Introduction to Collaborative Coding with GitHub

These practical exercises build on your existing knowledge on GitHub for collaborative coding by using universal Git commands in your system's terminal – a way to perform actions more quickly and use Git from any device or setup, because the terminal doesn't depend on RStudio. You will use the same local repository cloned during the lesson, originally forked from LMU OSC's Collaborative-RStudio-GitHub repository.

Using the terminal

- Every operating system uses a different terminal.

Windows – Git is used through Git Bash, which installs automatically with Git for Windows.
To find it, search for 'Git Bash' in the Start Menu.

MacOS – Git is used through the built-in Terminal app, which you can find by searching for 'Terminal'.

Linux – Git is used through the system's terminal. The name depends on which version of Linux you're using (for example, GNOME Terminal or Konsole), but all work the same for Git commands.

The interface may differ, but Git works similarly across all terminals. This consistency improves reproducibility as you can now run the same steps on any device.

- How does it work?

Simply type in the git command and hit 'Enter'.

- **Tip:** Keyboard shortcuts work differently in some terminals.

For these activities, follow this:
To copy, right-click → select 'Copy'
To paste, right-click → select 'Paste'

Setting the working directory

The **working directory** is the folder where the Git commands will be executed. This means the commands you enter will apply to the files in this folder.

- Do this by using the "change directory" command: `cd`

Type in `cd` followed by the path to your project.

For example:

```
$ cd /c/Projects/Collaborative-RStudio-GitHub
```

- **Tip:** If the path to your folder has spaces between the words, wrap the path in quotation marks.

For example:

```
$ cd "/c/Projects/Collaborative RStudio GitHub"
```

Practical exercise 1

Make and check changes with: `git status`

- First, you need to make your changes.

☐ In the project folder on your local device, open the "Params" folder.

☐ Right-click the "params_tmpl" R file and select 'Copy' then right-click again and select 'Paste' to place a copy of the R file in the local folder.

☐ Right click the **copied** R file then click 'Rename' to give it a new name, different from the one used in the lesson.

☐ Open the **copied** R file on your local device and make new parameters by editing the 'sig2', 'species.name', and 'color' elements.

☐ Save the edited file by clicking 'File' then 'Save'.

- Now we will check these changes in the terminal.

☐ Open your **Terminal** and set the working directory to the folder on your device that holds the cloned repository.

☐ Enter the command `git status` and hit 'Enter' to check what changes are there in the local copy that aren't on the remote copy.

- These changed files will show up in red.

For example:

```
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        params/My new params.R
        plot_trait_evolution.html
```

Untracked files in Git are files that exist in your project folder but Git is not yet monitoring them. For Git to track these files, you need to commit them.

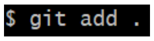
Introduction to Collaborative Coding with GitHub

Practical exercise 2

Stage and commit with: `git add` and `git commit -m`

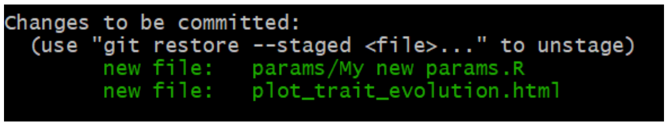
You have made new changes! Now you must commit them.

☐ In **the terminal**, stage all the changes by entering the command: `git add .`



```
$ git add .
```

When you enter the command `git status` again, the files will now be green:

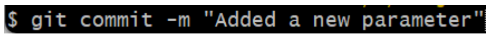


```
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   params/My new params.R
        new file:   plot_trait_evolution.html
```

This means they are staged to commit.

☐ Enter the command `git commit -m` followed by a commit message in quotation marks describing the changes made.

For example:



```
$ git commit -m "Added a new parameter"
```

These changes are now committed.

Practical exercise 3

Push and pull changes with: `git push` and `git pull`

☐ Push your changes to your forked remote repository on GitHub by entering the command: `git push`

☐ Now go to **GitHub**, refresh the page and your changes should appear.

To pull the changes made by other contributors from the remote repository on GitHub to your local copy on your device, enter the command: `git pull`

Extra 1: git clone

Clone a repo with: `git clone` + “`repository URL`”

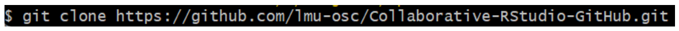
It is possible to use a simple Git command to clone a remote repository to your local device. Here’s how:

On **GitHub**, copy the repository URL

(**Hint:** Find the URL by clicking the Code button)

In the **terminal**, set the working directory to the destination on your device where you would like this clone to be.

Type the command `git clone` followed by the URL for the repository. Here is an example:



```
$ git clone https://github.com/lmu-osc/Collaborative-RStudio-GitHub.git
```

There should now be a copy of the repository in the folder on your local device.

Extra 2: Common Git commands

- Here is a summary of 5 common Git commands that you can use on all terminals.

`git status`: compares the files on the local and remote repositories to show the changes made, staged files, and untracked files.

`git add`: tells Git which changes you want to include in the next commit.

`git commit`: saves a snapshot of your staged changes to the project history.

`git push`: uploads the commits to the remote repository.

`git pull`: downloads and applies the latest changes from the remote repository to the local repository.

- For a quick overview of more Git commands, check out this Git Cheat Sheet: https://git-scm.com/cheat-sheet