Lecture 1: Introduction

Irina Gaynanova

Before/while we start

- Make sure you have a GitHub account
- Make sure you have accepted HW1 (following the link on eCampus)
- Make sure you have Git installed and have bash (for Windows users, Terminal for OS X users)
- Make sure you have Rstudio installed
- Troubleshooting: **HappyGitWithR**

If you haven't brought your laptop - > seat next to someone who has

About the course

Instructor: Dr. Irina Gaynanova (*random facts about me*)

Overall topic: statistical computing and optimization in *R*

More specifically:

- version control in R via git/github
- code profiling, timing, good coding practices
- convex optimization/duality
- optimization algorithms: gradient methods, coordinate-descent, proximal gradient, etc.
- cross-validation and splitting techniques
- interface to C/C++

Structure and policies

Homework (15%), roughly biweekly (6 total)

- Individual, primarily programming assignments in **R**
- Will be distributed via GitHub Classroom

Midterm (40%)

- In class, October 17th - **let me know if this is a problem this week**

Project (45%) - writing an R package

- Individual
- Several stages: Proposal, Intermediate and Final
- Final - Github repo that can be installed with **devtools**
- More info closer to the Proposal stage

Late submission policies

- Late HW/project will **never** be accepted. You are welcome to submit as early as you want.
- The lowest HW score will be dropped.
- GitHub has an automatic time stamp associated with all commits, even 1 min late is late. 1st HW will be accepted late (*exception*) with 10 points off if submitted within 24 hours of due date.

Collaboration policies

- You are welcome to discuss assignment with other students in person or via discussion boards but **without sharing any part of your code**
- You are welcome to use Google/StackOverflow for help, but if your code is from outside sources you have to acknowledge it (i.e. in code comments). These outside sources can not be other students
- **Exception:** If your code is not working and you can't figure out why, you can show your code to another student **in person** and ask for help in **identifying mistake**. Please acknowledge the student who helped you catch the mistake in the comments.

Plagiarism policies

- I will use automatic system to detect similar submissions and that system is smart (i.e. renaming won't help)
- The HWs/projects that are deemed very similar by the system will be reviewed by me
- If I believe the course policies are violated, **both** students get 0 irrespective of who did the work.

Electronics policies

- You are not allowed to be on your cell phone during the class. If you have a call you need to answer/make, please quietly exit the class and come back.
- You are encouraged to bring laptops so you can go along with some demonstrations in the class of git/R, but you are not allowed to use laptops to shop/answer personal emails/facebook during the class.
- Violators will be asked to sing in class for 30 seconds. I am serious.

Grading will be based on

- **correctness** (we will use automatic testing)
- **good version control practies** (this week plus next)
- **readability** (code style and quality of comments)
- **speed** (how fast is your implementation compared to others?)

There could be additional criteria depending on assignment.

Personalized grading: Starting with HW1, I will be giving you tests and correct code after HW due date, and ask you to evaluate your work on your own, and explain the grade you will give yourself. Each time, me and TA will sample 2/3 of the HWs to double check.

How to get help?

- eCampus has now discussion boards, you can post **anonymously**
- I will have office hours every Wed
- Yunfeng (TA) will have office hours every Mon

Questions?

Stare at students for 2 min in anticipation

So what is this thing Git and Github?

Git - version control system, allows to monitor changes to code/documents, can be easily integrated with Rstudio

Github - website that provides powerful integration with Git, and allows easy sharing of the code with others. Stores your code remotely, can be used as backup system.

Some helpful resources

- Software Carpentry Intro to Git
- Udacity short guide

What is Git?

- Git is a verson control system - lets you track your progress over time
- Check that you have GIT installed by typing in the terminal/bash

```
git --version
```

```
git version 2.17.2 (Apple Git-113)
```

Why version control?

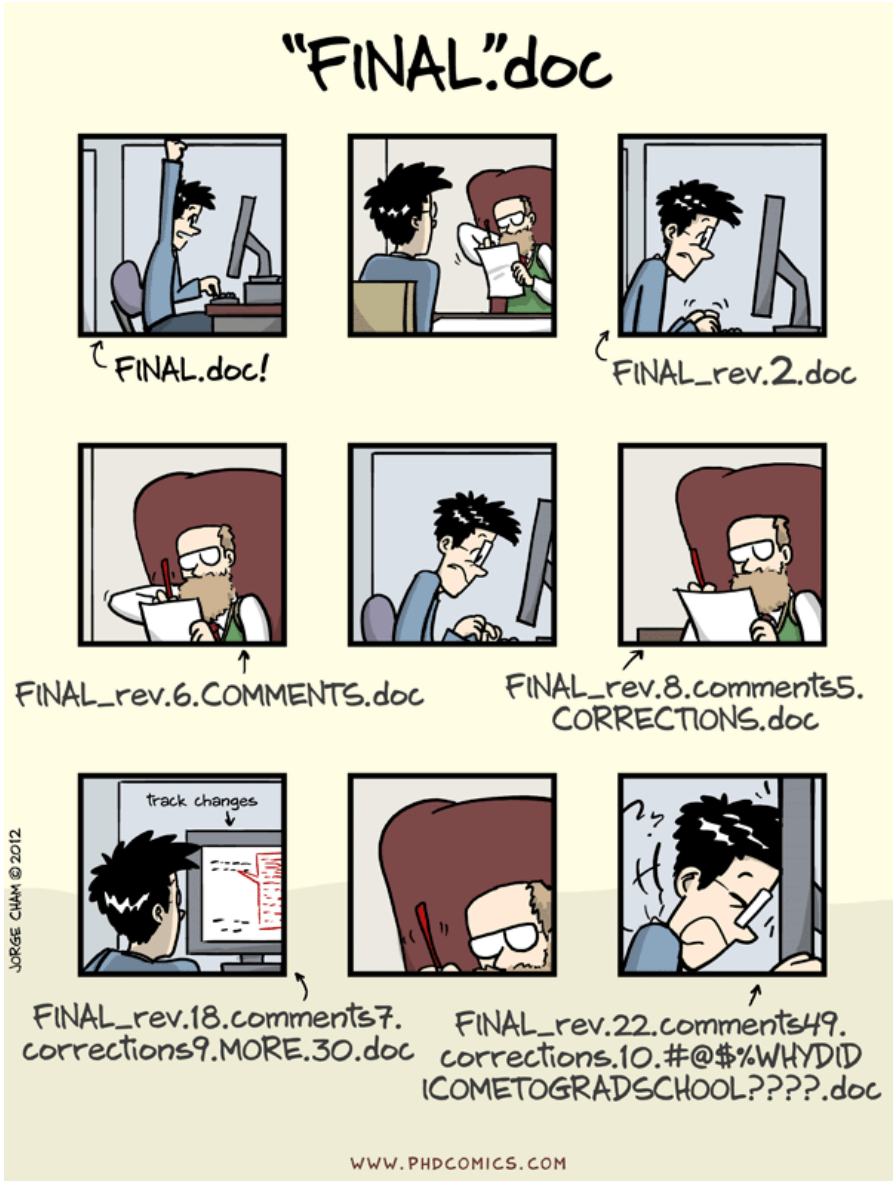

Figure 1: Version Control

Git configuration

- You need to tell Git who you are so that all changes are associated with you
- Type in the terminal

```
git config --global user.name "Irina Gaynanova"
```

```
git config --global user.email "irinag@stat.tamu.edu"
```

- If you work on classroom computers, you may not have rights to do global configuration and will need to do it in concrete repository.

Git philosophy

- One project - one repository (folder)
- To create a new repository, **change to the desired location**, and type

```
git init [Name of repository]
```

- Go into that repository and check what's inside by typing

```
ls -a
```

- To check that everything is good, type

```
git status
```

Version control basics with Git

- We will practice by working on a **readme.md** document
- *.md* indicates markdown syntax - lightweight markup language with plain text formatting, default for readme in Github
- Initialize empty readme

```
touch readme.md
ls -a
```

Edits/staging and commits

- Let's edit readme we just created from terminal using vim (you are welcome to use any other editor, from terminal or otherwise)

```
vim readme.md
```

- Add a sentence of your choice
- Save and exit. In vim - Esc, then :wq.
- Check status of git now

```
git status
```

Edits/staging and commits

- **Untracked** - files in the repo that Git is not keeping track of
- When ready, they need to be added for tracking in git

```
git add readme.md
git status
```

- The readme is tracked now, but not **committed** yet (Git knows it needs to keep track of changes, however it it didn't record these changes yet)
- To record the changes in git

```
git commit -m "Start of readme"
git status
```

Edits/staging and commits

- Git workflow

raw changes -> staged changes (tracking via add) -> committed changes (via commit)

- Each commit is a snapshot of a project version

```
git log
```

- Often staging/commit happen simultaneously, but not always (can you think of a reason?)

Edits/staging and commits

- Suppose we want to make more changes to readme. Add another sentence to it. In vim - Esc, then :wq.

```
vim readme.md
```

- Check status of git now

```
git status
```

- Can we check what are the changes?

```
git diff
```

- Let's commit the changes. What happened?

```
git commit -m "Added a sentence on test"
```

Git with Rstudio

- Working on terminal/bash can be a little painful
- Fortunately Rstudio can recognize Git
- File -> New Project -> Existing Directory -> [Pick the directory you just created]
- Note that you have Git icon appear next to Connections, Environment and History
- If no icon -> check **HappyGitWithR** and **H.Wickham's integration of Git and Rstudio**

Git with Rstudio

- Let's make edits to readme.md within Rstudio now and save
- Go to Git icon -> Here you can check file differences, add files for staging, and commit with the message
- Note new files/changes that happened behind the scenes: .gitignore and .Rproj
- .gitignore allows you to tell git to ignore some of the files completely -> helpful when you have configuration files like .Rproj and .Rhistory

Remote version control

- So far Git was doing version control locally - great if you are the only one working on the project
- Suppose you want to collaborate with someone else, or share your work -> need remote storage of the repository
- **GitHub**, **BitBacket** and **GitLab** are some commonly used hosting services for Git
- We will use **GitHub** in this class

Github

- Show my own account
- Mirrors your local repositories and let's you upload external changes -> tracks the projects
- Icon - **Octocat**

Github flows: From Local to Github

- Associate your directory with Rstudio project if not already
- Make sure the directory is setup with version control if not already

```
git init
```

- Make a new repo on Github (DO NOT INITIALIZE WITH README)
- Find your GitIntro repository from last class
- In the terminal/console

```
git remote add origin [URL path to created repo on GitHub]
git push --set-upstream origin master
```

Github flows: From Local to Github

- Instead of using terminal/console for the last step, you can use Rstudio
- In Rstudio, use Git pane, "two purple boxes and a white square" icon, and *Add remote*.
- Use origin as name, GitHub repo url, then use **master** branch name, click sync box, and then choose Overwrite.

Github flows: From Github to Local

- Pick an existing repository (HW1 is a good one if you accepted it)
- Use git clone to copy it locally to desired location

```
git clone [Repository path]
```

- Create an R project associated with the repository (if not one already)
- You can also use Rstudio directly to clone the project via NewProject->VersionControl->Git and entering the URL of the GitHub repository