



REPRODUCIBLE REPORT

Using R Quarto

14th August 2024



Dr Muhammad Saufi bin Abdullah

Doctor of Public Health (Epidemiology) candidate
from the Department of Community Medicine at the
School of Medical Sciences, Universiti Sains Malaysia.

PART 1

Creating Reports with R Quarto

1.1 Overview of Quarto

Quarto allows you to create documents that combine data analysis, results, and explanations all in one place. It ensures that your work is easily reproducible and can be converted into various formats like **HTML**, **PDF**, and **Word** files.

Quarto can be used in three main ways:

1
“
To share clear conclusions with decision-makers without including all the technical details.
”

2
“
To collaborate with other data researchers by sharing both findings and the underlying code.
”

3
“
As a digital notebook to document the entire data analysis process.
”

Quarto Document

To begin learning **Quarto**, let's create a new working directory and name this project '**Quarto Report**'.

In the **File** menu, click and select **Quarto Document**. Name the document '**Data Analysis**' and choose **HTML** as the output format.

New Quarto Document

Document
Presentation
Interactive

Title: Data Analysis

Author: (optional)

☒ HTML
Recommended format for authoring (you can switch to PDF or Word output anytime)

☐ PDF
PDF output requires a LaTeX installation (e.g. <https://yihui.org/tinytex/>)

☐ Word
Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux)

Engine: Knitr

Editor: ☒ Use visual markdown editor ?

? [Learn more about Quarto](#)

Create Empty Document Create Cancel

Save the Document

Save the file within the working directory by clicking **`File → Save`** or using the keyboard shortcut **`Ctrl + S`** (Windows) or **`Cmd + S`** (Mac).

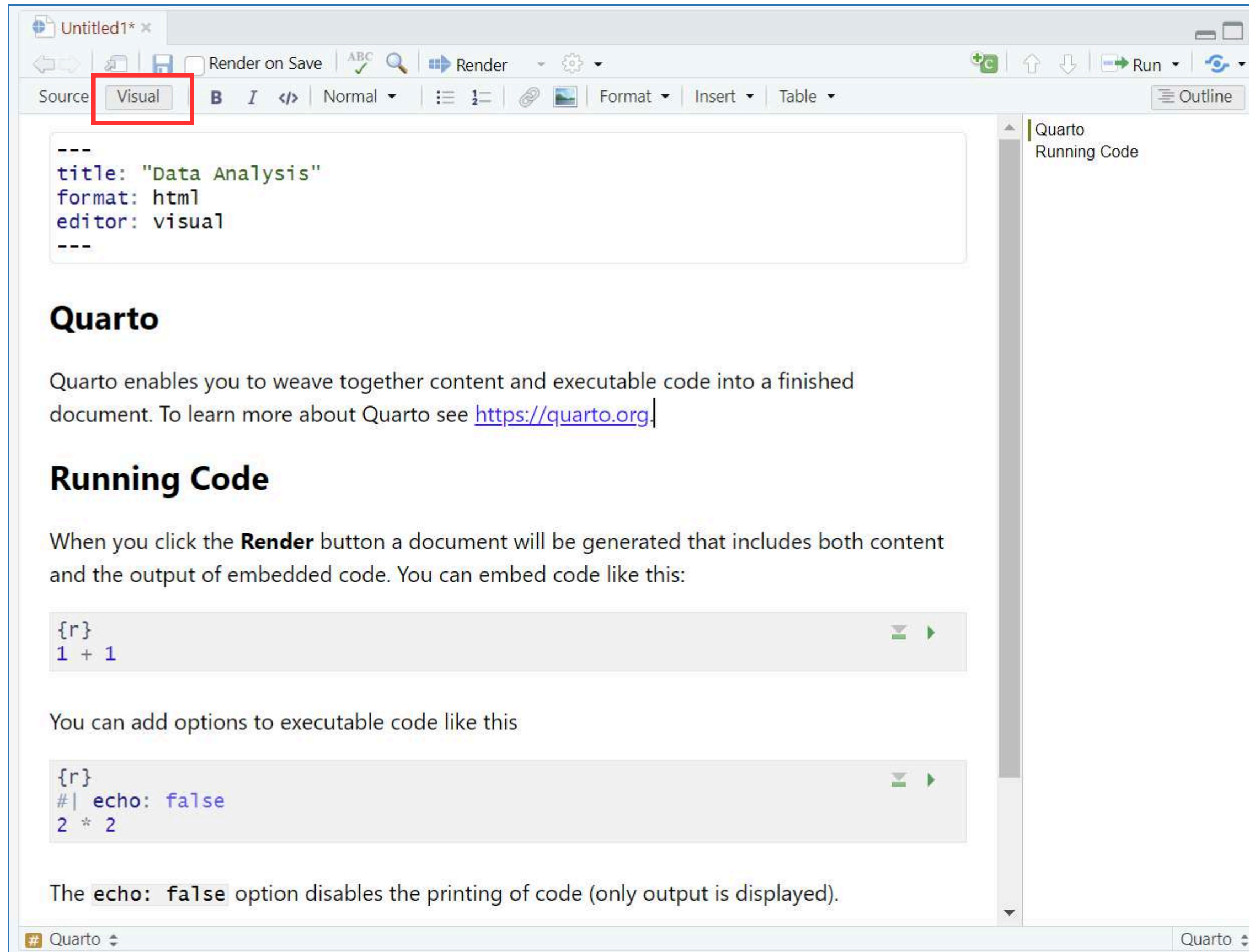
Name the file **`Analysis`**. Note that it will have a **` .qmd `** extension.

Visual and Source Editors

RStudio provides two editors for **Quarto** documents: the **Visual Editor** and the **Source Editor**.

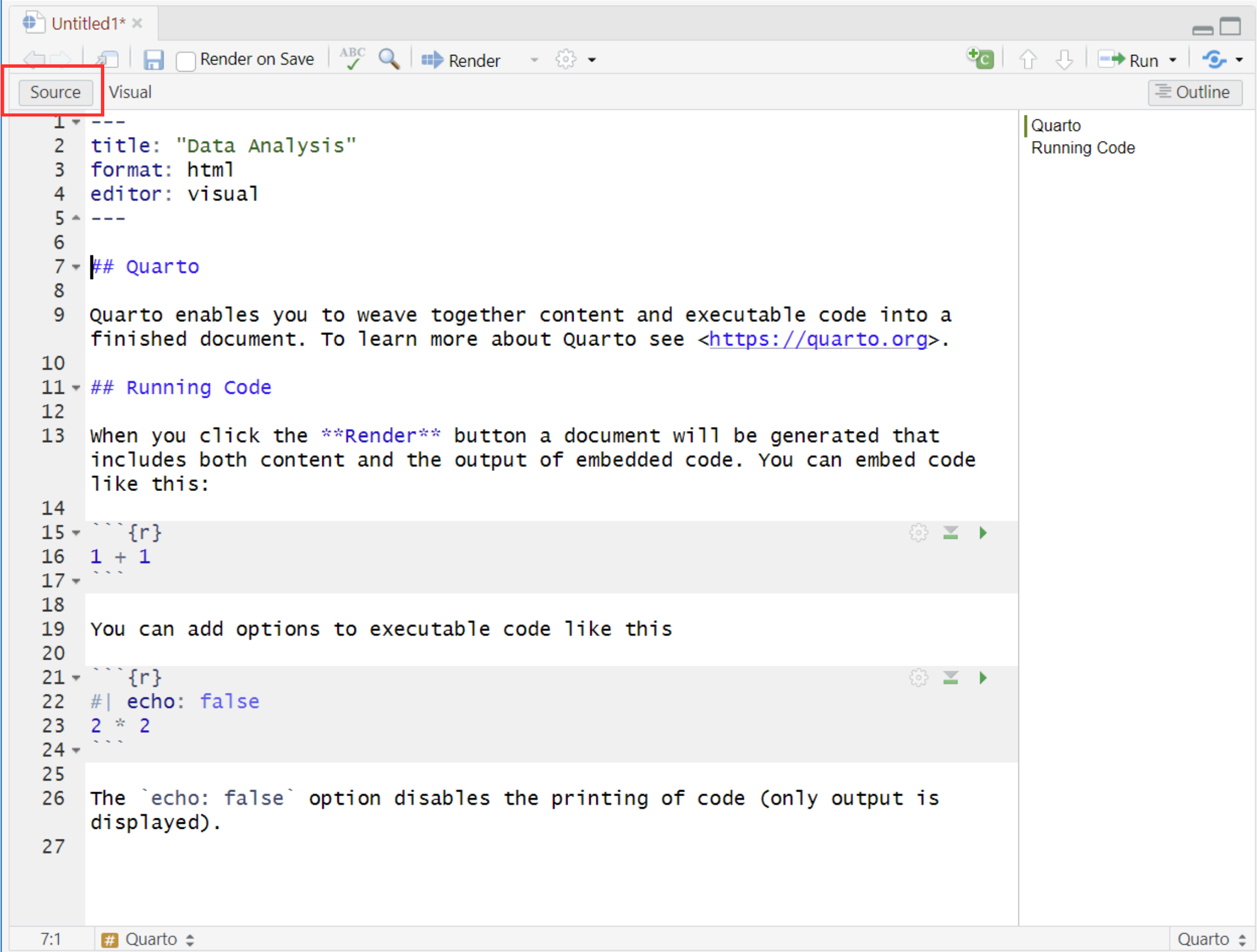


*You can seamlessly switch between these two editors using the buttons at the top left of the **Source** pane.*



***Visual Editor** offers an intuitive interface that allows you to format text and insert elements easily.*

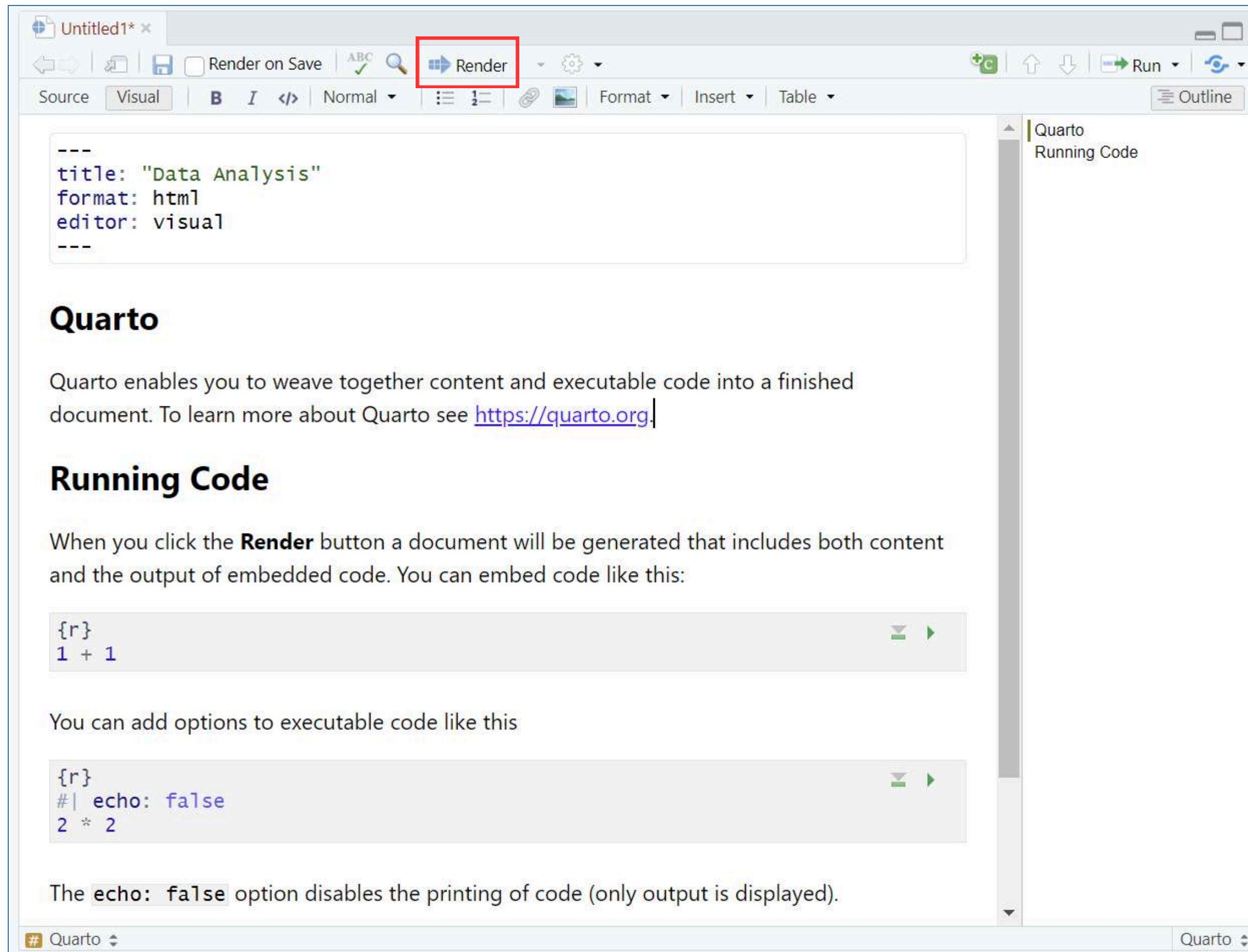
Source Editor, on the other hand, lets you manually enter syntax, giving you finer control over the document's formatting.



The screenshot displays the Quarto Source Editor interface. The top toolbar includes a 'Source' button, which is highlighted with a red rectangle. The editor is currently in 'Source' mode, as indicated by the 'Source' tab being active. The document content is as follows:

```
1 ---
2 title: "Data Analysis"
3 format: html
4 editor: visual
5 ---
6
7 ## Quarto
8
9 Quarto enables you to weave together content and executable code into a
10 finished document. To learn more about Quarto see <https://quarto.org>.
11
12 ## Running Code
13
14 When you click the Render button a document will be generated that
15 includes both content and the output of embedded code. You can embed code
16 like this:
17
18 ```{r}
19 1 + 1
20 ```
21
22 You can add options to executable code like this
23
24 ```{r}
25 #| echo: false
26 2 * 2
27 ```
28
29 The `echo: false` option disables the printing of code (only output is
30 displayed).
```

The right sidebar shows the 'Quarto Running Code' panel, which is currently empty. The status bar at the bottom indicates the current position is 7:1 in the Quarto document.



Render the Document

The ``rmarkdown`` package is required to render a **Quarto** file. If the package is not already installed on your machine, you can easily do so. Simply copy the code below, paste it into the **Console** pane, and press **`Enter`**.

In Console pane:

```
install.packages("rmarkdown")
```

By clicking the **`Render`** button at the top of the **Source** pane, you can transform the **Quarto** document into an **HTML** document.

Your browser should automatically open the **HTML** document in a new tab.

If it doesn't, you can find the **HTML** file in the same folder where your **Quarto** document is located.

*You should see the **HTML** document in your browser.*

Data Analysis

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see <https://quarto.org>.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
1 + 1
```

```
[1] 2
```

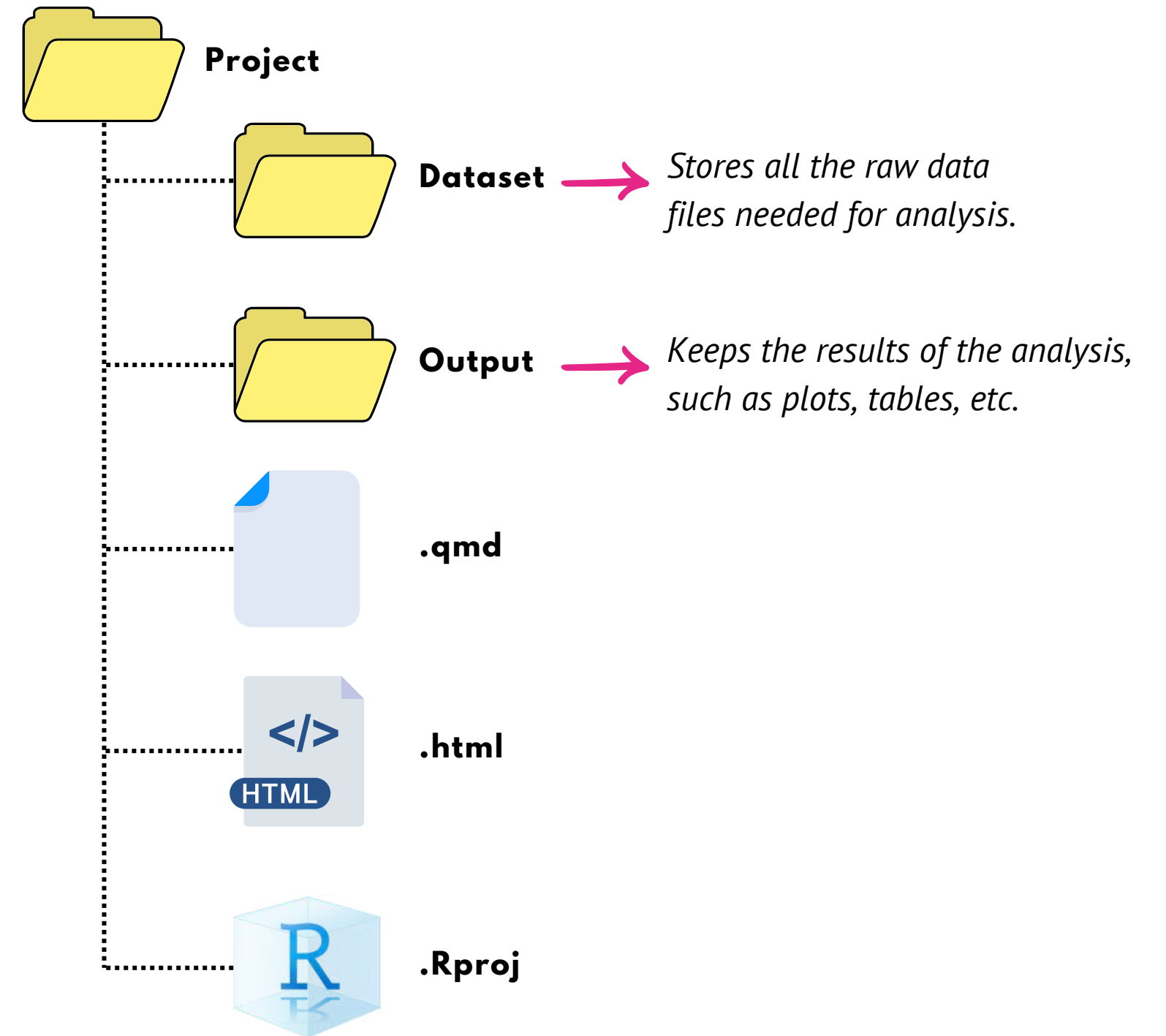
You can add options to executable code like this

```
[1] 4
```

The `echo: false` option disables the printing of code (only output is displayed).

You will find these files within your working directory.

Additionally, you can create **`Dataset`** and **`Output`** folders to organize your datasets and the results of your analysis as needed.



YAML Header

At the beginning of the document, you will notice a block of text enclosed by triple dashes (`---`).

This is the **YAML** header, which defines the settings and configurations for the document.

You can customize this header by adding details such as the author's name and the date.

```
---  
title: "Data Analysis"  
author: "Dr. Muhammad Saufi"  
date: 14 August 2024  
format: html  
editor: visual  
---
```

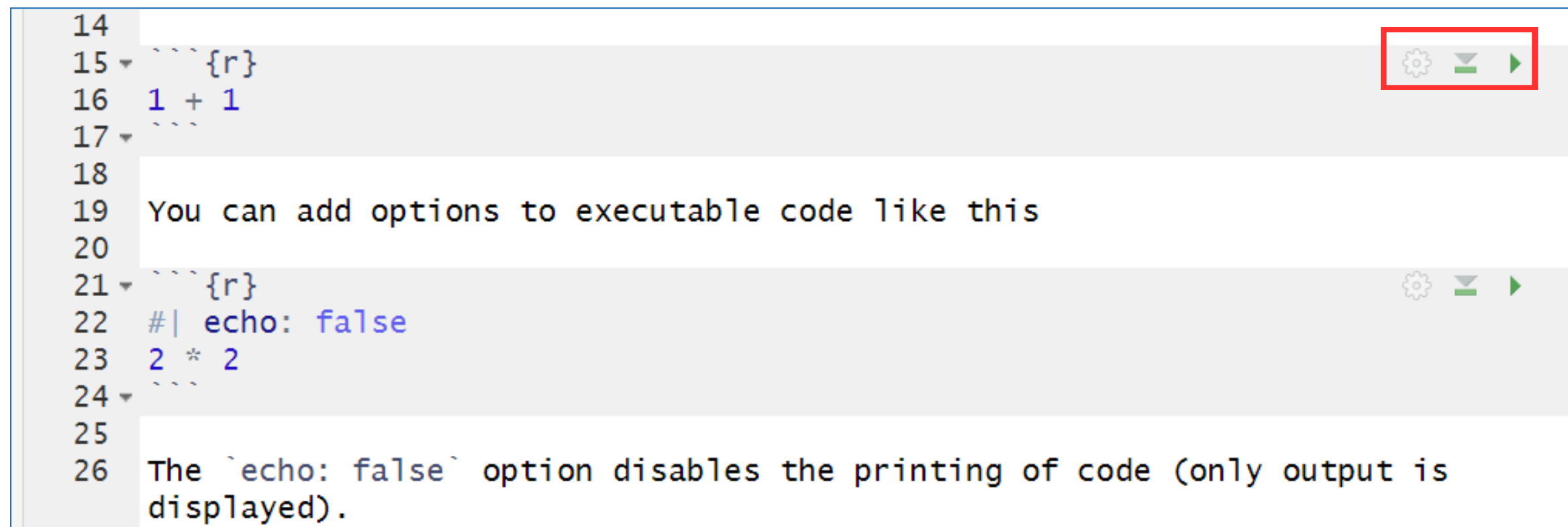
Code Chunks

Code chunks are sections of embedded code that allow you to execute code within **Quarto** document and include the results directly in the output.

To insert a code chunk, click the **Insert** button at the top of the **Source** pane, or use the keyboard shortcut **Ctrl + Alt + I** (Windows) or **Cmd + Option + I** (Mac).

You can run the code in two ways:

- Place the cursor on any line of the code and press **Ctrl + Enter** (Windows) or **Cmd + Enter** (Mac).
- Click the **Arrow** icon at the top right of the code chunk to run the entire chunk.



```
14
15 {r}
16 1 + 1
17
18
19 You can add options to executable code like this
20
21 {r}
22 #| echo: false
23 2 * 2
24
25
26 The `echo: false` option disables the printing of code (only output is displayed).
```

1.2 Data Analysis in Quarto

Let's proceed with performing some data analysis in **Quarto**.

First, delete everything in the **Quarto** document except for the **YAML** header.

Next, download the ``penguins.csv`` dataset from this [link](#) to use in our analysis.

It's important to keep your workflow organized. Here is an example of how to structure your analysis:

Loading the Packages

```
```{r}
library(tidyverse)
```
```

Reading the Dataset

```
```{r}
penguins <- read_csv("Datasets/penguins.csv")
```
```

Exploring the Dataset

```
```{r}
Get a quick overview of dataset
glimpse(penguins)
```
```

1.3 Headings

To organize **Quarto** document effectively, you can create headings by using the `#` symbol.

The number of `#` symbols will determine the heading level; more symbols indicate lower-level headings.

| | | |
|-----------------------------|---|-----------------|
| <code># Header 1</code> | → | Header 1 |
| <code>## Header 2</code> | → | Header 2 |
| <code>### Header 3</code> | → | Header 3 |
| <code>#### Header 4</code> | → | Header 4 |
| <code>##### Header 5</code> | → | Header 5 |
| <code>##### Header 6</code> | → | Header 6 |

*From now on, we will be using the **Source Editor** to better illustrate how coding works in **Quarto**.*

Let's create a structured **Quarto** document with headings and subheadings:

```
# Setting the Environment

## Loading the Packages

```{r}
Load the packages into R session
library(tidyverse)
```

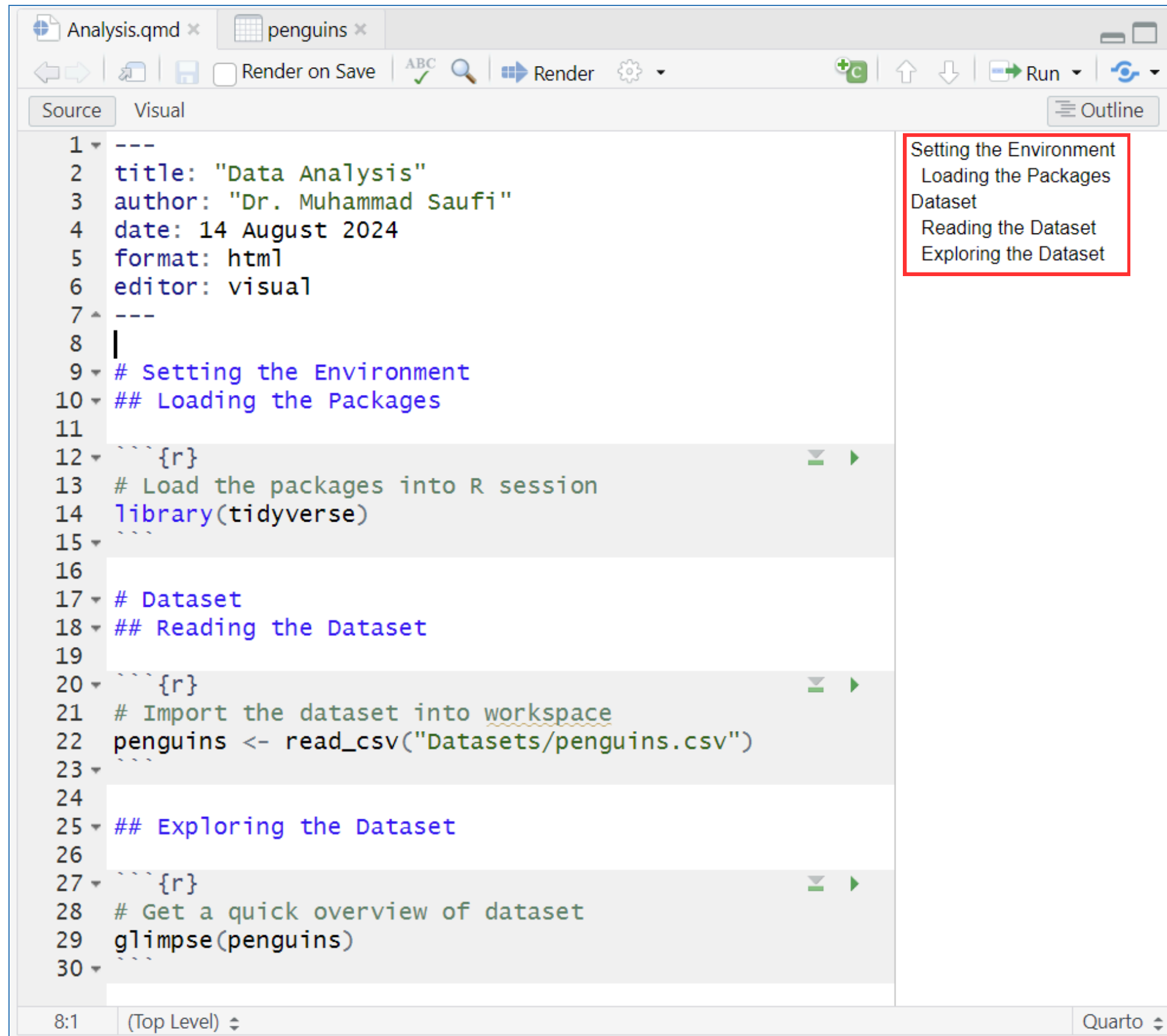
# Dataset

## Reading the Dataset

```{r}
Import the dataset into workspace
penguins <- read_csv("Datasets/penguins.csv")
```

## Exploring the Dataset

```{r}
Get a quick overview of dataset
glimpse(penguins)
```
```



The screenshot shows the Quarto editor interface. The top toolbar includes buttons for navigation, saving, rendering, and running. The main editor area is split into two panes: 'Source' on the left and 'Outline' on the right. The 'Source' pane displays R code with line numbers 1 through 30. The code includes a YAML header, a title, author, date, format, and editor, followed by three sections of R code: 'Setting the Environment', 'Loading the Packages', and 'Exploring the Dataset'. The 'Outline' pane on the right shows a list of these sections, with 'Setting the Environment' selected and highlighted by a red box. The bottom status bar shows '8:1' and '(Top Level)'.

```
1 ---  
2 title: "Data Analysis"  
3 author: "Dr. Muhammad Saufi"  
4 date: 14 August 2024  
5 format: html  
6 editor: visual  
7 ---  
8 |  
9 # Setting the Environment  
10 ## Loading the Packages  
11  
12 ```{r}  
13 # Load the packages into R session  
14 library(tidyverse)  
15 ```  
16  
17 # Dataset  
18 ## Reading the Dataset  
19  
20 ```{r}  
21 # Import the dataset into workspace  
22 penguins <- read_csv("Datasets/penguins.csv")  
23 ```  
24  
25 ## Exploring the Dataset  
26  
27 ```{r}  
28 # Get a quick overview of dataset  
29 glimpse(penguins)  
30 ```
```

The headings will appear under the **Outline** tab. You can click on these headings to directly navigate to the selected section. This is especially useful if you have a long document.

Note that if you include a `#` symbol followed by text within a code chunk, **Quarto** will recognize it as a comment instead of a heading.

Render your document to generate an **HTML** file.

The previously created **HTML** file will be replaced with the newly generated one.

You now have headings and subheadings in your **Quarto** document.

Data Analysis

AUTHOR

Dr. Muhammad Saufi

PUBLISHED

August 14, 2024

Setting the Environment

Loading the Packages

```
# Load the packages into R session
library(tidyverse)
```

Warning: package 'tidyverse' was built under R version 4.4.1

— Attaching core tidyverse packages — tidyverse 2.0.0 —

| | | | |
|-------------|-------|-----------|-------|
| ✓ dplyr | 1.1.4 | ✓ readr | 2.1.5 |
| ✓ forcats | 1.0.0 | ✓ stringr | 1.5.1 |
| ✓ ggplot2 | 3.5.1 | ✓ tibble | 3.2.1 |
| ✓ lubridate | 1.9.3 | ✓ tidyr | 1.3.1 |
| ✓ purrr | 1.0.2 | | |

— Conflicts — tidyverse_conflicts() —

✗ dplyr::filter() masks stats::filter()

✗ dplyr::lag() masks stats::lag()

ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

Dataset

Reading the Dataset

```
# Import the dataset into workspace
penguins <- read_csv("Datasets/penguins.csv")
```

1.4 Section Numbering

As your document grows longer, you might have several headings and subheadings to effectively organize your content.

Numbered headings can facilitate easier navigation of your output report.

Instead of manually typing the numbers into the headings, you can automatically create numbered headings by adding ``number-sections: true`` into the **YAML** header.

*Notice the change from the previous ``format: html``. This ensures that subsequent code is generated as **HTML** output.*

Ensure the position of each line of code is correct as shown above.

```
---
title: "Data Analysis"
author: "Dr. Muhammad Saufi"
date: 14 August 2024
format:
  html:
    number-sections: true
editor: visual
---
```

Data Analysis

AUTHOR
Dr. Muhammad Saufi

PUBLISHED
August 14, 2024

1 Setting the Environment

1.1 Loading the Packages

```
# Load the packages into R session
library(tidyverse)
```

Warning: package 'tidyverse' was built under R version 4.4.1

```
— Attaching core tidyverse packages — tidyverse 2.0.0 —
✓ dplyr      1.1.4    ✓ readr      2.1.5
✓ forcats    1.0.0    ✓ stringr    1.5.1
✓ ggplot2    3.5.1    ✓ tibble     3.2.1
✓ lubridate  1.9.3    ✓ tidyr      1.3.1
✓ purrr      1.0.2
— Conflicts — tidyverse_conflicts() —
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()     masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

2 Dataset

2.1 Reading the Dataset

```
# Import the dataset into workspace
penguins <- read_csv("Datasets/penguins.csv")
```

The numbered headings and subheadings appear in the **HTML** output after you render the document.

*Remember, you must render your document each time after you finish making changes in your report to generate a new **HTML** output.*

1.5 Table of Contents

You can automatically generate a table of contents in your output document by adjusting the **YAML** header. The following code demonstrates how to set up and customize the table of contents:

```
---
title: "Data Analysis"
author: "Dr. Muhammad Saufi"
date: 14 August 2024
format:
  html:
    toc: true
    toc-title: Contents
    toc-location: left
    toc-depth: 3
    toc-expand: 1
    number-sections: true
editor: visual
---
```

Below is an explanation of each line of code used to create the table of contents:

```
toc: true
```

Enables the table of contents

```
toc-title: Contents
```

Sets the title of the table of contents

```
toc-location: left
```

Positions the table of contents on the left side

```
toc-depth: 3
```

Specifies how many levels to be included

```
toc-expand: 1
```

Specifies how many levels to show initially

Below is the **HTML** output with the generated table of contents.

| | |
|---------------------------|--|
| Contents | <h2>2 Dataset</h2> |
| 1 Setting the Environment | |
| 2 Dataset | <h3>2.1 Reading the Dataset</h3> |
| 2.1 Reading the Dataset | <pre># Import the dataset into workspace penguins <- read_csv("Datasets/penguins.csv")</pre> <p>Rows: 340 Columns: 9</p> <p>— Column specification —</p> <p>Delimiter: ","</p> <p>chr (3): species, island, sex</p> <p>dbl (6): rowid, bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass...</p> <p>i Use <code>`spec()`</code> to retrieve the full column specification for this data.</p> <p>i Specify the column types or set <code>`show_col_types = FALSE`</code> to quiet this message.</p> |
| 2.2 Exploring the Dataset | <h3>2.2 Exploring the Dataset</h3> |
| | <pre># Get a quick overview of dataset glimpse(penguins)</pre> |

1.6 Folding Code

Sometimes, code can take up a lot of space in your document, which might overwhelm readers who are more interested in the results rather than the code itself.

To keep your document clean and focused, you can use the ``code-fold`` option. This feature allows you to hide the code by default, with an option to reveal it using a button if necessary.

This way, readers can focus on the output while still having access to the code when needed.

```
---  
title: "Data Analysis"  
author: "Dr. Muhammad Saufi"  
date: 14 August 2024  
format:  
  html:  
    toc: true  
    toc-title: Contents  
    toc-location: left  
    toc-depth: 3  
    toc-expand: 1  
    number-sections: true  
    code-fold: true  
    code-summary: "Show the Code"  
editor: visual  
---
```


Contents

1 Setting the Environment

1.1 Loading the Packages

2 Dataset

Data Analysis

AUTHOR

Dr. Muhammad Saufi

PUBLISHED

August 14, 2024

1 Setting the Environment

1.1 Loading the Packages

► Show the Code

Warning: package 'tidyverse' was built under R version 4.4.1

— Attaching core tidyverse packages — tidyverse 2.0.0 —

✓ dplyr 1.1.4 ✓ readr 2.1.5

✓ forcats 1.0.0 ✓ stringr 1.5.1

✓ ggplot2 3.5.1 ✓ tibble 3.2.1

✓ lubridate 1.9.3 ✓ tidyr 1.3.1

✓ purrr 1.0.2

— Conflicts — tidyverse_conflicts() —

✗ dplyr::filter() masks stats::filter()

✗ dplyr::lag() masks stats::lag()

ℹ Use the conflicted package (<<http://conflicted.r-lib.org/>>) to force all conflicts to be resolved

Readers can click the '**Show the Code**' button to view the hidden code when needed.

1.7 HTML Themes

Our eyes can sometimes get strained from looking at the monitor for too long. To reduce this strain, you can add a dark theme to your **HTML** output.

Even better, **HTML** allows you to switch between light and dark modes. You just need to specify this in the **YAML** header, as shown in the example below.

```
---
title: "Data Analysis"
author: "Dr. Muhammad Saufi"
date: 14 August 2024
format:
  html:
    toc: true
    toc-title: Contents
    toc-location: left
    toc-depth: 3
    toc-expand: 1
    number-sections: true
    code-fold: true
    code-summary: "Show the Code"
    theme:
      light: united
      dark: cyborg
editor: visual
---
```

Contents

1 Setting the Environment

1.1 Loading the Packages

2 Dataset

Data Analysis

AUTHOR

Dr. Muhammad Saufi

PUBLISHED

August 14, 2024

1 Setting the Environment

1.1 Loading the Packages

► Show the Code

Warning: package 'tidyverse' was built under R version 4.4.1

— Attaching core tidyverse packages ————— tid

✓ dplyr 1.1.4 ✓ readr 2.1.5

✓ forcats 1.0.0 ✓ stringr 1.5.1

✓ ggplot2 3.5.1 ✓ tibble 3.2.1

✓ lubridate 1.9.3 ✓ tidyr 1.3.1

✓ purrr 1.0.2

— Conflicts ————— tidyverse

✗ dplyr::filter() masks stats::filter()

✗ dplyr::lag() masks stats::lag()

i Use the conflicted package (<<http://conflicted.r-lib.org/>>) to



Click the button at the top right of the HTML file to enable or disable dark mode.

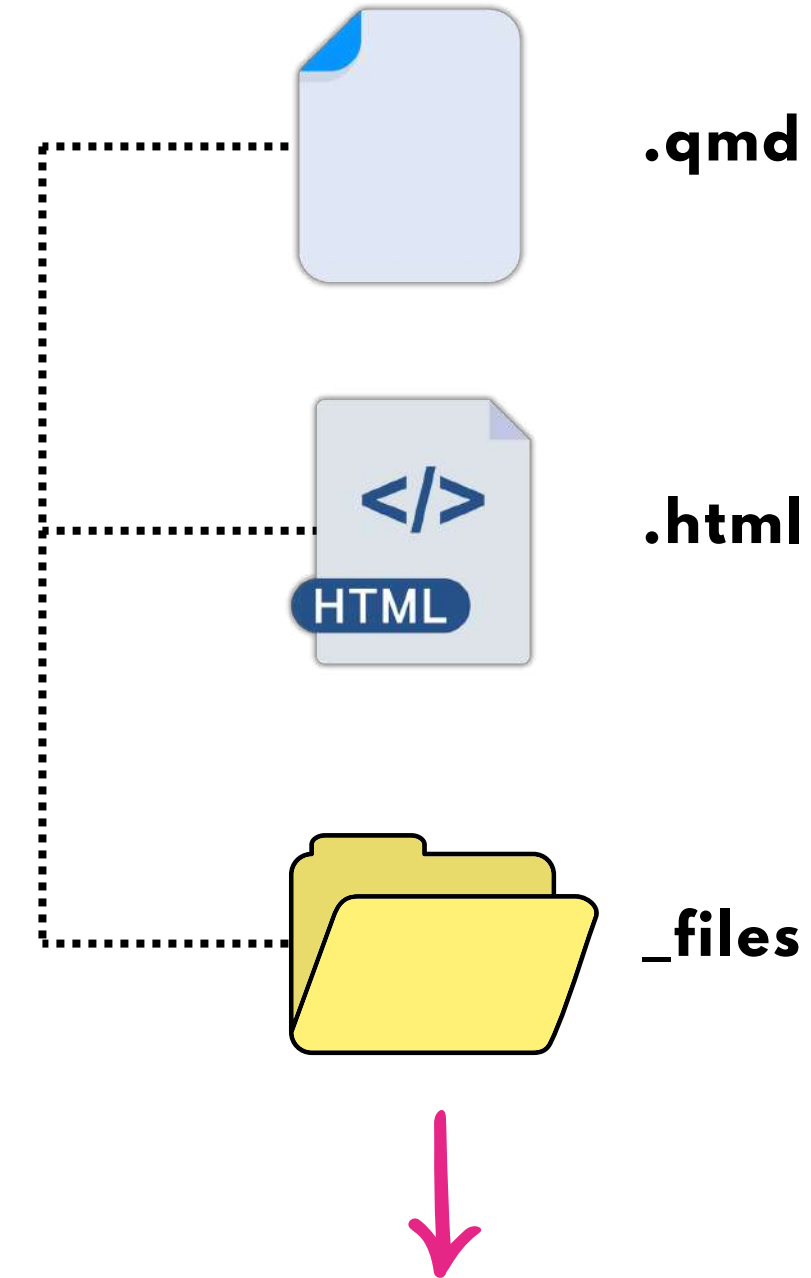
*You can explore a variety of themes provided in this [guide](#). **Quarto** offers at least 25 themes from the Bootswatch project.*

1.8 Self Contained

When you render a **Quarto** document into **HTML**, an additional folder will appear alongside the **HTML** file.

This folder typically contains assets like images and other resources referenced by the **HTML** file to ensure it displays correctly.

If you move or share the **HTML** file, you'll need to include this folder so that all associated content is correctly displayed.



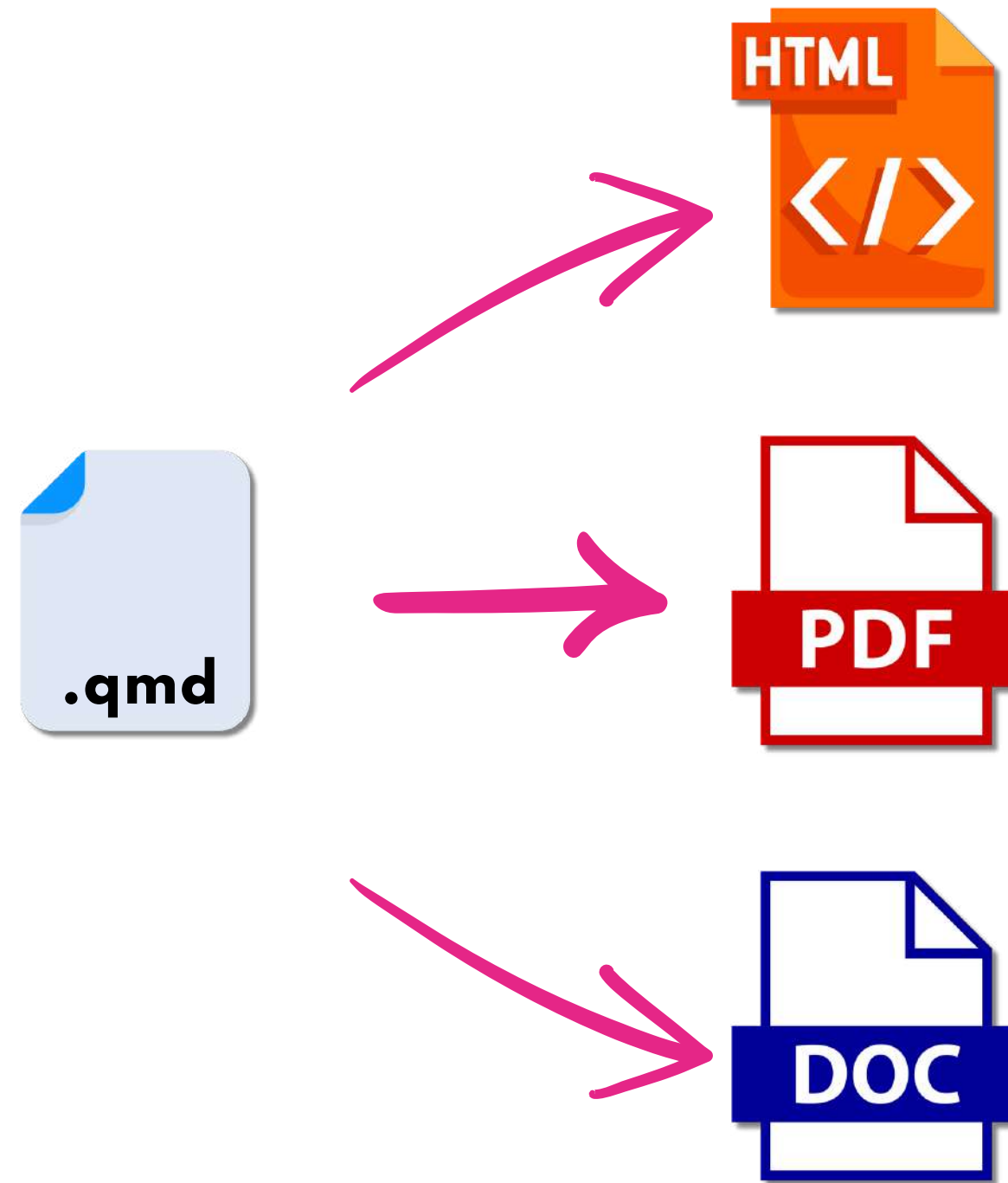
This folder contains external dependencies for the HTML file.

To create a fully self-contained **HTML** document, you can specify the ``embed-resources`` option in the **YAML** header.

```
---
title: "Data Analysis"
author: "Dr. Muhammad Saufi"
date: 14 August 2024
format:
  html:
    toc: true
    toc-title: Contents
    toc-location: left
    toc-depth: 3
    toc-expand: 1
    number-sections: true
    code-fold: true
    code-summary: "Show the Code"
    theme:
      light: united
      dark: cyborg
    embed-resources: true
editor: visual
---
```

*You should first delete the previous **HTML** document along with its external dependencies folder. Then, when you render the **Quarto** document again, it will generate only the **HTML** file.*

So far, we've focused on rendering **Quarto** documents into **HTML**. However, **Quarto** also allows you to generate **PDF** or **Microsoft Word** documents.

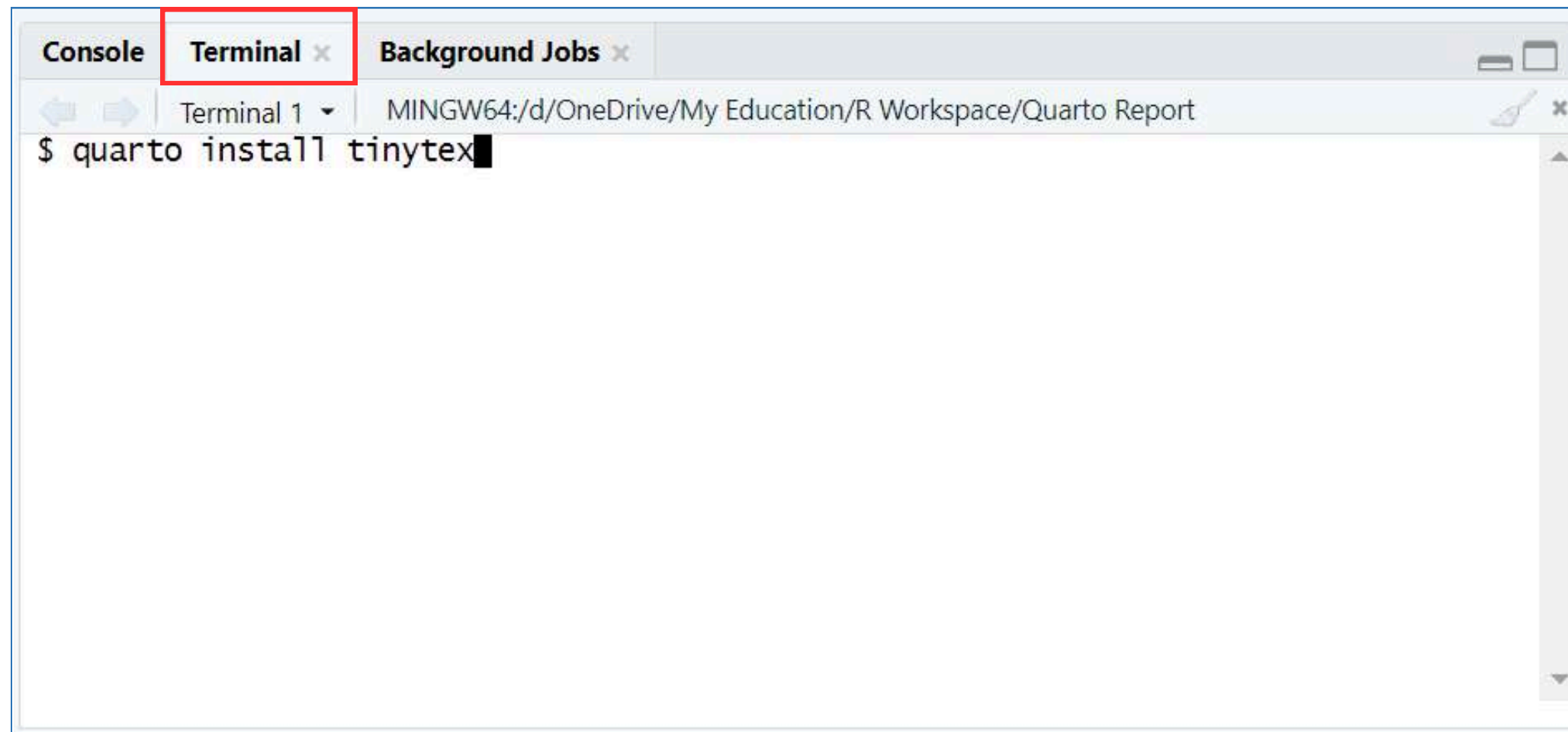


1.9 PDF

To create **PDF**, you'll need to install **TinyTeX** using the **Terminal** in **RStudio**. You can do this with the following command:

In Terminal:

```
quarto install tinytex
```



*Copy and paste the command into the **Terminal** and press **`Enter`**. The **TinyTeX** installation should start immediately.*

To create a **PDF** output, use the ``pdf`` format in the **YAML** header. The option ``toc: true`` will generate a table of contents in the **PDF**. If you want each heading numbered, include ``number-sections: true``.

```
---
title: "Data Analysis"
author: "Dr. Muhammad Saufi"
date: 14 August 2024
format:
  pdf:
    toc: true
    toc-title: Contents
    number-sections: true
editor: visual
---
```

*Once you've set these options, click **`Render`** to generate the **PDF** document. You should find the **PDF** within your working directory.*

1.10 Microsoft Word

To create a **Microsoft Word** output, use the ``docx`` format in the **YAML** header. For example:

```
---  
title: "Data Analysis"  
author: "Dr. Muhammad Saufi"  
date: 14 August 2024  
format:  
  docx:  
    toc: true  
    toc-title: Contents  
    number-sections: true  
editor: visual  
---
```

*Your **Microsoft Word** document will appear in your working directory after you render your **Quarto** document.*

As you've noticed, we render the **Quarto** document into each output format (**HTML**, **PDF**, **Microsoft Word**) individually, not simultaneously.

To create the desired output, you need to specify the format in the **YAML** header.

It's recommended to start by rendering your document into **HTML**, as it allows you to preview your report while making changes.

Once you're satisfied with the final version of your report, you can then render the document into other formats, such as **PDF** or **Microsoft Word**.



PART 2

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.

2.1 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](#).

2.2 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:



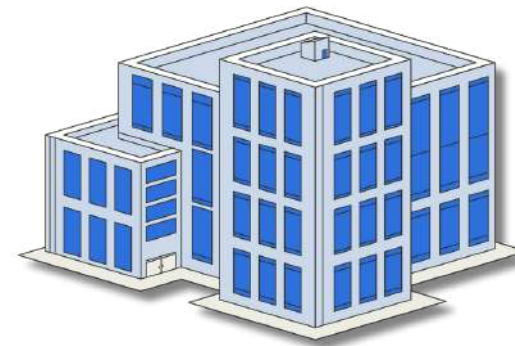
Include your real name

This makes it easier for others to recognize you.



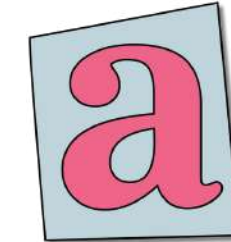
Keep it short

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``.

2.3 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`**:

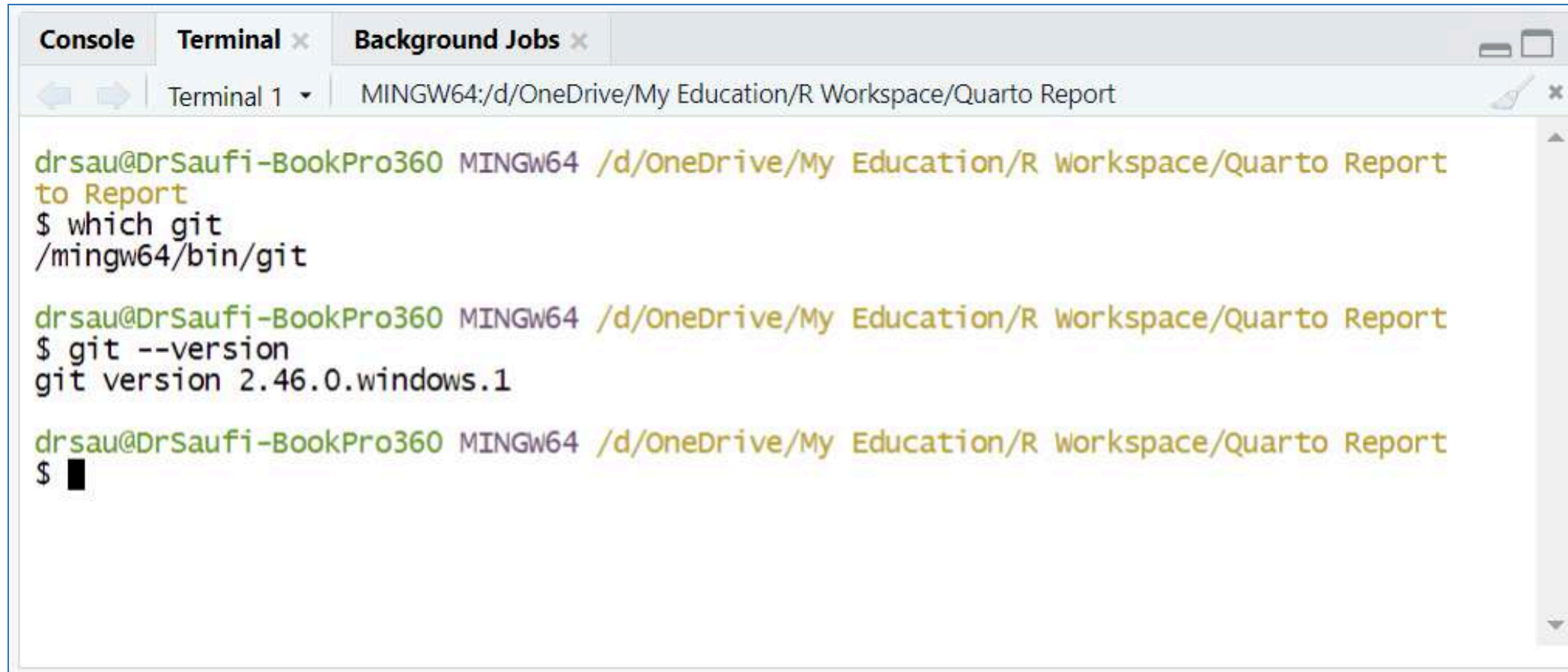
In Terminal:

```
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.

A screenshot of a terminal window with three tabs: 'Console', 'Terminal x', and 'Background Jobs x'. The 'Terminal' tab is active, showing a command prompt for 'Terminal 1' at the path 'MINGW64:/d/OneDrive/My Education/R Workspace/Quarto Report'. The prompt is 'drsau@DrSauFi-BookPro360 MINGW64 /d/OneDrive/My Education/R Workspace/Quarto Report'. The user enters '\$ which git' and the output is '/mingw64/bin/git'. Then the user enters '\$ git --version' and the output is 'git version 2.46.0.windows.1'. The prompt returns to '\$' with a cursor.

```
drsau@DrSauFi-BookPro360 MINGW64 /d/OneDrive/My Education/R Workspace/Quarto Report
$ which git
/mingw64/bin/git

drsau@DrSauFi-BookPro360 MINGW64 /d/OneDrive/My Education/R Workspace/Quarto Report
$ git --version
git version 2.46.0.windows.1

drsau@DrSauFi-BookPro360 MINGW64 /d/OneDrive/My Education/R Workspace/Quarto Report
$
```

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:

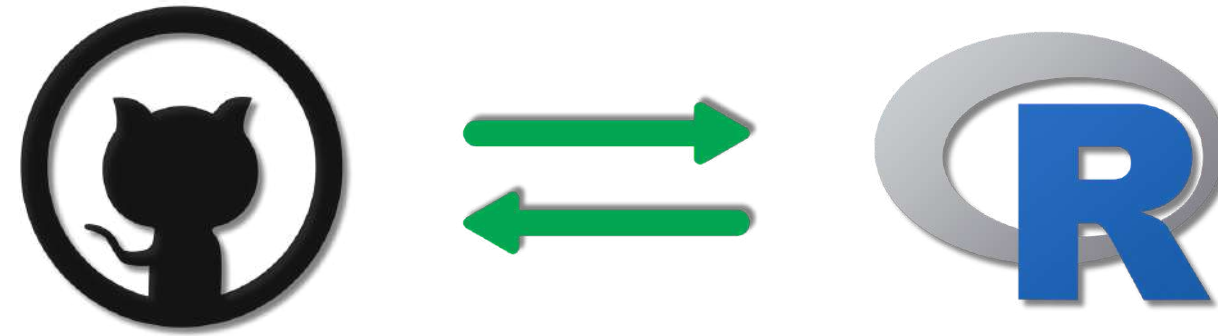
```
git config --global user.name "Your Name"  
git config --global user.email "email@example.com"  
git config --global --list
```

*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.*

2.4 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`.

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens

Tokens (classic)

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

BookPro 360

What's this token for?

Expiration *

No expiration

The token will never expire!

GitHub strongly recommends that you set an expiration date for your token to help keep your information secure. [Learn more](#)

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

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**:

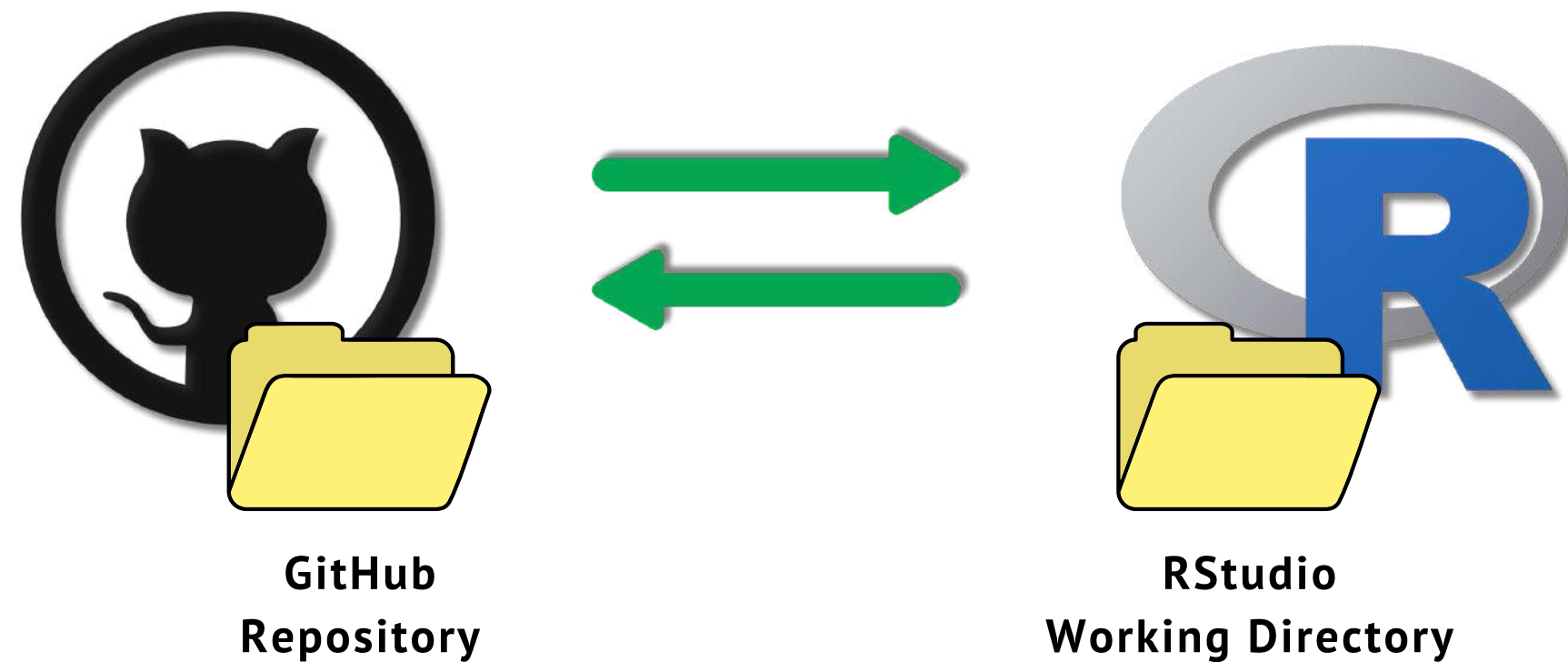
```
? Enter password or token: xxxxxxxxxxxxxxxxx  
-> Adding new credentials...  
-> Removing credentials from cache...  
-> Done.
```

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

2.5 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**.



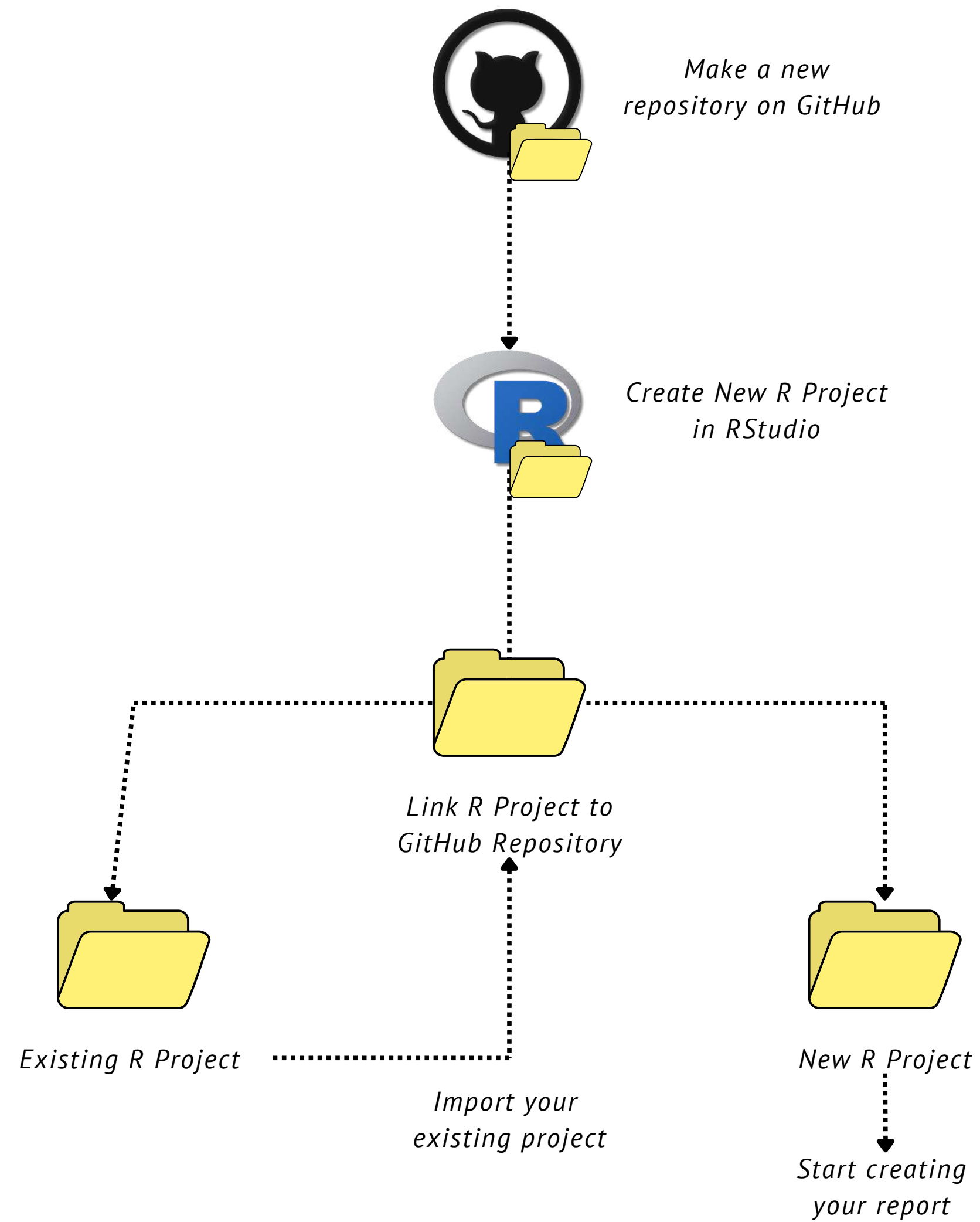
The process of integrating **GitHub** with **RStudio** begins by creating a new repository on **GitHub**.

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.

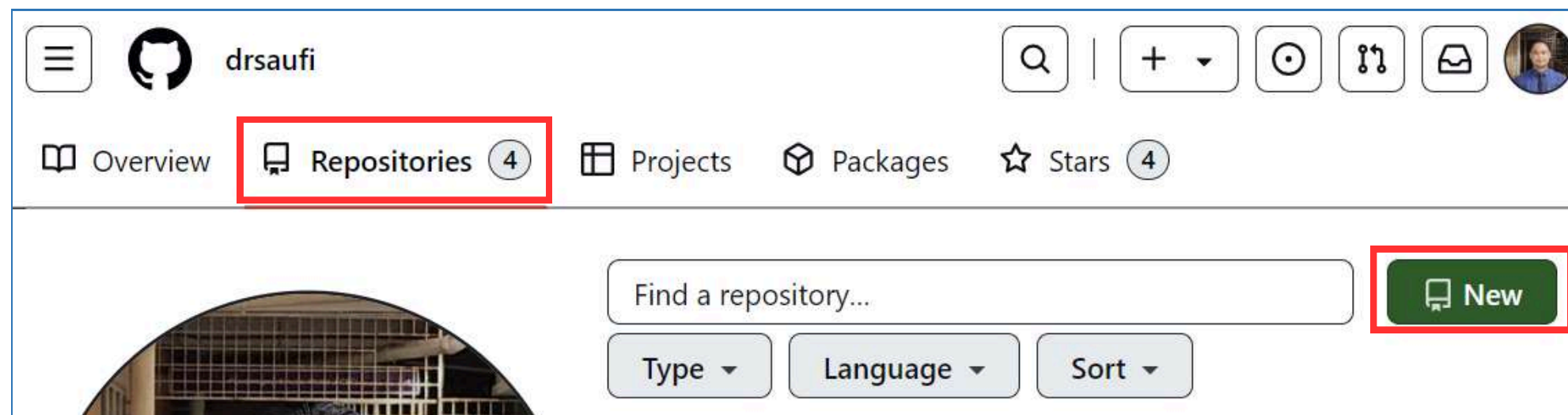
However, if you already have an existing **R** project, you can copy or move **R** documents and files into the directory for this new **R** project.



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.



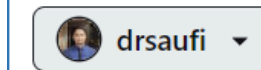
Give your repository a name. Keep in mind that **GitHub** does not allow spaces in repository names, so it automatically replaces them with hyphens ("-").

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner *



Repository name *



✔ Quarto-Report is available.

Great repository names are short and memorable. Need inspiration? How about **psychic-disco** ?

Description (optional)

This repository contains tutorials and guides on creating Quarto documents and publishing them to GitHub.



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:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▼

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None ▼

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)



You are creating a public repository in your personal account.

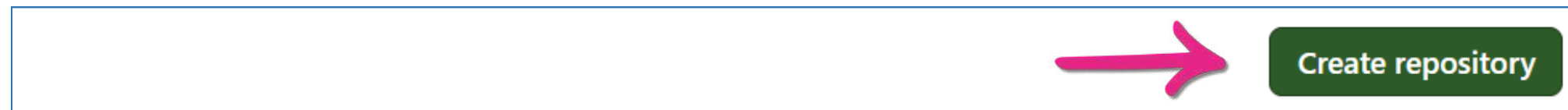
Create repository

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

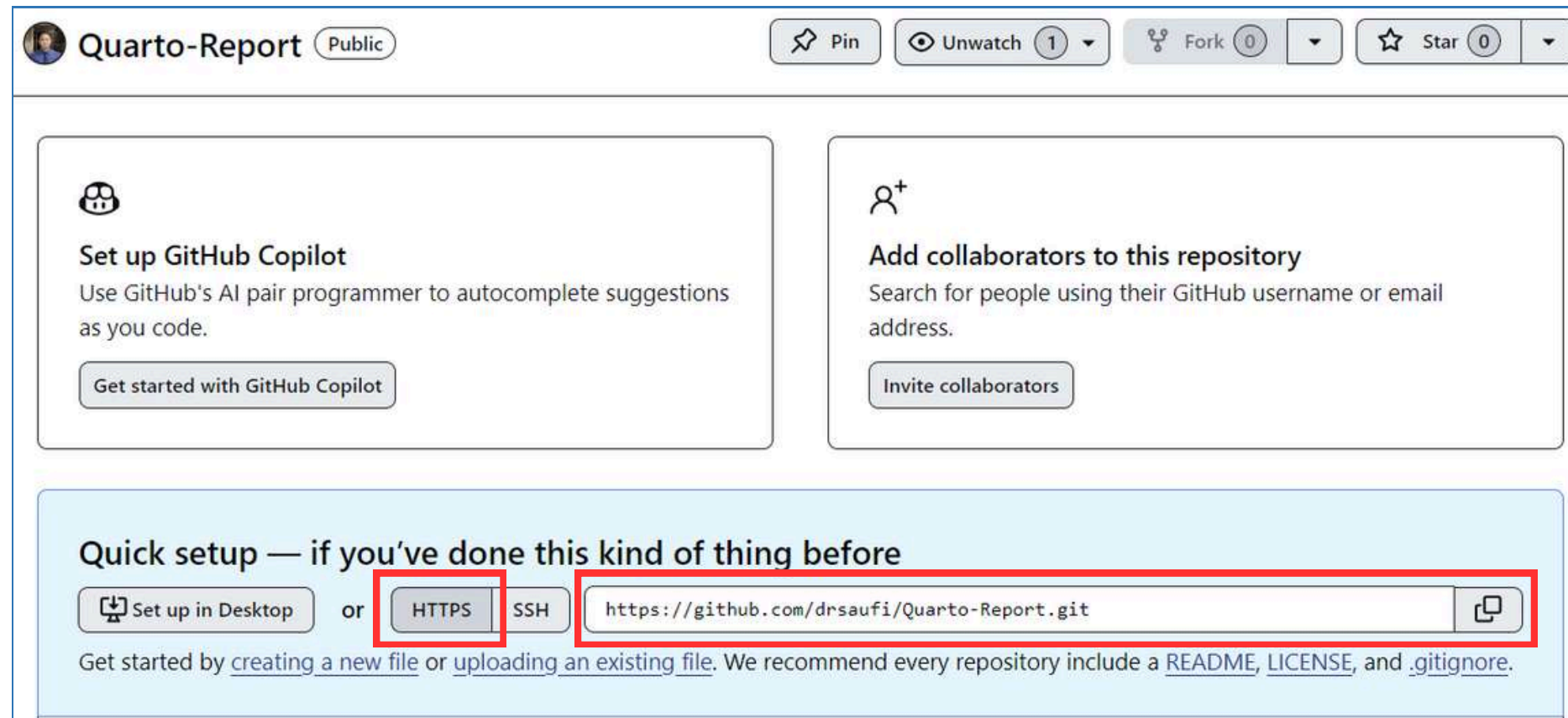
You can choose to set the repository as **`Public`** or **`Private`**, depending on its content.

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.



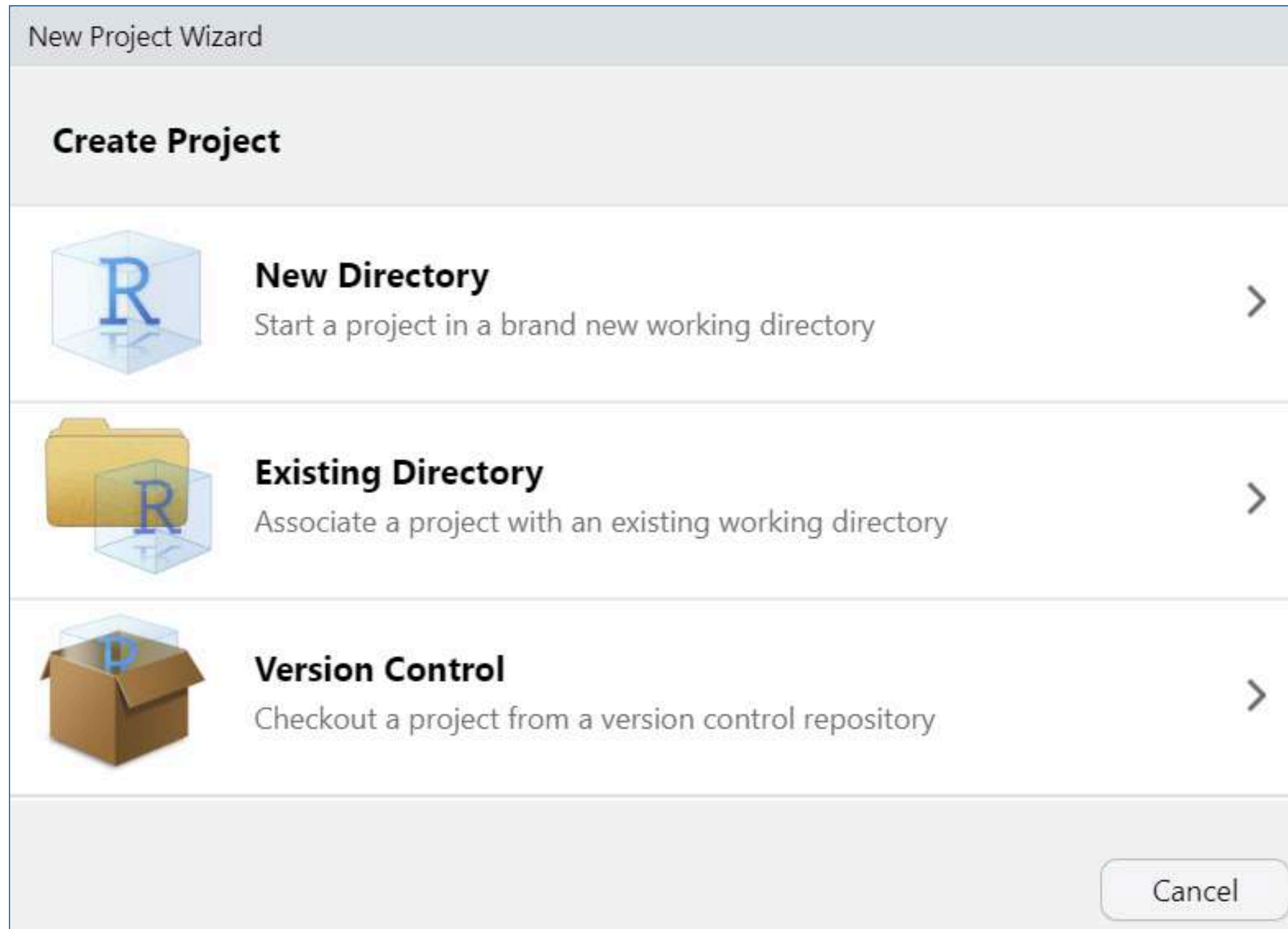
Congratulations! You have just created your first **GitHub** repository. You should now see a page like the one shown below.



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.

New Project Wizard


Back

Create Project from Version Control



Git
Clone a project from a Git repository

>




Subversion
Checkout a project from a Subversion repository

>

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

New Project Wizard

[Back](#) **Clone Git Repository**



Repository URL:

Project directory name:

Create project as subdirectory of:
 [Browse...](#)

☒ Open in new session

[Create Project](#) [Cancel](#)

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

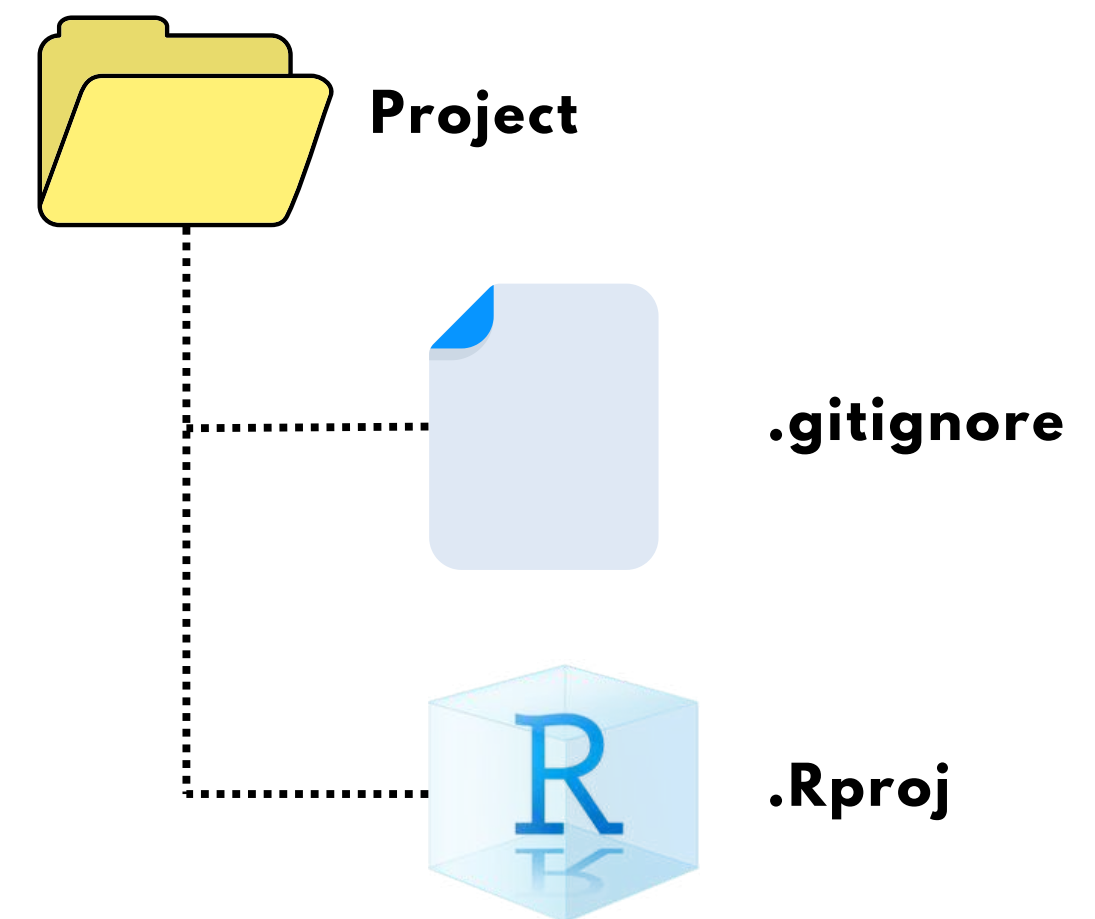
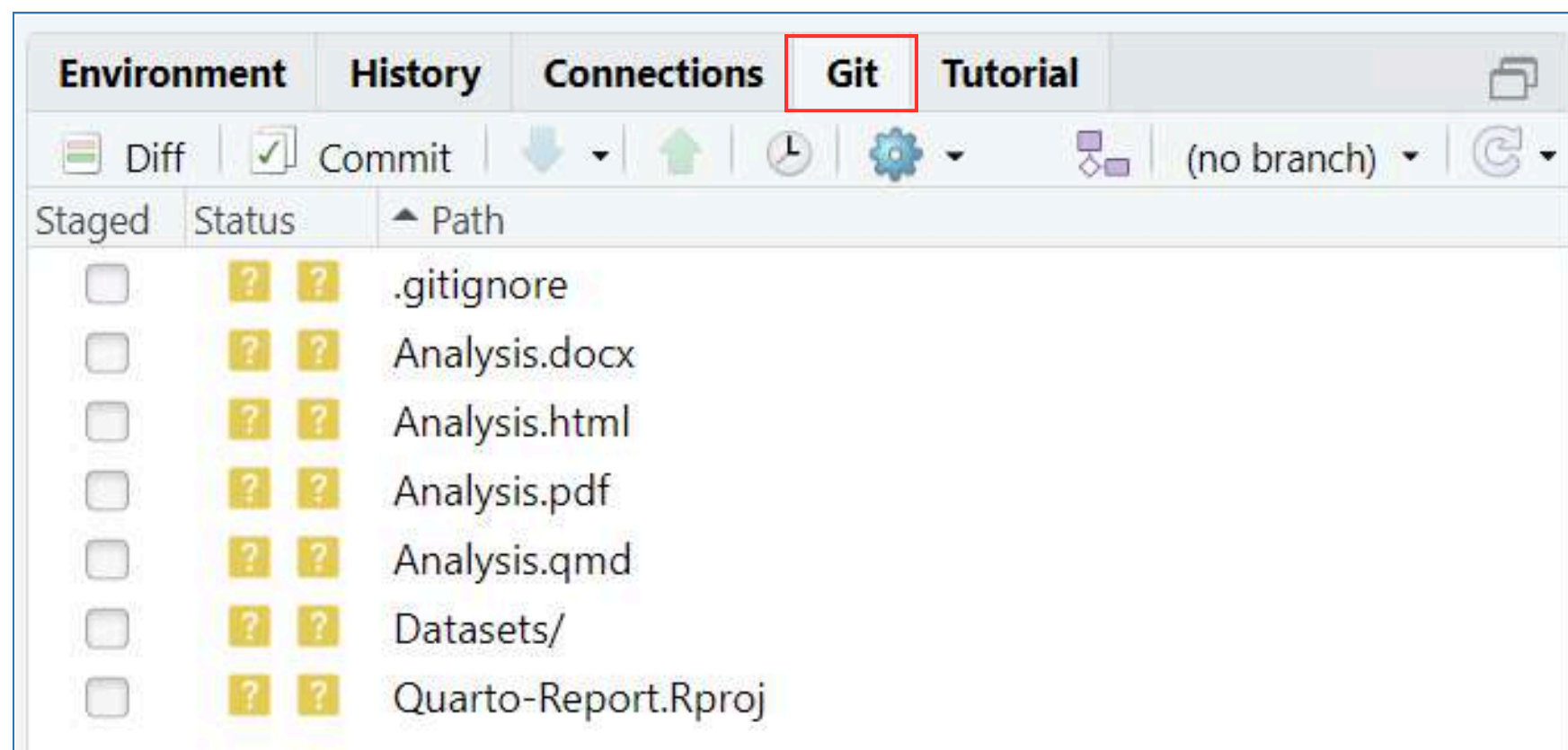
It is recommended that the name of the project directory be similar to your **GitHub** repository, although this will not affect the synchronization between your **R** project and **GitHub** repository.

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!

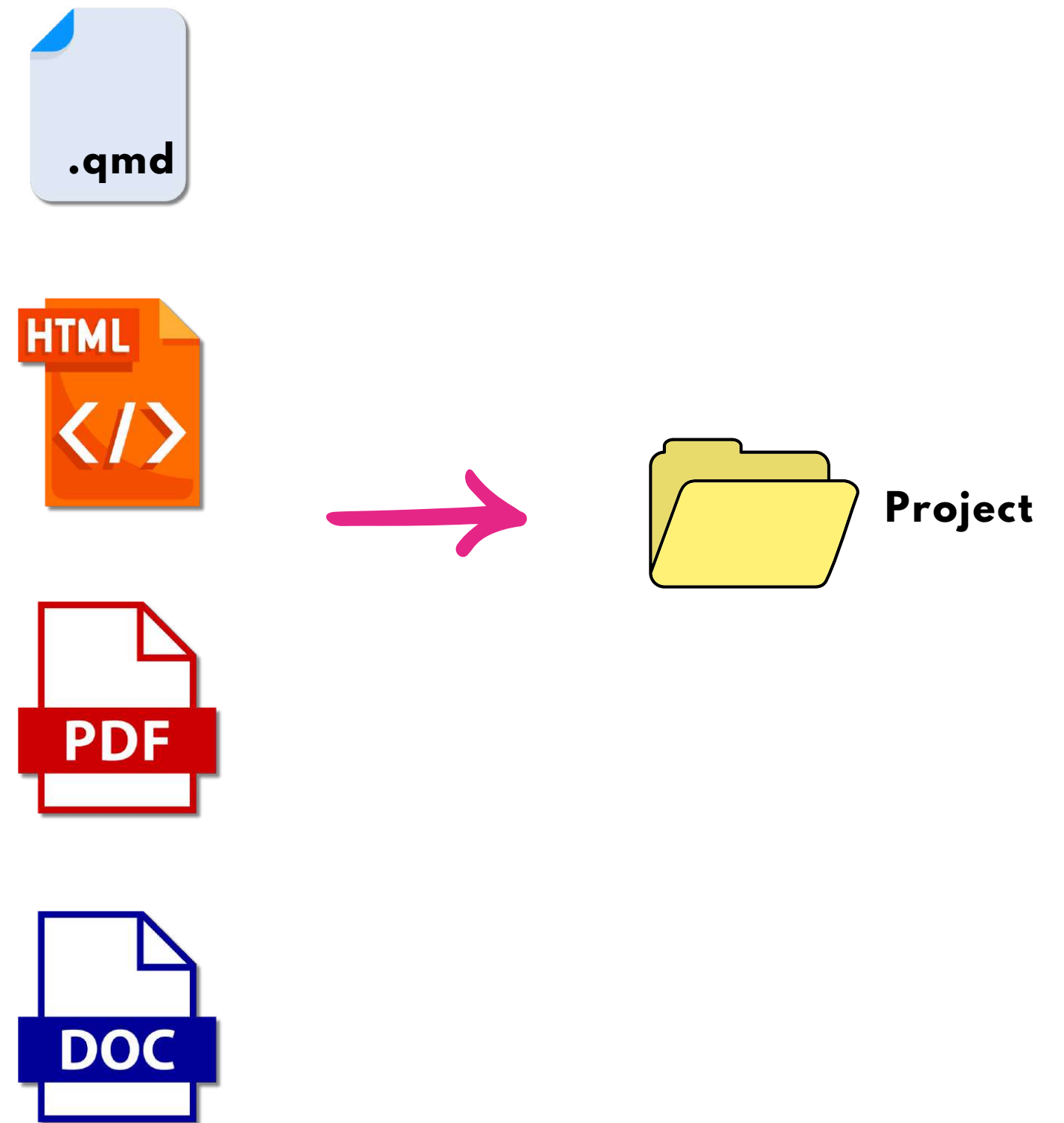
In the working directory or folder, you will find the following files:



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.



2.6 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.

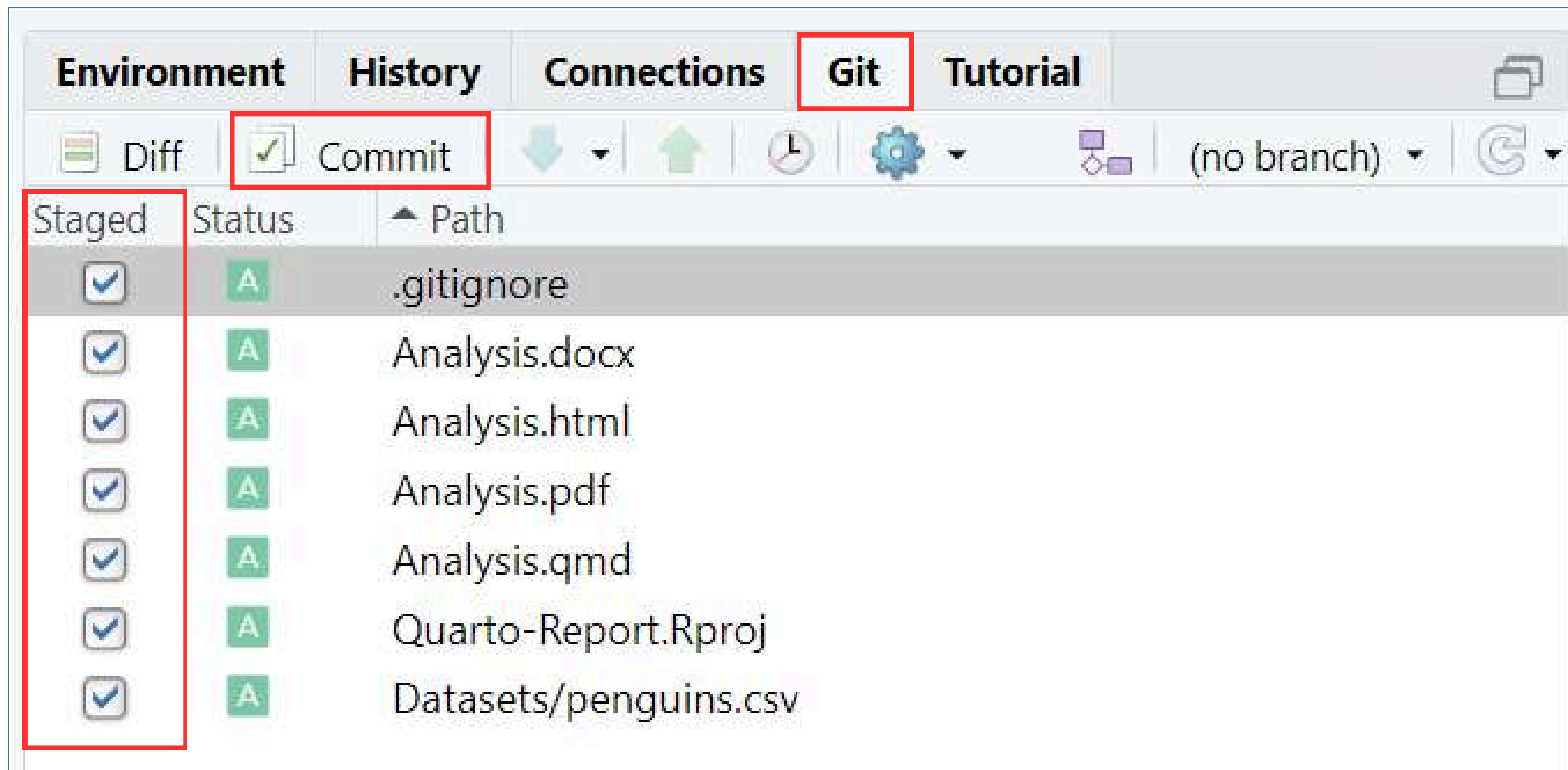
However, if your project is already stored in a cloud-based service or you choose to keep it there, it is **highly recommended** to pause the syncing before performing any **Git** operations.

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

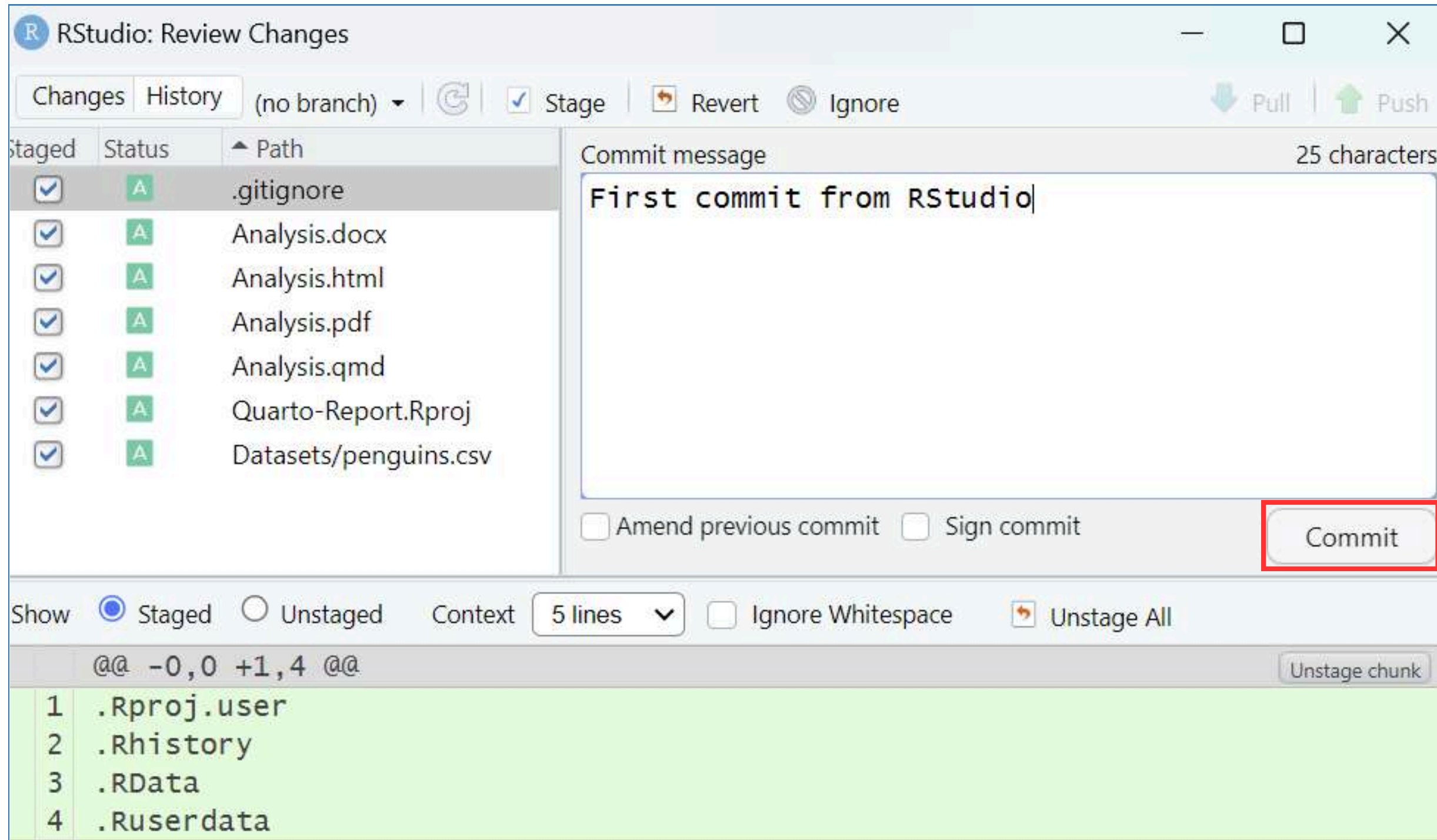
Commit

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.

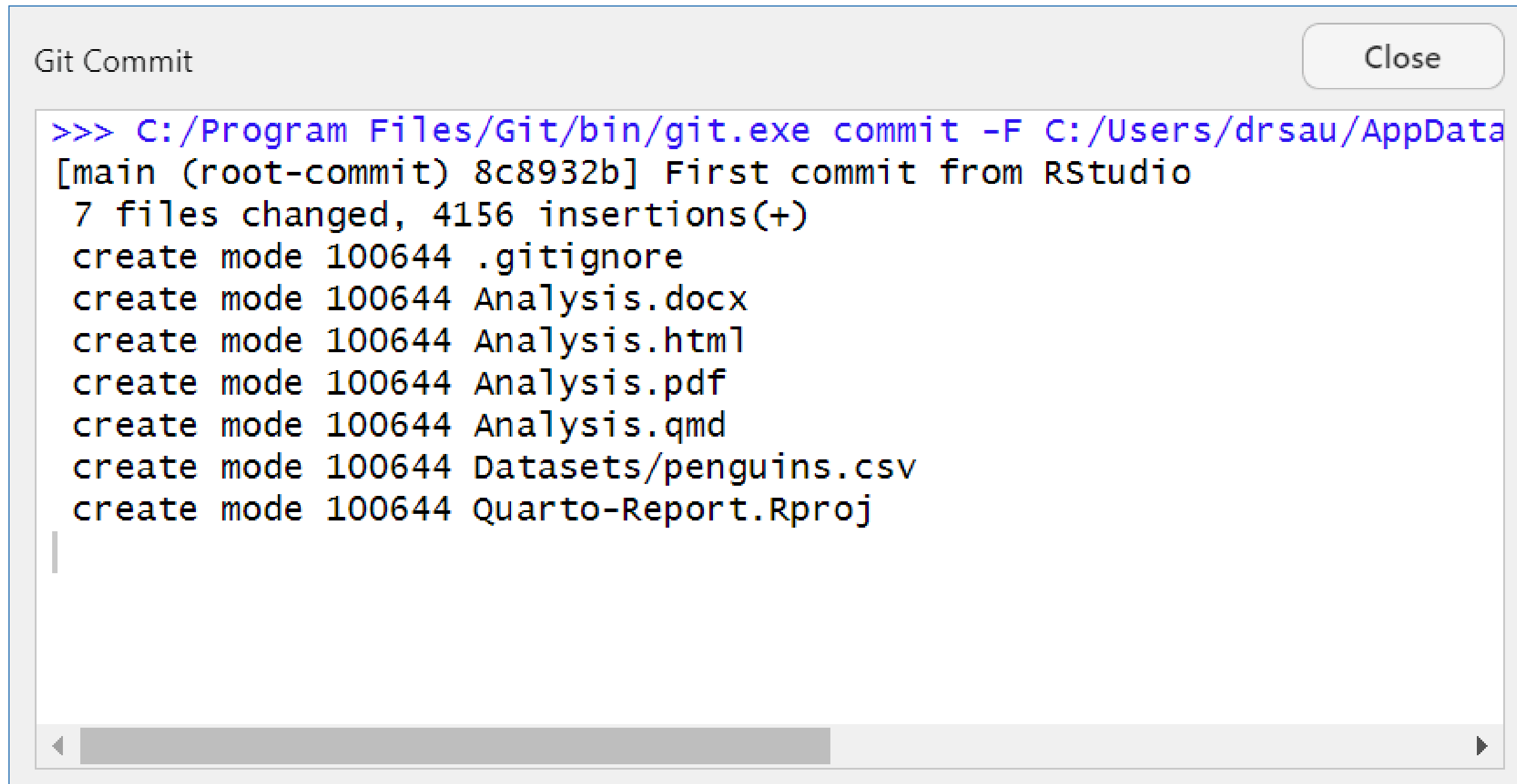
In **RStudio**, click the **`Git`** tab in the upper right pane. Check the **`Staged`** box for any files you want to **commit**, then click **`Commit`**.



A pop-up window will appear.



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

A screenshot of a 'Git Commit' dialog box. The title bar says 'Git Commit' and there is a 'Close' button in the top right. The main area contains a text report of a successful commit. The text is as follows:

```
>>> C:/Program Files/Git/bin/git.exe commit -F C:/Users/drsau/AppData
[main (root-commit) 8c8932b] First commit from RStudio
7 files changed, 4156 insertions(+)
create mode 100644 .gitignore
create mode 100644 Analysis.docx
create mode 100644 Analysis.html
create mode 100644 Analysis.pdf
create mode 100644 Analysis.qmd
create mode 100644 Datasets/penguins.csv
create mode 100644 Quarto-Report.Rproj
```

A vertical cursor is visible on the line following the last file entry. At the bottom of the text area is a horizontal scrollbar.

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**.

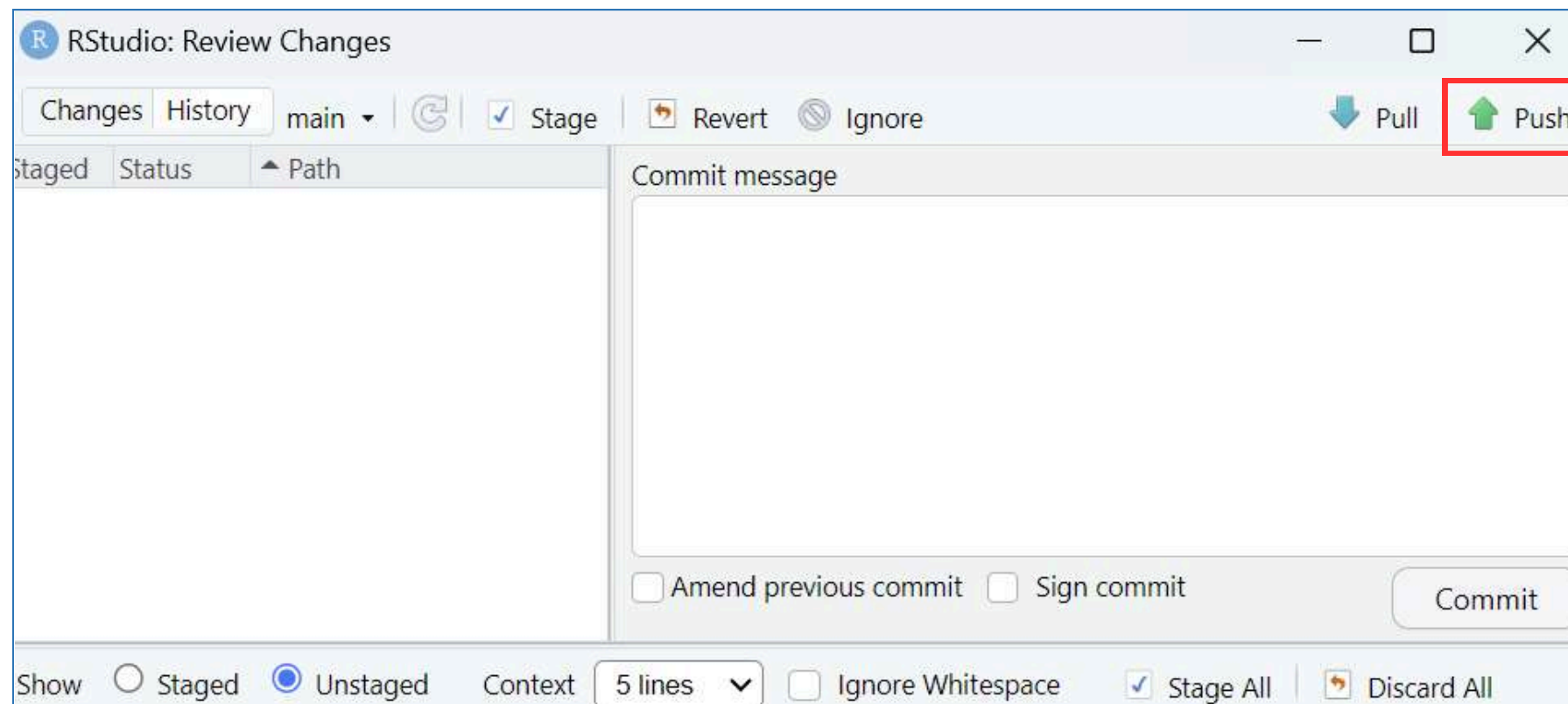
Push

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.

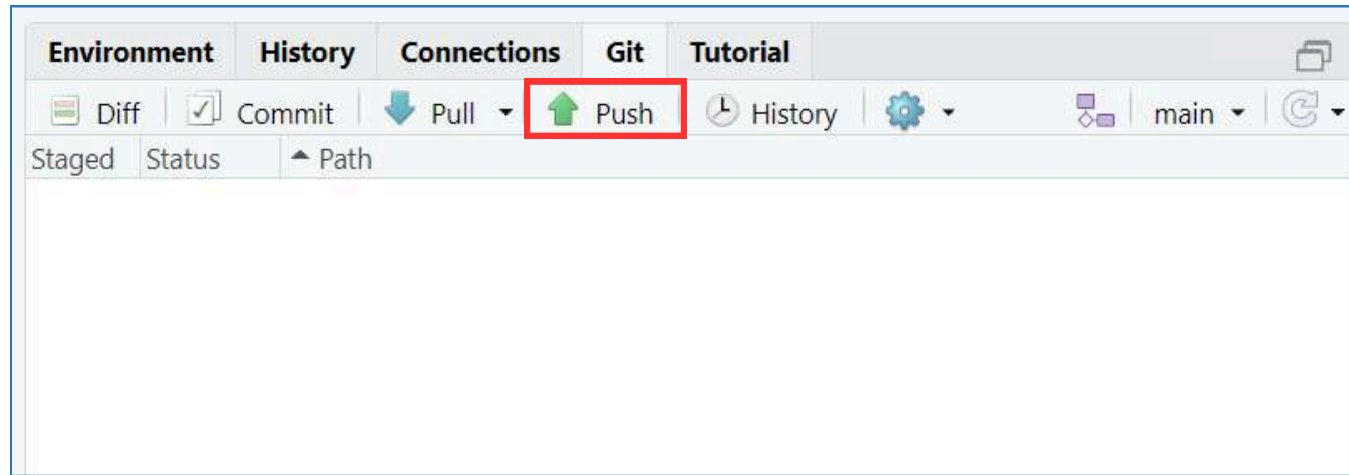
Now, it is time to **push** your local changes to **GitHub**.

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.



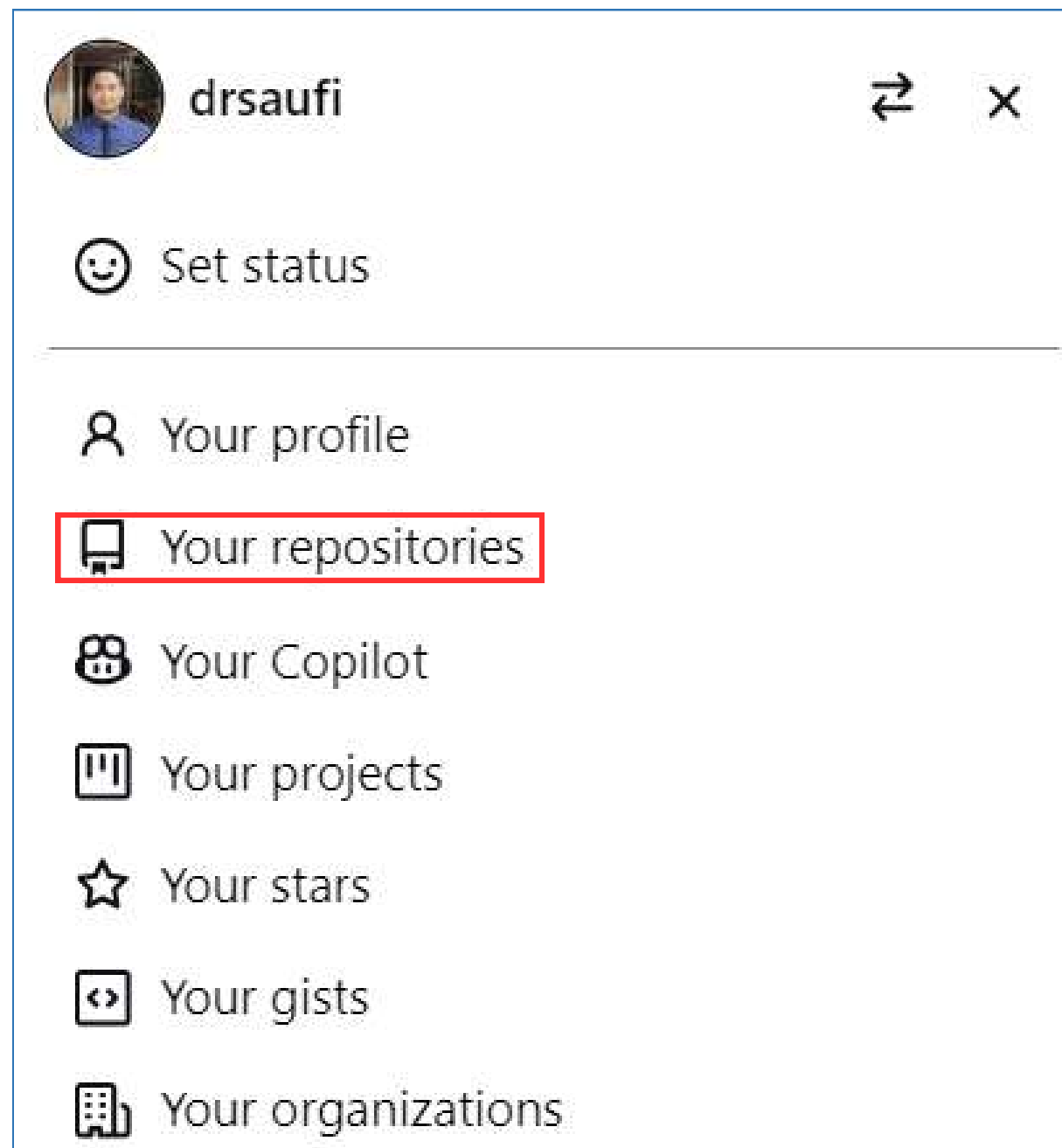
RStudio will display a report message like the one below, confirming that the changes have been successfully pushed to your **GitHub** repository.

```
Git Push
Close

>>> C:/Program Files/Git/bin/git.exe push origin HEAD:refs/heads/main
To https://github.com/drsaufi/Quarto-Report.git
 * [new branch]      HEAD -> main
```

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.

drsaufi / Quarto-Report

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Quarto-Report

Public

Pin

Unwatch 1

Fork 0

Star 0

main

Go to file

+

<> Code

About

drsaufi

First commit from RStudio

8c8932b · 22 minutes ago

| | | |
|---------------------------------|---------------------------|----------------|
| <div></div> Datasets | First commit from RStudio | 22 minutes ago |
| <div></div> .gitignore | First commit from RStudio | 22 minutes ago |
| <div></div> Analysis.docx | First commit from RStudio | 22 minutes ago |
| <div></div> Analysis.html | First commit from RStudio | 22 minutes ago |
| <div></div> Analysis.pdf | First commit from RStudio | 22 minutes ago |
| <div></div> Analysis.qmd | First commit from RStudio | 22 minutes ago |
| <div></div> Quarto-Report.Rproj | First commit from RStudio | 22 minutes ago |

About

This repository contains tutorials and guides on creating Quarto documents and publishing them to GitHub.

Activity

0 stars

1 watching

0 forks

Releases

No releases published

[Create a new release](#)

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.

The screenshot shows the GitHub interface for the repository 'drsaufi / Quarto-Report'. The repository is public and has 0 stars, 1 watch, and 0 forks. The file listing shows a commit from 'drsaufi' titled 'First commit from RStudio' made 22 minutes ago. The files listed are: Datasets, .gitignore, Analysis.docx, Analysis.html, Analysis.pdf, Analysis.qmd, and Quarto-Report.Rproj. A pink arrow points to the 'Analysis.qmd' file. The right sidebar contains an 'About' section with a description of the repository's purpose and an 'Activity' section showing 0 stars, 1 watching, and 0 forks. The 'Releases' section indicates that no releases have been published and provides a link to 'Create a new release'.

drsaufi / Quarto-Report

Code Issues Pull requests Actions Projects Wiki Security Insights

Quarto-Report Public

Pin Unwatch 1 Fork 0 Star 0

main Go to file + Code

drsaufi First commit from RStudio 8c8932b · 22 minutes ago

| | | |
|---------------------|---------------------------|----------------|
| Datasets | First commit from RStudio | 22 minutes ago |
| .gitignore | First commit from RStudio | 22 minutes ago |
| Analysis.docx | First commit from RStudio | 22 minutes ago |
| Analysis.html | First commit from RStudio | 22 minutes ago |
| Analysis.pdf | First commit from RStudio | 22 minutes ago |
| Analysis.qmd | First commit from RStudio | 22 minutes ago |
| Quarto-Report.Rproj | First commit from RStudio | 22 minutes ago |

About

This repository contains tutorials and guides on creating Quarto documents and publishing them to GitHub.

Activity

0 stars

1 watching

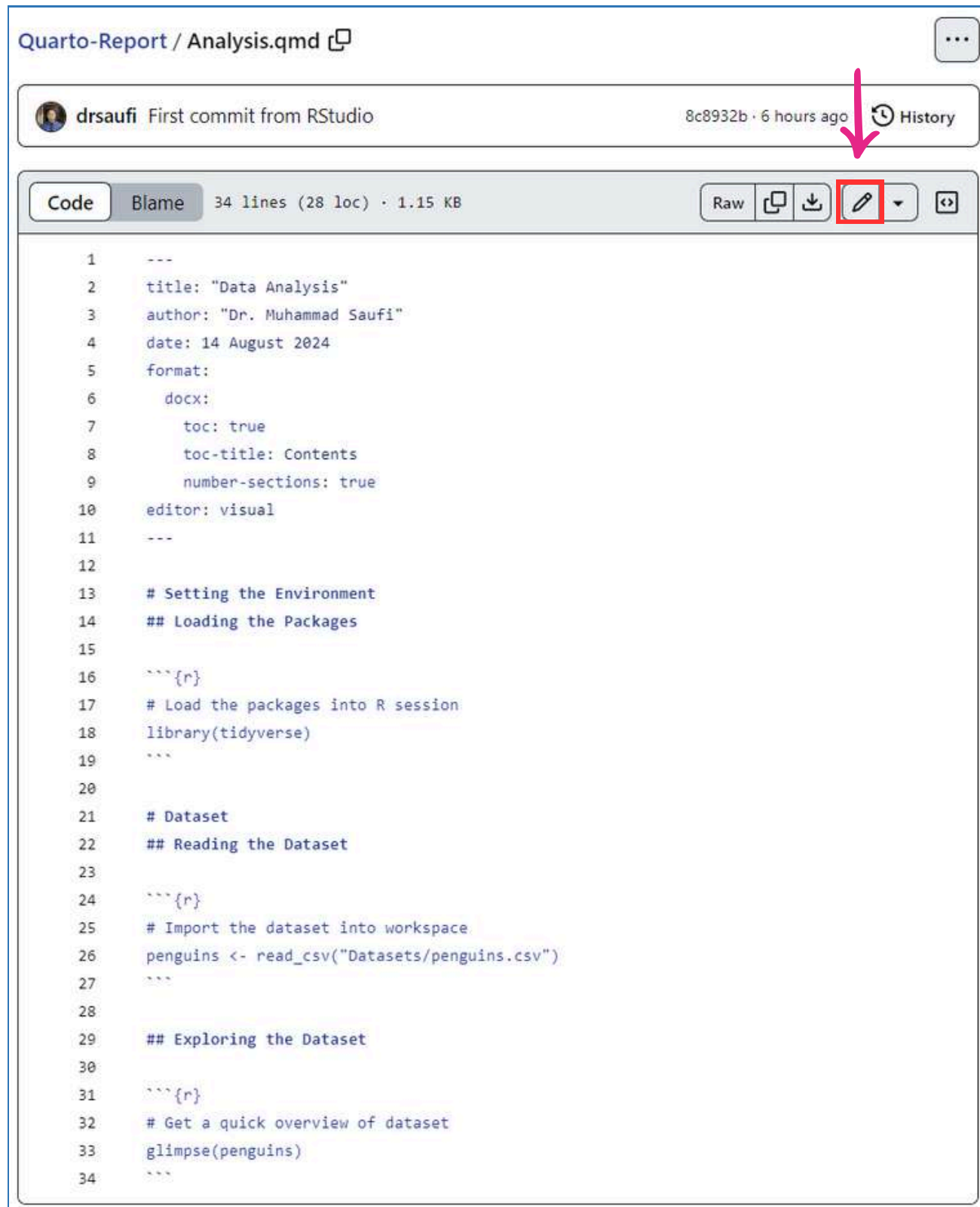
0 forks

Releases

No releases published

[Create a new release](#)

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



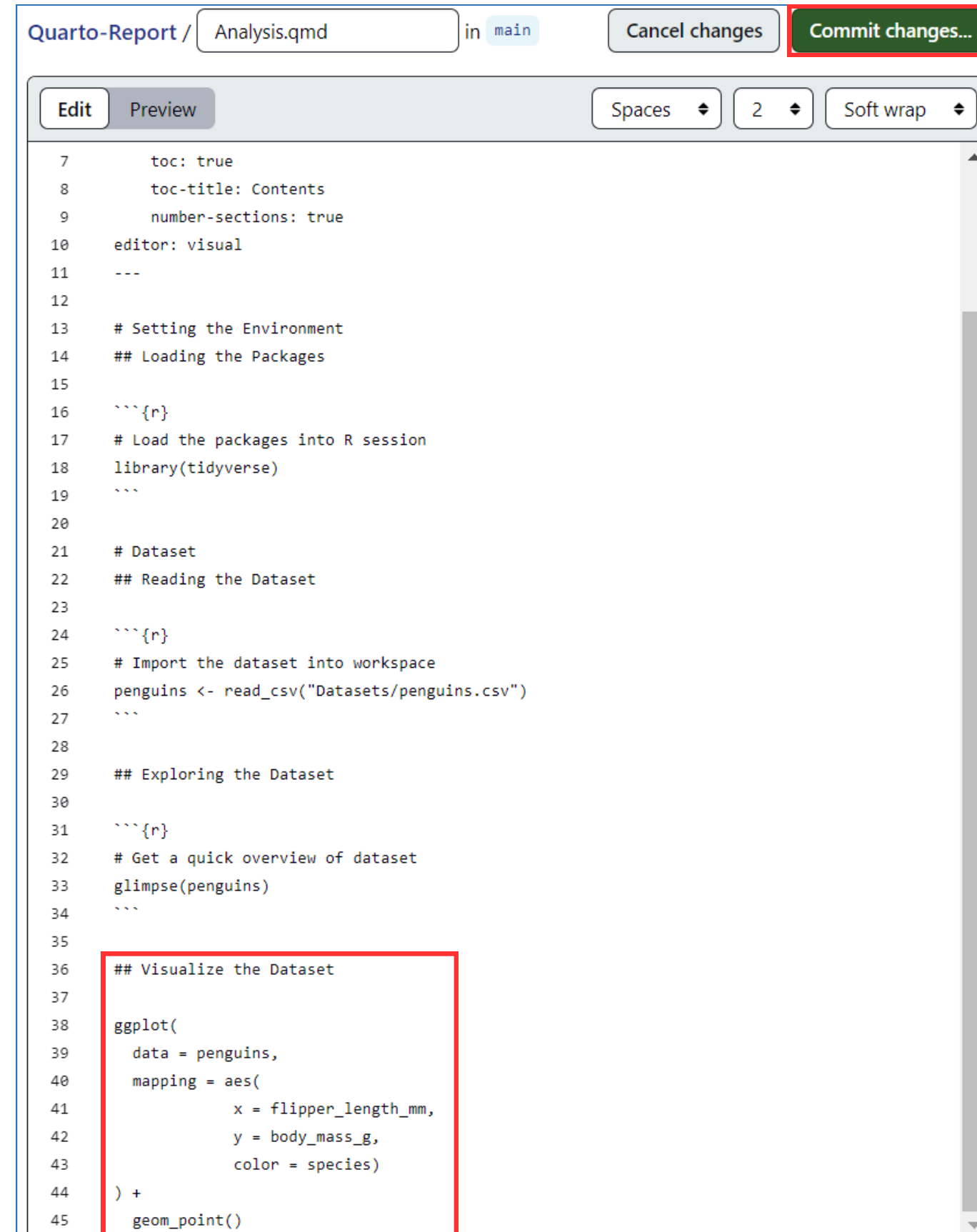
```
Quarto-Report / Analysis.qmd
drsaufi First commit from RStudio 8c8932b · 6 hours ago History
Code Blame 34 lines (28 loc) · 1.15 KB Raw Copy Download Edit this file
1 ---
2 title: "Data Analysis"
3 author: "Dr. Muhammad Saufi"
4 date: 14 August 2024
5 format:
6   docx:
7     toc: true
8     toc-title: Contents
9     number-sections: true
10 editor: visual
11 ---
12
13 # Setting the Environment
14 ## Loading the Packages
15
16 ```{r}
17 # Load the packages into R session
18 library(tidyverse)
19 ```
20
21 # Dataset
22 ## Reading the Dataset
23
24 ```{r}
25 # Import the dataset into workspace
26 penguins <- read_csv("Datasets/penguins.csv")
27 ```
28
29 ## Exploring the Dataset
30
31 ```{r}
32 # Get a quick overview of dataset
33 glimpse(penguins)
34 ```
```

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.



The screenshot shows the Quarto-Report editor interface. At the top, there's a header bar with the text "Quarto-Report / Analysis.qmd" and a "main" branch indicator. To the right of the header are two buttons: "Cancel changes" and "Commit changes...". The "Commit changes..." button is highlighted with a red border. Below the header is a toolbar with "Edit" and "Preview" tabs, and settings for "Spaces" (set to 2) and "Soft wrap". The main area is a code editor with line numbers on the left. The code is in R and includes comments for setting the environment, loading packages, reading a dataset, and visualizing it. The visualization code, starting with `ggplot()`, is highlighted with a red box. The code ends with `geom_point()` on line 45.

```
7     toc: true
8     toc-title: Contents
9     number-sections: true
10    editor: visual
11    ---
12
13    # Setting the Environment
14    ## Loading the Packages
15
16    ```{r}
17    # Load the packages into R session
18    library(tidyverse)
19    ```
20
21    # Dataset
22    ## Reading the Dataset
23
24    ```{r}
25    # Import the dataset into workspace
26    penguins <- read_csv("Datasets/penguins.csv")
27    ```
28
29    ## Exploring the Dataset
30
31    ```{r}
32    # Get a quick overview of dataset
33    glimpse(penguins)
34    ```
35
36    ## Visualize the Dataset
37
38    ggplot(
39      data = penguins,
40      mapping = aes(
41        x = flipper_length_mm,
42        y = body_mass_g,
43        color = species)
44    ) +
45    geom_point()
```

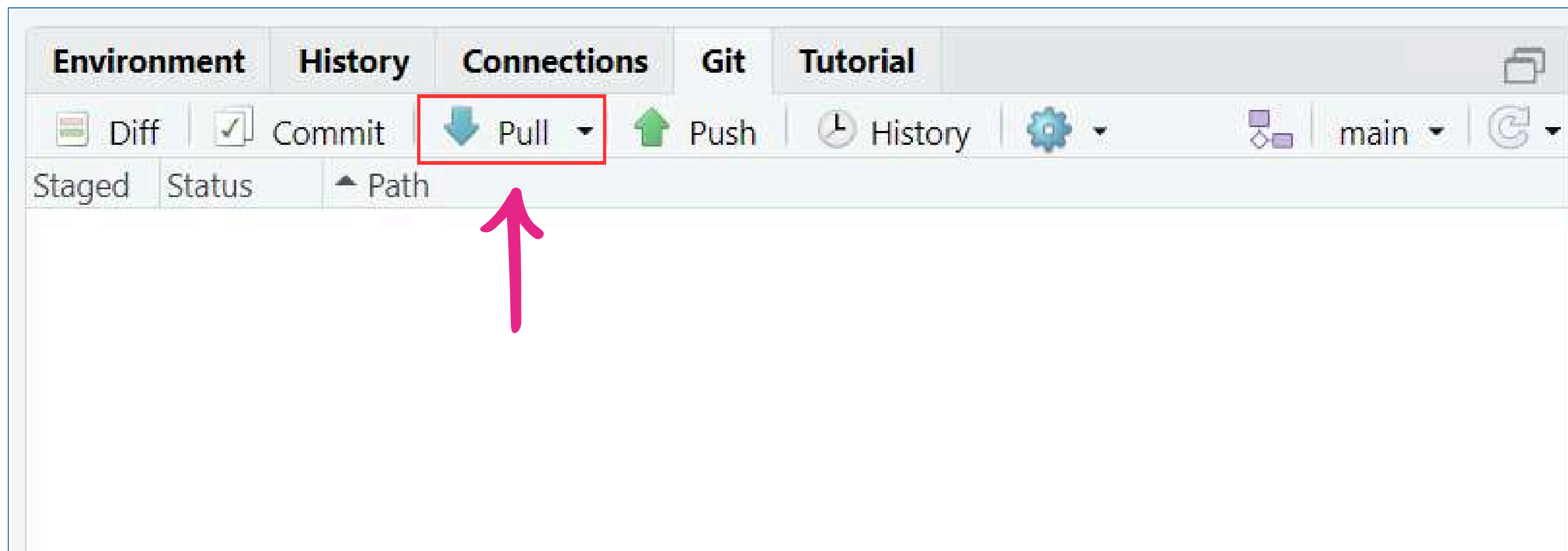
A window will appear where you can optionally add a short description, such as **`New code added from GitHub`**.

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

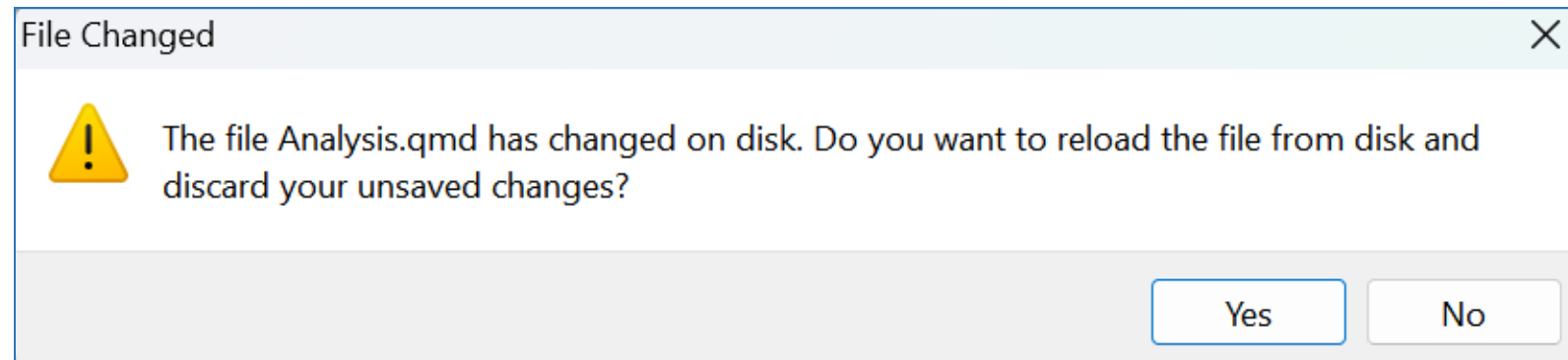
Now that you have committed the changes to your **GitHub** repository, they are not yet reflected in your local repository within **RStudio**.

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.

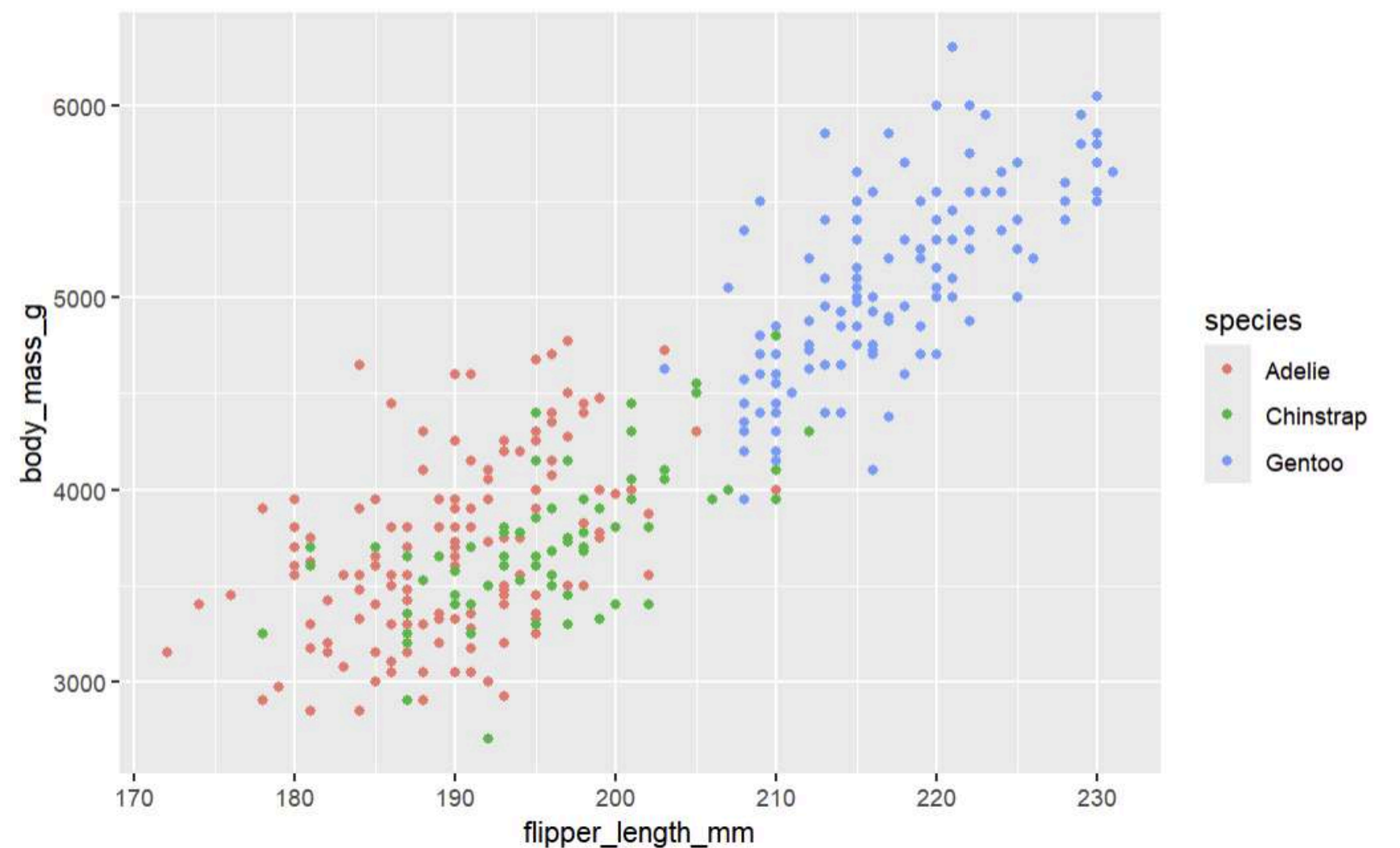


A warning message may appear.
Click **`Yes`** to accept the changes.



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.



2.7 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:

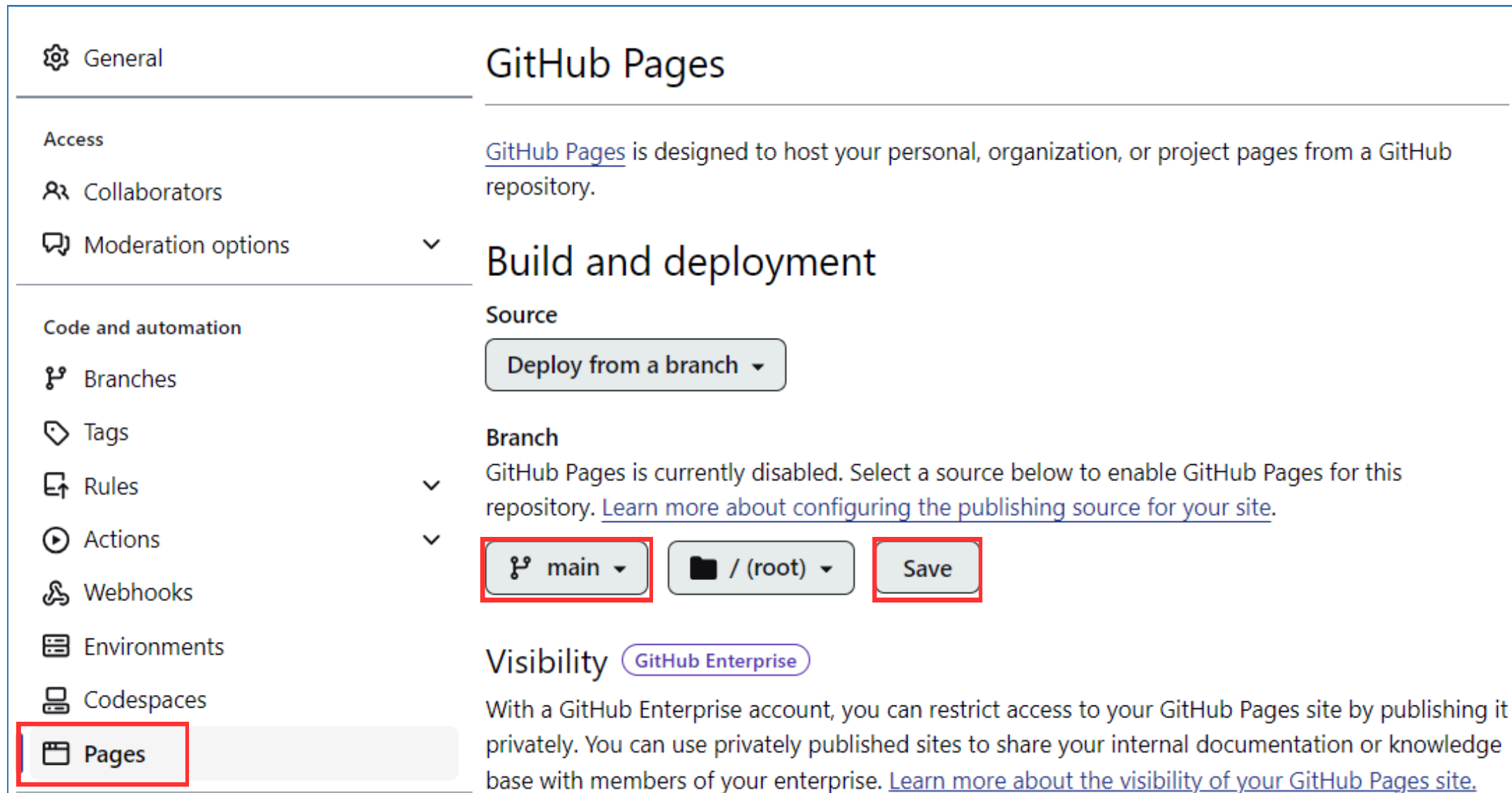
In your **GitHub** repository, click on **`Settings`**.

The screenshot shows the GitHub interface for the repository 'drsaufi / Quarto-Report'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The 'Settings' link is highlighted with a red box. Below the navigation bar, the repository name 'Quarto-Report' is shown with a 'Public' label. To the right of the repository name are buttons for Pin, Unwatch (1), Fork (0), and Star (0). Below this, there are buttons for 'main' branch, 'Go to file', and 'Code'. The main content area displays a commit by 'drsaufi' 4 hours ago (commit hash 9b756dc). The commit message is 'Commit changes'. The file list shows the following files and their commit times:

| File | Commit Message | Commit Time |
|---------------------|---------------------------|--------------|
| Datasets | First commit from RStudio | 13 hours ago |
| .gitignore | First commit from RStudio | 13 hours ago |
| Analysis.docx | Commit changes | 4 hours ago |
| Analysis.html | Commit changes | 4 hours ago |
| Analysis.pdf | Commit changes | 4 hours ago |
| Analysis.qmd | Commit changes | 4 hours ago |
| Quarto-Report.Rproj | First commit from RStudio | 13 hours ago |

On the right side of the repository page, there is an 'About' section with the text: 'This repository contains tutorials and guides on creating Quarto documents and publishing them to GitHub.' Below this, there are statistics: Activity, 0 stars, 1 watching, and 0 forks. At the bottom, there is a 'Releases' section with the text: 'No releases published' and a link to 'Create a new release'.

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



General

GitHub Pages

[GitHub Pages](#) is designed to host your personal, organization, or project pages from a GitHub repository.

Build and deployment

Source

Deploy from a branch

Branch

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more about configuring the publishing source for your site.](#)

main / (root) Save

Visibility [GitHub Enterprise](#)

With a GitHub Enterprise account, you can restrict access to your GitHub Pages site by publishing it privately. You can use privately published sites to share your internal documentation or knowledge base with members of your enterprise. [Learn more about the visibility of your GitHub Pages site.](#)

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](#).

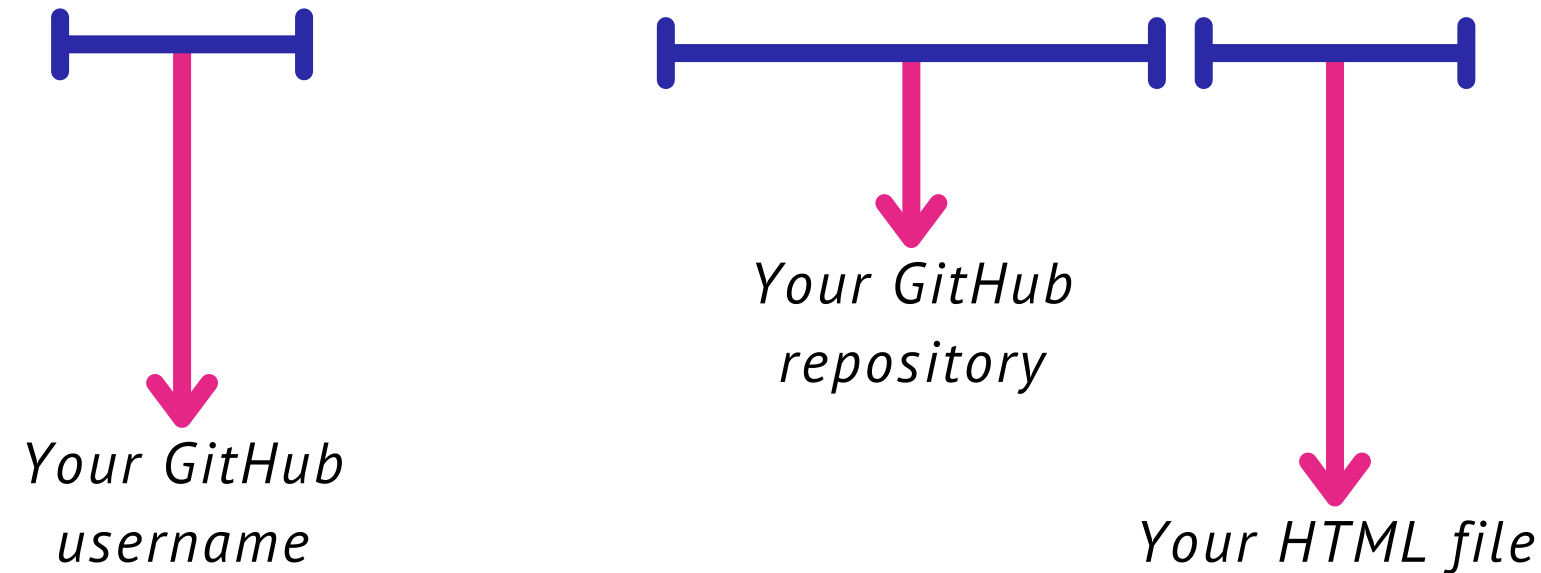
The **URL** for the website is based on your **GitHub** repository's **URL**. For example:

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`**.

Resources:

1. [Data Analysis in Medicine and Health using R](#)
2. [R for Data Science](#)
3. [Quarto](#)
4. [Happy Git and GitHub for the useR](#)