

Git

https://xkcd.com/1597/

(Please be better than this guy)

Your first Homework Task

Your first homework Task

"Familiarize yourself with Git & GitHub"

(You will not get extra points for this task, but boy will you regret not doing this.)

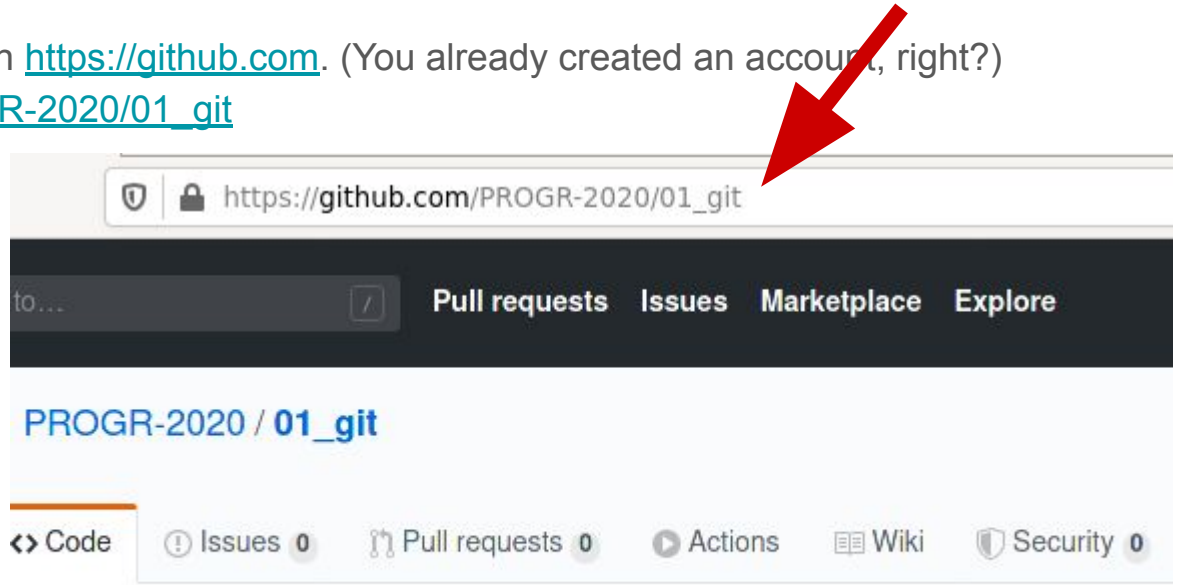
1. Log in to you GitHub account on https://github.com. (You already created an account, right?)

Your first homework Task

"Familiarize yourself with Git & GitHub"

(You will not get extra points for this task, but boy will you regret not doing this.)

1. Log in to you GitHub account on https://github.com. (You already created an account, right?)
2. Go to https://github.com/PROGR-2020/01_git

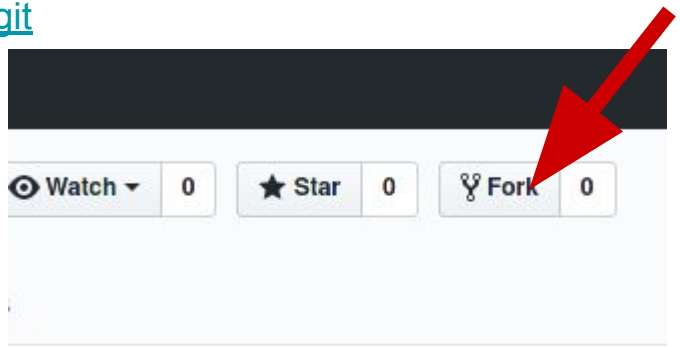


Your first homework Task

"Familiarize yourself with Git & GitHub"

(You will not get extra points for this task, but boy will you regret not doing this.)

1. Log in to you GitHub account on https://github.com. (You already created an account, right?)
2. Go to https://github.com/PROGR-2020/01_git
3. Click on "Fork" to create your own copy



Your first homework Task

"Familiarize yourself with Git & GitHub"

(You will not get extra points for this task, but boy will you regret not doing this.)

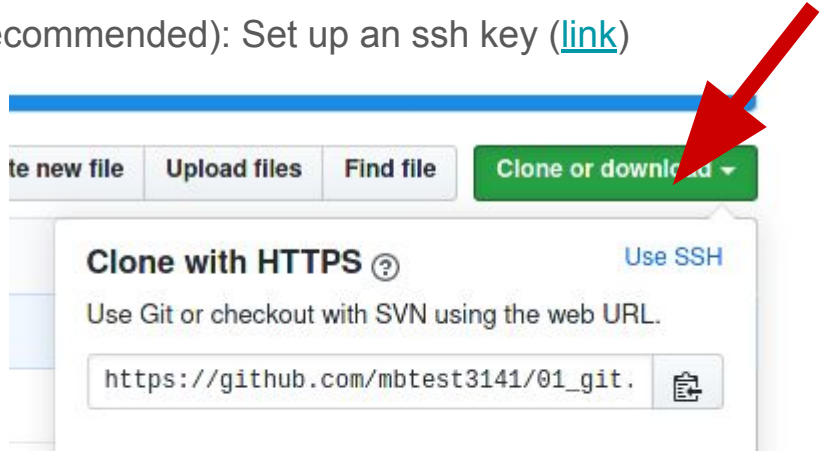
1. Log in to you GitHub account on https://github.com. (You already created an account, right?)
2. Go to https://github.com/PROGR-2020/01_git
3. Click on "Fork" to create your own copy
4. Learn what `git clone` does. Optional (but recommended): Set up an ssh key (link)

Your first homework Task

"Familiarize yourself with Git & GitHub"

(You will not get extra points for this task, but boy will you regret not doing this.)

1. Log in to you GitHub account on https://github.com. (You already created an account, right?)
2. Go to https://github.com/PROGR-2020/01_git
3. Click on "Fork" to create your own copy
4. Learn what `git clone` does. Optional (but recommended): Set up an ssh key (link)
5. Clone the repo to your own computer

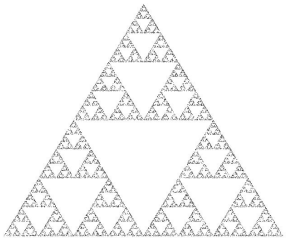


Your first homework Task

"Familiarize yourself with Git & GitHub"

(You will not get extra points for this task, but boy will you regret not doing this.)

1. Log in to you GitHub account on https://github.com. (You already created an account, right?)
2. Go to https://github.com/PROGR-2020/01_git
3. Click on "Fork" to create your own copy
4. Learn what `git clone` does. Optional (but recommended): Set up an ssh key (link)
5. Clone the repo to your own computer
6. Do "The Task" as described in the repository!



Your second Homework Task

Your second homework Task

- ... will be published on Friday 2020-04-24
- ... and is due Thursday 2020-04-30, 23:59:59 CEST
- ... and will assume that you know the basics of `git`