Module: Collaboration and Workflow

Rebecca C. Steorts

Expectations

- Assume that you have caught up from last week if you weren't in class
- Assume you have read the syllabus and know course expectations
- Assume you have joined Canvas, Piazza, Gradescope, etc.
- Assume you have started homework 1 and know how to submit it

I forgot to do something

- If you forgot to do something today, don't panic!
- If you have not done any of these items, please make sure to do these after class.
- Please also make sure that you're doing these each week and keeping up with the course.

Expectations

- Come prepared and ready to interact
- Ask questions!
- Please give me feedback if something is NOT working for you regarding how I'm teaching the course. I want to know what's going well and what can be improved.

Goals

My personal goal is that everyone does well in this course and gets practical applied experience to take away to industry, academia, etc.

My personal goal for myself is that I am an effective instructor and help you meet your goals and objectives of the course!

Finally, if you happen to have a difficult time during the semester, **please let me know**.

Flow of Class

In today's lecture, we're going to review the following concepts:

- organization
- reproducibility
- version control

We will also perform an interactive exercise of setting up a repository (individually).

Then we will move into how we can work together collaboratively on projects and talk about why this is important.

Agenda

- Organization
- Reproducibility
- Version Control

Organization

One asset of being an efficient data scientist is being organized.

Organization

Why is organization an important skill?

Organization

Messy files and poorly documented code mean that:

- you will need to often dig through your code
- you will have trouble communicating with your collaborative team
- you will have a difficult workflow for a industry/academic project
- you will likely have trouble reproducing your results

These can all be disasterous if you're working on product development for a large company and they are depending on your skills, expertise, and analysis!

Self-Assessment

Think about a recent programming assignment from the past year.

- Do you think you can reproduce the results exactly?
- Do you think that anyone in the class could easily understand your code (i.e., is it well documented)?

Give a **short assessment** regarding one strength of your coding and one weakness that you plan to work on in this course.

Collaborative Programming

How can we work collaboratively on a project involving code?

1. Create a strong team with a team leader to keep the group **organized** and **on track**
2. Making sure that our code is **reproducible** and using software such as R markdown as one such option
3. Using **version control** (github) for our collaborative work in order to **effectively communicate** and **work together**

Reproducibility

What is reproducibility and why is it important?

The Reproducibility Crisis

"The terms "reproducibility crisis" and "replication crisis" gained currency in conversation and in print over the last decade (Pashler & Wagenmakers, 2012) as disappointing results emerged from large scale reproducibility projects in various medical, life and behavioural sciences (Open Science Collaboration, OSC 2015)."

The Reproducibility Crisis

What does this mean?

- Researchers are finding that when they try and replicate papers, their results are not the same as in the published work.
- This has led to the above crisis.
- The response is to push data scientists to publish work that is **fully transparent** and can be **fully checked** when we pass it to someone else.

Reproducibility

- Work is considered reproducible if the entire process can be replicated from start to finish.
- This includes the data, code, figures — everything.

Reproducibility

- Why is reproducibility important for our community?
- Why is reproducibility hard to achieve?

Reproducibility

We just talked about **why** it's important for our work to be both organized and reproducible.

Let's talk about one tool we can use to make our work reproducible.

Rmarkdown

- R is a statistical programming language
- RStudio is a convenient interface for R (an integrated development environment, IDE)

Put simply:

- R is like a car's engine
- RStudio is like a car's dashboard

R packages

To make our work the most reproducible as possible, best practice is writing our finished product into an R package available on CRAN and making this accessible to the community.

R packages

1. CRAN packages have strict criteria for acceptance so this forces you to write your code in best practice
2. You can tie in your R packages to your paper/project to make it reproducible from start to finish

This is the most idealized way of writing software and working with a team.

To learn more about R packages, you can check out https://r-pkgs.org/.

Version Control

What is version control?

Version Control

Version control is the practice of tracking and managing changes to software code.

Version control systems (github) are software tools that help software teams manage changes to source code over time.

Version Control

- GitHub is a platform for collaboration
- It's really designed for version control

Version Control

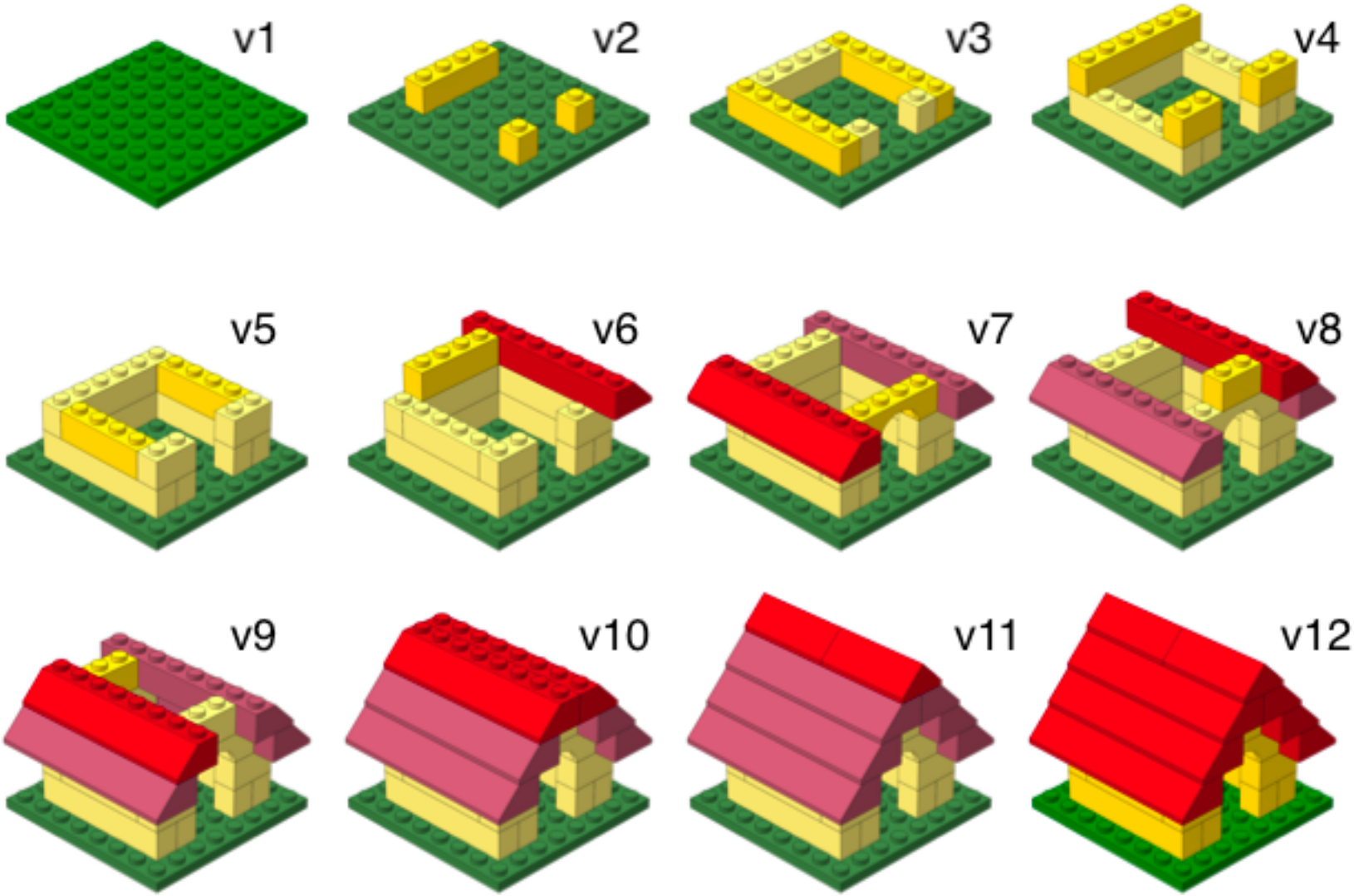


Figure 1: Lego steps

Version Control with commit messages

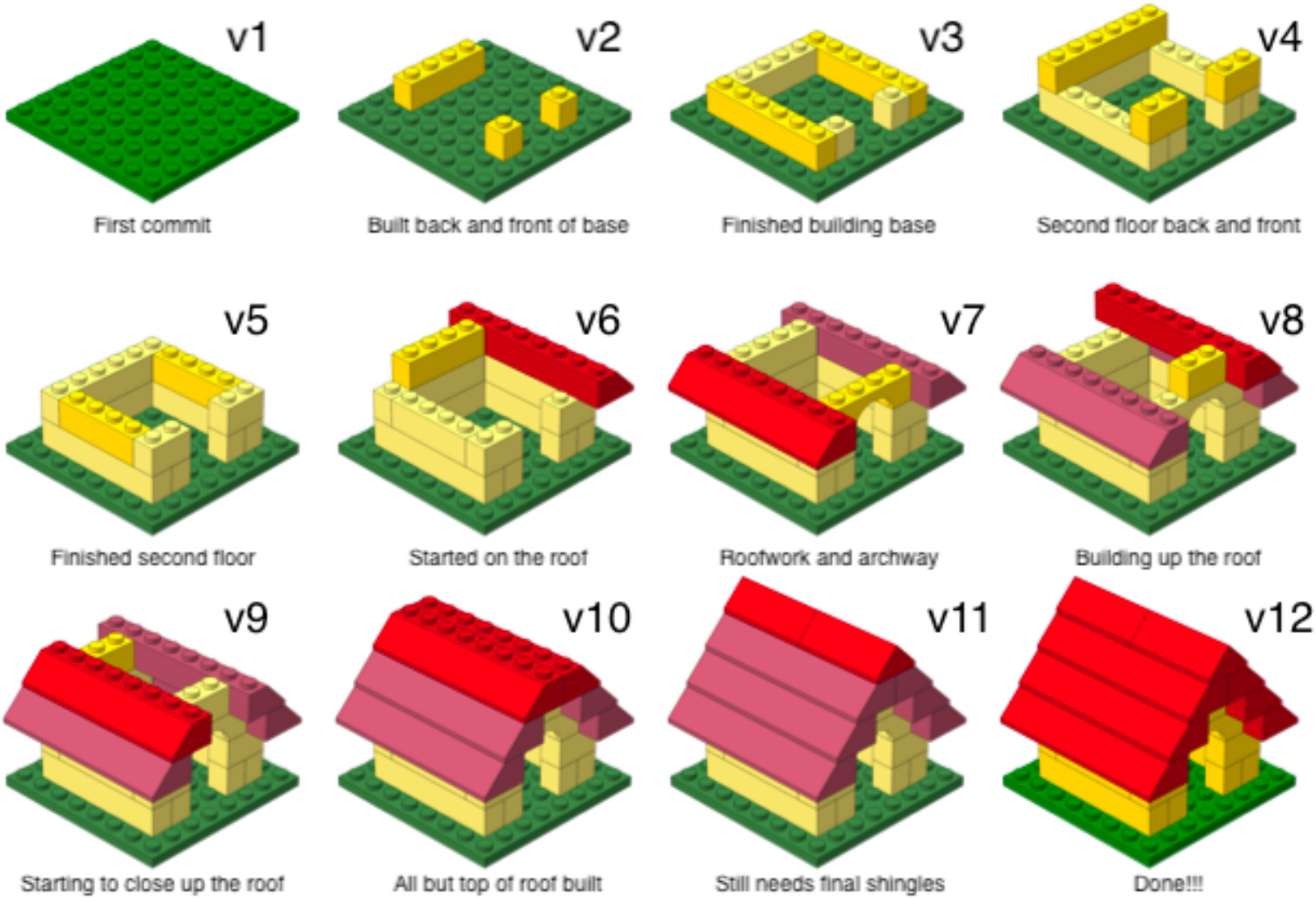
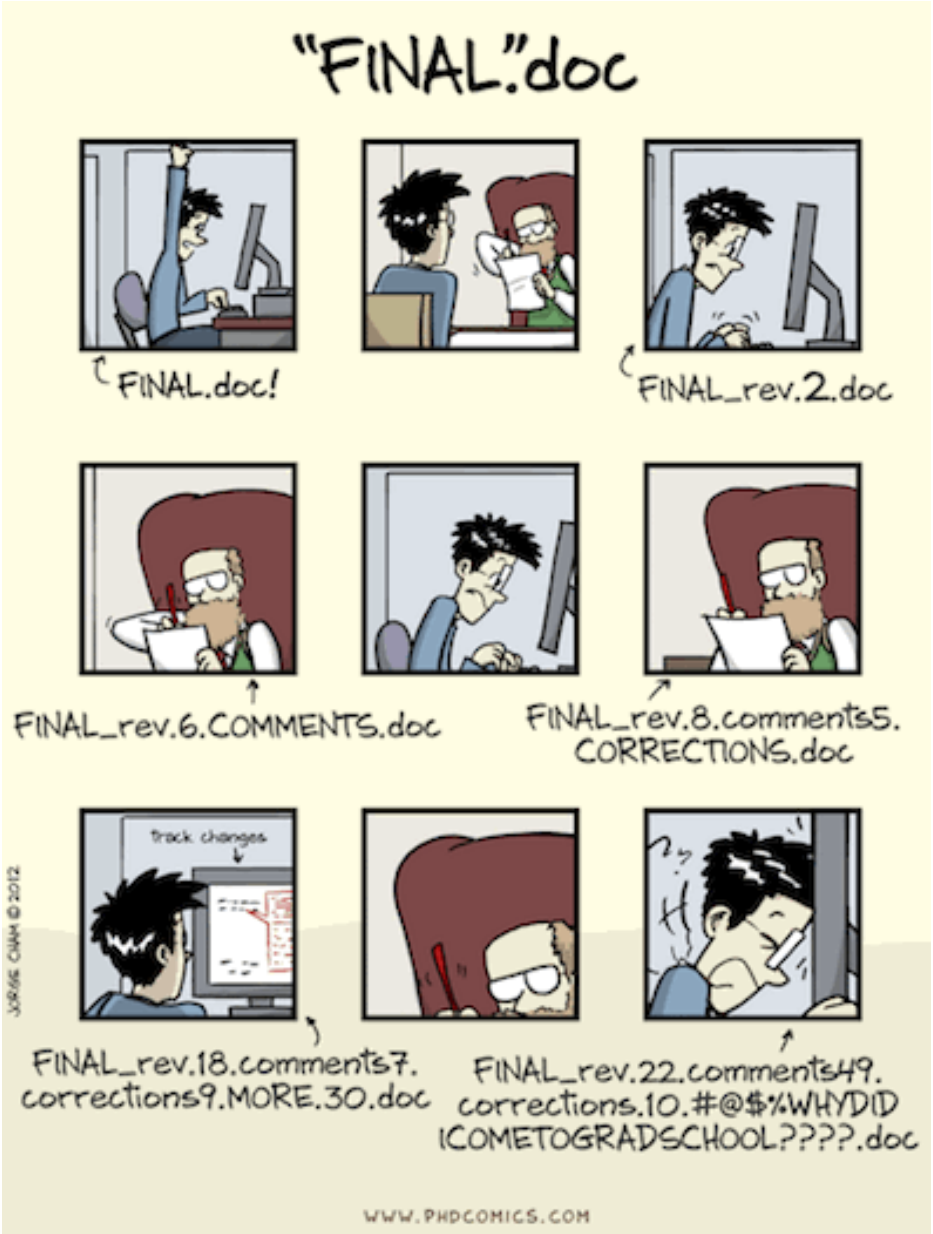


Figure 2: Lego steps with commit messages

Why do we need version control?



Github and Tips

- Git is a version control system – like "Track Changes" features from Microsoft Word.
- GitHub is the home for your Git-based projects on the internet (like DropBox but much better).
- There are a lot of Git commands and very few people know them all. 99% of the time you will use git to add, commit, push, and pull.

Github Terminology

Repository: A repository (repo) is similiar to the project's folder. It contains all of the project's files and stores each file's revision history. A repo can either be private or public.

Example: I'd like for your to make a repository called `hello-world`.

Github Terminology

Clone: A clone is the act of making a copy of a repository from a webpage or server.

Example: I'd like for your to clone an existing repository called `data-clean`, which is the repo for this class.

Github Terminology

Remote: The **remote or global version** is hosted on a server or webpage. It can be connected to the local copy on your computer so that changes can be synced.

Example: The remote or global version of the repository is hosted on github.

Github Terminology

Pull: This is the act of fetching information from the **global (remote) server** with your local copy.

Example: I am ready to pull the global changes that I made online to my local computer.

Github Terminology

Commit: An individual change to a file (or set of files). Every time you save the file(s), a a unique ID is created that keeps a record of what the change was and who made the change.

Example: I've fixed some typos to my repository and I'm going to commit the change, indicating that typos were fixed.

Github Terminology

Push: Sending your committed changes to a remote repository such as gitHub.com.

Example: I'm ready to push my changes to the global repository.

Github Terminology

Fork: A fork is a personal copy of another user's repository that lives on your account. Forks allow you to freely make changes to a project without affecting the original.

Example: I'm going to fork a repository so that I can make changes to it without affecting the original repository since I don't have permissions to change it!

Github Terminology

Pull Request: Proposed changes to a repository submitted by a user and accepted or rejected by a repository's collaborators.

Example: I've found a typo in a repository. I'm going to submitted a pull request to propose a change to the repository and improve the quality of open source code.

Github Terminology

Issue: Issues are suggested improvements, tasks or questions related to the repository.

Example: There seems to be an issue with my collaborators code, so I open an issue so that my collaborator can take a closer look.

When should I commit?

- You should commit early and often to your repository.

Think of commits as a checkpoint in a video game. This is a point in time when you want to save your status so that you can come back to it later if need be. Otherwise, everything might be gone!

Things will blow up

- Sometimes your local repo gets borked and you might have merge conflicts.
- My main advice is that sometimes you can work very hard to fix them and sometimes you have to stash (store your work) and manually resolve your merge conflicts.
- This is **the most frustrating part of using version control**, which is why I commit early and often.

Git and Github Tour

[DEMO]

In this demo, we will do the following:

- Create a new repository called `hello-world`
- We will work with our local and remote (global) repository
- Make a change locally, committing, and pushing to the global repository
- Make a change on GitHub and pull to our local repository

Your turn

1. Create a github account and username
2. Consider your username, want to be identiable, professional and probably include your actual name. Do you have other handles? Twitter?
3. Don't worry about paying for a plan now, stick with the free one
4. Create a `hello-world` repo
5. Play around with your `hello-world` repository and try out the same demo that I did

Questions?

Recap

Can you answer these questions?

- Explain why code needs to be organized
- Explain the reproducibility crisis
- Explain reprodubile workflow
- Explain version control
- How can we put all these tools together?