CHAPTER 10

Using GitHub to Manage Versions

OBJECTIVE

This chapter introduces you to the modern and popular techniques of version control with GitHub. GitHub is a code hosting platform for version control and collaboration. As you start developing R scripts, you will find yourself managing several versions of the same code. Having to remember what changes were made to what version for what purpose can become overwhelming at some point. Here is where GitHub can help. Becoming familiar with it is highly recommended, particularly if you will occasionally share your code with others.

Contents

10.1 *Introduction* 375
10.2 *What is GitHub?* 376
10.3 *Using the GitHub Desktop* 377
10.3.1 *Creating a GitHub Account* 378
10.3.2 *Installing GitHub Desktop* 382
10.3.3 *Basic Use of GitHub Desktop* 387
10.4 *How to Use GitHub for Version Control?* 395
10.4.1 *Committing, Reviewing and Pushing Changes* 397
10.4.2 *Branching in GitHub* 403

10.4.3 *Merging and Pulling Branches* 406
10.5 *Cloning and Forking* 415
10.5.1 *Cloning / Forking Your Own Repositories* 416
10.5.2 *Cloning repositories created by others* 419
10.6 *Alternatives to GitHub* 422
10.7 *Concluding Remarks* 425

10.1 Introduction

This chapter is different from the previous ones. It focuses on one popular approach for managing the different versions of your script files in a very effective manner. I will expose you to the *Version Control System* (VCS) called `GitHub`. Although this system was primarily designed to allow large teams of developers to work on the same projects in a collaborative environment, I found it very useful even for single individuals who can always keep track of all the changes they make to the same script file.

Each time you write a script to accomplish a task, you will often modify it either to add a new feature to it or to change the way previous tasks are performed. If you do not manage the different versions of your script, you will run into trouble. Occasionally, you will look at a report you previously created with one version of the script file that you can no longer recover. Which function did you use to create the report? What formula did you implement? Which change did you make to the script and for what purpose? If you have many different script files, such problems can compound to the point where you can not answer version-related questions.

`GitHub` is a vast and complex system that I do not intent to cover in-depth. This clearly is out of the scope of this book. But using this system for yourself as a solo professional is trivial and very useful. I strongly advise that you become familiar with it to some extent now and may improve your knowledge of it overtime.

Can you do without GitHub, by regularly saving various versions of your script file? Technically, it can be done. However, there are a few serious problems that will come with this option, which you will avoid altogether by using GitHub.

- Without GitHub, you will need to open each version of the saved script files so that you can read its content and understand what it is doing.

With GitHub, you always add comments describing the changes made. These comments are always visible on the GitHub's platform `GitHub.com`, along with the file name. No need to open the file to see what changes were made. More important is the fact that `GitHub.com` keeps tract of every single change you make to your script file.

- When adding new features to your script file, you would typically make a backup copy of the original file before you start making the changes. With GitHub, 2 persons (or the same person) can independently work on adding 2 new features to 2 separate backup copies of the script file. Once satisfied with the changes, GitHub allows you to automatically merge the 2 modified copies into a single new version, which will incorporate the 2 new features. Sometimes, GitHub cannot merge both versions of the file due to conflicting changes, in which case a manual intervention becomes necessary. All previous versions always stay on `GitHub.com`. Nothing (almost) gets lost on `GitHub.com`.

  You can always go back to any previous version you want, and start adding new features to it. All this would be impossible with a manual creation of several versions of a script file.

- `GitHub.com` is a website that can be accessed from anywhere in the world. It gives you the ability to decide which files are made public for others to use, and which ones remain private. Therefore, you can share your files with others to use, verify and comment. You will then decide whether or not you will merge these comments to the main script file or not.

10.2 What is GitHub?

How version control systems such as GitHub operate can be quite confusing at first. Nevertheless, I found it very useful to the point where I concluded that having a basic understanding of it was worth the hassle. My goal in this

chapter is to help you get your feet wet. First of all, let me clear up some basic terms that you will often encounter. Whenever people talk about `GitHub`, they actually refer to 2 different software products. These are `Git` and `GitHub`, 2 separate products that work together.

`Git` is a free, stand-alone open-source Distributed Version Control System (DVCS), designed to manage the various versions of your script file. Its keeps tracks of the changes you make, and can reverse these changes. You can install it locally on your system and you would not need another software. That is, `Git` does not need another software to operate. However, having direct interaction with Git can be a very unfriendly experience, and I do not recommend taking that route. GitHub makes your life easier in addition to adding more features that Git does not have.

GitHub is a web-based hosting service for Git repositories, which resides in the cloud. Therefore, it requires the Internet and can be accessed from anywhere. Since GitHub runs Git in the background, you will not need to deal with Git at all in this chapter. But you need to know that it exists in the background and is not to be confused with GitHub, which is a centralized location in the cloud where you maintain your files and their different versions.

Although GitHub is much easier to use than Git, I still found it to be not so friendly. However, there is a `GitHub Desktop` version, which I love very much. In this chapter, the focus will be on `GitHub Desktop`, which you will use to interact with GitHub in the cloud, which in turn will deal with Git. But `GitHub Desktop` must be installed locally. In the next few sections, you will see concrete examples that will show you how useful this can be.

10.3 Using the GitHub Desktop

Before you can use GitHub Desktop to manage the different versions of your script files, you need to have a GitHub account. The procedure for setting up a GitHub account is described in section 10.3.1. Once you have your GitHub

account set up, the next step is to download and install the GitHub Desktop software. Section 10.3.2 presents an overview of this installation process. In section 10.3.3, I will show you some basic use of GitHub Desktop for managing your project directories and their different versions.

10.3.1 *Creating a GitHub Account*

To create your first GitHub account, follow the 3 steps that are described in this section:

1. GitHub is a commercial software, which provides free and paid accounts. You may check their pricing with the link `https://github.com/pricing`. I advise to always start with the free account. As an individual coder, it should be more than sufficient to help you manage your file versions. The pricing webpage is updated often, and you may want to visit it occasionally to stay informed about the changes. Figure 10.1 shows you a summary of the benefits you get from the free plan.

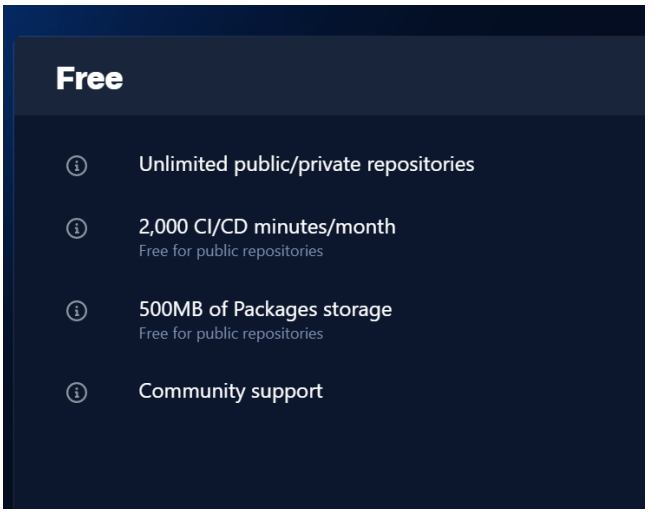

Figure 10.1: Benefits associated with the free account

Having "Unlimited public/private repositories" is likely the single most important benefit you get. For our purpose, a GitHub repository will essentially be any R project directory you will create. I will show you later how to convert a project directory to a Git repository. Moreover, you would make your repository public if you want others to access it and perhaps make changes and suggest these changes to you. Otherwise, you may keep your repository private if you are actively working on it, or you do not want to share it with others for various reasons.

2,000 CI/CD minutes refer to Continuous Integration (CD) and Continuous Delivery (CD). These are concepts you should not worry about, as they apply to teamwork and to a collaborative work environment.

"500MB of Packages storage" is how much free space you have for your repositories. If you eventually need more, you will need to upgrade your plan and possibly pay for it.

2. On the GitHub home page (https://github.com/), click on the sign-up button Sign up.

The first dialog form that you will get is shown in Figure 10.2. You will need to provide an email address that will be tied to your GitHub account.

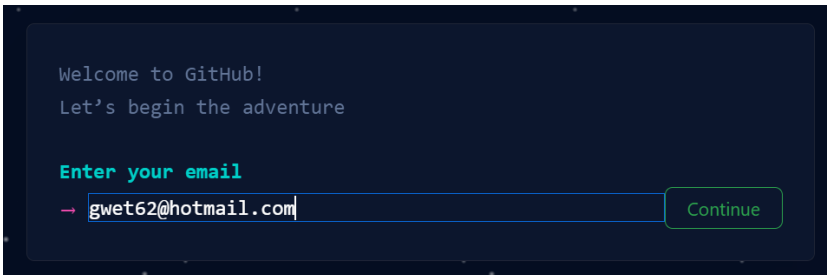

Figure 10.2: Provide email address that will be tied to your account

In the second dialog form of Figure 10.3, you will need to supply your

password, before clicking on the button Continue. I keyed in my password `XXXXXXXXX`, which I expect you to change.

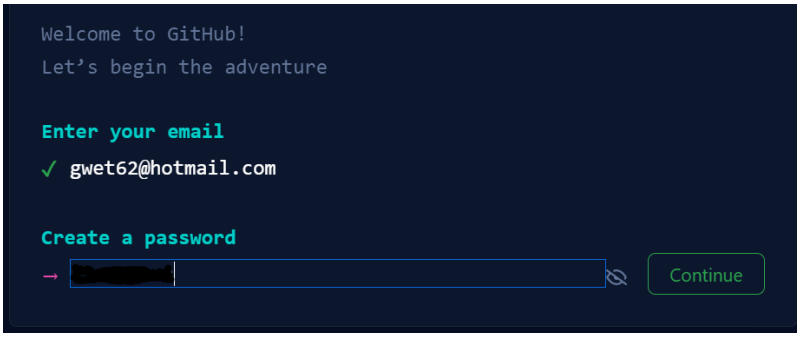

Figure 10.3: Creating your GitHub's account password

Figure 10.4 shows the next dialog form where you need to provide your username. My personal username in this case is `gwet237`.

Figure 10.4: Creating your GitHub's account username

In Figure 10.5, you can see a window containing the `"Create account"` button. Click on that button to create your GitHub account. A verification code will be sent to your registered email address. You will need to enter that code in the window shown in Figure 10.6 in order to complete

the account creation process.

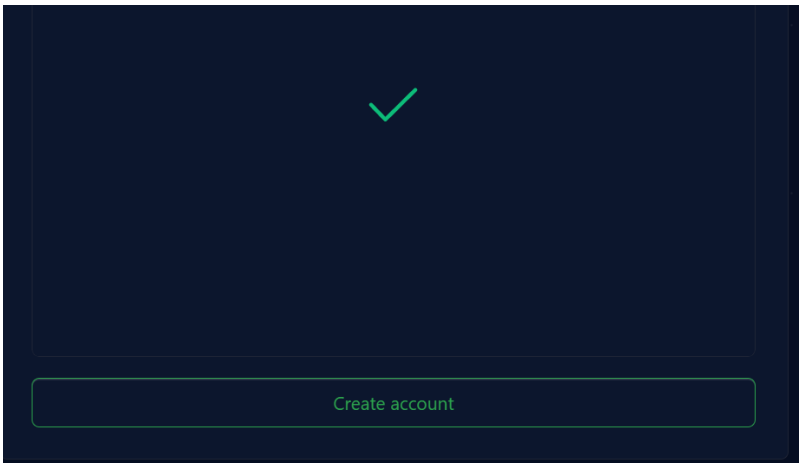

Figure 10.5: Creating your GitHub's account

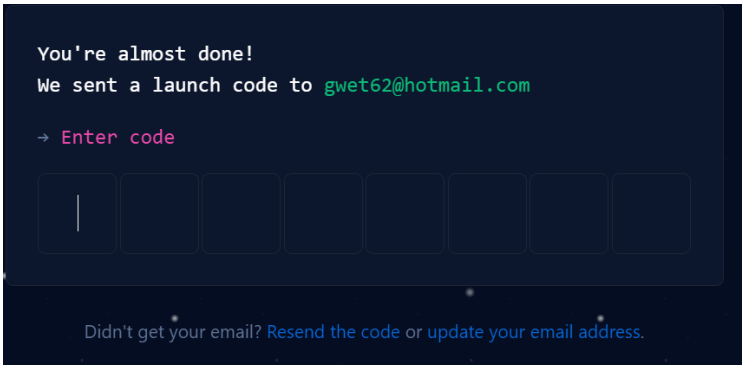

Figure 10.6: Entering the e-mail verification code

3. After providing your verification code as shown in Figure 10.6, you will be given the option to "Continue for free." Select this option. After a few seconds, GitHub will create your dashboard similar to what is shown in Figure 10.7. Although some users work directly from this dashboard, I found it easier to work from the more user-friendly menu-driven GitHub Desktop. I occasionally visit my `GitHub.com` account to check that all the

remote versions of my repositories are in place, and to do some cleaning when I no longer need some repositories.

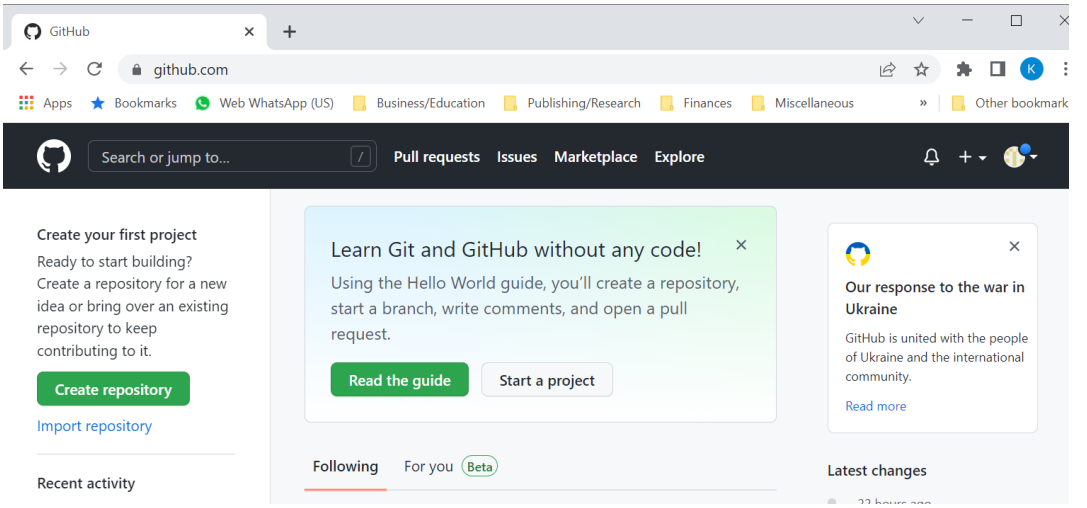

Figure 10.7: Initial GitHub.com's dashboard

10.3.2 *Installing GitHub Desktop*

To install GitHub Desktop, visit the webpage `https://desktop.github.com/` and click on the "Download" button. As an example, if you are using the Windows system, your webpage should appear as shown in Figure 10.8.

After downloading the Desktop GitHub installer in your `Downloads` directory, run it. The installation process is generally completed within a short period of time. Before you create the first Git repository on your hard drive, the dashboard of your GitHub Desktop will have the appearance shown in Figure 10.9. On this dashboard are 3 options, which give some suggestions regarding the different actions you can take at this stage. Before, I review these options, you need to know what a repository is and why it is needed.

Figure 10.8: Entering the e-mail verification code

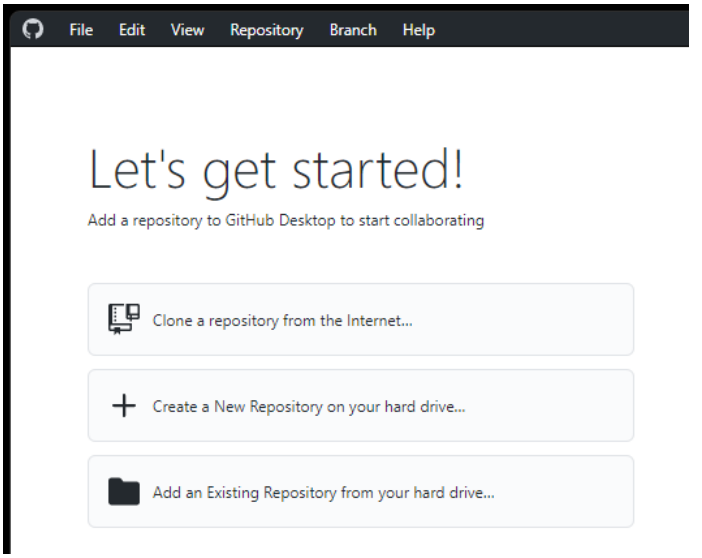

Figure 10.9: GitHub Desktop's dashboard prior to creating your first repository.

Each time I use to the word "`repository`", I will be referring to the `Git Repository`. A Git repository, is a special project directory containing files, datasets, and R script files and which tracks and saves the history of all changes made to the project files. It saves this historical data in a special subdirectory named `.git`, also known as the `Repository Folder`. In chapters 6-9, I created and used several project directories, which are not yet Git repositories (see

Figure 10.10). I will show you in the next few sections how to create new Git repositories based on existing project directories.

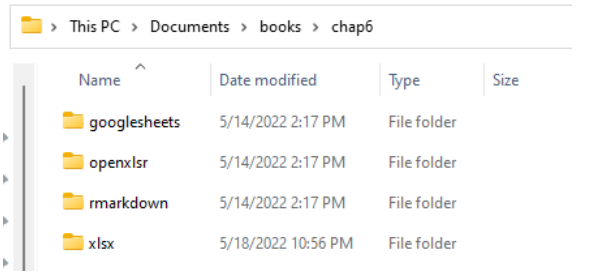

Figure 10.10: Project directories created in chapters 6-9

Getting Started ···

Let us review the 3 options for creating a Git repository with GitHub Desktop:

- The first option that GitHub Desktop offers you is to "Clone a repository from the Internet." Since you have not yet created any repository, none is available, neither locally on your hard drive, nor remotely on your `GitHub.com` account. Therefore, there is nothing to clone.

  > *Cloning a repository means taking a remote version you have on your GitHub.com account, creating a local copy on your computer hard drive, and syncing between the two locations.*

  I will show you later how this can be done, once you have created a remote repository on your `GitHub.com` account. You will generally be asked to sign into your `GitHub.com` account as shown in Figure 10.11. The `"GitHub.com"` tab will need to be selected, and not the `"GitHub Enterprise"`.

  You can already see one key benefit of using GitHub. It does not matter where you are in the world, you can always clone a particular repository

you want and start working on it. Moreover, you can clone someone else's Git repository and start working on it, as long as it was made public. On the same token, if you make your repository public, others may clone it and start making changes, although they may not modify the remote version you have on your `GitHub.com` account. This could be an effective way of collaborating with others on the same large and complex scripts. Note that GitHub provides a convenient way to merge changes made by multiple users, although this feature will not be covered in this book.

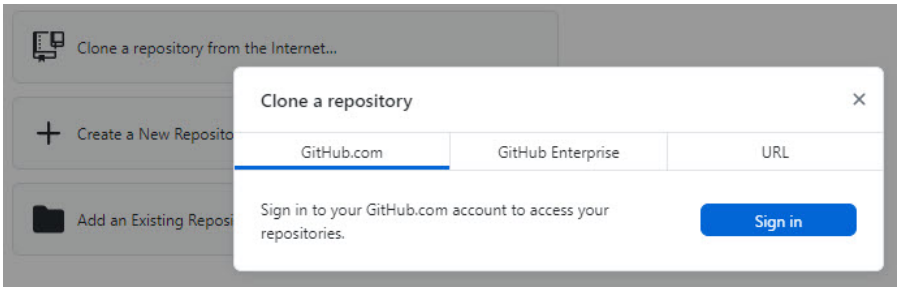

Figure 10.11: Cloning a repository from GitHub.com

- The second option that GitHub Desktop offers for you to get started is to "Create a New Repository on your hard drive." Suppose that you want to use the 4 project directories of Figure 10.10 to create 4 Git repositories that will track all changes made. By selecting this option, GitHub Desktop will display the dialog form of Figure 10.12 for you to fill out. In section 10.3.3, you will see how this is done.

- The third option is to "Add and Existing Repository from your Hard drive." This option is primarily useful if you already have a copy of your Git repository on your hard drive that you like to manage using GitHub Desktop. After selecting this option, GitHub Desktop will display the dialog form of Figure 10.13 for you to fill out. You will hardly need this option, which may prove more useful to those who began using GitHub.com as their primary way of managing their Git repositories and want to switch to GitHub Desktop.

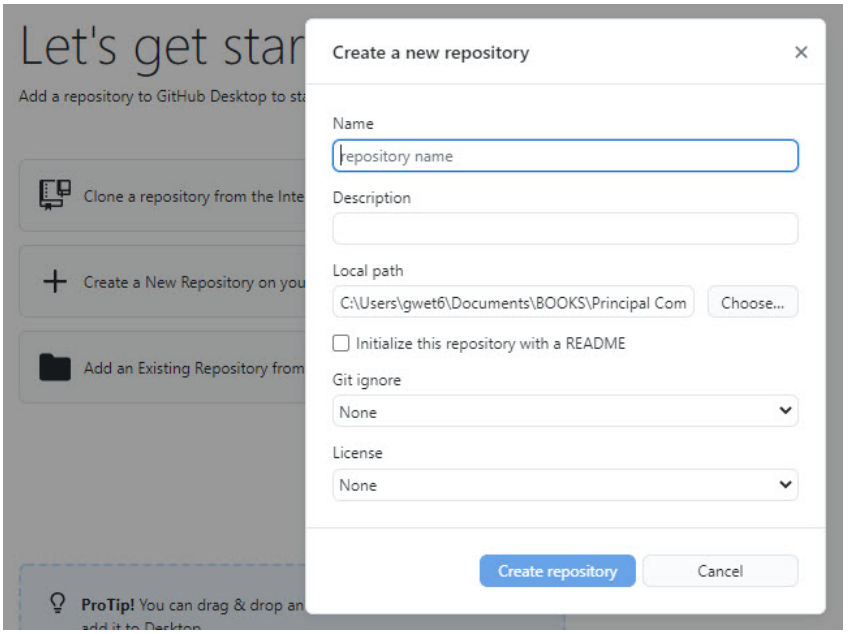

Figure 10.12: Creating a new repository on your hard drive

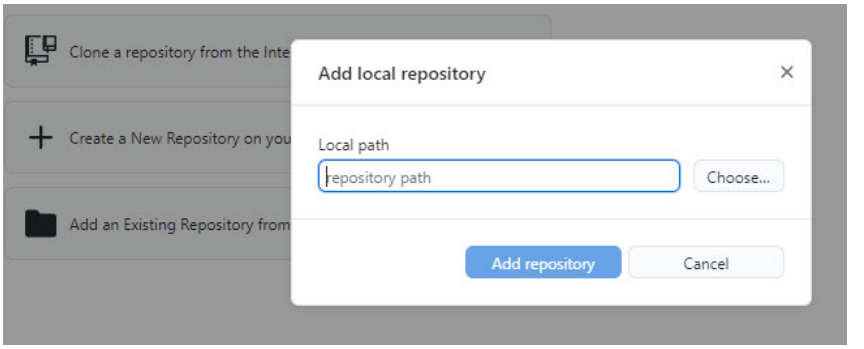

Figure 10.13: Adding an existing repository from your hard drive

10.3.3 *Basic Use of GitHub Desktop*

Now is the time to learn how you can mange the different versions of our project directories. Let us consider the `xlsx` project directory that was created in section 8.2 of Chapter 8 (see Figure 8.1). In this section, I will show you how to convert this project directory to a Git repository using GitHub Desktop and how to publish it to `GitHub.com`.

The best way to follow the guidelines provided in this section, is to create your own copy of the `xlsx` project directory on your hard drive. You can download this directory with the link `https://bit.ly/3BGcgxA`, along with its subdirectories and root directory files. The `"data"` subdirectory contains 2 Excel workbooks `chap6datasets.xlsx` and `chap6datasets1.xlsx`. The `"scripts"` subdirectory contains an R script file named `scr@6xlsx1.r`, listed as Script 8.1 in chapter 8.

Create your first Git repository

From the GitHub Desktop's dashboard (see Figure 10.14), proceed as follows:

1. Click on the + sign to open the `"Create a new repository"` dialog form seen on the right side of Figure 10.14.

2. Type in the repository name. This name cannot contain blank characters, but may include hyphens. In this case, I chose the name `xlsx`, which matches the name of my project directory. Remember that I want my existing project directory to become a Git repository. If you have not yet created a project directory, then you can still create an empty Git repository and make it a project directory at a later time.

3. Provide a short description of the main task that the scripts in this project directory perform. This description will prove useful as the number of

repositories grows, because you will be able to tell what each repository is all about.

4. The fourth step consists of selecting the directory of which `xlsx` is a subdirectory. In my case, I used the following directory:

   `"c:\Users\gwet6\Documents\books\chap6"`.

5. Select the README checkbox, so that a README file can be created for the repository. This important file contains a more detailed description of your project. You could modify it later and use it as a place holder where you could add technical details about your project.

   The `"Git ignore"` field allows you to select the type of files that you do not want to add to the repository. For example some files may be too big, and you may not want GitHub to track each change you make to them. I advise that you ignore this field as a beginner. The `"License"` field is another one that I advise beginners to ignore. If your repository is going to be made public (more on this later), then this field is where to tell the general public what they are authorized to do with your script file.

6. In step 6, click on the `Create repository` button to proceed. This action will change the dashboard appearance as shown in Figure 10.15. Verify that the name of your newly-created repository is displayed at the top-left corner of the dashboard as your "Current repository." If so, then you have just created the local version of your Git repository.

   If you look at the `xlsx` directory on your hard drive again, you will notice that a new subdirectory named `.git` and a file named `.gitattributes` have been created. This is another sign that your directory is now a Git repository that will regularly keep a record of all the changes you make to your project directory.

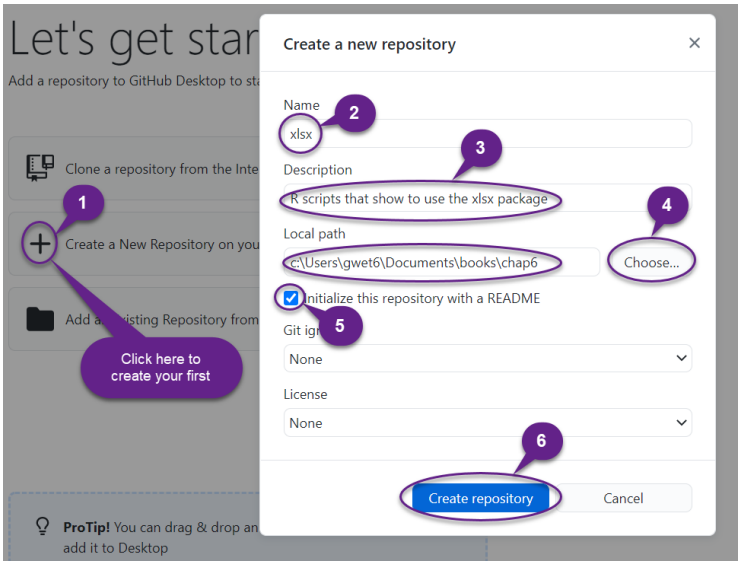

Figure 10.14: Dialog form for creating your first Git repository

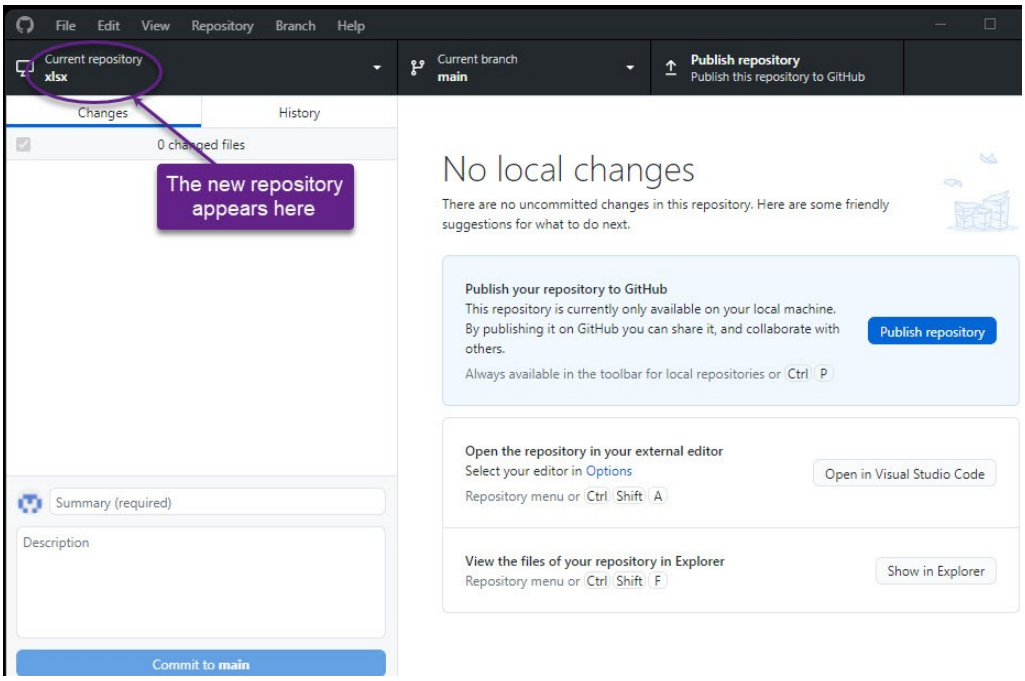

Figure 10.15: Publishing your first repository to `GitHub.com`