# Enhancing Research Reproducibility and Collaboration with RStudio, Git, and GitHub

JP Courneya

May, 2023

# Table of contents

**Preface** **4**
Acknowledgements . . . 4

**1 Introduction to R and RStudio** **5**
1.1 Learning Objectives . . . 5
1.2 Why learn R? . . . 5
1.3 Starting out in R . . . 6
1.3.1 Downloading, Installing and Running R . . . 6
1.3.2 RStudio . . . 7
1.3.3 Posit Cloud (formerly RStudio Cloud) . . . 7
1.4 Using this book . . . 8
1.5 Working in the Console . . . 9
1.6 Working in the Terminal . . . 10

**2 Reproducible Project Management** **12**
2.1 Learning Objectives . . . 12
2.2 RStudio Projects . . . 12
2.3 Creating an Posit Cloud project . . . 12

**3 Version Control and RStudio** **14**
3.1 Learning Objectives . . . 14
3.2 Why Git? . . . 15
3.3 What's GitHub? . . . 17
3.4 Setting up a remote repository on GitHub . . . 18
Connecting Rstudio to GitHub . . . 19
3.5 Introduce yourself to Git . . . 19
3.6 Get a personal access token (PAT) . . . 19
3.6.1 Create the PAT . . . 19
3.6.2 Put your PAT into the Git credential store . . . 20
3.7 Getting started with GitHub . . . 21
3.8 Clone the new GitHub repository to your computer via RStudio . . . 21
3.9 Making some changes, save, commit. . . . 22
3.9.1 How often to commit? . . . 22
3.9.2 What to commit? . . . 22
3.9.3 Using .gitignore . . . 23

3.10 Push your local changes online to GitHub . . . . . 23
3.11 Confirm the local change propagated to the GitHub remote . . . . . 24
3.12 Clean up . . . . . 24
3.13 Clone a template repository from MRP-Bioinformatics . . . . . 24

**References** **26**

# Preface

Welcome to the Malaria Research Program at The University of Maryland Baltimore - Center for Vaccine Development and Global Health https://www.medschool.umaryland.edu/malaria/.

These training materials are developed and made publicly available for increasing awareness of reproducible science and enhancing data and programming skills.



## Acknowledgements

**Git and Github lessons** adapt material from:

- Happy Git with R
- Software Carpentry git-novice

This is a Quarto book. To learn more about Quarto books visit https://quarto.org/docs/books. (Allaire 2022)

# 1 Introduction to R and RStudio

The following chapter will provide you with a hands on opportunity to familiarize yourself with RStudio. Learning RStudio is a big topic and we will not be able to cover everything, by the end of this session we hope that you will feel comfortable starting to use R on your own for working with Git and GitHub.

## 1.1 Learning Objectives

- Navigate RStudio
- Use Posit Cloud (formerly RStudio Cloud)

## 1.2 Why learn R?

- **R is free, open-source, and cross-platform.** Anyone can inspect the source code to see how R works. Because of this transparency, there is less chance for mistakes, and if you (or someone else) find some, you can report and fix bugs. Because R is open source and is supported by a large community of developers and users, there is a very large selection of third-party add-on packages which are freely available to extend R's native capabilities.
- **R code is great for reproducibility**. Reproducibility is when someone else (including your future self) can obtain the same results from the same dataset when using the same analysis. R integrates with other tools to generate manuscripts from your code. If you collect more data, or fix a mistake in your dataset, the figures and the statistical tests in your manuscript are updated automatically.
- **R relies on a series of written commands, not on remembering a succession of pointing and clicking.** If you want to redo your analysis because you collected more data, you don't have to remember which button you clicked in which order to obtain your results; you just have to run your script again.
- **R is interdisciplinary and extensible** With 10,000+ packages that can be installed to extend its capabilities, R provides a framework that allows you to combine statistical approaches from many scientific disciplines to best suit the analytical framework you

need to analyze your data. For instance, R has packages for image analysis, GIS, time series, population genetics, and a lot more.

- **R works on data of all shapes and sizes.** The skills you learn with R scale easily with the size of your dataset. Whether your dataset has hundreds or millions of lines, it won't make much difference to you. R is designed for data analysis. It comes with special data structures and data types that make handling of missing data and statistical factors convenient. R can connect to spreadsheets, databases, and many other data formats, on your computer or on the web.
- **R produces high-quality graphics.** The plotting functionalities in R are endless, and allow you to adjust any aspect of your graph to convey most effectively the message from your data.
- **R has a large and welcoming community.** Thousands of people use R daily. Many of them are willing to help you through mailing lists and websites such as Stack Overflow, or on the RStudio community. Questions which are backed up with short, reproducible code snippets are more likely to attract knowledgeable responses.

## 1.3 Starting out in R

R is both a programming language and an interactive environment for data exploration and statistics.(R Core Team 2022)

Working with R is primarily text-based. The basic mode of use for R is that the user provides commands in the R language and then R computes and displays the result.

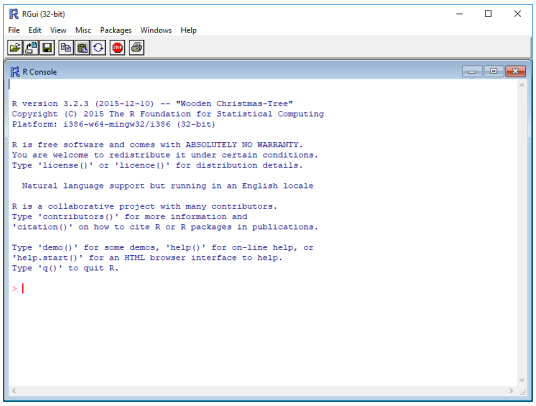
### 1.3.1 Downloading, Installing and Running R

**Download**
R can be downloaded from CRAN (The Comprehensive R Archive Network) for Windows, Linux, or Mac.

**Install**
Installation of R is like most software packages and you will be guided. Should you have any issues or need help you can refer to R Installation and Administration

**Running**
R can be launched from your software or applications launcher or When working at a command line on UNIX or Windows, the command `R` can be used for starting the main R program in the form `R`

You will see a console similar to this appear:




While it is possible to work solely through the console or using a command line interface, the ideal environment to work in R is RStudio.

### 1.3.2 RStudio

RStudio is a user interface for working with R. It is called an Integrated Development Environment (IDE): a piece of software that provides tools to make programming easier. RStudio acts as a sort of wrapper around the R language. You can use R without RStudio, but it's much more limiting. RStudio makes it easier to import datasets, create and write scripts, and makes using R much more effective. RStudio is also free and open source. To function correctly, RStudio needs R and therefore both need to be installed on your computer. For this training we'll be using a browser based version called Posit Cloud (see directions in the Posit Cloud section below.

RStudio interface is conveniently organized into four divisions called "panes".

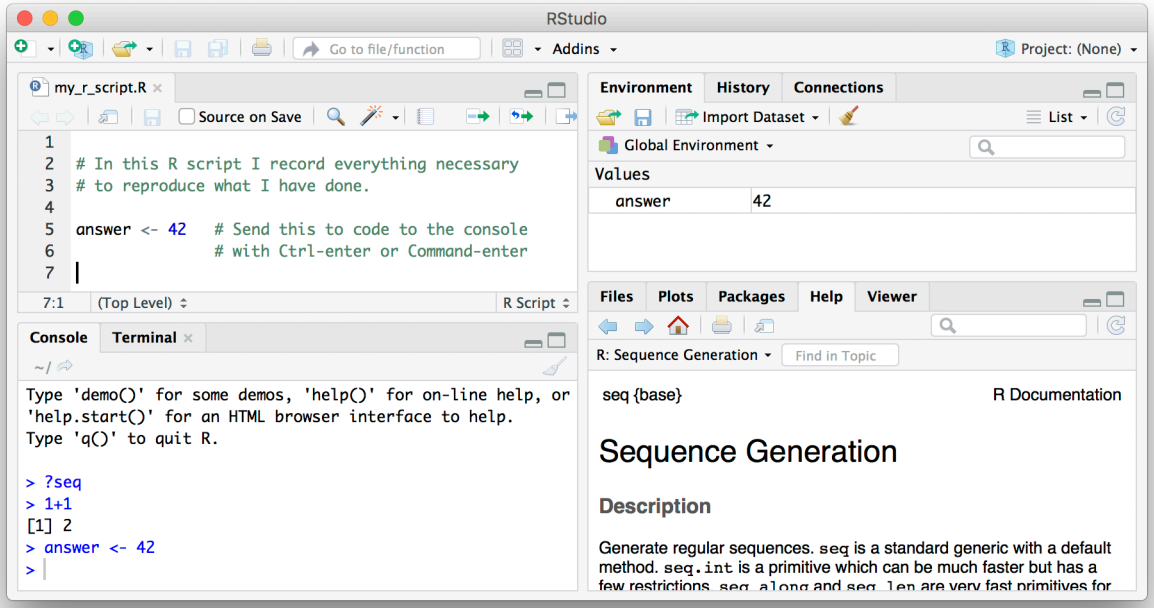
The Default Layout is:

- Top Left - **Source**: your scripts and documents
- Bottom Left - **Console**: what R would look and be like without RStudio
- Top Right - **Environment/History**: look here to see what you have done
- Bottom Right - **Files** and more: see the contents of the project/working directory here, like your Script.R file

The placement of these panes and their content can be customized (see menu, Tools -> Global Options -> Pane Layout)

### 1.3.3 Posit Cloud (formerly RStudio Cloud)

Posit Cloud is a web browser-based version of RStudio. It will allow you to use RStudio without needing to download anything to your computer. You can also easily share your R




projects with others. To use Posit Cloud a user account is required. While we recommend downloading RStudio for routine R programming use, we will be using Posit Cloud for this training.

To access Posit Cloud

1. In a new browser window or tab create your account at https://posit.cloud/plans/free.
2. Log in with Goolge or GitHub or however you choose!
3. Create a new project.
4. Write amazing code!

## 1.4 Using this book

**For these instructions code will appear in the gray box as follows:**

```
fake code
```

To run the code you can copy and paste the code and run it in your RStudio session console at the prompt > which looks like a greater than symbol.

```
> fake code
```

The code can also be added to an R Script to be run.

When the code is run in RStudio the console prints out results like so:

```
[1] Result
```

In this tutorial results from code will appear like so:

```
## [1] Result
```

## 1.5 Working in the Console

The console is an interactive environment for RStudio, click on the "Console" pane, type `3 + 3` and press enter. R displays the result of the calculation.

```
3 + 3
```

```
[1] 6
```

`+` is called an operator. R has the operators you would expect for for basic mathematics:

**Arithmetic operators**

| operator | meaning |
|---|---|
| + | plus |
| - | minus |
| * | times |
| / | divided by |
| ^ | exponent |

**Logical Operators**

| operator | meaning |
|---|---|
| == | exactly equal |
| != | not equal to |
| < | less than |
| <= | less than or equal to |

| operator | meaning |
|---|---|
| > | greater than |
| >= | greater than or equal to |
| x\|y | x or y |
| x&y | x and y |
| !x | not x |

Spaces can be used to make code easier to read.

```
2 * 2 == 4
```

```
[1] TRUE
```

You can also run commands in the consile for working with your computers filesystem.

```
getwd() # similar to UNIX PWD
```

## 1.6 Working in the Terminal

The embedded Terminal in RStudio is a command-line interface within the IDE, allowing users to execute system commands and interact with the operating system directly.

- It shares the same working directory as the RStudio session and supports various commands for file management, package installation, and more.
- Integration into the IDE streamlines workflows by eliminating the need to switch between applications, enhancing productivity and enabling seamless interaction between R programming and system administration tasks.


Console
Terminal
Background Jobs
Terminal 1
~/Library/CloudStorage/OneDrive-UniversityofMarylandSchoolofMedicine/Projects/BC01_Bioinformat
(base) MEDM-58458:MRP_git_training jean-paul.courneya$

# 2 Reproducible Project Management

R and RStudio allow you to tackle a wide variety of analytic and programmatic tasks not all corresponding to the same project. Knowing what really needs to be saved and establishing a procedure to keep all of the stuff together for any given project will help in preventing a data loss or information loss.

## 2.1 Learning Objectives

- What are RStudio Projects
- Creating a Posit Cloud project

## 2.2 RStudio Projects

R experts keep all the files associated with a project together. This is such a wise and common practice that RStudio has built-in support for this via Rprojects.

What are they?

- RStudio feature enabling users to create self-contained environments for their data analysis and coding projects.
- Provide a dedicated workspace where researchers and developers can organize their R scripts, data files, and related resources, simplifying project management, version control, and collaboration among team members.
- Encapsulate everything within a project, including configurations and dependencies
- Because of this they promote reproducibility and streamline the development process.

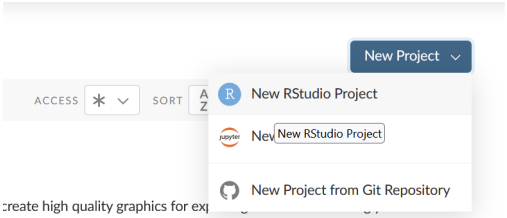
## 2.3 Creating an Posit Cloud project

Posit Cloud organizes everything into projects automatically.

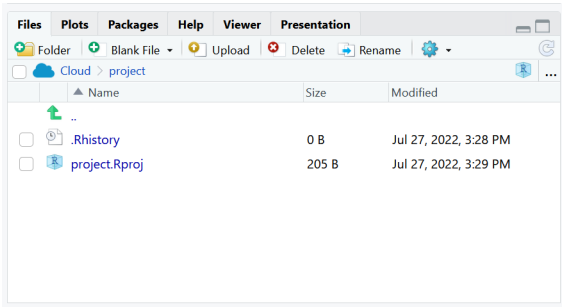
Let's create a new project in Posit Cloud now.

First make sure you are logged in to your Posit Cloud account (or create a free account if you haven't yet.)

Then from `Your Workspace`, select `New Project > New RStudio Project`.



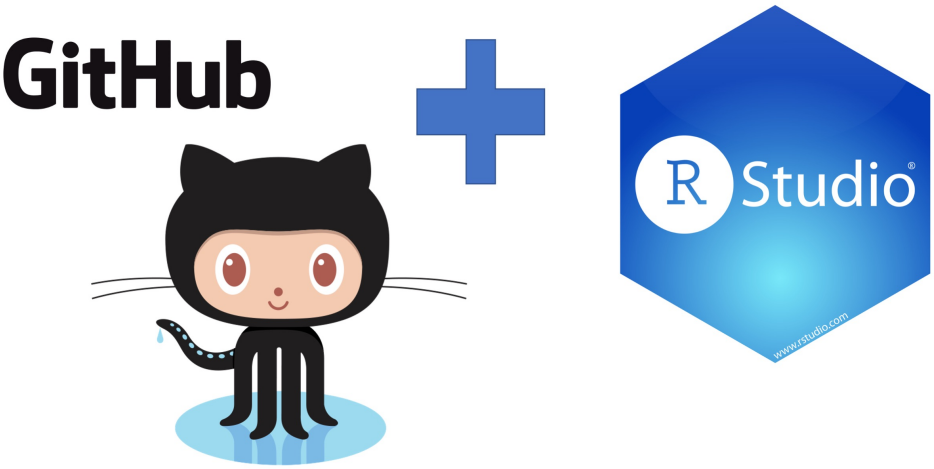

It will take a moment for your project to launch. Then you can give it a name. Let's call this `Reproducible Project Example`. Note that you have a `.Rproj` in your file pane.




Whenever you are in a project, the project is your working directory.

Now that you have mastered this skill we will switch gears and talk about version control with git and GitHub and how to intergrate them with RStudio projects!

# 3 Version Control and RStudio



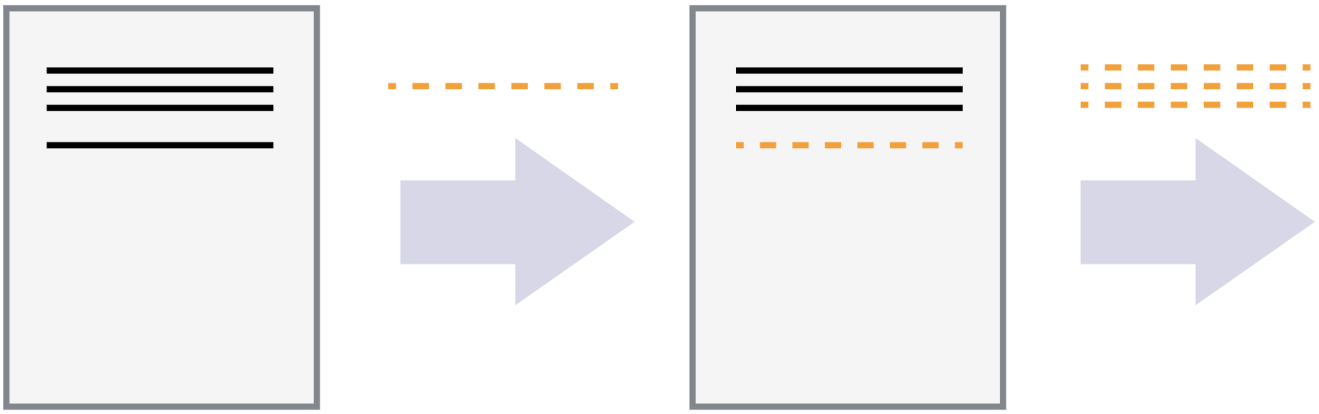

## 3.1 Learning Objectives

- What is version control
- What is GitHub
- How to set up a remote repository on GitHub
- Connecting Rstudio to GitHub
- Create a personal access token for pushing changes to GitHub
- What is a commit
- Push changes to a remote repository
- How to delete a repository
- Use a template to create a repository on GitHub

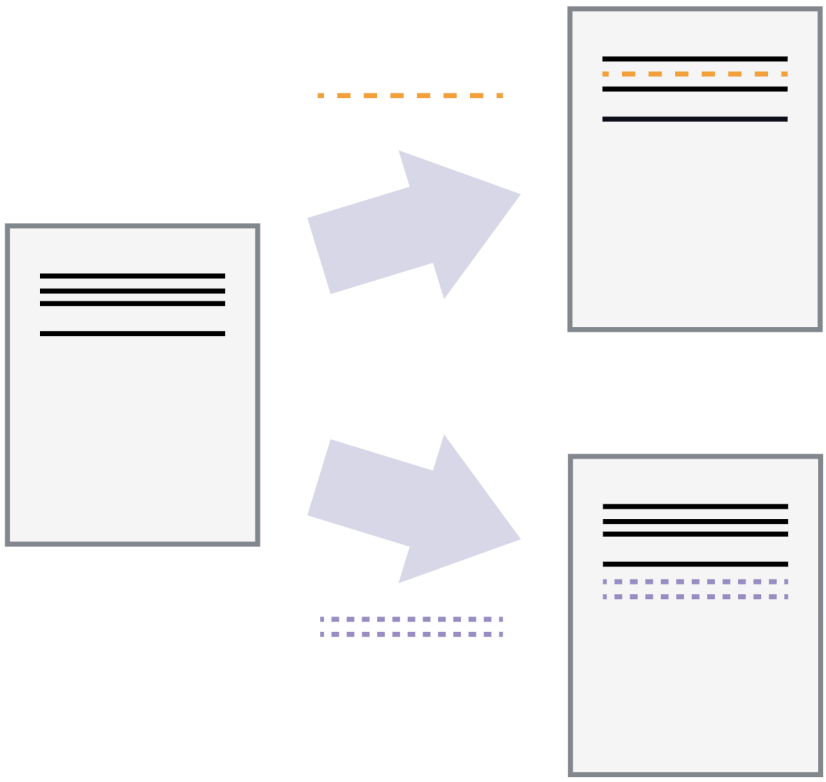
## 3.2 Why Git?

Git is a version control system. Its original purpose was to help groups of software developers work collaboratively on big software projects. Git manages the evolution of a set of files – called a repository – in a sane, highly structured way. If you have no idea what I'm talking about, think of it as the "Track Changes" features from Microsoft Word but supercharged.

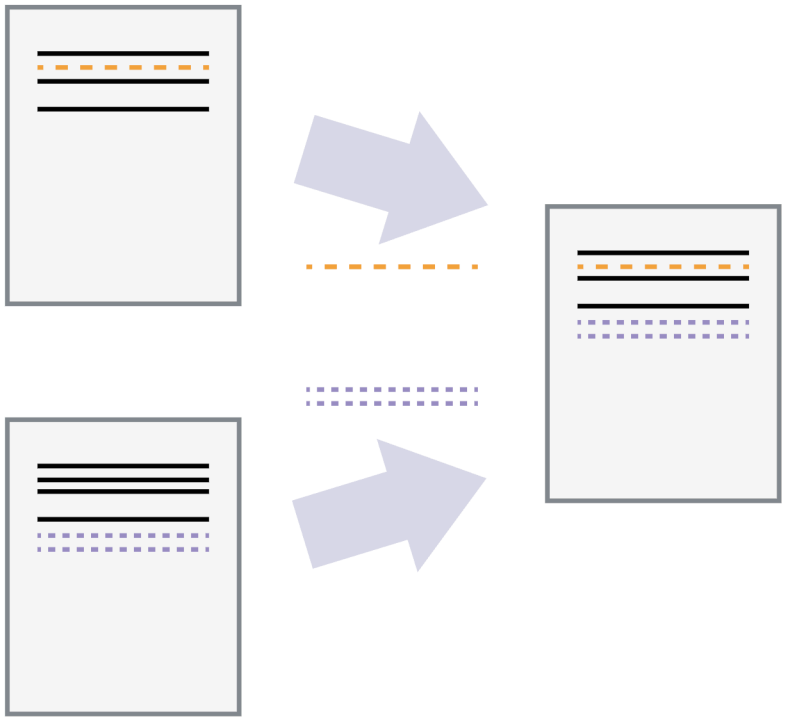
Version control systems start with a base version of a document and then record changes you make each step of the way. You can think of it as a recording of your progress: you can rewind to start at the base document and play back each change you made, eventually arriving at your more recent version.



Once you think of changes as separate from the document itself, you can then think about "playing back" different sets of changes on the base document, ultimately resulting in different versions of that document. For example, two users can make independent sets of changes on the same document.



Unless multiple users make changes to the same section of the document - whats known as a conflict - you can incorporate two sets of changes into the same base document.



A version control system is a tool that keeps track of these changes for us, effectively creating different versions of our files. It allows us to decide which changes will be made to the next version (each record of these changes is called a commit), and keeps useful metadata about them. The complete history of commits for a particular project and their metadata make up a repository. Repositories can be kept in sync across different computers, facilitating collaboration among different people.

Because of its usefulness git has been re-purposed by the data science community. In addition to using it for source code, we use it to manage the motley collection of files that make up typical data analysis projects, which often consist of data, figures, reports, and, yes, source code.((eds) 2019)

## 3.3 What's GitHub?

GitHub, Bitbucket, and GitLab are online services that provide a home for your Git-based projects on the internet. If you have no idea what I'm talking about, think of them as DropBox but much, much better. The remote host acts as a distribution channel or clearinghouse for your Git-managed project. It allows other people to see your stuff, sync up with you, and perhaps even make changes. These hosting providers improve upon traditional Unix Git servers with well-designed web-based interfaces.

Even for private solo projects, it's a good idea to push your work to a remote location for peace of mind. Why? Because it's fairly easy to screw up your local Git repository, especially when you're new at this. The good news is that often only the Git infrastructure is borked up. Your files are just fine! Which makes your Git pickle all the more frustrating. There are official Git solutions to these problems, but they might require expertise and patience you can't access at 3a.m. If you've recently pushed your work to GitHub, it's easy to grab a fresh copy, patch things up with the changes that only exist locally, and get on with your life. Don't get too caught up on public versus private at this point. There are many ways to get private repositories from the major providers for low or no cost. Just get started and figure out if and how Git/GitHub is going to work for you!

We will not be covering all the in's and outs of version control with Git, GitHub and all the resources to be found there since our time is limited. Instead you will learn how to:

- Set up a remote repository on GitHub
- Connecting RStudio to GitHub
- Set up a personal access token on GitHub
- Checking out a project from a version control remote repository
- Making some changes, using the Rstudio Git controls and pushing those changes to GitHub

## 3.4 Setting up a remote repository on GitHub

First thing we do is navigate to GitHub, make sure you're logged in.

Click green "New repository" button. Or, if you are on your own profile page, click on "Repositories", then click the green "New" button.

How to fill this in:

- Repository name: myrepo (or whatever you wish, we'll delete this soon anyway).
- Description: "testing my setup" (or whatever, but some text is good for the README).
- Public.
- YES Initialize this repository with a README.
- For everything else, just accept the default.

Click big green button "Create repository."

Copy the HTTPS clone URL to your clipboard via the green "Clone or Download" button.

## Connecting Rstudio to GitHub

Here we verify that RStudio can issue Git commands on your behalf. This means you'll be able to pull from and push to GitHub from RStudio.

## 3.5 Introduce yourself to Git

Now we will configure git to talk with GitHub from RStudio.

The usethis package (Wickham, Bryan, and Barrett 2022) offers an approach to set your Git user name and email from within R as well as setting up a personal access token. Why does this matter? Because we want to be able to push changes to our remote repository (save what we work on locally, on GitHub).

To configure git:

```
## install if needed (do this exactly once):
install.packages("usethis")

library(usethis)
use_git_config(user.name = "Jane Doe", user.email = "jane@example.org")
```

Check what you just did `usethis::git_sitrep()` generates a git situation-report. It can help you confirm things will work as expected; it can also help you diagnose problems:

```
library(usethis)

use_git_config(core.editor = "nano")
```

## 3.6 Get a personal access token (PAT)

### 3.6.1 Create the PAT

In this step you will be creating a PAT. Since GitHub is moving away from username + password to push to their server it will be inevitable that you must set this up to work with GitHub from your local computer and RStudio. To create the token using `usethis` run:

```
create_github_token()
```

You will see running this command takes you to a pre-filled form to create a new PAT. You can get to the same page in the browser by clicking on "Generate new token" from https://github.com/settings/tokens. The advantage of `create_github_token()` is that the `usethis` maintainers have pre-selected some recommended scopes, which you can look over and adjust before clicking "Generate token". It is a very good idea to give the token a descriptive name, because one day you might have multiple PATs, e.g., one that's configured on your main work computer and another that you use from a secondary computer or VM. Eventually, you'll need to "spring clean" your PATs and this is much less nerve-wracking if you know which PAT is being used where and for what.

You must **store this token somewhere**, because you'll never be able to see it again, once you leave this browser window. If you somehow goof this up, just generate a new PAT and, so you don't confuse yourself, delete the lost token. In the moment, we usually copy the PAT to the clipboard, anticipating what we'll do next: trigger a prompt that lets us store the PAT in the Git credential store.

### 3.6.2 Put your PAT into the Git credential store

Next you need to run:

```
gitcreds::gitcreds_set()
```

You will have the `gitcreds` package installed, as of `usethis v2.0.0`, because usethis uses gh, and gh uses gitcreds.

If you don't have a PAT stored already, it will prompt you to enter your PAT. **Paste!**

If you do already have a stored credential, `gitcreds::gitcreds_set()` reveals this and will even let you inspect it. This helps you decide whether to keep the existing credential or replace it. When in doubt, embrace a new, known-to-be-good credential over an old one, of uncertain origins.

Here are two great ways to check that all is well:

```
gh::gh_whoami()

usethis::git_sitrep()
```

Both of these functions reveal whether a GitHub PAT is discovered and provide information about the associated user, the PAT's scopes, etc.

This step is something you do once. Or, rather, once per machine, per PAT. From this point on, `usethis` and its dependencies should be able to automatically retrieve and use this PAT.

## 3.7 Getting started with GitHub

Generally speaking a user is provided 3 workflows for working with GitHub.

- New project, GitHub first is the easiest way to get a working project.
- Existing project, GitHub first is a deeply pragmatic way to get pre-existing work onto GitHub. You can follow these instructions to do this yourself
- Existing project, GitHub last is the more proper way to connect existing local work to a remote on GitHub, especially if there's already a Git history. Checking out a project from a version control remote repository. You can follow these instructions to do this yourself

We will cover New project, GitHub first. Picking up were we left off when Setting up a remote repository copy the HTTPS clone URL to your clipboard via the green "Clone or Download" button.

## 3.8 Clone the new GitHub repository to your computer via RStudio

In RStudio, start a new Project:

- RStudio > New Project > New Project from Git Repository. In "Repository URL", paste the URL of your new GitHub repository. It will be something like this: `https://github.com/j-p-courneya/GSF_GitHub_Practice.git`.
- Accept the default project directory name, e.g. GSF_GitHub_Practice, which coincides with the GitHub repo name.
- Take charge of – or at least notice! – where the Project will be saved locally. A common rookie mistake is to have no idea where you are saving files or what your working directory is. Pay attention. Be intentional.
- I suggest you check "Open in new session", as that's what you'll usually do in real life.
- Click "Create Project".

You should find yourself in a new local RStudio Project that represents the new test repo we just created on GitHub. This should download the README.md file from GitHub. Look in RStudio's file browser pane for the README.md file.

## 3.9 Making some changes, save, commit.

Remember, you can work with git and GitHub via the command line or a GUI (were using RStudio). The routine use of git involves just a few commands: principally add, commit, and push, but also status and diff. We will use these commands integrated into RStudio as a set of buttons and check boxes and dialog boxes.

From RStudio, modify the `README.md` file, e.g., by adding the line "This is a line from RStudio". Save your changes.

Commit these changes to your local repo. How?

From RStudio:

- Click the "Git" tab in upper right pane.
- Check "Staged" box for README.md.
- If you're not already in the Git pop-up, click "Commit".
- Type a message in "Commit message", such as "Commit from RStudio".
- Click "Commit".

A few questions that typically come up: how often to commit, what to commit, how do i not track something.

### 3.9.1 How often to commit?

I prefer to do many small commits, each for a set of related changes:

Think of something that needs to be fixed, or a feature to add. Do the work. Test that it is okay. Add and commit. Look at others' projects on GitHub, to see what they do and what sort of commit messages they write.

### 3.9.2 What to commit?

Don't include files that are derived from other files in the repository.

For example, for this quarto book, I wouldn't include all the .quarto, _book, etc., files. And if I have R code to generate a figure, I'll include the R code but not the figure.

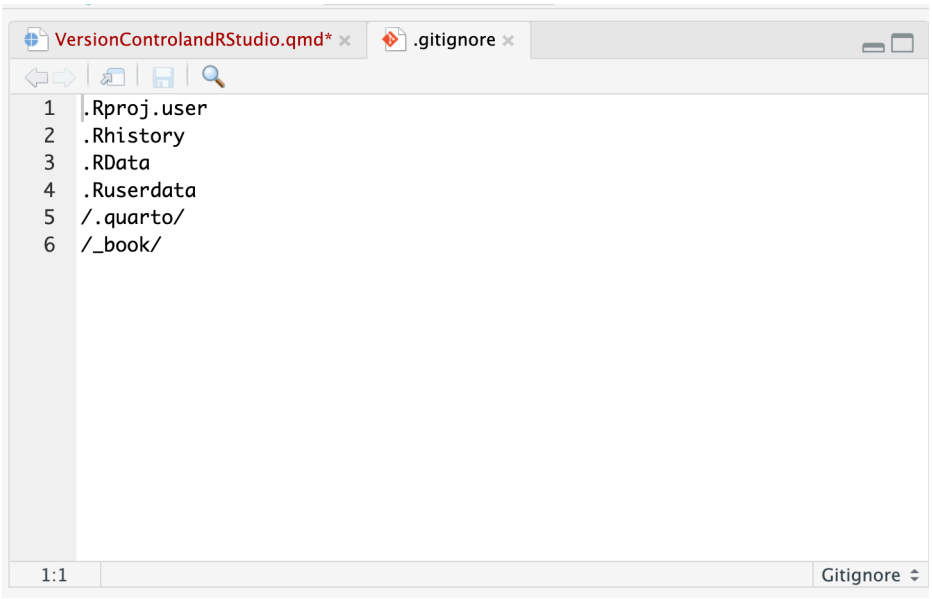
Be careful about committing binary files, or really big files. Git works best with text files (like source code), as you can see just the lines that were changed. A new copy of a file will get added to the repository every time you change it. For small text files, that's no big deal; for big images, you'll get a bloated repository.

And once you've committed a big file to your repository, it's there forever, even if you use git rm to remove it later.

For big data files that are changing, you'll want to track a text-based version (not .xls!), and you may want to make a fully separate git repository for the data.(Broman 2021)

### 3.9.3 Using .gitignore

When you dont want to track changes of something you can add it to the .gitignore




## 3.10 Push your local changes online to GitHub

Click the green "Push" button to send your local changes to GitHub. You should see some message along these lines.

```
[master dc671f0] blah
 3 files changed, 22 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 myrepo.Rproj
```

## 3.11 Confirm the local change propagated to the GitHub remote

Go back to the browser. I assume we're still viewing your new GitHub repo.

Refresh.

You should see the new "This is a line from RStudio" in the README.

If you click on "commits", you should see one with the message "Commit from RStudio".

If you have made it this far, you are DONE with set up.

## 3.12 Clean up

At this point since this was all practice you might be interested in deleting the practice repo. It is also good to learn how to do this for real-life scenarios.

**Local** When you're ready to clean up, you can delete the local repo any way you like. It's just a regular directory on your computer.

**GitHub** In the browser, go to your repo's landing page on GitHub. Click on "Settings".

Scroll down, click on "delete repository," and do as it asks. GitHub really wants to make sure you definitely want to get rid of the repository.

## 3.13 Clone a template repository from MRP-Bioinformatics

The bioinformatics core has set up a organization account on GitHub. A GitHub organization is a shared workspace on the GitHub platform where individuals and teams collaborate on projects, manage repositories, and coordinate their development efforts.

GitHub organizations have repositories just like personal accounts do. Also for routine structured repositories, like for when youre going to work on a code project, a user can create a "template" repository which can be cloned to a persons user account or to another organization. Lets practice doing that

- Navigate to MRP-Bioinformatics organization page https://github.com/mrp-bioinformatics
- Under repositories look for "new_project_template" and select it
- Feel free to examine its contents clicking around in the files and directories.
- Next we will click "Use this template" and choose "Create a new repository" to clone it to our personal GitHub accounts.

- Fill out the form naming the new repository. This is a great point to start from for version controlled projects because you are inheriting an ideal directory structure and creating a repository at the same time instead of starting frorm scratch.
- Select whether you want a Public or Private repository
- Leave "Include all branches" unchecked
- Mash the green button "Create repository from template"
- Bask in glory of your amazing GitHub skills!

This brings us to the conclusion of this training. Stay tuned for more learning opportunities and dont forget to like and subscribe (JK!)

# References


Allaire, JJ. 2022. *Quarto: R Interface to 'Quarto' Markdown Publishing System*. https://CRAN.R-project.org/package=quarto.

Broman, Karl. 2021. *Git/Github Guide: Karl's Minimal Tutorial*. https://kbroman.org/github_tutorial.

(eds), Ivan Gonzalez; Daisie Huang; Nima Hejazi; Katherine Koziar; Madicken Munk. 2019. *Software Carpentry: Version Control with Git*. https://github.com/swcarpentry/git-novice.

R Core Team. 2022. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Wickham, Hadley, Jennifer Bryan, and Malcolm Barrett. 2022. *Usethis: Automate Package and Project Setup*. https://usethis.r-lib.org.