EEB603 – Chapter 11: Version Control for Projects

Ann Wozniak, Annie Taylor, and Phebian Odufuwa

2020-11-09

Contents

Day Two

- **Application** *Using Git and GitBash*

Outline

The following are what will be covered in this tutorial:

- Making your research reproducible using GitHub
- Create a Repository on GitHub
- Cloning your repository from GitHub environment into your local computer
- Writing your code in R **(note that as example, the presenters have provided all data needed)**
- Collaborating on project
- Merging codes from different collaborators into one
- Cleaning up your GitHub repository
- Creating a README.md file on GitHub
- Commiting your document as a GitHub page

Introduction

The fundamental idea behind a robust, reproducible analysis is **a clean, repeatable, script-based workflow (i.e. the sequence of tasks from the start to the end of a project), linking raw data through to clean data and to final analysis outputs**. Working on GitHub will ensure that your research is fully reproducible for collaborators, and for the public.

In our previous tutorial for Git version control software, we learned the essential basic commands for using git, as well as how to work with GitHub to establish a repository and push our project to the website; and we also looked at a model GitHub page.

Now it is time to start working with GitHub using git: making changes in the project safely off to one side, and merging them back into the original project once they have proved to be correct, or at least not disastrous.

Aim of tutorial and learning outcomes

This tutorial will teach you how to implement a version control protocol (with Git). By the end of this tutorial, you will be able to:

- Clone a repository from GitHub to RStudio
- Pull an existing repository from GitHub to RStudio
- Commit changes made to 'branch' or 'main/master' to GitHub
- Merge branches, and clean duplicated branches
- Best practices on GitHub and collaborating on projects

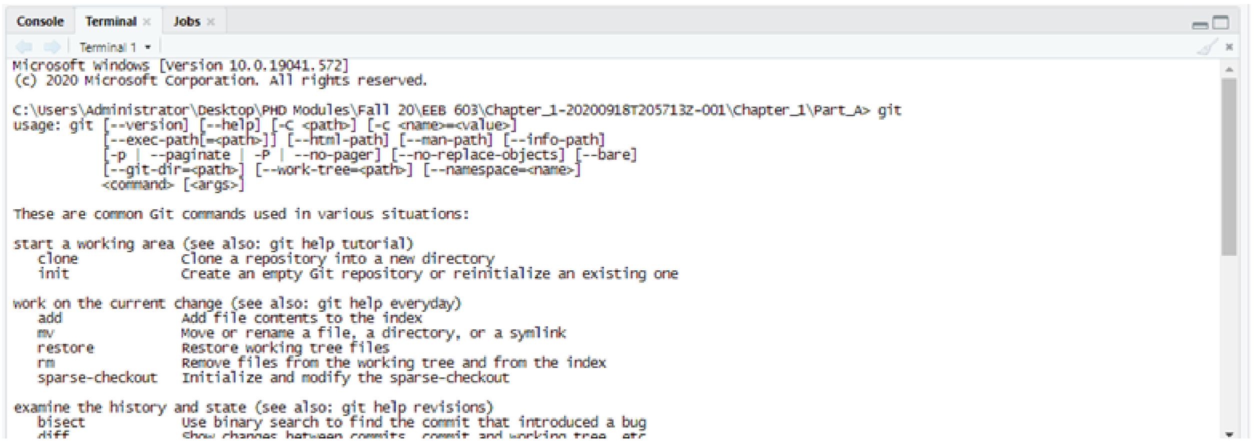
Getting Started: Linking GitHub and RStudio

Step 1

Before we start, you need to check if git is installed on your windows machine, for mac, it is already installed, but for windows, no! You should have installed it, but there is no harm in checking, so from RStudio, click on Terminal, and type:

```
git
```

if git is already installed, you will see a window that looks like this:



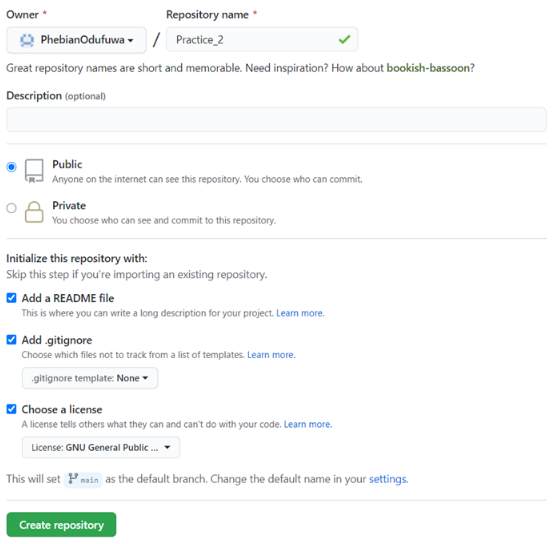
```
Microsoft Windows [Version 10.0.19041.572]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Administrator\Desktop\PHD Modules\Fall 20\EEB 603\Chapter_1-20200918T205713Z-001\Chapter_1\Part_A> git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone             Clone a repository into a new directory
   init              Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add               Add file contents to the index
   mv                Move or rename a file, a directory, or a symlink
   restore           Restore working tree files
   rm                Remove files from the working tree and from the index
   sparse-checkout   Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
   bisect            Use binary search to find the commit that introduced a bug
   diff              Show changes between commits, commit and working tree, etc
```

Figure 1: Screen capture to show that git is installed on your machine

and you are all set! But if it isn't, you will receive this error message **('git' is not recognized as an internal or external command, operable program or batch file)**. To install git on your computer, copy this link https://git-scm.com/downloads into your browser, and click on "download git for windows", install git, repeat the same process as above, by opening RStudio and typing the command 'git' into your Terminal, it should now be installed.

Step 2

Sign into your account on GitHub https://github.com/login, and create a new repository, name it ("practice_2_**yourinitials**").



Owner *
PhebianOdufuwa / Repository name * Practice_2

Great repository names are short and memorable. Need inspiration? How about bookish-bassoon?

Description (optional)

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☑ Add a README file
This is where you can write a long description for your project. Learn more.

☑ Add .gitignore
Choose which files not to track from a list of templates. Learn more.
.gitignore template: None

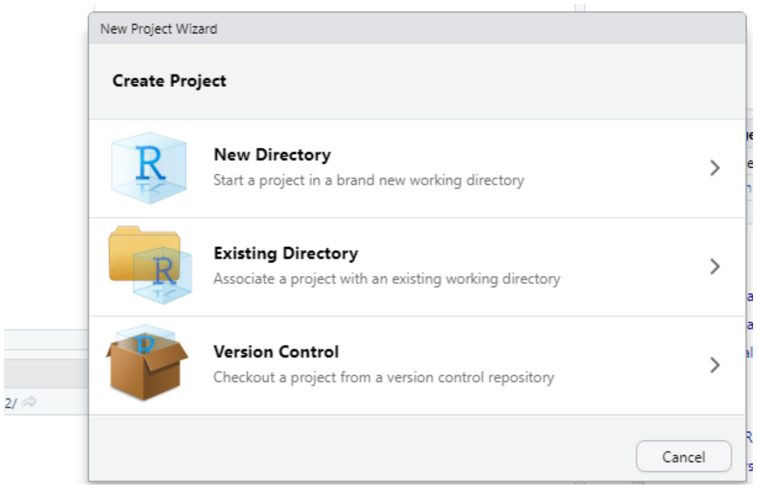
☑ Choose a license
A license tells others what they can and can't do with your code. Learn more.
License: GNU General Public ...

This will set main as the default branch. Change the default name in your settings.

Figure 2: Screen capture showing how to create a repository on GitHub

Step 3

Copy the URL of the repository, open RStudio and click on **'file'**, click on **'New Project'**, from the drop-down menu, click on **'version control'** and select **'Git'**.



New Project Wizard

Create Project

New Directory
Start a project in a brand new working directory

Existing Directory
Associate a project with an existing working directory

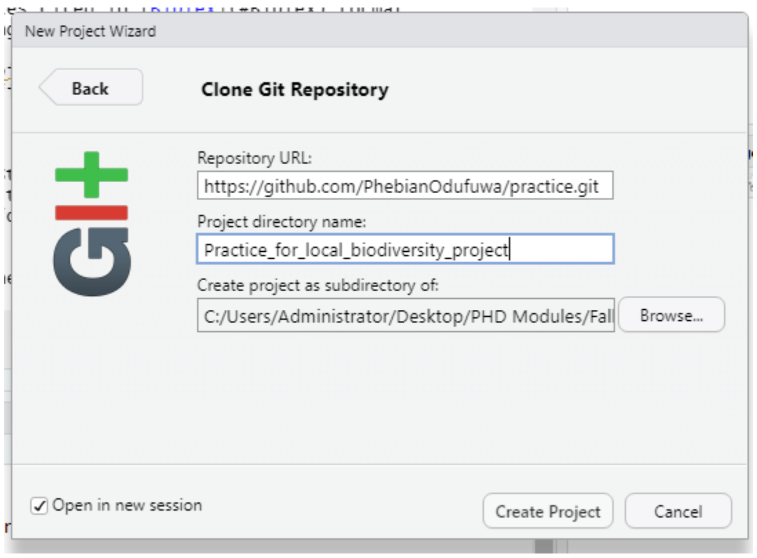
Version Control
Checkout a project from a version control repository

Cancel

Figure 3: Screen capture showing how to create version control on RStudio

Step 4

Paste the copied url, in the **'project directory name'** box, type in 'Practice_for_local_biodiversity_project_**yourinitials**', select **'open in new session'** and click on **'create project'** (see screenshot below)



New Project Wizard

Back

Clone Git Repository

Repository URL:
https://github.com/PhebianOdufuwa/practice.git

Project directory name:
Practice_for_local_biodiversity_project

Create project as subdirectory of:
C:/Users/Administrator/Desktop/PHD Modules/Fall

Browse...

☑ Open in new session

Create Project

Cancel

Figure 4: Screen capture showing how to create a GitHub project on RStudio

You should now have your repository cloned into your local computer!

Project: an example

Preamble:

As an example for this tutorial, we are using the code written for Dr Leonora Bittleston's Advance Ecology class this semester, full permission has been taken from the course instructor, and we are permitted to use this code.

Note that the instruction for this tutorial is in the "README.md" file, but we will be doing everything hands-on in class.

See this in a practical light thus: ("you have just completed your project on"Idaho's Local Biodiversity", you have written your code, you have your chapter written out, and you want to make it available on GitHub for your collaborators, and for the public). Let's get on with this!

Exercise

Instruction:

1. We will be doing this exercise by opening a new R Script, save the 'R Script' into the Folder where you have all your Docs stored (that is, the folder you cloned from GitHub), save your 'R script' with the name 'local_biodiv_practice_**yourinitials**'.

2. From our Google Drive entitled **'GitTutorial_2'**, please copy the folders with the names **'Figures'**, **'Data'**, and **'Documentation.html'** and paste in your own directory where you will be working from.

3. Open our R Markdown file entitled **"Chapter_11_part_2.Rmd"**

4. Copy the code in the code chunk below, and paste in your R script.

5. Run your code to make sure that it is working perfectly

```r
# R code for local biodiversity project - an example
## Load the necessary packages
if (!require("ggplot2")) {
    install.packages("ggplot2")
    require("ggplot2")
}
if (!require("vegan")) {
    install.packages("vegan")
    require("vegan")
}
if (!require("gridExtra")) {
    install.packages("gridExtra")
    require("gridExtra")
}
library(vegan)
library(ggplot2)
library(gridExtra)
## Set your working directory setwd to source File Location
## Read in your data
dat <- read.csv("Data/Test_data_local_biodiv.csv", head = T,
    row.names = 1)
### View table
dat
```

```r
### Look at row and column sums
summary(rowSums(dat))
summary(colSums(dat))
### Most analyses will want data transposed (rows as 'sites'
### and columns as 'species')
datt <- t(dat)
#### Species-individual curve (optional)
dat.specaccum <- specaccum(datt, method = "rarefaction")
plot(dat.specaccum)
#### Measures of alpha diversity
dat.specnumber <- specnumber(datt)  ## Number of species
dat.rowsums <- rowSums(datt != 0)  ## Number of non-zero elements in each row.
## Note that this is the same as above.
dat.shannon <- diversity(datt)  ## default is Shannon diversity
dat.ens <- exp(dat.shannon)  ## Effective number of species
### Combine into one table for easy graphing
dat.alpha <- cbind.data.frame(Quadrat = c(1:6), dat.specnumber,
    dat.shannon, dat.ens)
### This is an example of just plotting in basic R
plot(dat.alpha$Quadrat, dat.alpha$dat.specnumber)
### Make graphs in ggplot2
plot1 <- ggplot(data = dat.alpha, aes(x = Quadrat, y = dat.specnumber)) +
    geom_point()
plot2 <- ggplot(data = dat.alpha, aes(x = Quadrat, y = dat.shannon)) +
    geom_point()
plot3 <- ggplot(data = dat.alpha, aes(x = Quadrat, y = dat.ens)) +
    geom_point()
### plot the 3 graphs next to each other
grid.arrange(plot1, plot2, plot3, ncol = 3)
## why do we see that a quadrant with the 2nd lowest species
## number has the second highest effective number of species?
### Do not answer the question, as the solutions have been
### provided in the documentation see folder entitled
### 'Documentation.html'.
```

Push code to GitHub

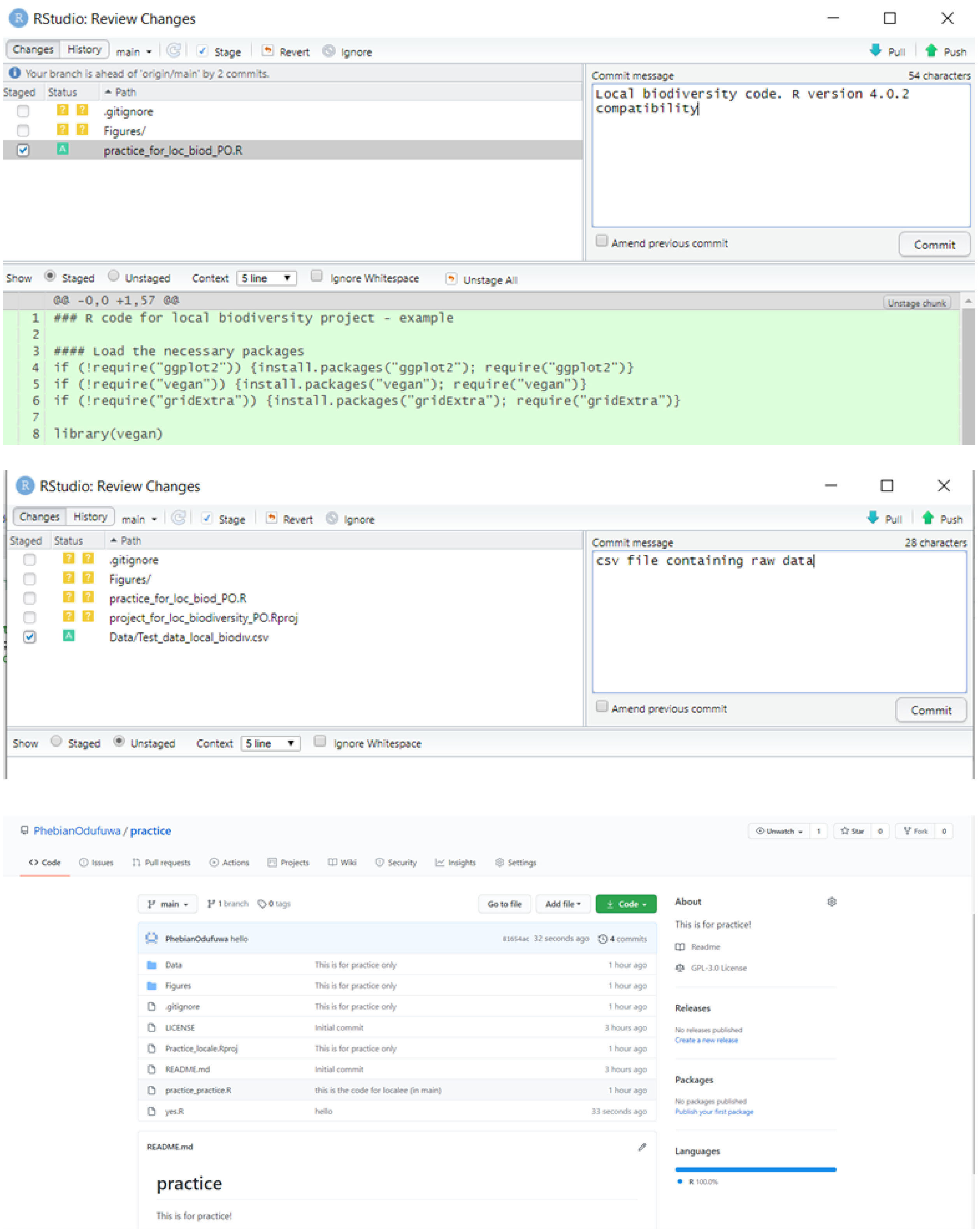
Now that you have your code written out, you should now push your code to GitHub from your local computer

How to do that:

- To push your code to GitHub from your local computer, you need to:

1. Click on ‘Tools’,
2. Click on ‘Version Control’,
3. Click on ‘Commit’,
4. From the pop-up window, select what you want to add to GitHub, it is good practice to add your files individually, this will enable you to comment on each individual file. This is to ensure reproducibility, so that whoever sees those files on GitHub at first glance will be able to understand what each file does.

So click on each individual file and type out a comment (individually), you have to do this in a succession. See the screenshots below:



```
Local biodiversity code. R version 4.0.2
compatibility
```

```
@@ -0,0 +1,57 @@
### R code for local biodiversity project - example

#### Load the necessary packages
if (!require("ggplot2")) {install.packages("ggplot2"); require("ggplot2")}
if (!require("vegan")) {install.packages("vegan"); require("vegan")}
if (!require("gridExtra")) {install.packages("gridExtra"); require("gridExtra")}

library(vegan)
```

```
csv file containing raw data
```

PhebianOdufuwa hello 81654ac 32 seconds ago 4 commits

Data	This is for practice only	1 hour ago
Figures	This is for practice only	1 hour ago
.gitignore	This is for practice only	1 hour ago
LICENSE	Initial commit	3 hours ago
Practice_locale.Rproj	This is for practice only	1 hour ago
README.md	Initial commit	3 hours ago
practice_practice.R	this is the code for localee (in main)	1 hour ago
yes.R	hello	33 seconds ago

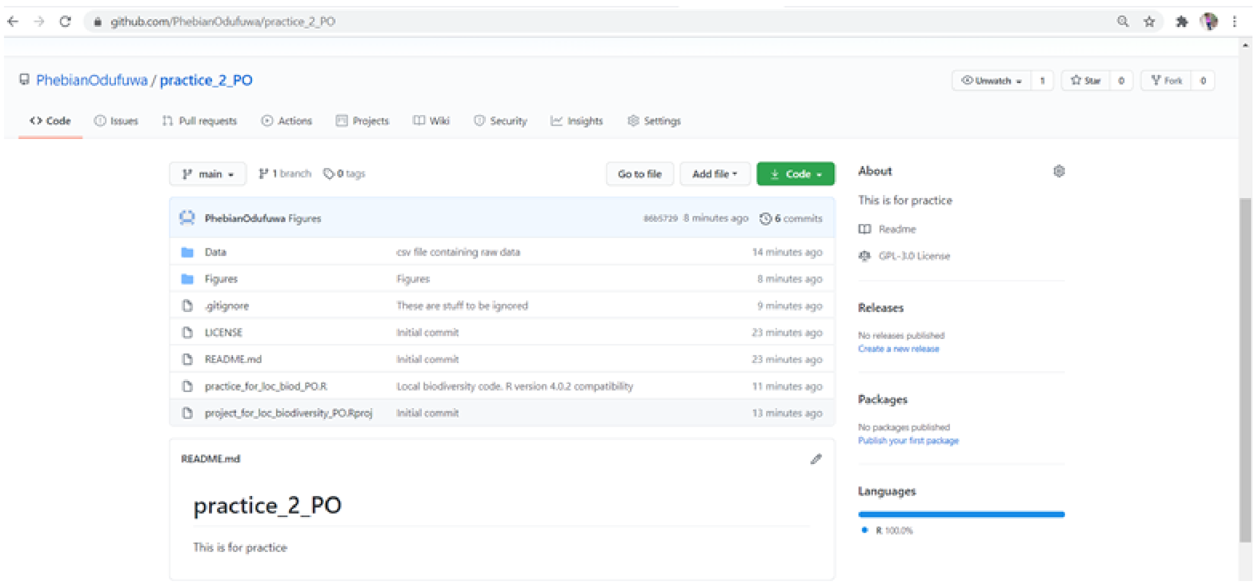
README.md

practice

This is for practice!

Figure 5: Screen capture showing files with comment

5. Click on ‘Commit’ each time
6. Once you have finished committing all changes, close the window
7. Click on ‘Push’ to push ALL (the commits you have made) to GitHub
8. Now check your GitHub page, refresh your GitHub page, and you will see that your repository has been populated with your new branch.



PhebianOdufuwa Figures 86b5729 8 minutes ago 6 commits

Data	csv file containing raw data	14 minutes ago
Figures	Figures	8 minutes ago
.gitignore	These are stuff to be ignored	9 minutes ago
LICENSE	Initial commit	23 minutes ago
README.md	Initial commit	23 minutes ago
practice_for_loc_biod_PO.R	Local biodiversity code. R version 4.0.2 compatibility	11 minutes ago
project_for_loc_biodiversity_PO.Rproj	Initial commit	13 minutes ago

README.md

practice_2_PO

This is for practice

Figure 6: Screen capture showing branch committed to GitHub

You are now done with your project. Close R Studio in preparation for the next stage!

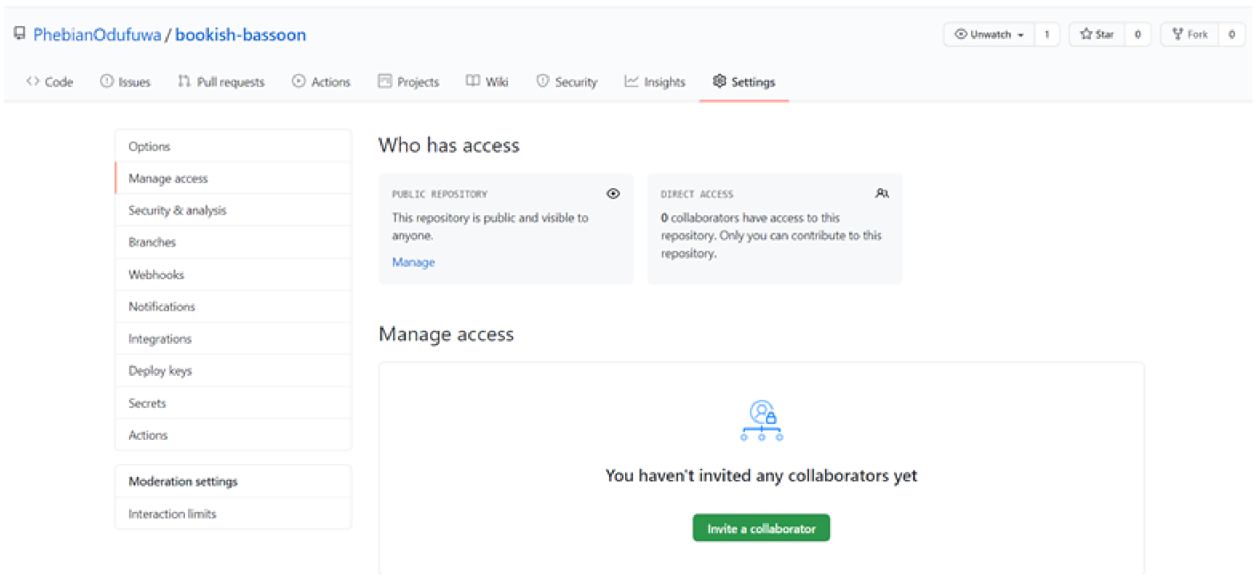
In the next stage, you will be working as a collaborator, and not on your project anymore.

Collaborations and making changes to code

For collaborations, the first step is to give access to collaborators using either their username or email address. To do this;

1. Go to your GitHub repository, click on your repository (i.e. the repository you need collaboration on)
2. Go to **‘settings’**, click on **‘manage access’**, GitHub will prompt you for your login password, once you enter your password, click on **‘Invite a collaborator’**
3. Enter the details of your collaborator, and send the invite
4. The collaborator will be nudged via email; once the collaborator accepts your invite to collaborate on the repository, you can now collaborate on the repository (or project).

See the figure below:



This repository is public and visible to anyone.

0 collaborators have access to this repository. Only you can contribute to this repository.

You haven't invited any collaborators yet

Figure 7: Screen capture showing how to add collaborators on GitHub

(In your own case, enter the username of your group collaborator as provided in the file entitled **'GitTutorial_2_Collaborations Group'** on our Google drive, and grant them access to your repository)

To make changes to an existing code as a collaborator, it is best to follow these steps:

1. Clone the repository into your local computer by doing the following:

- Copy the URL of the repository you are collaborating on.
- Open R Studio, and click on 'Terminal',
- For the purpose of this exercise, we will be cloning our repository to Desktop; to change out of your previous directory, using the terminal, do;

```
cd ~/Desktop
```

- Clone the repository to Desktop by doing;

```
# paste the url by right clicking and pressing 'paste'
git clone <paste the repository url>
```

- By typing

```
ls
```

you should now be able to see your cloned repository on your Desktop.

Now that you have the repository cloned into your local computer, you are now ready to make changes to the code as a collaborator.

Best Practice for effective collaboration:

By now you understand that git saves each version of your project as a snapshot of the code exactly as it was at the moment you committed it. Essentially creating a timeline of versions of a project as it progresses, so that you can roll back to an earlier version in the event disaster strikes.

The way git, and GitHub, manage this timeline — especially when more than one person is working in the project and making changes — is by using branches. A branch is essentially a unique set of code changes with a unique name. Each repository can have one or more branches. The main branch — the one where all changes eventually get merged back into, and is called master. This is the official working version of your project, and the one you see when you visit the project repository at **github.com/yourname/projectname**.

Do not mess with the master. If you make changes to the master branch of a group project while other people are also working on it, this could spell, well in one word - doom! In other words, do not mess with the master!

Instead, everyone uses branches created from master to experiment, make edits and additions and changes, before eventually rolling that branch back into the master once they have been approved and are known to work. Master then is updated to contain all the new stuff.

Note: A well commented code is very important to you as a collaborator, and to others who will want to use the code. Therefore, for all the commits you will be making, ensure that they are properly commented.

2. To begin working on anything new in a project, or to change existing things, you create a branch off the stable master branch.

(a) you need to change to the newly cloned repository

```
cd practice_2_yourcollaboratorinitials
```

(b) Prior to creating new branches, we want to check for any other existing branches. We know about the master, but who knows what other project collaborators may be up to? We can view all existing branches by typing

```
git branch -a
```

into terminal, which tells git that we want to see ALL the branches in this project, even ones that are not in our local workspace.

(c) To create this new branch type "git checkout -b branchNameHere" (in our case, we will call the branch name "betabiodiv_**yourinitials**")

```
git checkout -b betabiodiv_yourinitials
```

This command creates a new branch and changes to the new branch automatically. You can now see that you are no longer on the main or master, you are now on the branch you created

Note: To see the log of ALL that you are doing, type

```
git log --oneline
```

This opens the log of your file, and shows you where you are currently at (not that this isn't obvious, but just to make double sure that you are on the branch you created and not on the main file!)

(d) Open the code to be made changes to on visual studio (visual studio code is a text editor that allows you to easily make changes to code, although it can be used for much more, but that will not be discussed in this class).

To open, visual studio code, type

```
code .
```

Make changes to code as a Collaborator

The repository you want to work on is now opened in visual studio code from the new branch you created, and you are now ready to make changes to code.

Instruction:

Step 1: Click on the script you want to work on

Step 2: Copy the function we already created (from our R markdown file entitled **"Chapter_11_part2.Rmd"**), and paste at the bottom of previous code.

Hypothetically speaking, you as a collaborator feels there is a need to measure the beta diversity for the project you are collaborating on, so, you are updating the code with this function:

```
#### Measures of beta diversity NMDS First using Bray-Curtis
## set a seed to make the results reproducible
set.seed(2)
dat.bc.nmds <- metaMDS(datt, k = 2, trymax = 100)  ### Bray-Curtis is the default metric,
### k = 2 dimensions
ordiplot(dat.bc.nmds, type = "t", display = "sites")  ## Plot that shows names
stressplot(dat.bc.nmds)
## optional, this just shows how your dissimilarity fits with
## ordination distance
## Now using the binary version of Jaccard
dat.jb.nmds <- metaMDS(datt, k = 2, trymax = 100, distance = "jaccard",
    binary = T)
ordiplot(dat.jb.nmds, type = "t", display = "sites")
## you can use cex = 0.6 to make text smaller
## Why do we see quadrants collapsing into each other?
stressplot(dat.jb.nmds)
## optional, this just shows how your dissimilarity fits with
## ordination distance
```

Step 3: Save the changes you made, by typing 'ctrl+s'

Step 4: Navigate to R Terminal, commit changes, and upload to GitHub by doing:

- Save the change you made to the code, by typing:

```
git add local_biodiv_practice_yourcollaboratorinitials.R
```

- Commit changes, by typing the command: git commit -m "comment (in branchName)", thus:

```
git commit -m "yourinitials adding this code to the initial code to measure beta biodiversity (in betabi
```

- Push your branch “betabiodiv_**yourinitials**” with the edited code to GitHub by typing

```
git push --set-upstream origin betabiodiv_yourinitials
```

- Go to your GitHub page, and click on your own repository (that is the repository name with **yourinitials**), you can now see that the branch (created by your collaborator) has been added.

Merging branches to the main repository

Remember that you are now working from your repository

We will now look at how to merge branches into the main repository. To merge the two separate branches as one, you will go back to your R terminal and follow these steps:

Step 1: From your project repository, bring in the changes and test.

```
git fetch origin
```

```
git checkout -b betabiodiv_yourcollaboratorinitials origin/betabiodiv_yourcollaboratorinitials
```

```
git merge main
```

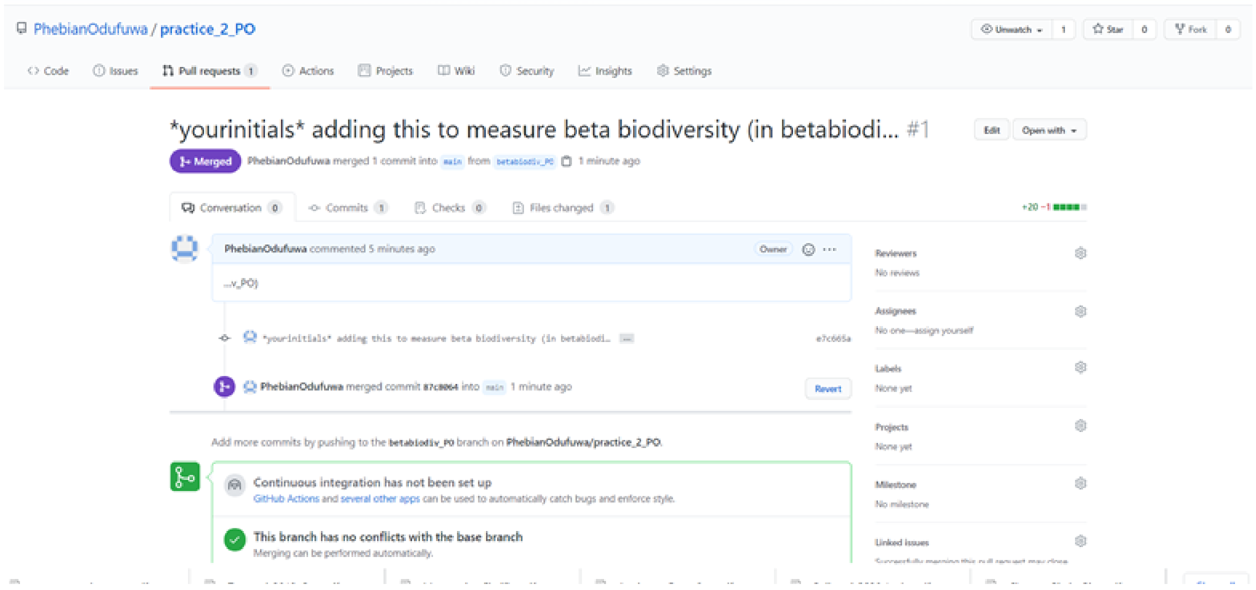
Step 2: Merge the changes and update on GitHub.

```
git checkout main
```

```
git merge --no-ff betabiodiv_yourcollaboratorinitials
```

```
git push origin main
```

Step 3: Go to GitHub, you should now see that the changes made by your collaborator have now been merged to your main branch. See screenshot below:



yourinitials adding this to measure beta biodiversity (in betabiodi... #1

PhebianOdufuwa merged 1 commit into main from betabiodiv_PO 1 minute ago

PhebianOdufuwa commented 5 minutes ago

...v_PO)

yourinitials adding this to measure beta biodiversity (in betabiodi... e7c665a

PhebianOdufuwa merged commit 87c8064 into main 1 minute ago

Add more commits by pushing to the betabiodiv_PO branch on PhebianOdufuwa/practice_2_PO.

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch
Merging can be performed automatically.

No reviews
No one—assign yourself
None yet
None yet
No milestone

Figure 8: Screen capture showing merges

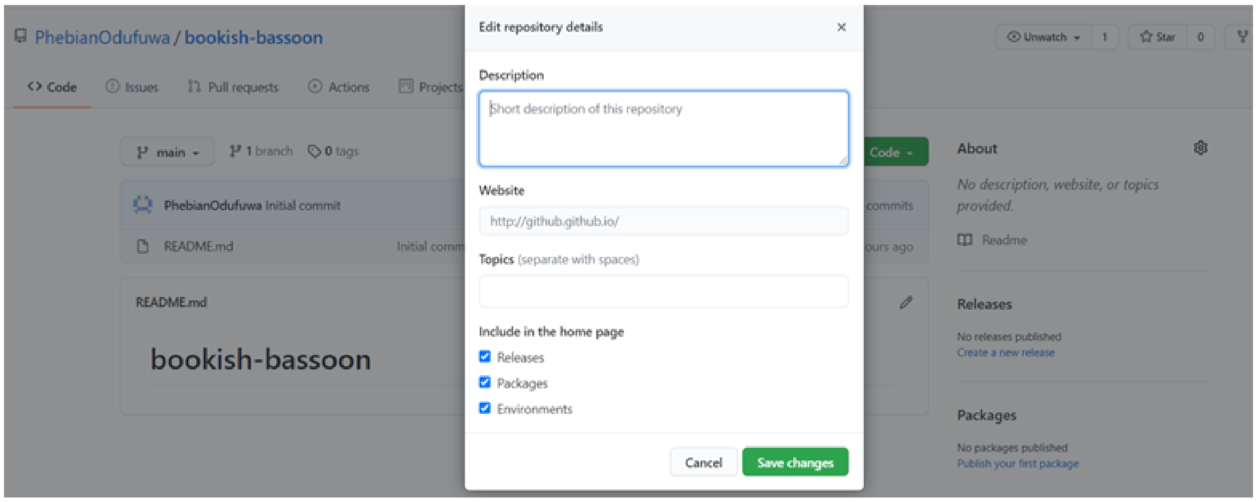
Cleaning up data

Once you have merged your branches, the next thing is to delete extra branch(es). To delete extraneous branch(es) from GitHub,

- From main repository, click on **'branch'**
- Select the branch you want to delete
- Click on the **delete icon** in front of the selected branch
- Navigate to **'Main'** and you can now see that the additional branch has been deleted.

Organizing your data

It is important that your description is well laid out. So write a description that gives an overview of your project. By scrolling to **"About"** and clicking on the **cogwheel symbol** on GitHub, type in a concise description. See the screenshot below



☑ Releases
☑ Packages
☑ Environments

Figure 9: Screen capture showing how to add description to your project on GitHub

Adding README and wiki

README files are a quick and simple way for other users to learn more about your work. Wikis on GitHub help you present in-depth information about your project in a useful way. To find out more about wiki https://docs.github.com/en/free-pro-team@latest/github/building-a-strong-community/adding-or-editing-wiki-pages

Adding README and wiki follows the same process as above: see our README and our wiki pages on https://github.com/PhebianOdufuwa/GitTutorial_2

Commiting your document as a GitHub page

GitHub Pages are public webpages hosted and easily published through GitHub. The quickest way to get up and running is by using the Jekyll Theme Chooser to load a pre-made theme. You can then modify your GitHub Pages' content and style remotely via the web. See https://guides.github.com/features/pages for more information.

For this class, we will be using the “Documentation.html” which you have committed to GitHub, as an example.

- Click on Settings
- Scroll down to the ‘GitHub Pages’ tab and follow the instruction. We will do this together in class.

References

[1] Gandrud, C.. 2015. Reproducible research with R and RStudio. 2nd ed. CRC Press, Taylor & Francis Group, Boca Raton.Anon.

[2] Anon. git-help Documentation. Git. Available at: https://git-scm.com/docs/git-help [Accessed November 4, 2020].

[3] Anon. Hello World. Hello World · GitHub Guides. Available at: https://guides.github.com/activities/hello-world/#commit [Accessed November 4, 2020].

[4] Jenny Bryan, the S.T.A.T.545 T.A.. Happy Git and GitHub for the useR. A The shell. Available at: https://happygitwithr.com/shell.html#shell [Accessed November 4, 2020].

[5] Jenny Bryan, the S.T.A.T.545 T.A.. Happy Git and GitHub for the useR. Chapter 9 Connect to GitHub. Available at: https://happygitwithr.com/push-pull-github.html#push-pull-github [Accessed November 4, 2020].

[6] Jenny Bryan, the S.T.A.T.545 T.A.. Happy Git and GitHub for the useR. Chapter 7 Introduce yourself to Git. Available at: https://happygitwithr.com/hello-git.html [Accessed November 4, 2020].

[7] Anon. SSH Keys for GitHub. Available at: https://jdblischak.github.io/2014-09-18-chicago/novice/git/05-sshkeys.html [Accessed November 4, 2020].

Appendix 1

Citations of all R packages used to generate this report.

[1] J. Allaire, Y. Xie, J. McPherson, et al. *rmarkdown: Dynamic Documents for R*. R package version 2.5. 2020. <URL: https://github.com/rstudio/rmarkdown>.

[2] B. Auguie. *gridExtra: Miscellaneous Functions for "Grid" Graphics*. R package version 2.3. 2017. <URL: https://CRAN.R-project.org/package=gridExtra>.

[3] C. Boettiger. *knitcitations: Citations for Knitr Markdown Files*. R package version 1.0.10. 2019. <URL: https://github.com/cboettig/knitcitations>.

[4] J. Cheng, C. Sievert, W. Chang, et al. *htmltools: Tools for HTML*. R package version 0.5.0. 2020. <URL: https://github.com/rstudio/htmltools>.

[5] I. Lyttle. *vembedr: Embed Video in HTML*. R package version 0.1.4. 2020. <URL: https://github.com/ijlyttle/vembedr>.

[6] J. Oksanen, F. G. Blanchet, M. Friendly, et al. *vegan: Community Ecology Package*. R package version 2.5-6. 2019. <URL: https://CRAN.R-project.org/package=vegan>.

[7] Y. Qiu. *prettydoc: Creating Pretty Documents from R Markdown*. R package version 0.4.0. 2020. <URL: https://github.com/yixuan/prettydoc>.

[8] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2020. <URL: https://www.R-project.org/>.

[9] D. Sarkar. *Lattice: Multivariate Data Visualization with R*. ISBN 978-0-387-75968-5. New York: Springer, 2008. <URL: http://lmdvr.r-forge.r-project.org>.

[10] D. Sarkar. *lattice: Trellis Graphics for R*. R package version 0.20-41. 2020. <URL: http://lattice.r-forge.r-project.org/>.

[11] G. L. Simpson. *permute: Functions for Generating Restricted Permutations of Data*. R package version 0.9-5. 2019. <URL: https://github.com/gavinsimpson/permute>.

[12] K. Ushey, J. McPherson, J. Cheng, et al. *packrat: A Dependency Management System for Projects and their R Package Dependencies*. R package version 0.5.0. 2018. <URL: https://github.com/rstudio/packrat/>.

[13] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN: 978-3-319-24277-4. <URL: https://ggplot2.tidyverse.org>.

[14] H. Wickham and J. Bryan. *usethis: Automate Package and Project Setup*. R package version 1.6.3. 2020. <URL: https://CRAN.R-project.org/package=usethis>.

[15] H. Wickham, W. Chang, L. Henry, et al. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. R package version 3.3.2. 2020. <URL: https://CRAN.R-project.org/package=ggplot2>.

[16] H. Wickham, J. Hester, and W. Chang. *devtools: Tools to Make Developing R Packages Easier*. R package version 2.3.2. 2020. <URL: https://CRAN.R-project.org/package=devtools>.

[17] Y. Xie. *bookdown: Authoring Books and Technical Documents with R Markdown*. ISBN 978-1138700109. Boca Raton, Florida: Chapman and Hall/CRC, 2016. <URL: https://github.com/rstudio/bookdown>.

[18] Y. Xie. *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.20. 2020. <URL: https://github.com/rstudio/bookdown>.

[19] Y. Xie. *Dynamic Documents with R and knitr*. 2nd. ISBN 978-1498716963. Boca Raton, Florida: Chapman and Hall/CRC, 2015. <URL: https://yihui.org/knitr/>.

[20] Y. Xie. *formatR: Format R Code Automatically*. R package version 1.7. 2019. <URL: https://github.com/yihui/formatR>.

[21] Y. Xie. “knitr: A Comprehensive Tool for Reproducible Research in R”. In: *Implementing Reproducible Computational Research*. Ed. by V. Stodden, F. Leisch and R. D. Peng. ISBN 978-1466561595. Chapman and Hall/CRC, 2014. <URL: http://www.crcpress.com/product/isbn/9781466561595>.

[22] Y. Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.30. 2020. <URL: https://yihui.org/knitr/>.

[23] Y. Xie, J. Allaire, and G. Grolemund. *R Markdown: The Definitive Guide*. ISBN 9781138359338. Boca Raton, Florida: Chapman and Hall/CRC, 2018. <URL: https://bookdown.org/yihui/rmarkdown>.

[24] Y. Xie, C. Dervieux, and E. Riederer. *R Markdown Cookbook*. ISBN 9780367563837. Boca Raton, Florida: Chapman and Hall/CRC, 2020. <URL: https://bookdown.org/yihui/rmarkdown-cookbook>.

Appendix 2

Version information about R, the operating system (OS) and attached or R loaded packages. This appendix was generated using `sessionInfo()`.

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19041)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
```

```
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] gridExtra_2.3       vegan_2.5-6         lattice_0.20-41
##  [4] permute_0.9-5       ggplot2_3.3.2       packrat_0.5.0
##  [7] htmltools_0.5.0     prettydoc_0.4.0     vembedr_0.1.4
## [10] devtools_2.3.2      usethis_1.6.3       formatR_1.7
## [13] knitcitations_1.0.10 bookdown_0.20       rmarkdown_2.5
## [16] knitr_1.30
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.5        lubridate_1.7.9   prettyunits_1.1.1 ps_1.3.4
##  [5] assertthat_0.2.1  rprojroot_1.3-2   digest_0.6.25     R6_2.4.1
##  [9] plyr_1.8.6        backports_1.1.10  evaluate_0.14     httr_1.4.2
## [13] pillar_1.4.6      rlang_0.4.7       callr_3.4.4       Matrix_1.2-18
## [17] splines_4.0.2     desc_1.2.0        RefManageR_1.2.12 stringr_1.4.0
## [21] munsell_0.5.0     compiler_4.0.2    xfun_0.19         pkgconfig_2.0.3
## [25] pkgbuild_1.1.0    mgcv_1.8-31       tidyselect_1.1.0  tibble_3.0.3
## [29] codetools_0.2-16  fansi_0.4.1       crayon_1.3.4      dplyr_1.0.2
## [33] withr_2.3.0       MASS_7.3-51.6     grid_4.0.2        nlme_3.1-148
## [37] jsonlite_1.7.1    gtable_0.3.0      lifecycle_0.2.0   magrittr_1.5
## [41] scales_1.1.1      bibtex_0.4.2.3    cli_2.0.2         stringi_1.5.3
## [45] fs_1.5.0          remotes_2.2.0     testthat_2.3.2    xml2_1.3.2
## [49] ellipsis_0.3.1    vctrs_0.3.4       generics_0.0.2    tools_4.0.2
## [53] glue_1.4.2        purrr_0.3.4       processx_3.4.4    pkgload_1.1.0
## [57] parallel_4.0.2    yaml_2.2.1        colorspace_1.4-1  cluster_2.1.0
## [61] sessioninfo_1.1.1 memoise_1.1.0
```