

Acknowledgement

- The content in this week's exercises are heavily inspired by
 - RLadies Sydney July 2021 DIY R package workshop by Fonti Kar (UNSW)

Skills

This week we will target the following skills:

1. To gain a basic understanding of Git and Github
2. To apply version control to an R package

Applied skills, exercises, resources.

The required readings for this week are:

- Github for Supporting, Reusing, Contributing, and Failing Safely (<https://www.openscapes.org/blog/2022/05/27/github-illustrated-series/>)
- “R packages”, by Hadley Wickham and Jenny Bryan (<https://r-pkgs.org/index.html>)
 - Section 18
- Happy Git and GitHub for the useR website (<https://happygitwithr.com>)
- As of July 2022, Hadley & Jenny are working on a second edition of this book (their first edition (aka the stable version) is no longer available online) and so some section could look unfinished and it's also entirely possible they could make large, systematic changes to the book during the semester. If the section we referenced here doesn't make sense anymore, please let us know.

1. Using Git and Github

Learning outcomes:

- Deploy your package on Github.

Exploratory activity

We already went through a rather detailed example of putting your own R packages on Github during Topic 06 notes.

If you think you have mastered this, you can attempt Exercise 1.

2. Explore other functionality of Github.

Learning outcomes:

- Explore other functionality on Github
 - issues/bug report.
 - forking & pull request.
 - conflict

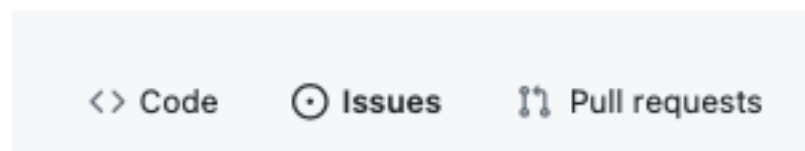
Exploratory activity

Github works the best with friends or in a collaborating setting. The following activities work the best if you can form teams.

1) Issues/bug report.

Issues are a great way to keep track of tasks, enhancements, and bugs for your projects. They're kind of like email – except they can be shared and discussed with the rest of your team. You can read more about it here: <https://docs.github.com/en/issues/tracking-your-work-with-issues/about-issues>.

There is a dedicated section on every repository:



In this activity, head to one of the repositories of your team members and create an issue. You can ask for a bug fix or enhancement (or even have a conversation there). It's also a good idea to test out the repress package but with the `gh` option to see how they all tie in together.

In return, you can respond to your team members' posts in various ways:

- learn how to tag someone with @ and reply with a message;
- assign the task to someone, i.e. yourself in this case;
- close the request (in a typical setting, you will push changes to the repo before closing).

2) Forking & pull request.

After using GitHub for a while, you may find yourself wanting to contribute to someone else's project. This usually happens when you tried to extend someone's work but found some bugs at the same time. You need a copy of someone else's project as the starting point.

You can, of course, just download a copy of the repo from Github, but that workflow does not encourage you to stay up-to-date with the ongoing development. It's better to use Github to manage this, and the process is called **forking**.

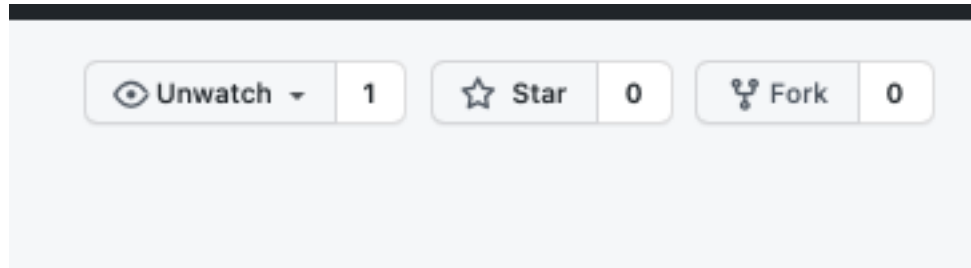
Creating a *fork* is producing a personal copy of someone else's project. Forks act as a sort of bridge between the original repository and your personal copy.

And the process of pushing changes to a repo that you don't own is called a **pull request**. If you think you have a brilliant idea and would benefit the community, this is how you contribute.

You can read more about the process here: <https://docs.github.com/en/get-started/quickstart/contributing-to-projects>.

In this activity, fork one of the repositories of your team members and try to create a pull request. You will have to

- fork the repo



- clone the repo down to your computer;
- make some changes/improvements,
- commit and push them back to Github;
- initiate a pull request on the Github webpage.

Your team member should then be able to review the changes on Github and

- reject the request;
- discuss the changes;
- merge the request to the main branch.

If you are looking for a more bare bone repo for some easy enhancements (create a package/add README, etc) before submitting a pull request, you can use one of mine here: <https://github.com/thomas-fung/shiny-disco>.

3) Conflict

This one is a bit more tricky to create because we are mainly flying solo in this unit. Most of the time, Github will merge changes from multiple users automatically, but sometimes it will need to seek clarification from you.

To artificially create a conflict, you have to

1) invite someone else to work on your project

- go to the Github page of your repo of choice;
- go to **Setting**;
- go to **Manage access**;
- **Invite a collaborator** and follow the instructions there.

2) create a conflict

- your teammate will need to accept the invitation and clone a copy of the repo;

3. Research and find solutions to the following problems

- both you and your teammate will need to make different changes to the same line of code;
- your teammate will commit and push the changes first;
- it's your turn to commit and push the changes, but Github should reject it with an error message. You should see an error message as well if you try to do a pull.

I found using a client to be the easiest way to manage conflicts, but this is beyond the scoops of this unit, so we will just leave it here.

If you want to know more about collaborating with Github, you can read this as well: <https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests>.

3. Research and find solutions to the following problems

Exercise 1 Your task for this exercise is to put the `ohwhaley` package on Github.

You will need to

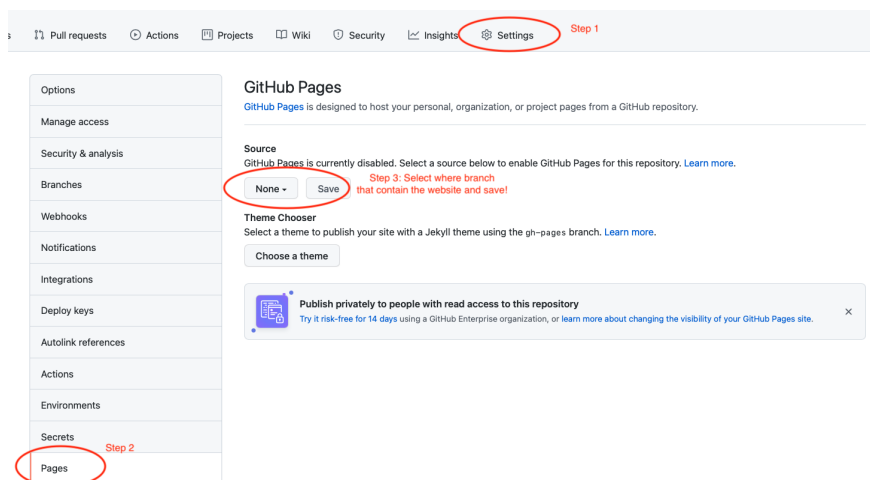
- create a `README` file in `rmd` and then `md` format.
 - a sample file can be downloaded from iLearn.
 - See Section 20.5.1–20.5.2 for more details.
- push the package to Github by using a combination of
 - `usethis::use_git()`;
 - `usethis::use_github()`.
- try to download and install the package from Github (Hint: see `README` for the instruction).
- delete your local copy of the package (Yes, be brave!) and `clone` a copy back to your computer.

Exercise 2 In this exercise, we will add more features to the `ohwhaley` or the `praiseme` package we created earlier.

1. Use the `pkgdown` package to create a new website of your package that will be hosted on Github. You can read more about it here: <https://pkgdown.r-lib.org/index.html>.
 - It takes minimal effort to create a basic website for your package.

```
# Run once to configure package to use pkgdown
usethis::use_pkgdown()
# Run to build the website
pkgdown::build_site()
```

- You will need to push the website to Github.
- Assuming your repo is a public one (**Warning: don't perform this step for your assessments!**), you can then head to Github to activate it via `Settings -> Pages -> Select which branch contains the files`.



2. Use Github Actions.

In the previous exercise, the `pkgdown` package used existing information within your package to create a website. Supposed you made some changes to improve the documentation of your package. This means you will need to run `pkgdown::build_site()` to ensure the information on the webpage will be in sync with the rest of the package before you can release the updated version on Github.

Wouldn't it be nice if the webpage would update itself automatically whenever you have a new commit?

Yes, it would be and that's called **GitHub Actions**. Github Actions makes it easy to automate our workflows.

Besides publishing your `pkgdown` site to Github automatically, another very useful **GitHub Actions** is to set up continuous integration (CI). CI in our context means we will conduct a `check()` after every commit. You may ask shouldn't we run multiple `check()`s before releasing our work on Github? Yes, you should, but **GitHub Actions** can do so much more.

So far, our check is limited to your local machine which could use a particular OS (for instance Windows or MacOS) and a particular version of R. Given R supports multiple platforms, you need to test out other OS as part of the quality assurance (QA) process. You may also want to test your package on some older or the development version of R to ensure most users can use your package properly. This QA process can be demanding if you have to conduct it manually.

Setting up **GitHub Actions** is remarkably easy.

- 1) Just run the following code and commit the changes to Github.

```
# Sets up automatic publishing of your pkgdown site to GitHub pages:
usethis::use_pkgdown_github_pages()
```

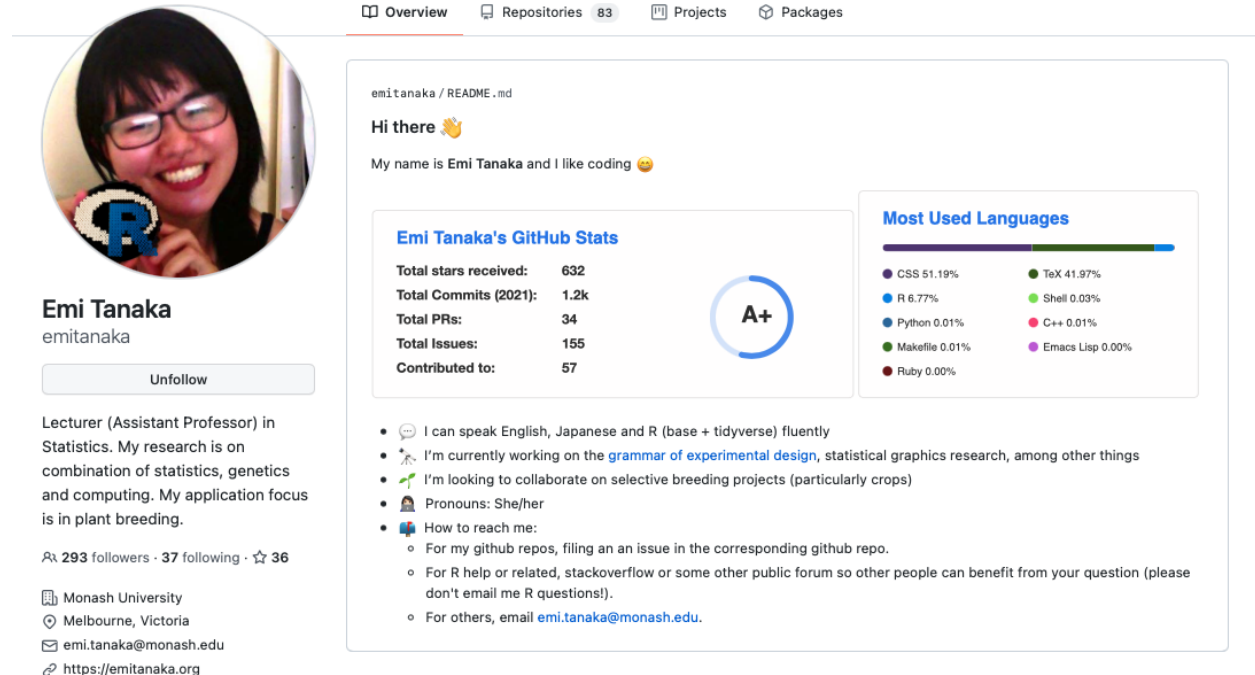
To set up continuous integration (CI) so that it will check after every commit with Github, please refer to Section 19.3 of R-Packages for more details.

- 2) Head to Github webpage and look under the **Actions** tab. Your tasks will be running in the background.
- 3) Feel free to make further changes to your package and experience how **GitHub Actions** would make our life so much easier.

When reading some of the reference materials, you may come across the term **Travis-CI**. **Travis-CI** is a third party CI service that's popular for a while, but that's no longer recommended by the tidyverse team.

3. Research and find solutions to the following problems

Exercise 3 In this exercise, we will attempt to add a README to your profile. It looks something like this:



The screenshot shows a GitHub profile for **Emi Tanaka** (username: `emitanaka`). The profile includes a circular profile picture of a woman with glasses and a blue 'R' logo. The README file content is as follows:

Hi there 🙌

My name is Emi Tanaka and I like coding 🤖

Emi Tanaka's GitHub Stats

Total stars received:	632
Total Commits (2021):	1.2k
Total PRs:	34
Total Issues:	155
Contributed to:	57

Most Used Languages

Language	Percentage
CSS	51.19%
TeX	41.97%
R	6.77%
Shell	0.03%
Python	0.01%
C++	0.01%
Makefile	0.01%
Emacs Lisp	0.00%
Ruby	0.00%

About Emi Tanaka:

- I can speak English, Japanese and R (base + tidyverse) fluently
- I'm currently working on the [grammar of experimental design](#), statistical graphics research, among other things
- I'm looking to collaborate on selective breeding projects (particularly crops)
- Pronouns: She/her
- How to reach me:
 - For my github repos, filing an issue in the corresponding github repo.
 - For R help or related, stackoverflow or some other public forum so other people can benefit from your question (please don't email me R questions!).
 - For others, email emi.tanaka@monash.edu.

Contact Information:

- Monash University
- Melbourne, Victoria
- emi.tanaka@monash.edu
- <https://emitanaka.org>

All you have to do is to create a repository named exactly the same as your GitHub username. You should receive a little message telling you that you've found a secret!

Make sure you make the repo public and also initialise it with a README.

You can then - edit the README online; or - clone the repo down to your local computer first before editing.

Feel free to check out the code used by Emi here: <https://github.com/emitanaka/emitanaka> or this profile by Allison Horst: <https://github.com/allisonhorst>.