R Workflow Essentials

Publishing Reports on GitHub

Dr Muhammad Saufi Abdullah
DrPH (Epidemiology) Candidate

Publishing Reports on GitHub

You should now be able to create **HTML** reports or other output formats using **Quarto**. However, several issues could happen in your work:

If your computer crashes unexpectedly, you could lose all your work. That's why it is important to use cloud storage as a backup.

If you want to share your work with colleagues, sending it via email might seem convenient, but it is an outdated method.

Your colleagues might want to contribute to your project, so it is essential to use a collaborative platform where your team can work together seamlessly.

This is where **GitHub** comes in.

1 Introduction to GitHub

GitHub is a platform where you can store and manage your code online.

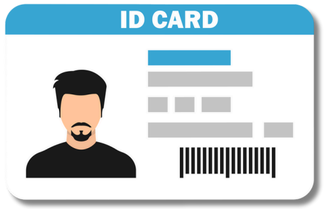
Think of it as a home for your projects on the internet, similar to cloud-based services like **Dropbox** or **OneDrive**.

GitHub acts as a distribution channel, making it easy to share your work with others. Check out my **GitHub** profile here.

2 GitHub Profile

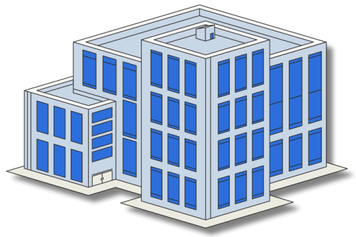
Register for your **GitHub** account and create your profile here. When setting up your **GitHub profile**, it's important to choose a **username** with care, as it may represent you in professional settings later on.

Here are a few tips for selecting a good username:



Include your real name
This makes it easier for others to recognize you.

Keep it short
A concise username is easier to remember.



Timeless
Avoid including references to your current university, employer, or location.

Use all lowercase
Recommended for simplicity

For example, my **GitHub** username is **`drsaufi`**.

3 Install Git

Git is a version control system that you need to connect **RStudio** on your laptop to your **GitHub** account.

First, check if **Git** is already installed on your machine. In **RStudio**, open the **Terminal** and type the following commands, then press `**Enter**`:

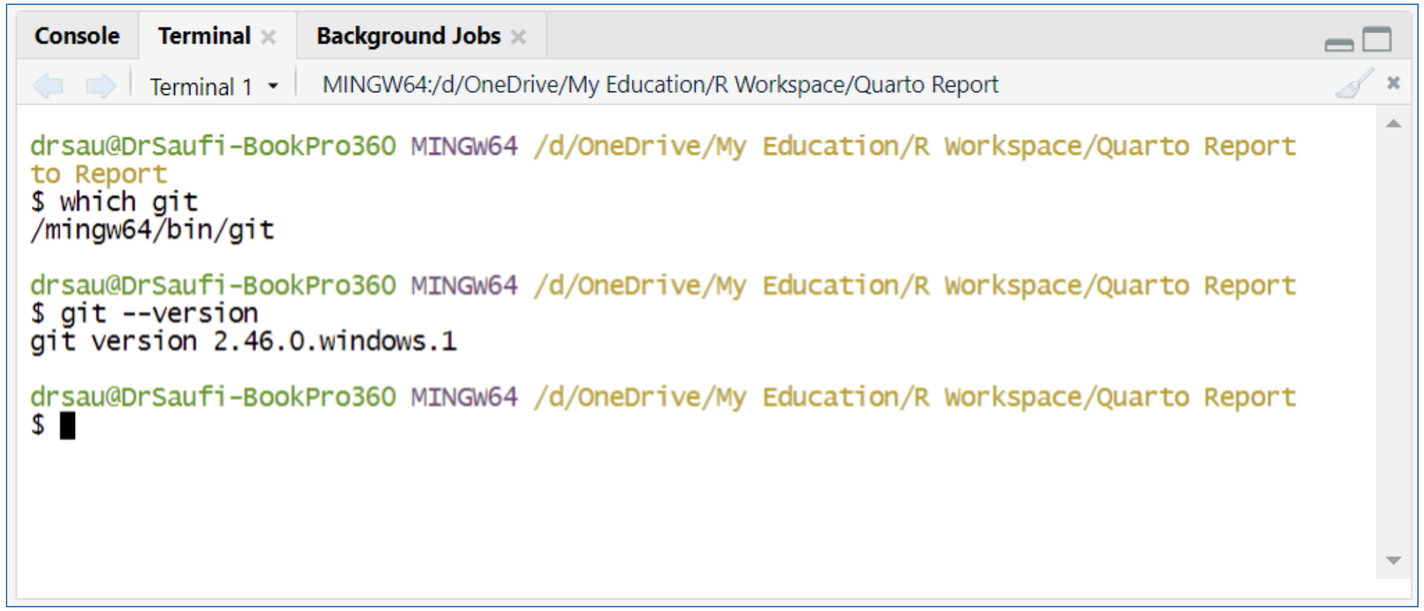
In Terminal:

```
which git
```

In Terminal:

```
git --version
```

If you receive a response showing the **Git version**, as in the example image below, it means **Git** is already installed.



However, if you see something like **`git: command not found`**, it means **Git** is not installed on your computer.

In that case, visit this link to download and install **Git** on your machine.

After successfully installing **Git**, you'll need to configure it with your name and email address.

In the **Terminal** within **RStudio**, type each of the following commands one at a time, pressing **`Enter`** after each line:

In Terminal:

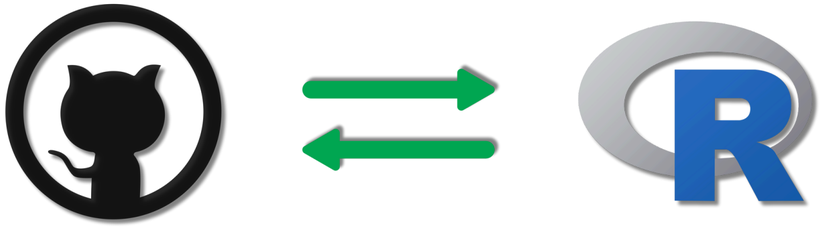
```
git config --global user.name "Your Name"
git config --global user.email "email@example.com"
git config --global --list
```

*Your username can be either your actual name or the name from your **GitHub** profile.*

*Make sure to use the email address that is linked to your **GitHub** account.*

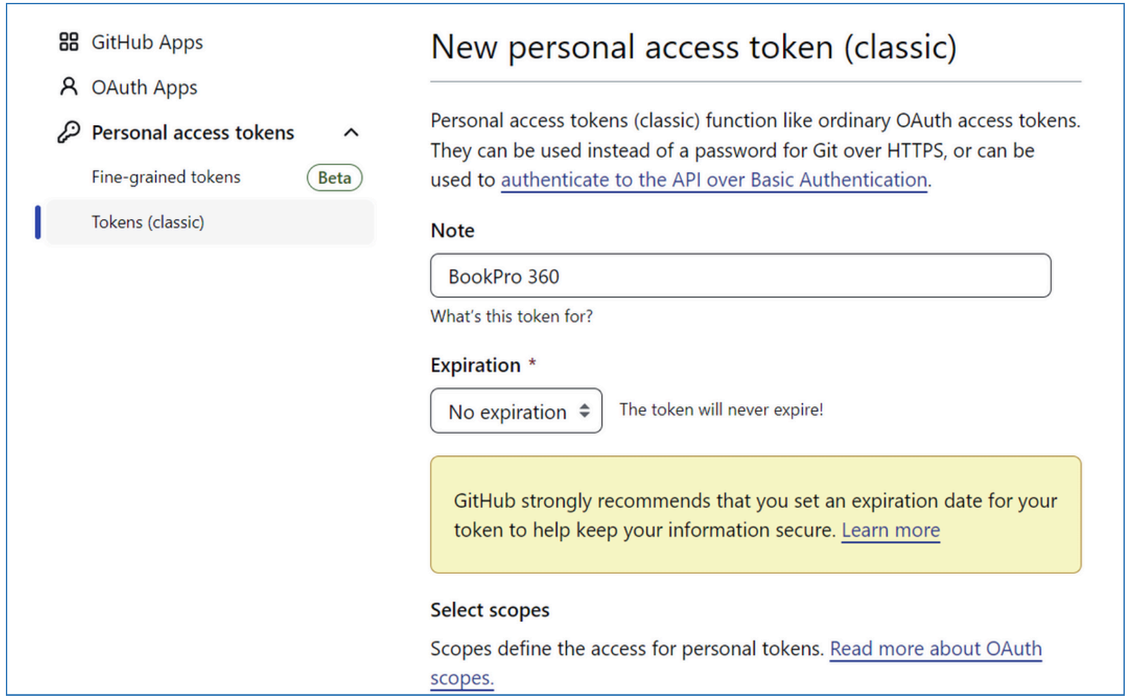
4 Link GitHub to RStudio

To connect your **GitHub** account to **RStudio** on your machine, you will need to create a **personal access token** or **PAT** on **GitHub**.



Start by opening your **GitHub** profile or visit this link to create **PAT**, also reachable via:

`Settings → Developer settings → Personal access tokens`.



*In the **Note** section, briefly describe the purpose of the token.*

For example, I labeled mine as `**BookPro 360**` since it is the token for my laptop.

If you're using this token on your personal laptop, you might choose to set no expiration.

*However, for better security, **GitHub** recommends setting an expiration date for your token.*

*Once you're satisfied with the **Note**, **Expiration**, and **Scopes**, click `**Generate token**`.*

Once you have generated your **PAT**, make sure to save it somewhere safe before closing or navigating away from the browser, as you will not be able to view this token again.

Next, you will need to run a few commands in the **Console** within **RStudio**.

Start by installing the **`gitcreds`** package. Type the following command in the **Console** and press **`Enter`**:

In Console:

```
install.packages("gitcreds")
```

Once the package is installed (or if you already have it installed), proceed to the next step.

In the **Console**, type the following commands and press `**Enter**` after each one:

In Console:

```
library(gitcreds)
gitcreds_set()
```

You will then see a prompt asking for your password or token. Paste your **PAT** here and press `**Enter**`:

In Console:

```
? Enter password or token:
```

If you see the following response, you have successfully stored the **PAT** in your **RStudio**:

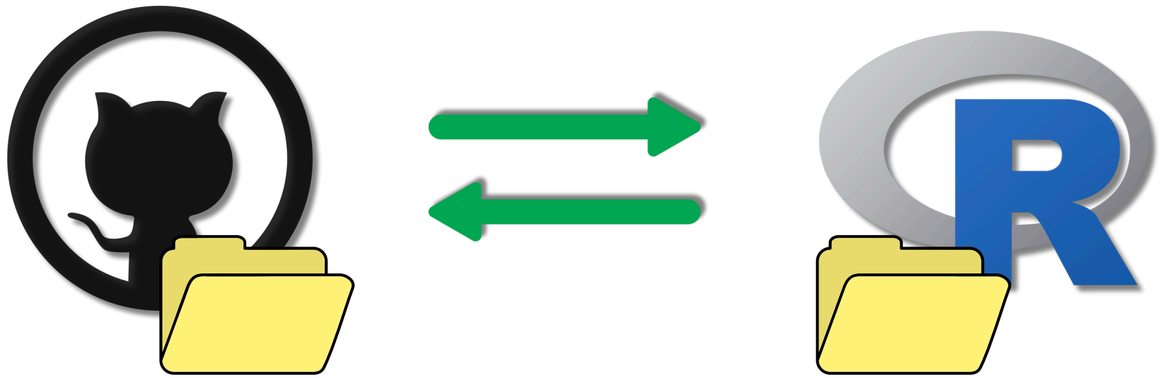
```
? Enter password or token: xxxxxxxxxxxxxxx
-> Adding new credentials...
-> Removing credentials from cache...
-> Done.
```

*Now, you have successfully linked your **GitHub** account to **RStudio** on your computer.*

5 GitHub Repository

A repository in **GitHub** is similar to a working directory in **RStudio**. Essentially, it's a folder on **GitHub**.

The current objective is to synchronize your working directory in **RStudio** with a repository on **GitHub**.



GitHub Repository

RStudio Working Directory

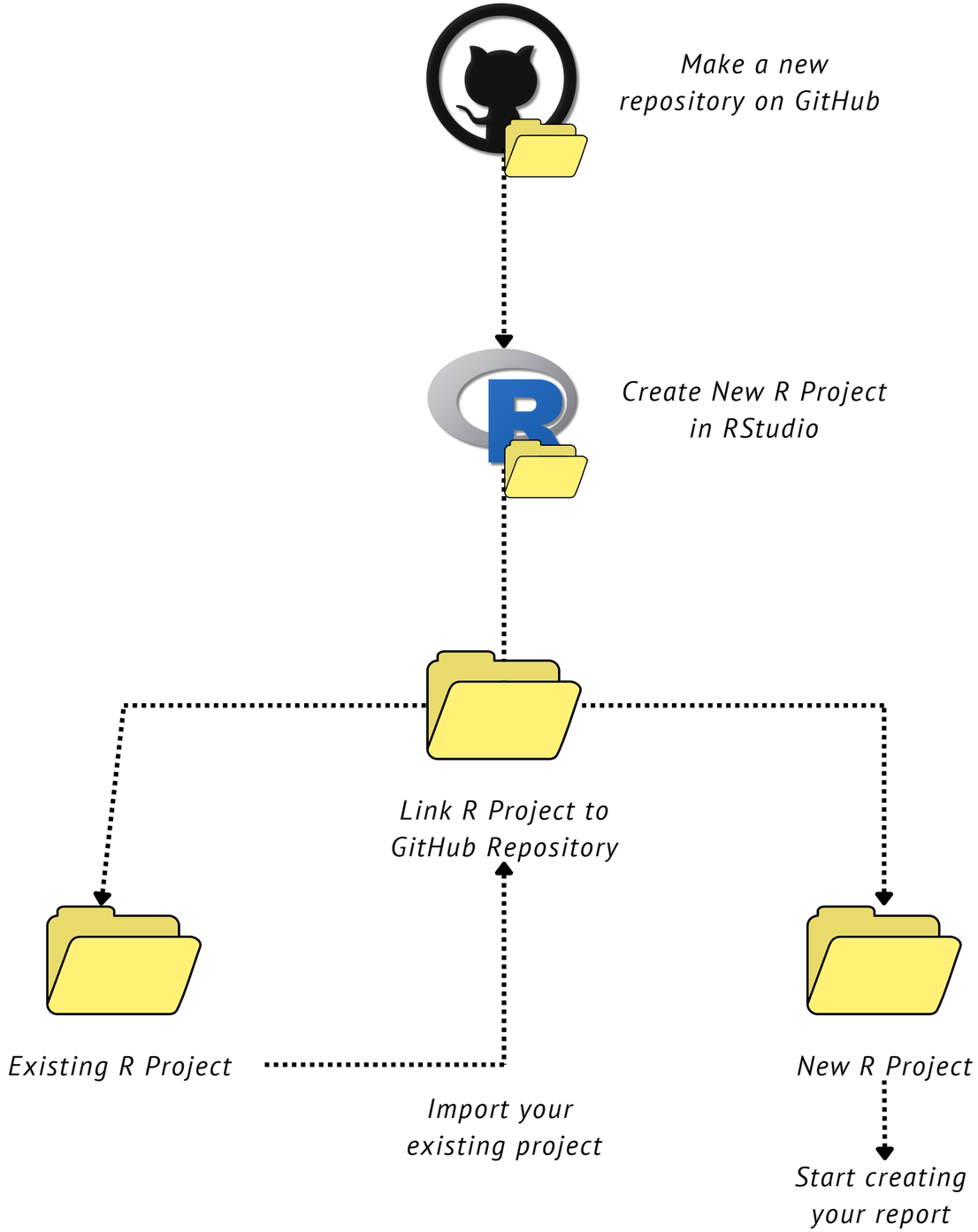
The process of integrating **GitHub** with **RStudio** begins by creating a new repository on **GitHub**.

After setting up the **GitHub** repository, the next step is to create a new project in **RStudio**.

When creating this project, you need to link it directly to the **GitHub** repository to ensure synchronization between your **RStudio** and the **GitHub** repository.

Once the new **R** project is created, you have a working directory in **RStudio** that is synced to your **GitHub** repository. You can start creating your reports within the **R** project.

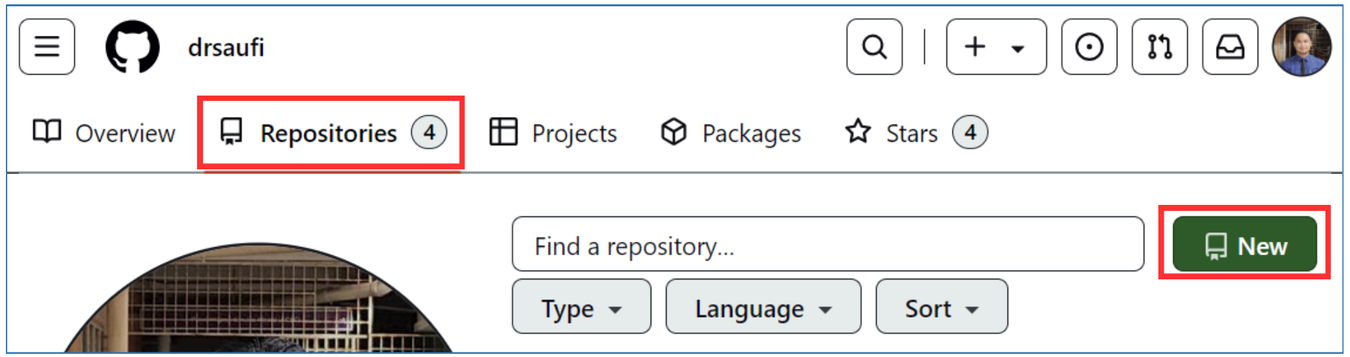
However, if you already have an existing **R** project, you can copy or move **R** documents and files into the directory for this new **R** project.


Make a new
repository on GitHub
Create New R Project
in RStudio
Link R Project to
GitHub Repository
Existing R Project
Import your
existing project
New R Project
Start creating
your report

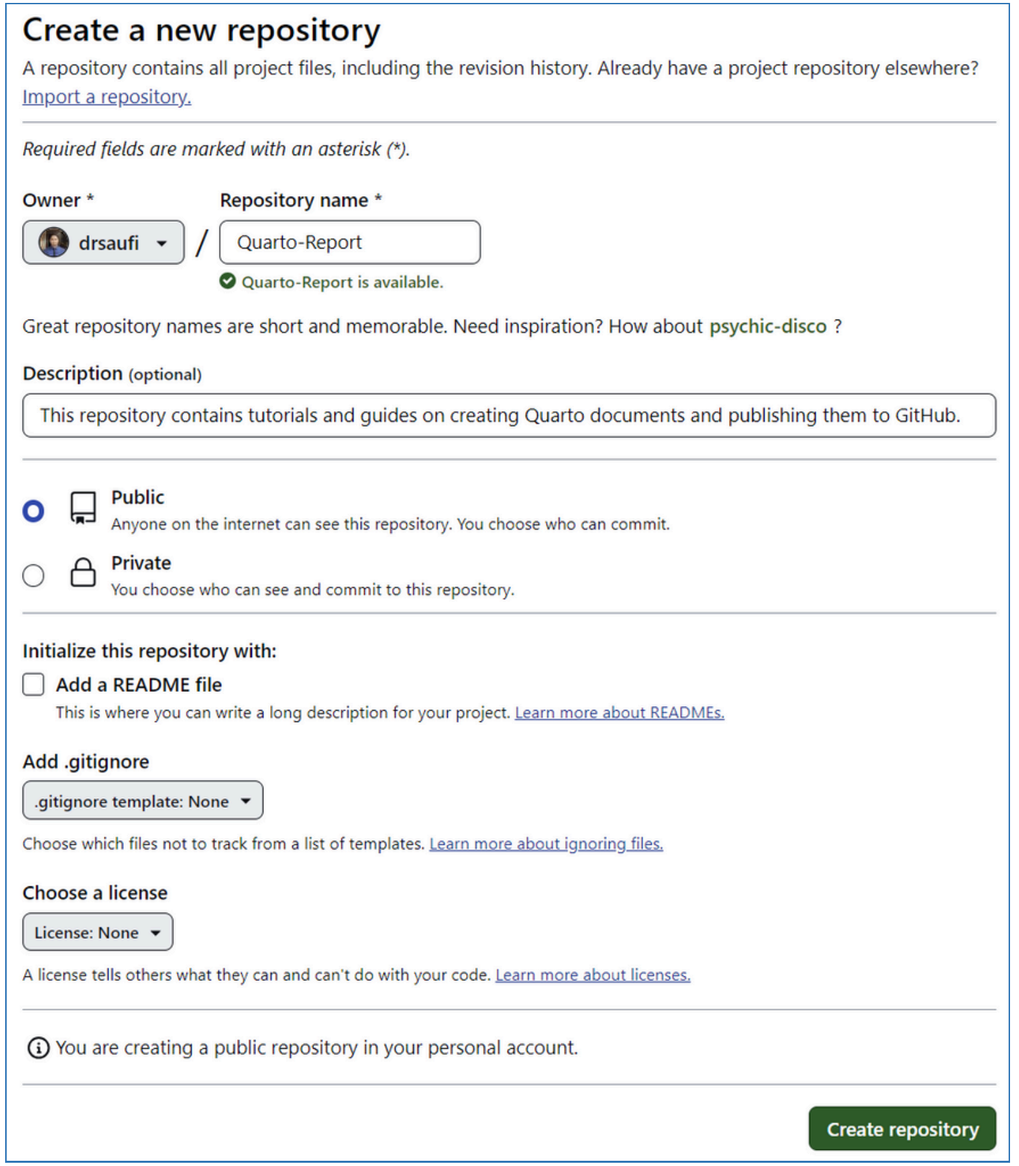
Create a GitHub Repository

Let’s start by creating a new repository on **GitHub**.

In your **GitHub** profile, navigate to the **`Repositories`** tab and click the **`New`** button.



Give your repository a name. Keep in mind that **GitHub** does not allow spaces in repository names, so it automatically replaces them with hyphens ("-").



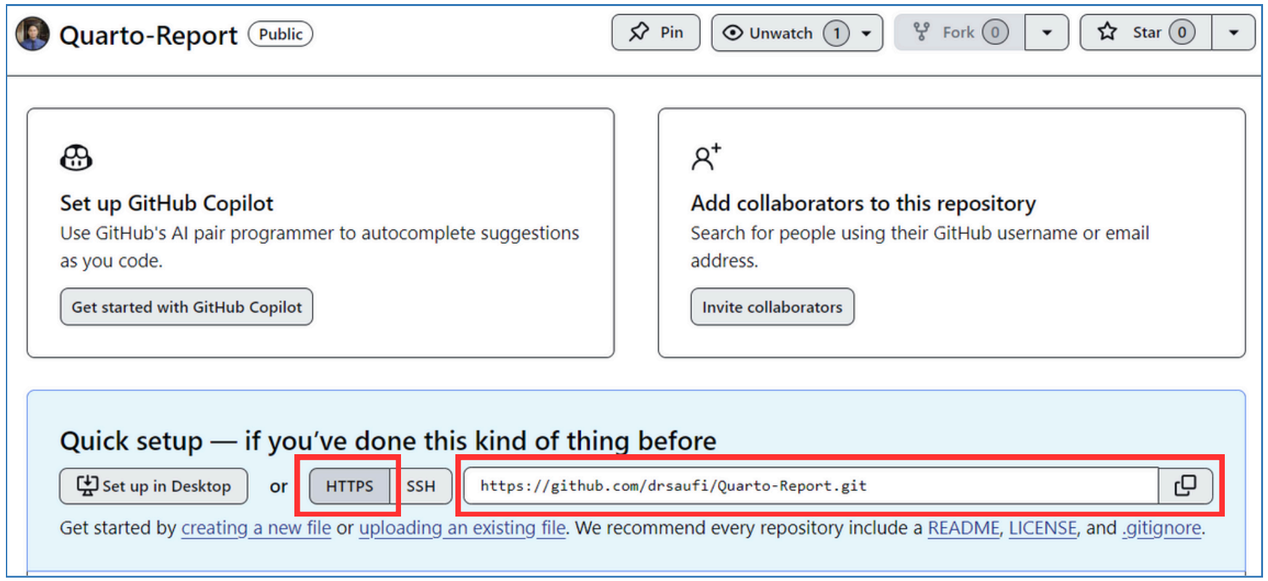
It is a good practice to include a brief description of the purpose of your repository.

You can choose to set the repository as `**Public**` or `**Private**`, depending on its content.

For the purpose of learning and sharing knowledge, let's set it to `**Public**`.

Once you have filled in all the details, click the `**Create repository**` button.

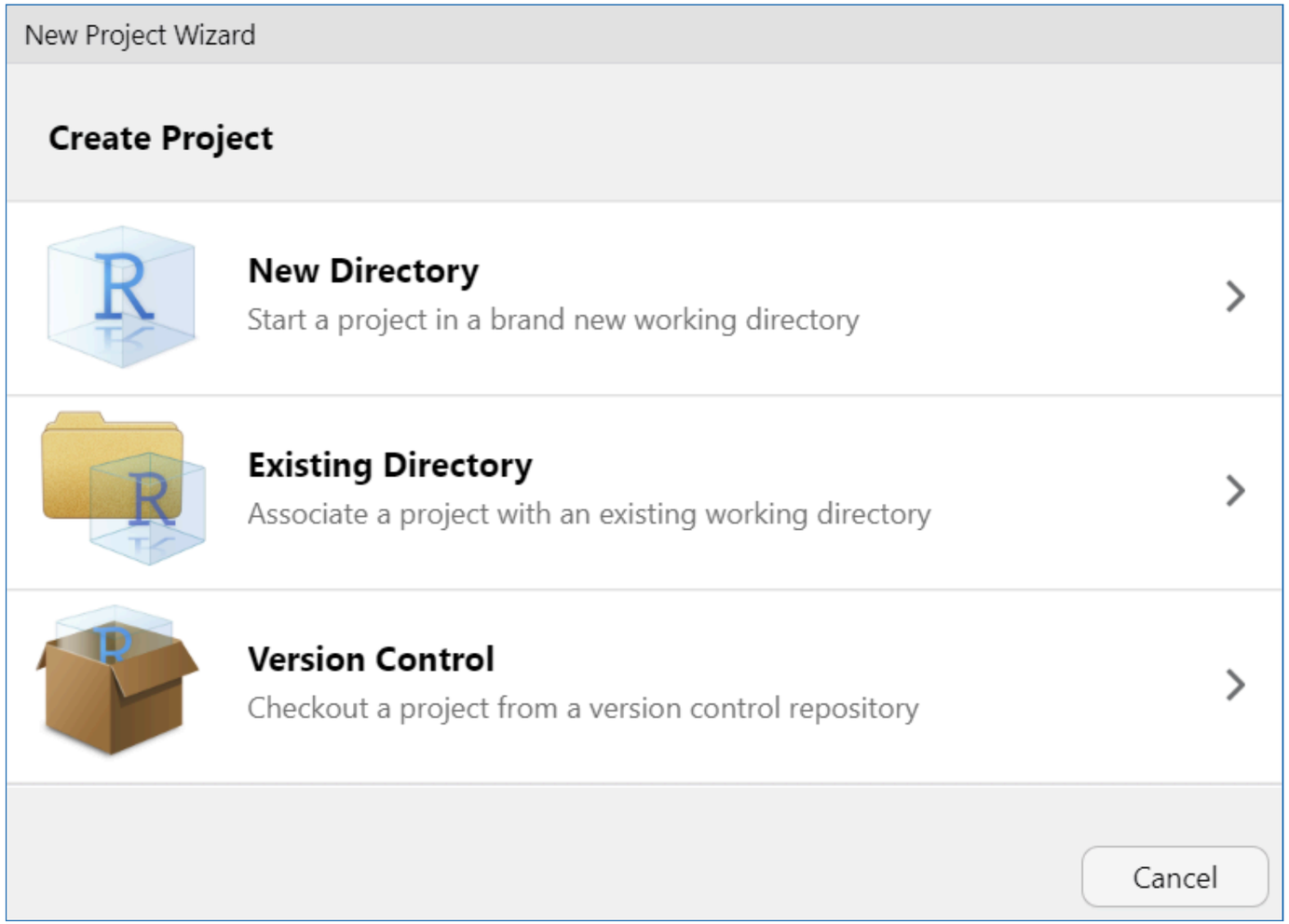
Congratulations! You have just created your first **GitHub** repository.
You should now see a page like the one shown below.



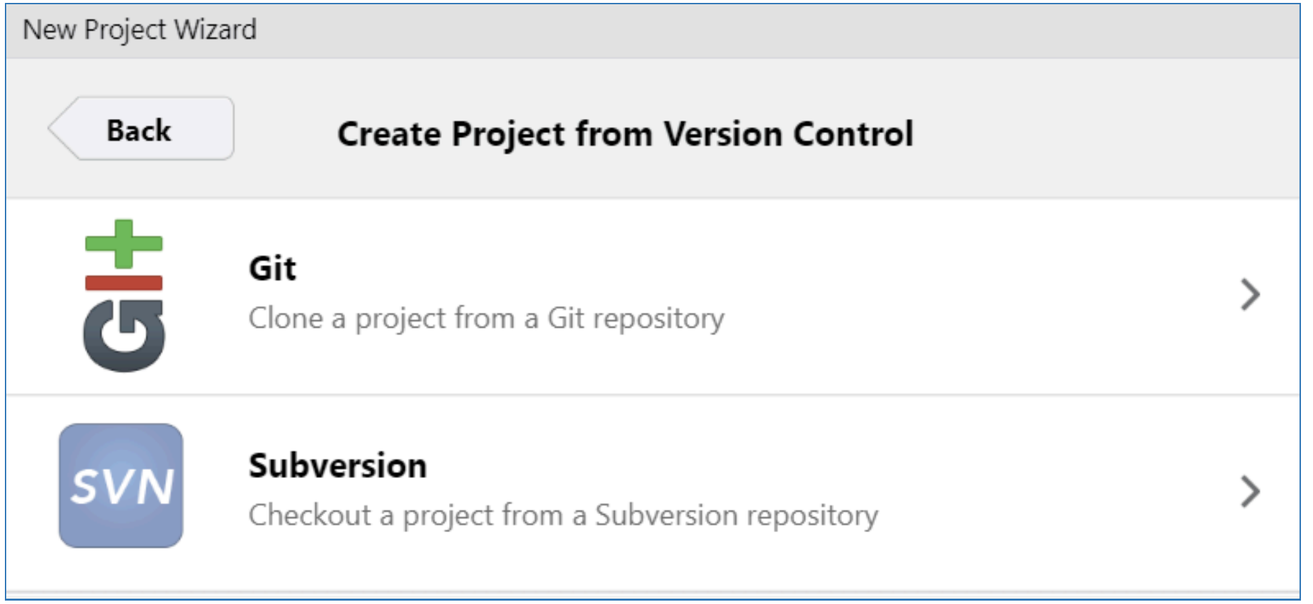
Select `**HTTPS**` and copy the link to your repository.

Set Up a New R Project in RStudio

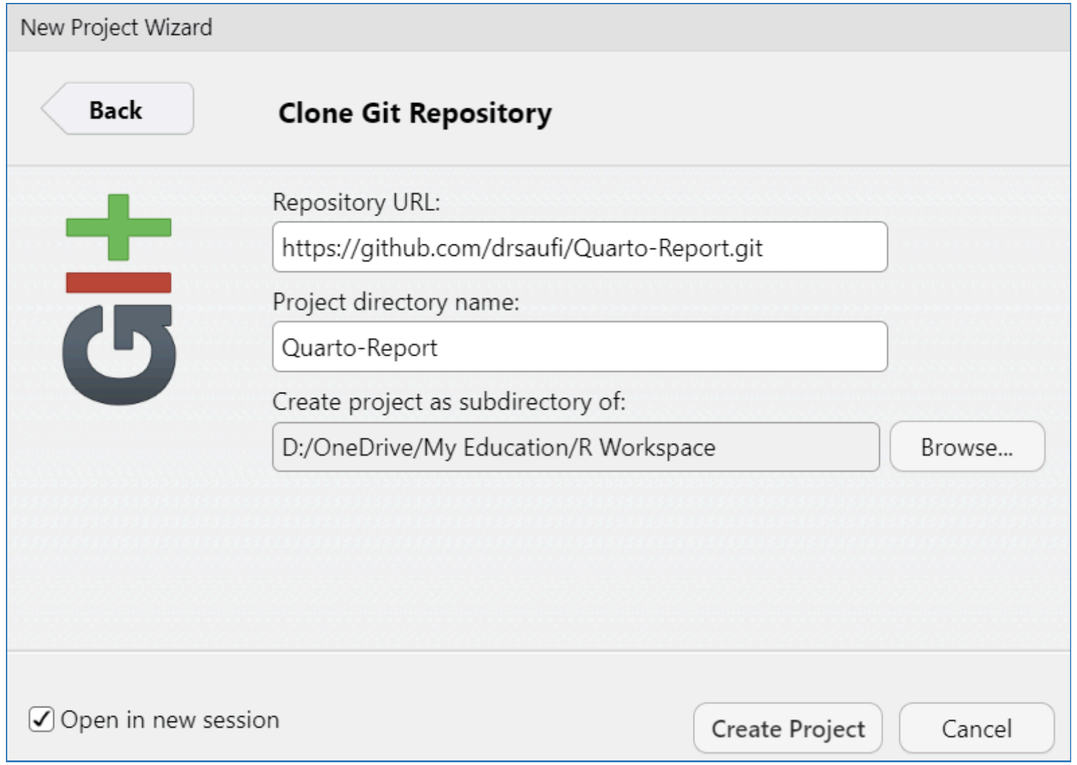
In **RStudio**, go to **`File → New Project`**.



Select **`Version Control`** from the options.



Choose `**Git**` as the version control system.



In the `**Repository URL**` field, paste the URL of your new **GitHub** repository.

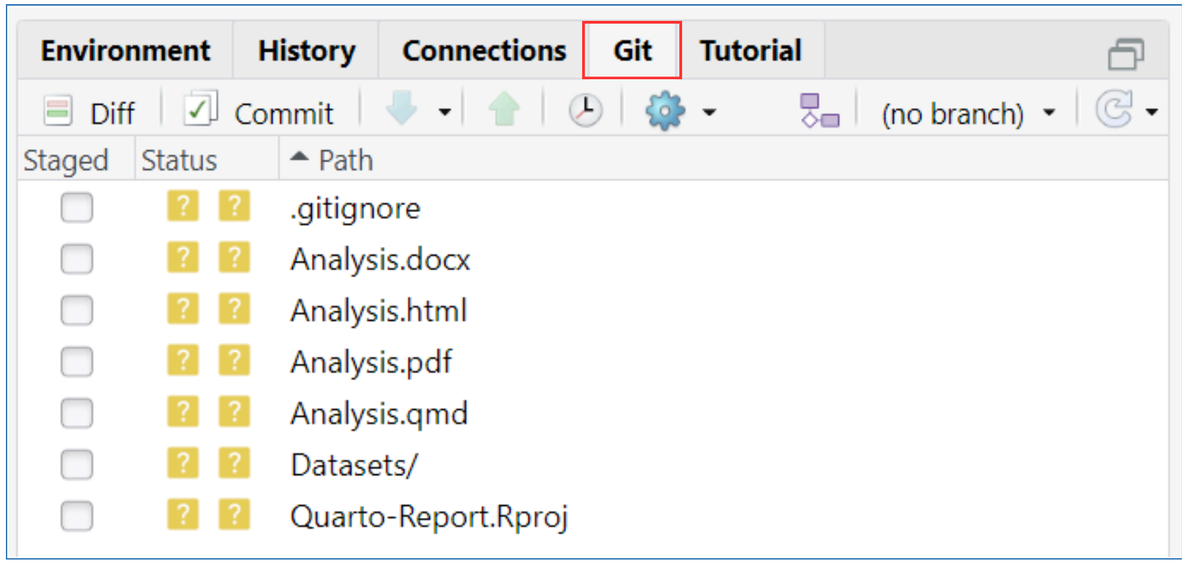
It is recommended that the name of the project directory be similar to your **GitHub** repository, although this will not affect the synchronization between your **R** project and **GitHub** repository.

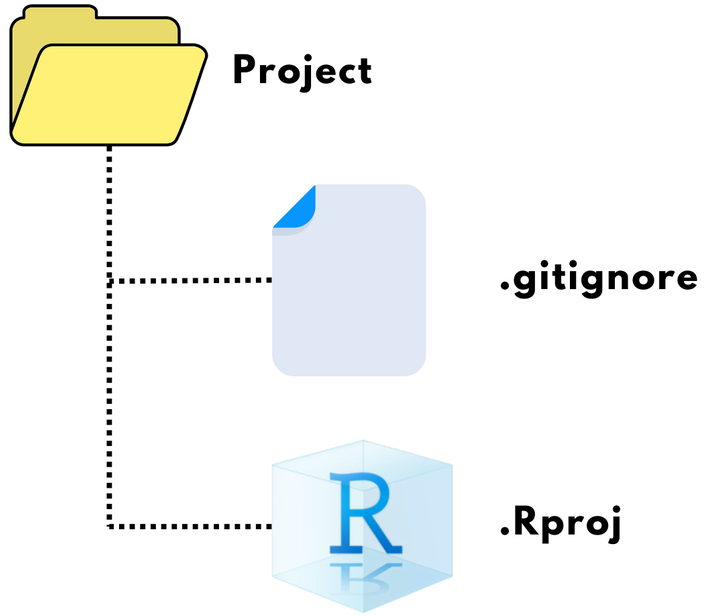
If you have an existing **R** project with the same name, avoid conflicts by renaming the new project directory with a slightly different name.

Click the `**Create Project**` button to proceed.

Finally, you have created **R** project that is linked to your **GitHub** repository. Congratulations!

In the working directory or folder, you will find the following files:

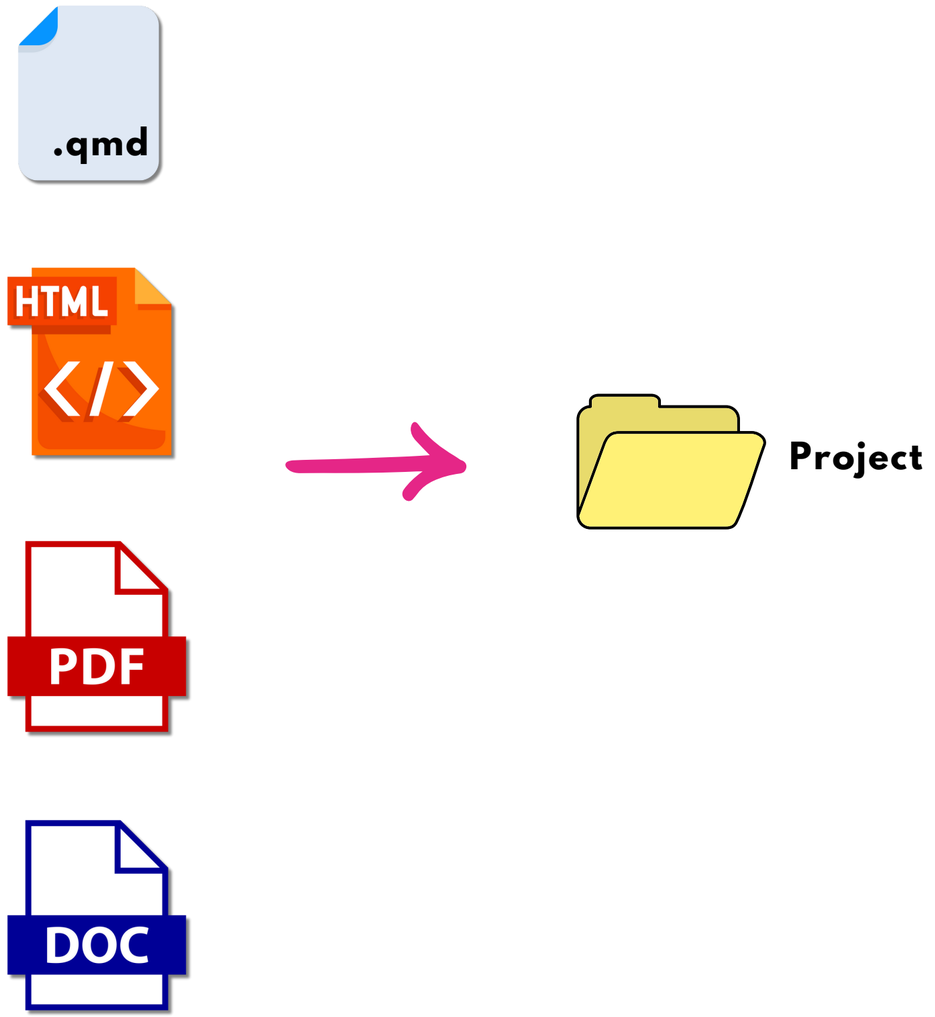




The **Git** pane will appear in your **RStudio** environment.

You can now start creating your report. In **RStudio**, click on the **`File`** menu, then choose **`Quarto Document`** to begin crafting your code.

If you already have an existing **R** project, you can move your **R** documents and files into this new working directory.



6 Commit, Push and Pull

Linking a **GitHub** repository to an **R** project does not automatically upload your **R** files and documents to the repository.

To synchronize your **R** project with the **GitHub** repository, you need to perform the following actions in the **Git** pane: **commit**, **push**, and **pull**.

Commit is the process of recording changes to your project's files. However, these changes remain only in your local repository until you push them to the **GitHub** repository.

Push command is used to upload your committed changes from your local repository to the **GitHub** repository.

Pull command is used to fetch changes from the **GitHub** repository into your local repository, which will be reflected in your **R** project.

Warning

If you save an **R** project with **Git** in cloud-based services like **Dropbox** or **OneDrive**, potential issues may arise during file synchronization.

Cloud-based services like **Dropbox** or **OneDrive** continuously sync files between your local machine and the cloud.

This real-time syncing can cause conflicts, leading to corrupted files or the creation of conflict copies.

Recommendation

To avoid these issues, it's best to keep your **R** projects with **Git** in a local directory that is not linked to any cloud-based sync service.

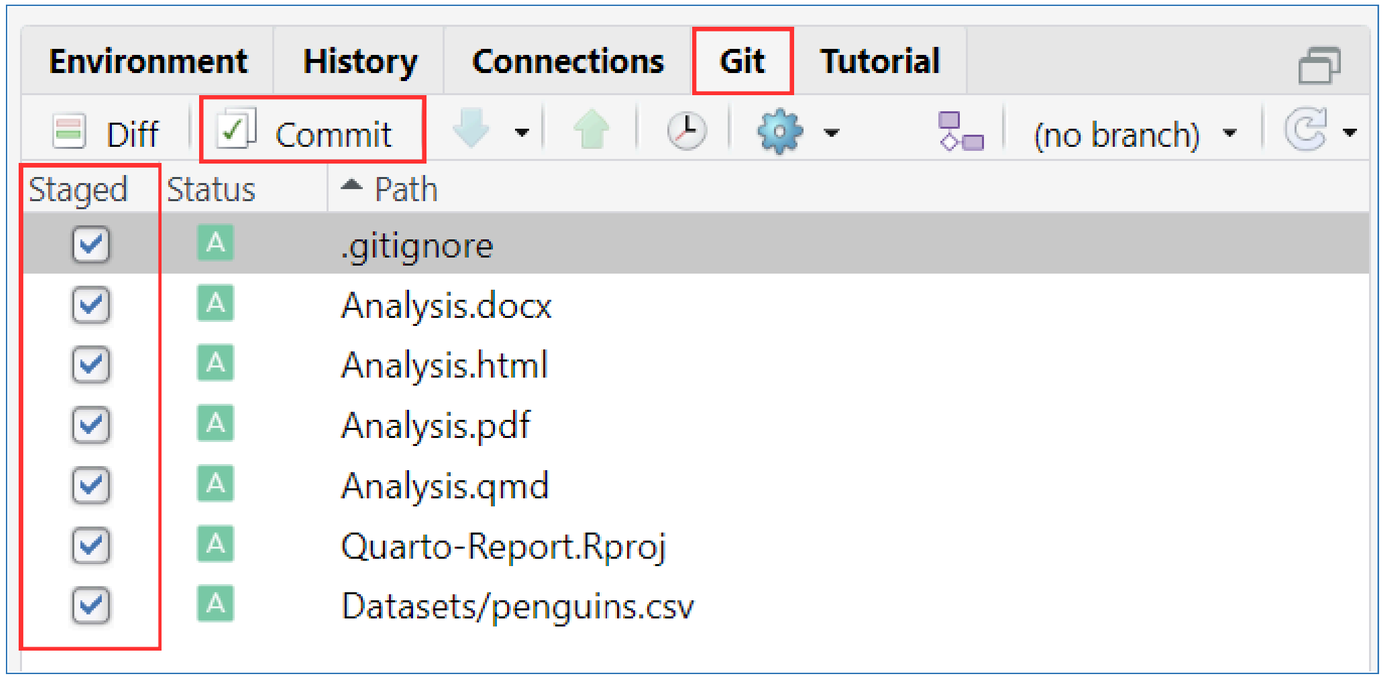
However, if your project is already stored in a cloud-based service or you choose to keep it there, it is **highly recommended** to pause the syncing before performing any **Git** operations.

This precaution helps prevent conflicts and ensures the integrity of your files.

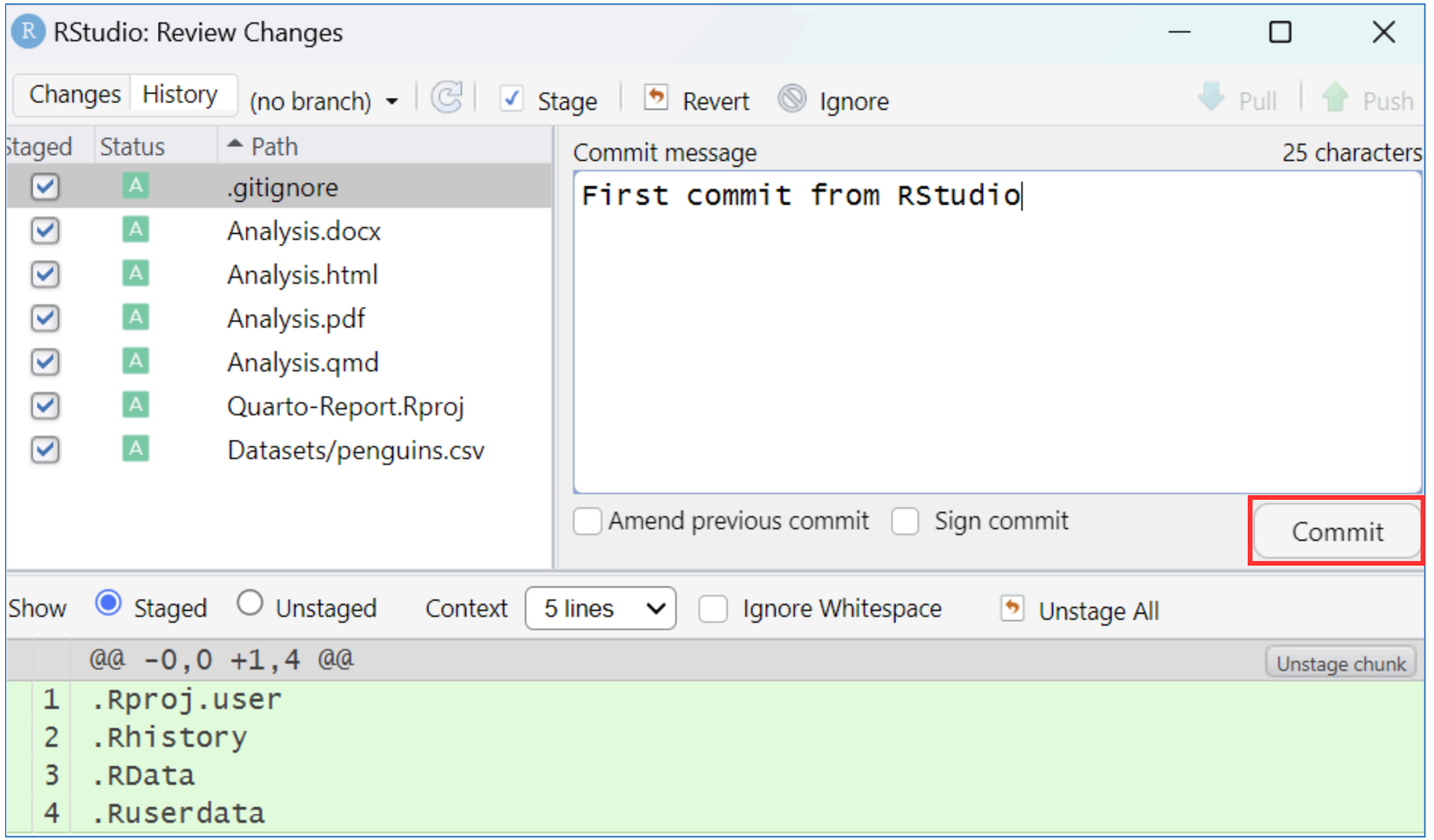
Commit

You are encouraged to **commit**, which means saving your changes every time you complete valuable work. It is similar to saving your progress when writing a **Word** document.

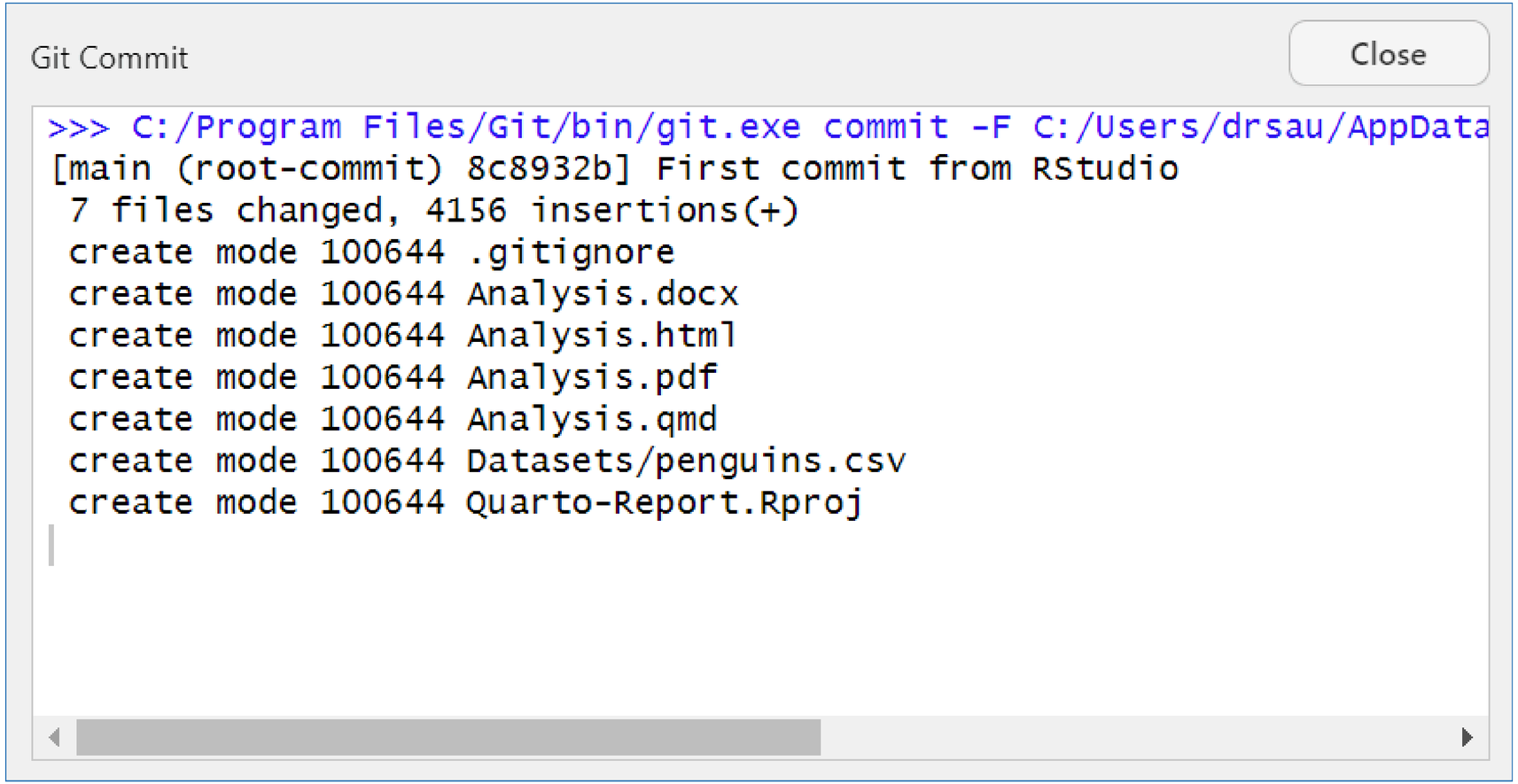
In **RStudio**, click the **`Git`** tab in the upper right pane. Check the **`Staged`** box for any files you want to **commit**, then click **`Commit`**.



A pop-up window will appear.



Make it a habit to write a short message in the **`Commit message`** box, then click **`Commit`**.



You will see a report message confirming that you have successfully committed your changes.

You can close these pop-up windows and return to **RStudio**.

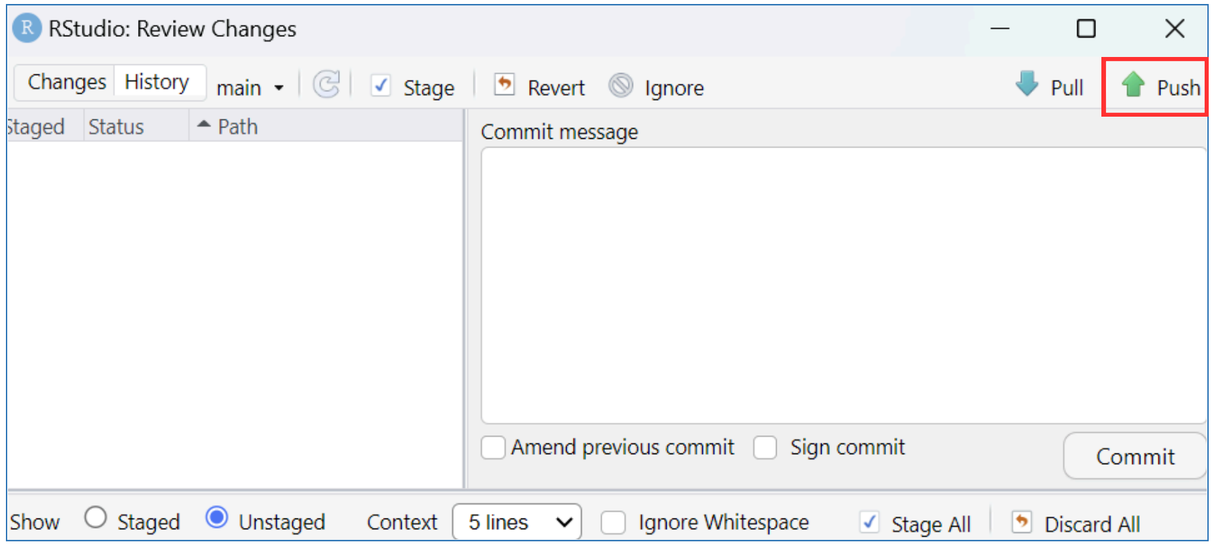
Push

While working on your report in **RStudio**, you might **commit** several times.

However, these changes are only stored in your local repository and are not yet reflected in your **GitHub** repository.

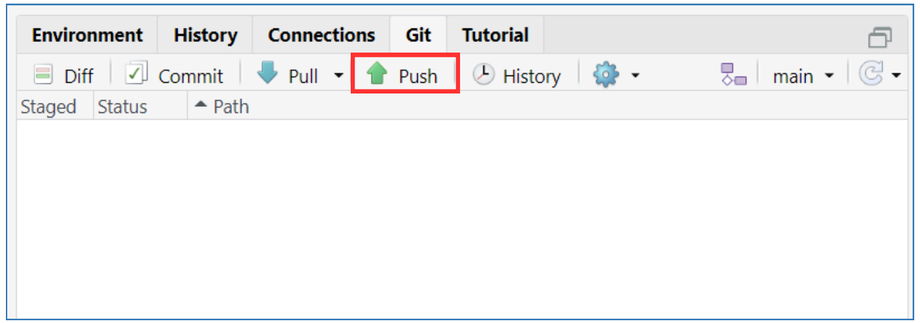
Now, it is time to **push** your local changes to **GitHub**.

If you still have the pop-up window open, you will notice it is blank, indicating that there are no changes left to **commit**.

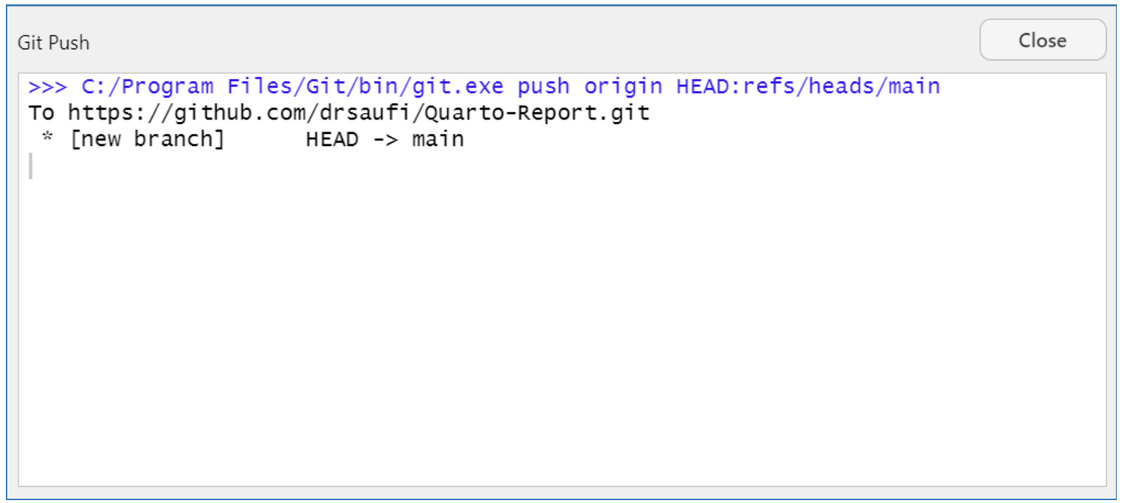


Proceed by clicking the `**Push**` icon, which is the up arrow in the upper right corner.

Alternatively, you can click the `**Push**` button in the `**Git**` pane.

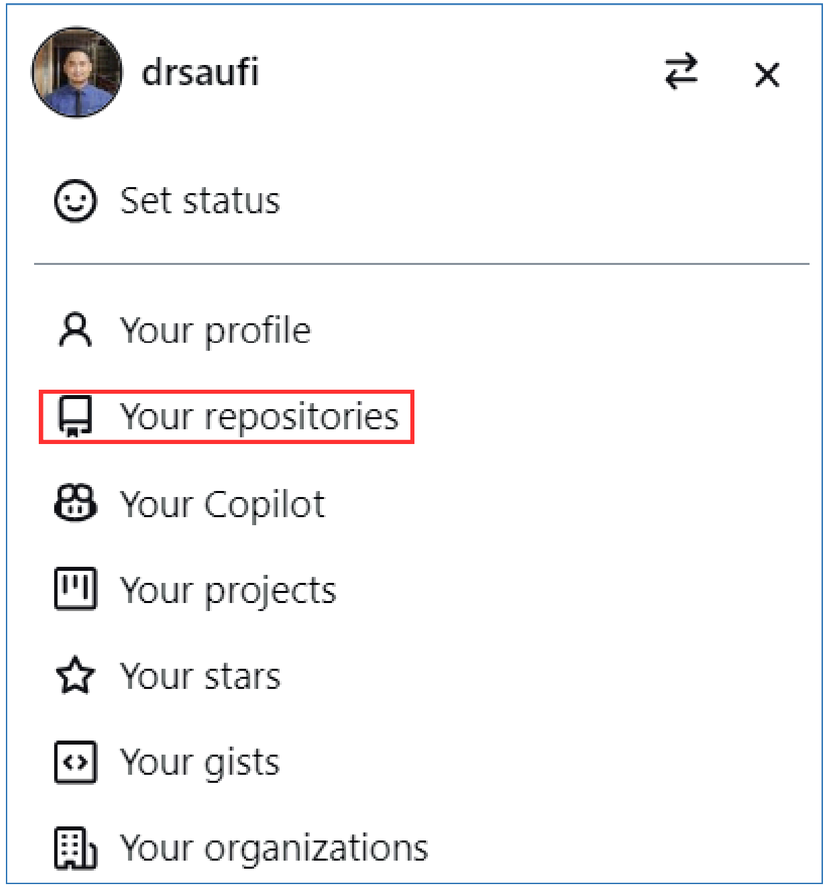


RStudio will display a report message like the one below, confirming that the changes have been successfully pushed to your **GitHub** repository.

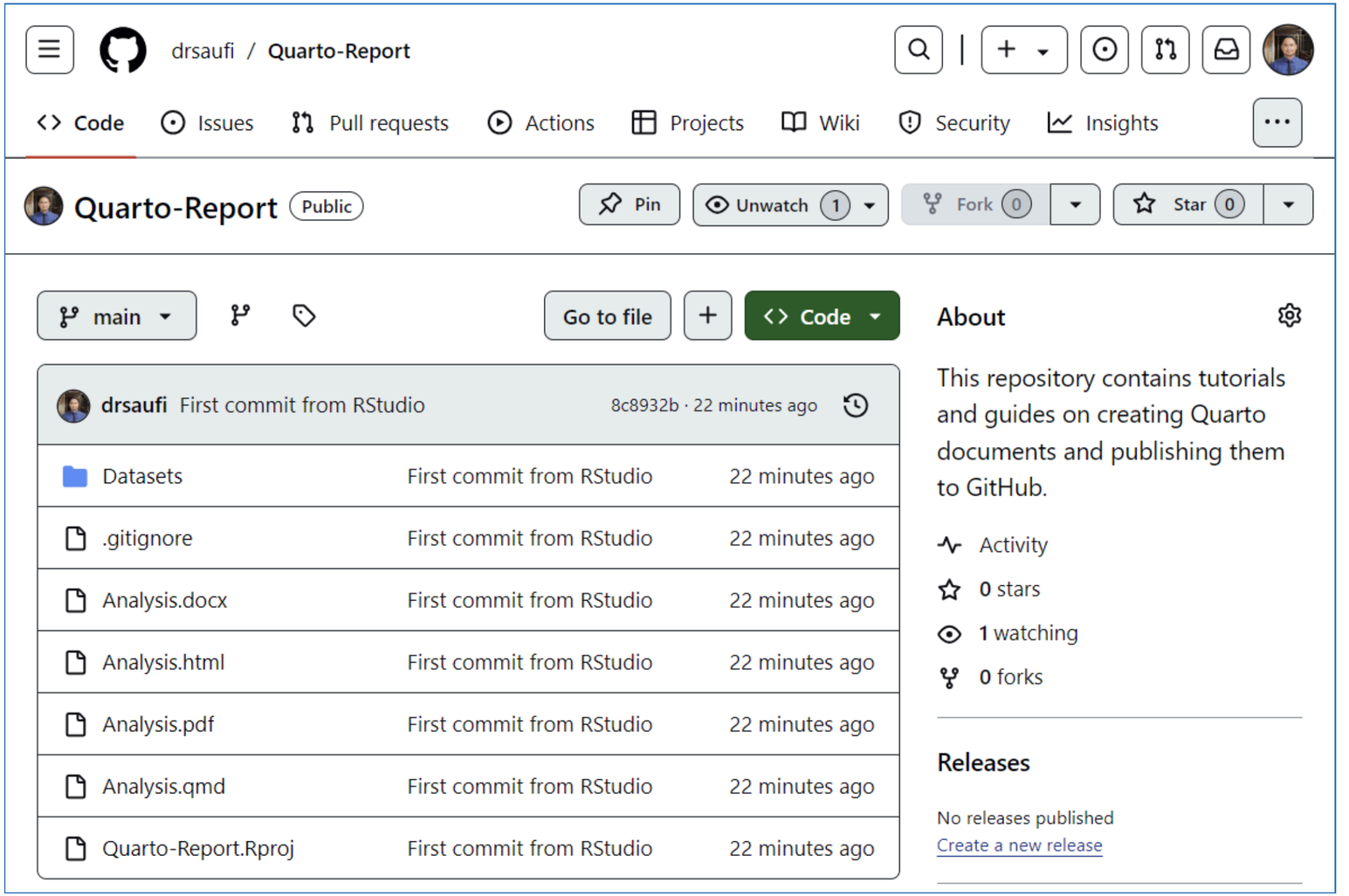


To verify these changes, return to your **GitHub** repository in your browser.

If you've already closed the browser or can not find your repository, simply go to your **GitHub** profile and click `**Your repositories**`. Then, select your **GitHub** repository.



In your **GitHub** repository, you will see that all your **R** files and documents are safely stored.



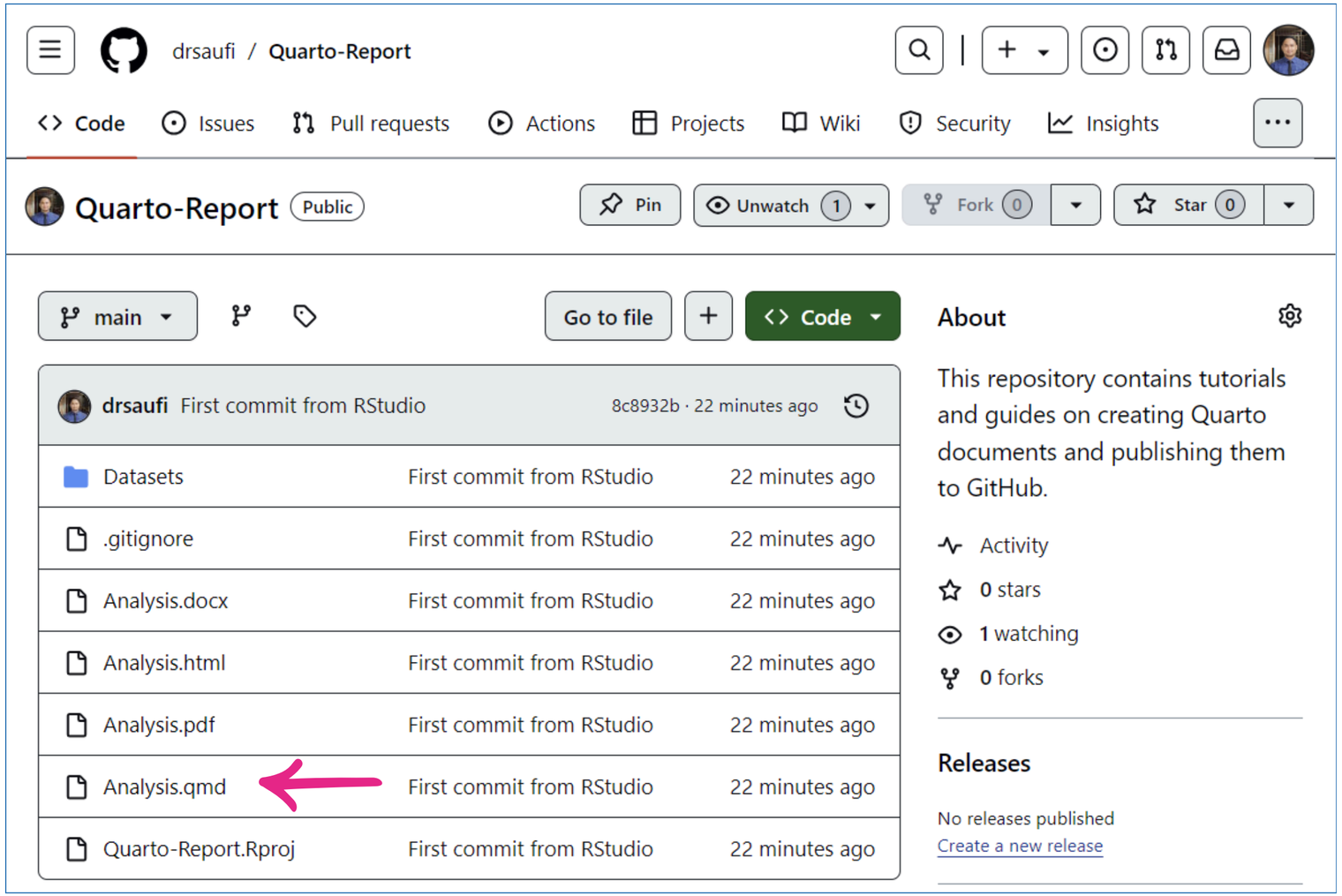
Pull

This action is necessary when there are changes or updates in your **GitHub** repository that you want to be reflected in your local repository in **RStudio**.

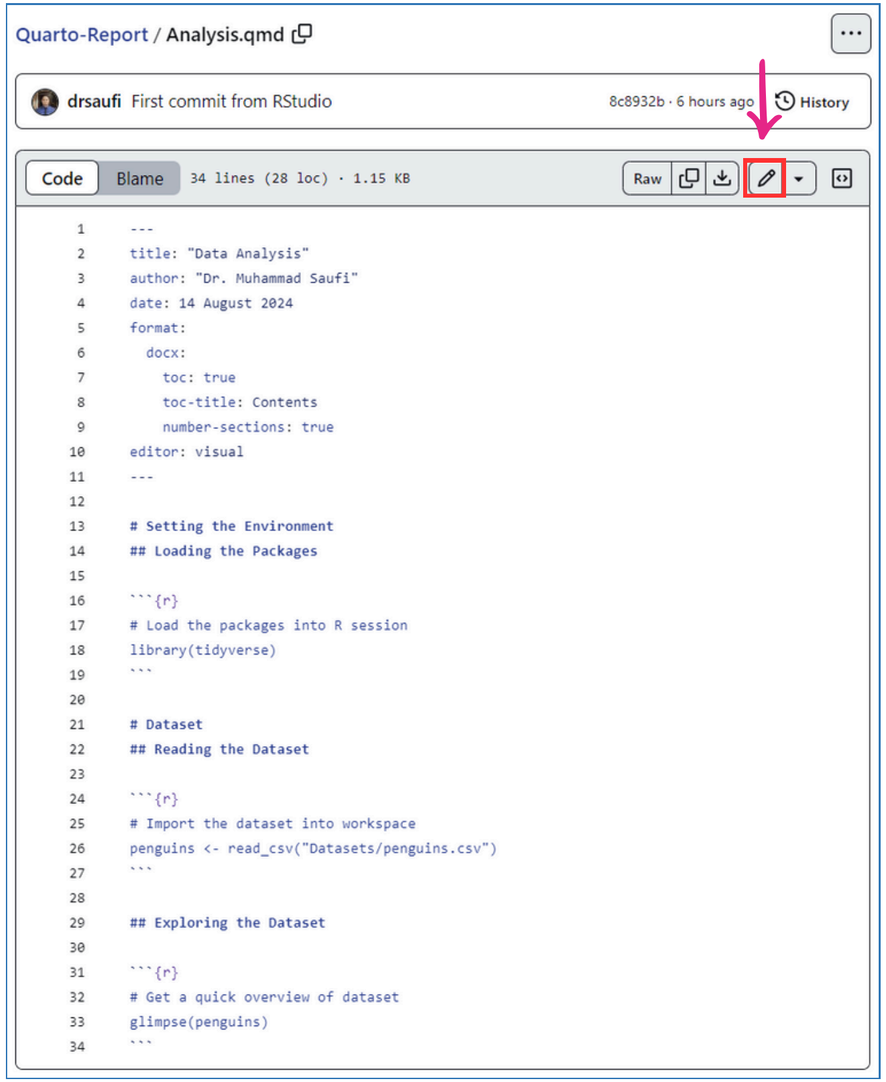
Imagine in the future, you are working on a project with one or more collaborators using a shared **GitHub** repository.

Any progress and updates to the project will be pushed to the repository. You will need to pull these changes into your **RStudio** to keep your local version up to date.

Now, let's make some changes in your **GitHub** repository.
From the file listing, click on your **Quarto** document.



In the upper right corner, click on the pencil icon labeled `**Edit this file**`.

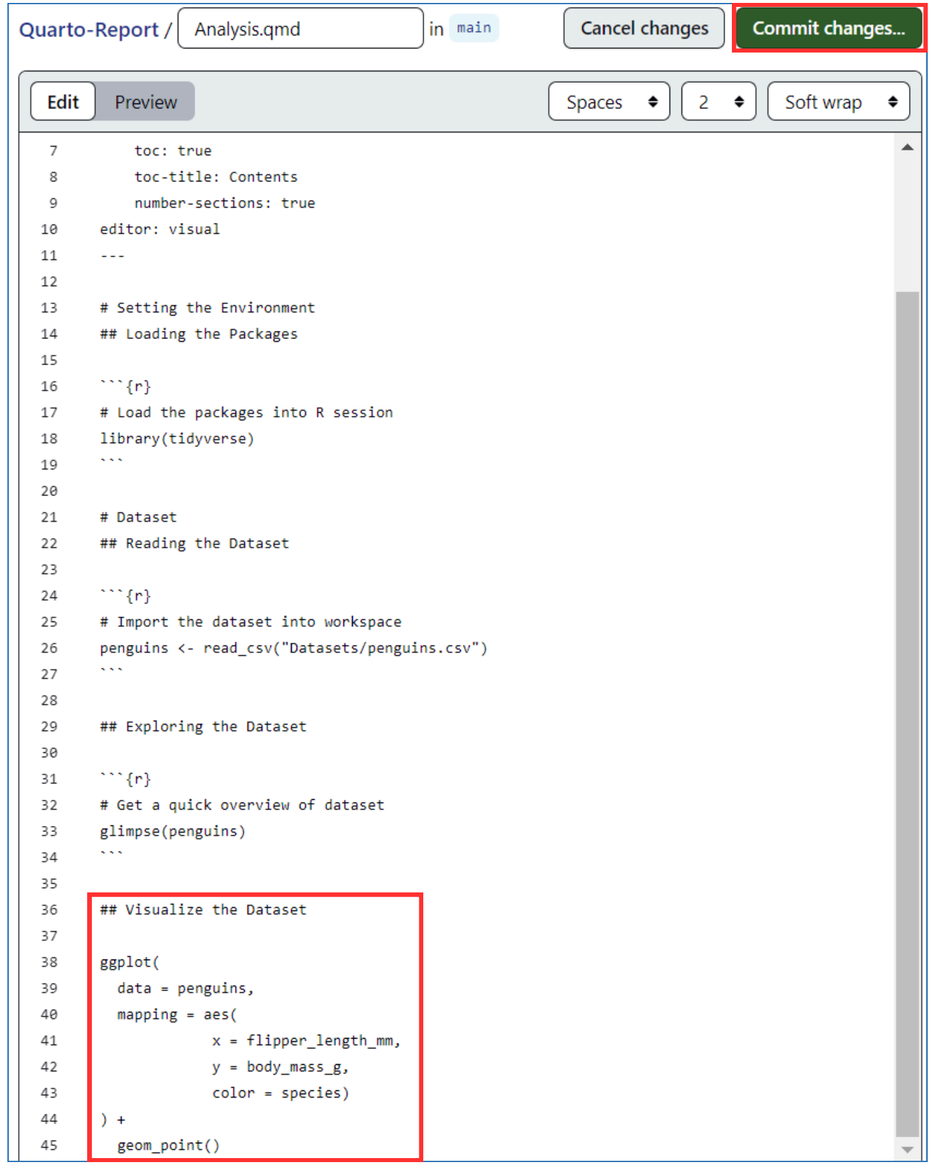


Add the following code. Copy the heading, code chunk, and **R** code, then paste them at the bottom of the **Quarto** document in **GitHub** repository.

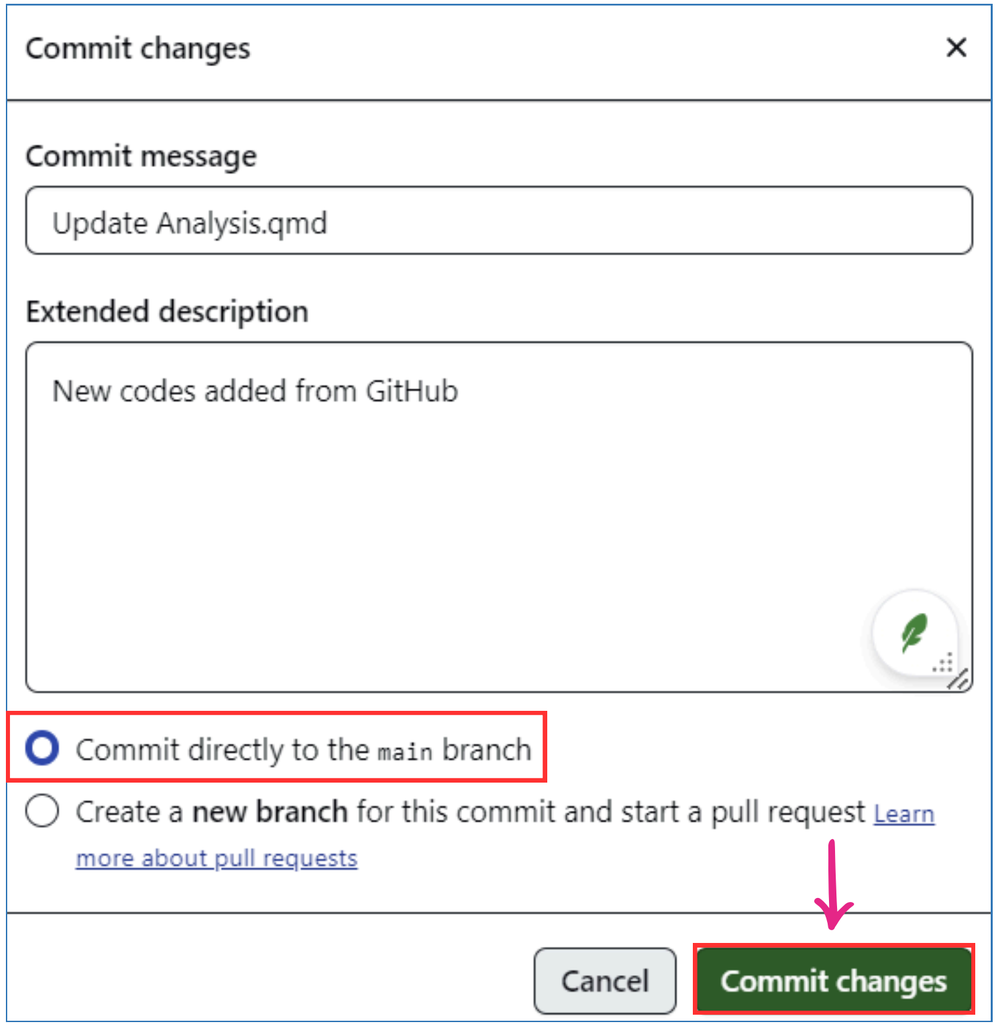
````
## Visualize the Dataset

```{r}
ggplot(
  data = penguins,
  mapping = aes(
              x = flipper_length_mm,
              y = body_mass_g,
              color = species)
) +
  geom_point()
```
````

After adding the code, click the **`Commit changes`** button in the upper right corner.



A window will appear where you can optionally add a short description, such as **`New code added from GitHub`.**

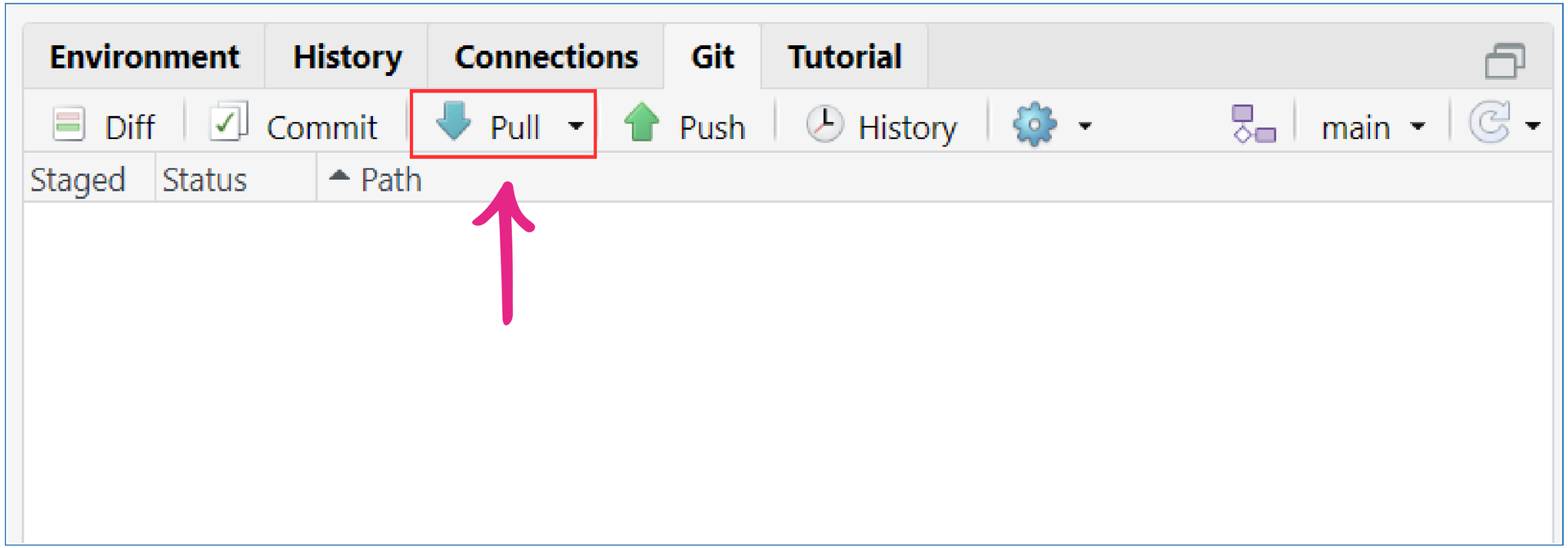


Make sure to select the **`Commit directly to the main branch`** option, then click the **`Commit changes`** button.

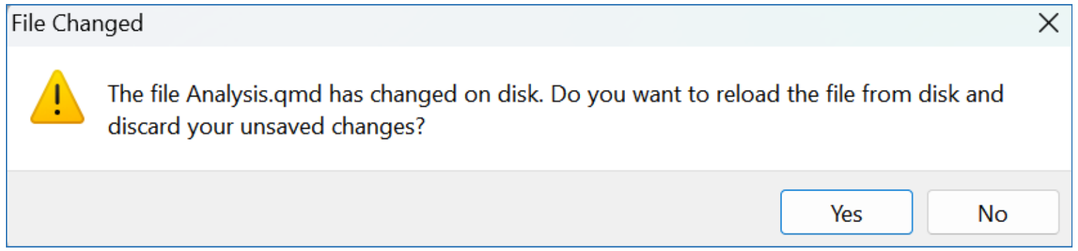
Now that you have committed the changes to your **GitHub** repository, they are not yet reflected in your local repository within **RStudio**.

To do this, you need to perform a pull action from **RStudio**.

Back in **RStudio**, go to the **`Git`** pane and click the **`Pull`** button.

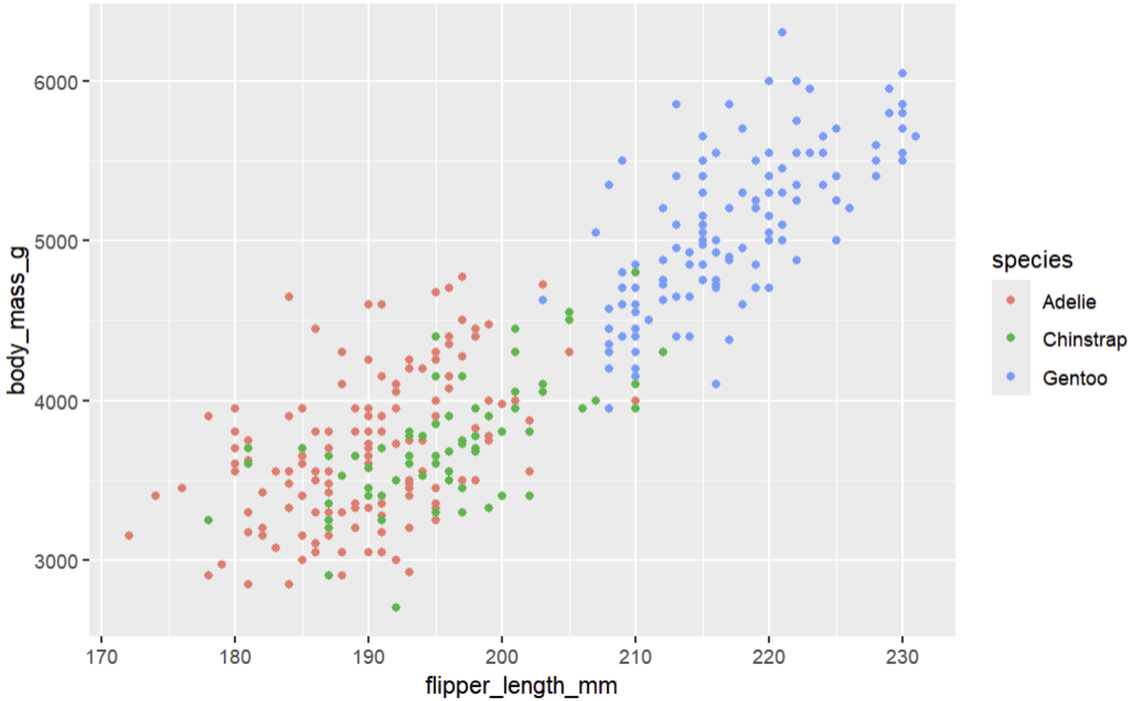


A warning message may appear.
Click `**Yes**` to accept the changes.



Now, inspect your **Quarto** document in **RStudio**, and you will see the new code added at the bottom of the document.

You can run this new code in **RStudio**, and it will generate a scatter plot based on the dataset.



7 Publishing Reports as Websites

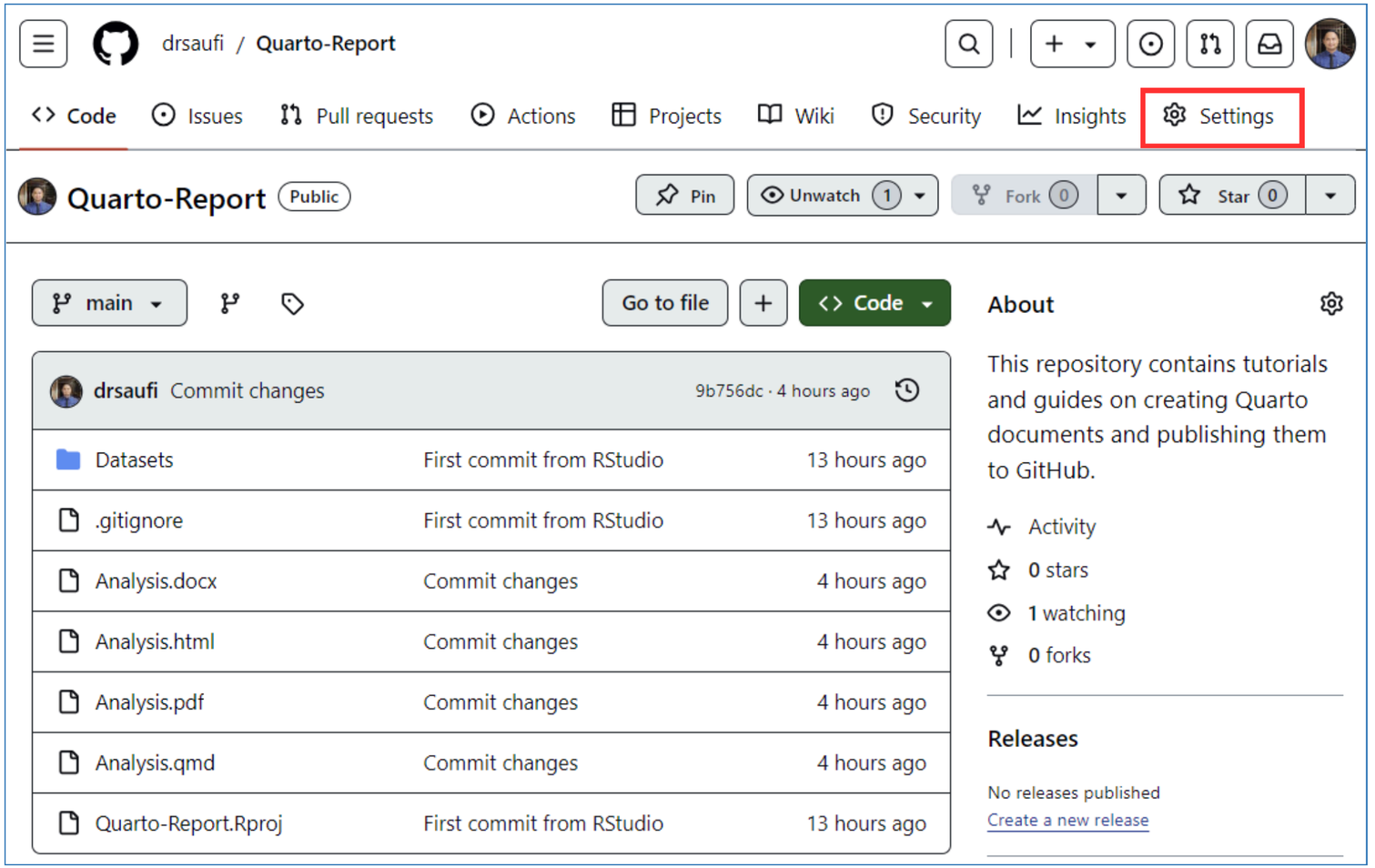
Your work is now accessible through your **GitHub** repository, but currently, it is only available in a text format.

GitHub has the capability to transform an **HTML** report into a website, accessible via a unique **URL**.

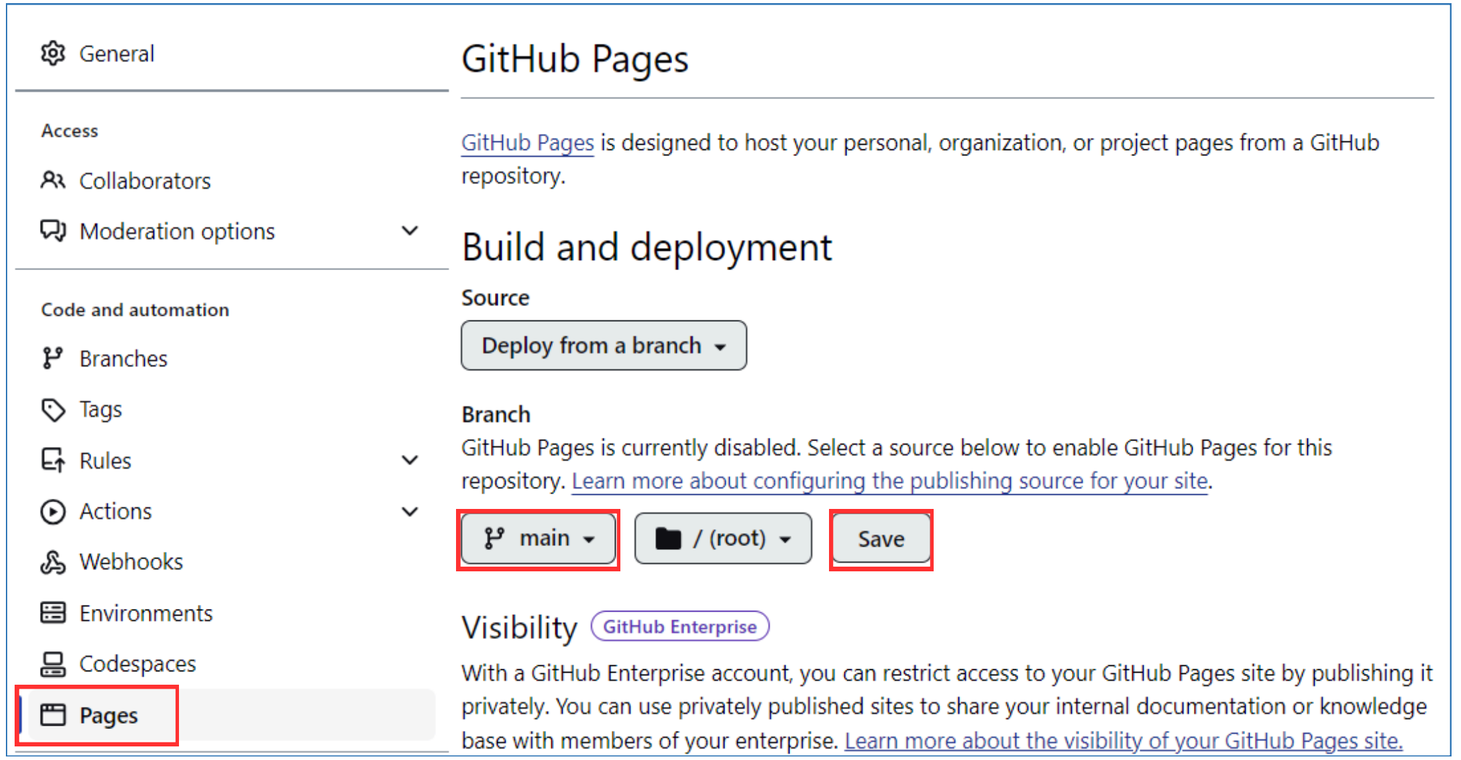
Before proceeding with this transformation, it is recommended to **commit** any changes in your **RStudio** environment and push them to your **GitHub** repository.

To enable the website feature in your **GitHub** repository, follow these steps:

In your **GitHub** repository, click on **`Settings`**.



In the left pane, select `**Pages**`.



In the `**Branch**` section, choose `**main**` instead of `**none**` from the dropdown menu, then click `**Save**`.

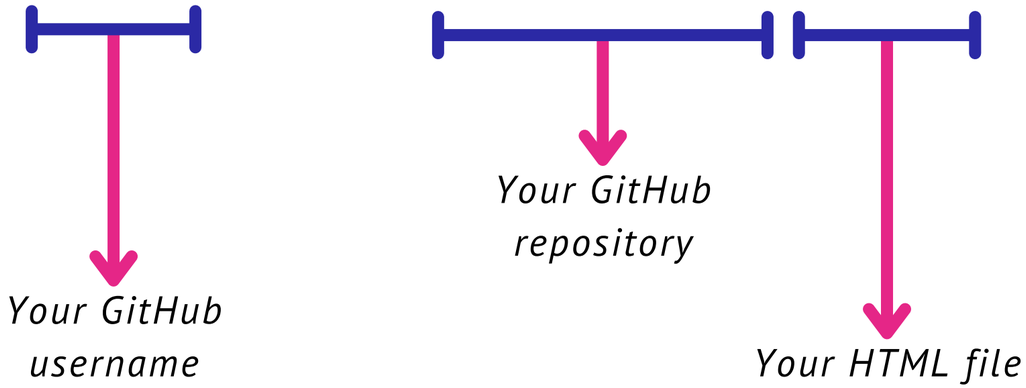
Wait for approximately 2 minutes for **GitHub** to transform your **HTML** report into a website.

Congratulations! Your **HTML** report is now accessible as a website.

The **URL** for the website is based on your **GitHub** repository's **URL**. For example:

GitHub repository URL:
https://github.com/drsaufi/Quarto-Report

Quarto HTML URL:
https://drsaufi.github.io/Quarto-Report/Analysis



A few important considerations:

1. You can have multiple **Quarto HTML** files within your repository.
For instance, if you have another **Quarto HTML** report named **`Analysis-2`**, the **URL** would be:

   https://drsaufi.github.io/Quarto-Report/Analysis-2

2. If your **HTML** file is stored inside a folder (e.g., a folder named **`R`**) within your repository, the **URL** need to include that folder. Your **URL** would be:

   https://drsaufi.github.io/Quarto-Report/R/Analysis

3. Remember that the **URL** is case-sensitive, and the **HTML** file name cannot contain spaces (e.g., **`Analysis 2`**). Instead, use hyphens, such as **`Analysis-2`**.


Thank
You