Introduction to Git using R

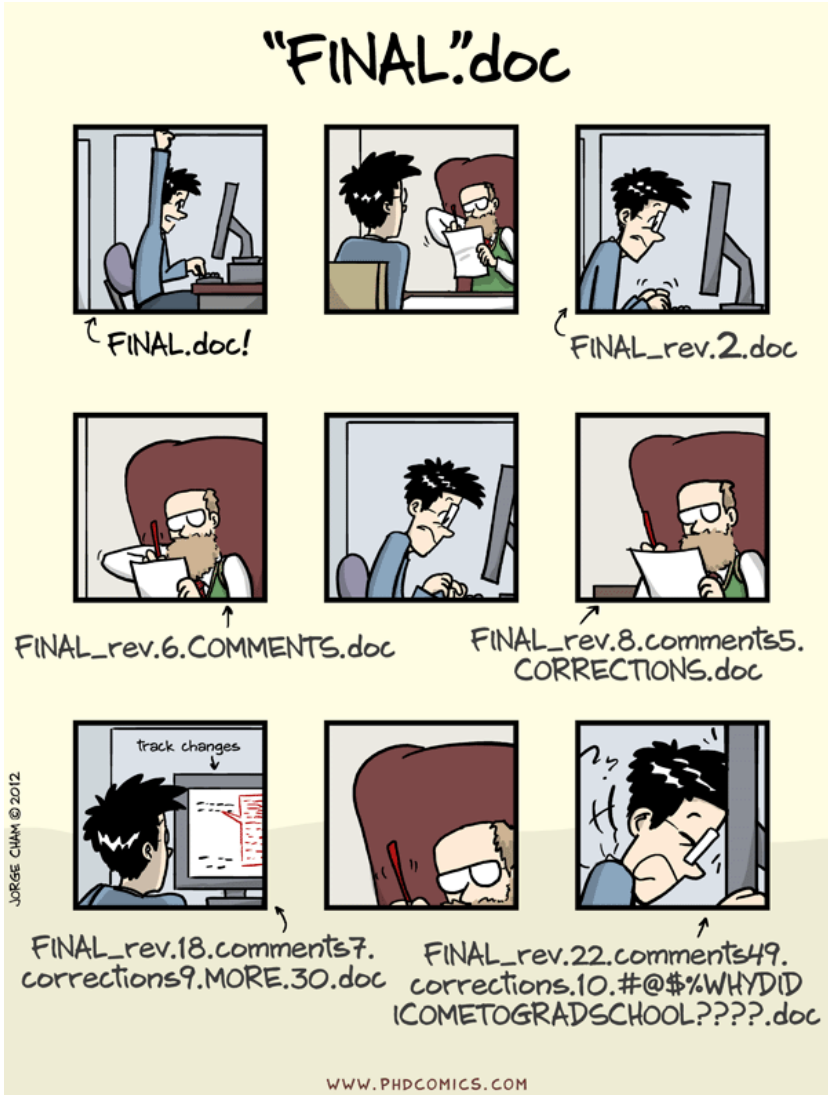
CDSI COMPUTATIONAL AND DATA SYSTEMS INITIATIVE

Dr. Tim Elrick | 26 March 2024

What you will learn today

- what are the benefits of using Git
- how to configure Git and set it up in general and in R
- how to connect to Git, GitHub and GitHub repositories
- how to use basic Git commands on the command line
- how to use basic Git commands in R

Ever been in this situation?



Source: http://phdcomics.com/comics/archive_print.php?comicid=1531

Why Git?

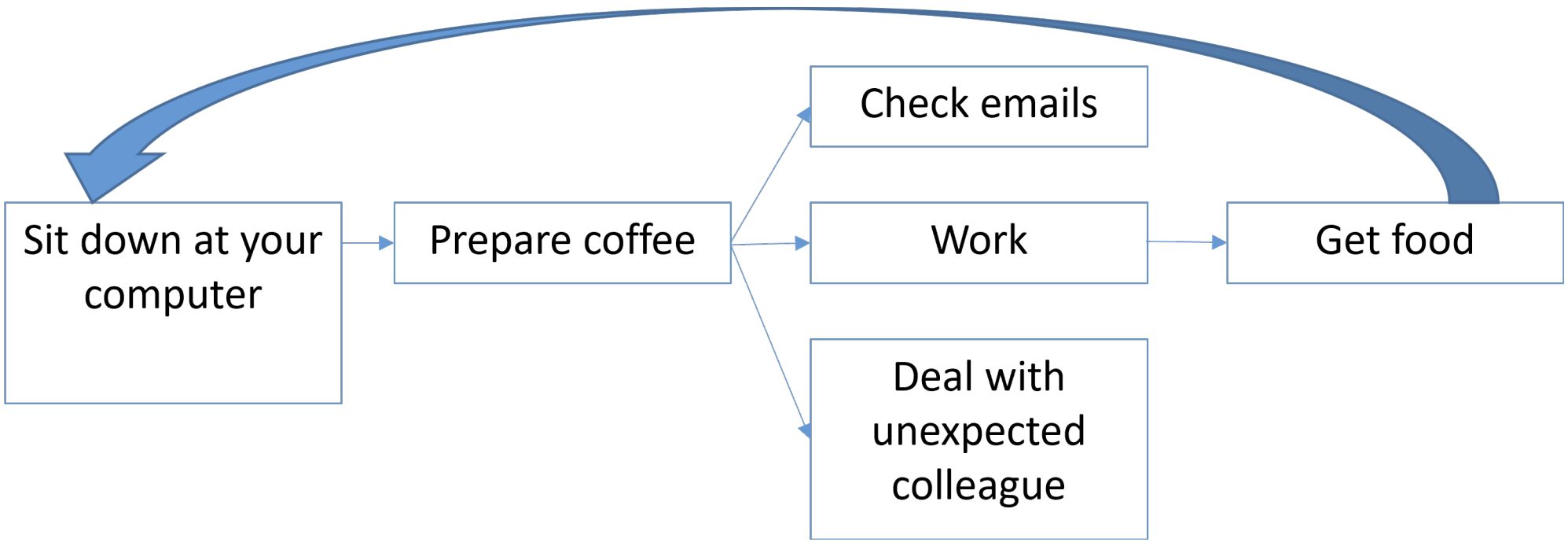
- Git is an industry standard version control system (VCS) for tracking changes in code and text files
- supports non-linear (i.e. branching) development (i.e. many versions of one thing)
- free and open-source
- you can use GitHub (or the like) to store your data online

Why not use Dropbox?

- Trust in our first filename
- Unlimited undo/redo/saves/getting back previous versions
- Keep a ‘master’ file while allowing others to branch on their own
- Selectively allow changes
- Easily share files through GitHub, promote open-access/ reproducibility
- Make use of workflows

Workflows

Workflows can be anything:

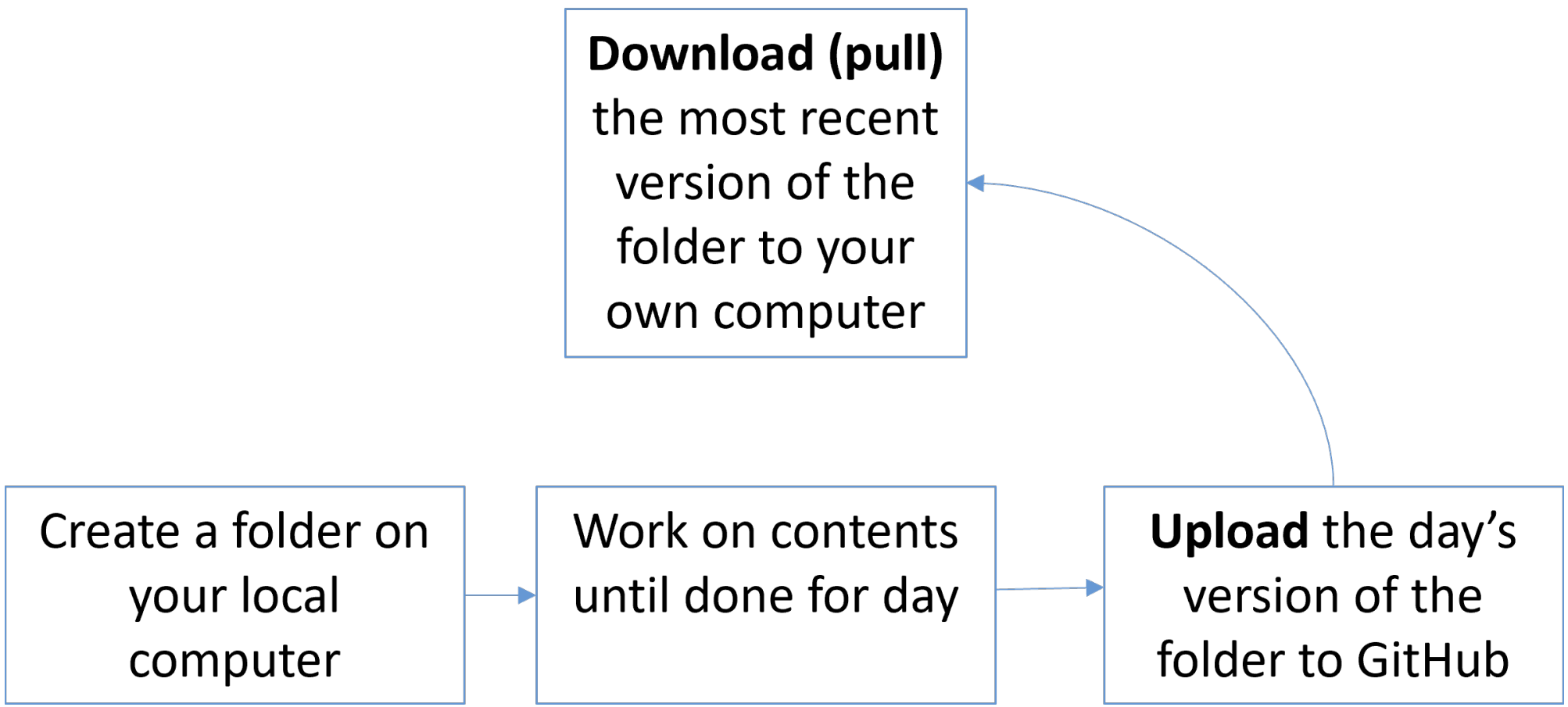


or

https://rvprasad.medium.com/a-git-workflow-for-writing-papers-in-latex-4cfb31be4b06

Workflows

What your (basic) GIT workflow might look like:



What's the catch?

- Git was developed to keep track of changes of the Linux kernel (not for data science projects)
- Git works best with text files (R, Python, SQL, CSV, …)
- Git is a command-line tool

Working with Git prerequisites

- Installation of Git

  local version control system (VCS)

- Optional: GitHub/BitBucket/SVN/… account

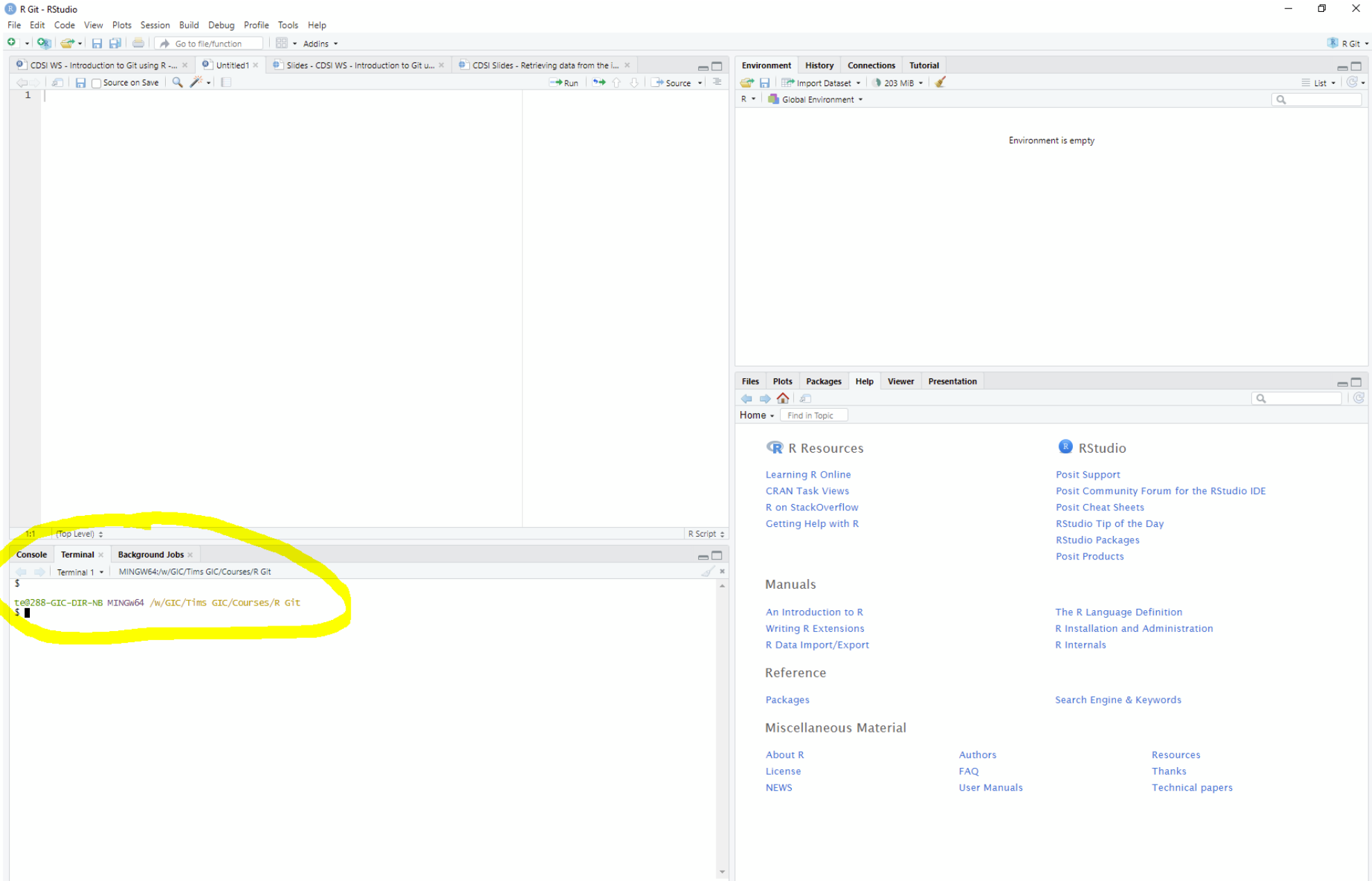
  online extension of your VCS

- Optional: Git client (GitKraken, GitHub Desktop, …)

  local graphical user interface (GUI) of your VCS

Verifying that Git works I

After installation of Git, (re)open RStudio and go to the **Terminal** window.


R Git - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
R Git
CDSI WS - Introduction to Git using R -...
Untitled1
Slides - CDSI WS - Introduction to Git u...
CDSI Slides - Retrieving data from the i...
Source on Save
Run
Source
1:1
(Top Level)
R Script
Console
Terminal
Background Jobs
Terminal 1
MINGW64:/w/GIC/Tims GIC/Courses/R Git
$
te@288-GIC-DIR-NB MINGW64 /w/GIC/Tims GIC/Courses/R Git
$
Environment
History
Connections
Tutorial
Import Dataset
203 MiB
List
R
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Presentation
Home
Find in Topic
R Resources
Learning R Online
CRAN Task Views
R on StackOverflow
Getting Help with R
RStudio
Posit Support
Posit Community Forum for the RStudio IDE
Posit Cheat Sheets
RStudio Tip of the Day
RStudio Packages
Posit Products
Manuals
An Introduction to R
Writing R Extensions
R Data Import/Export
The R Language Definition
R Installation and Administration
R Internals
Reference
Packages
Search Engine & Keywords
Miscellaneous Material
About R
License
NEWS
Authors
FAQ
User Manuals
Resources
Thanks
Technical papers

Verifying that Git works II

In the **terminal** type:

- `which git`
- `where git`
- `git -v`

For the last command the reply should be something along these lines:

```
te@288-GIC-DIR-NB MINGW64 /w/GIC/Tims GIC/Courses/R Git
$ git -v
git version 2.43.0.windows.1
```

Let's introduce ourselves to Git

In the **terminal**, type:

```
git config --global user.name "Jane Doe"
git config --global user.email
"jane.doe@mail.mcgill.ca"
```

then

```
git config --global --list
```

alternatively, in the **R console**, type:

```r
library(usethis)
use_git_config(user.name = "Jane Doe",
               user.email = "jane.doe@mail.mcgill.ca")
# then
git_sitrep()
```

Establishing connection to GitHub

For security reasons, GitHub has stopped using passwords when connecting programmatically from your local machine to GitHub.

You can either use **tokens with HTTPS** or **keys with SSH**.

For ease in this workshop, we will be using a **personal access token (PAT)**:

```
usethis::create_github_token()
```

alternatively, go to https://github.com/settings/tokens and click *Generate token*.

Note, you need this token! Copy it to your clipboard now!

Generating a token

Invoking `usethis::create_github_token()` will send you to the GitHub website.

In the `Note` section you should describe what you want the token to use for.

Set an `Expiration` date.

Leave the other settings as set by the default.

Click `Generate token` at the bottom of the web page.

Working with tokens

Now, you can either save your PAT somewhere savely on your computer (e.g. in a password manager) …

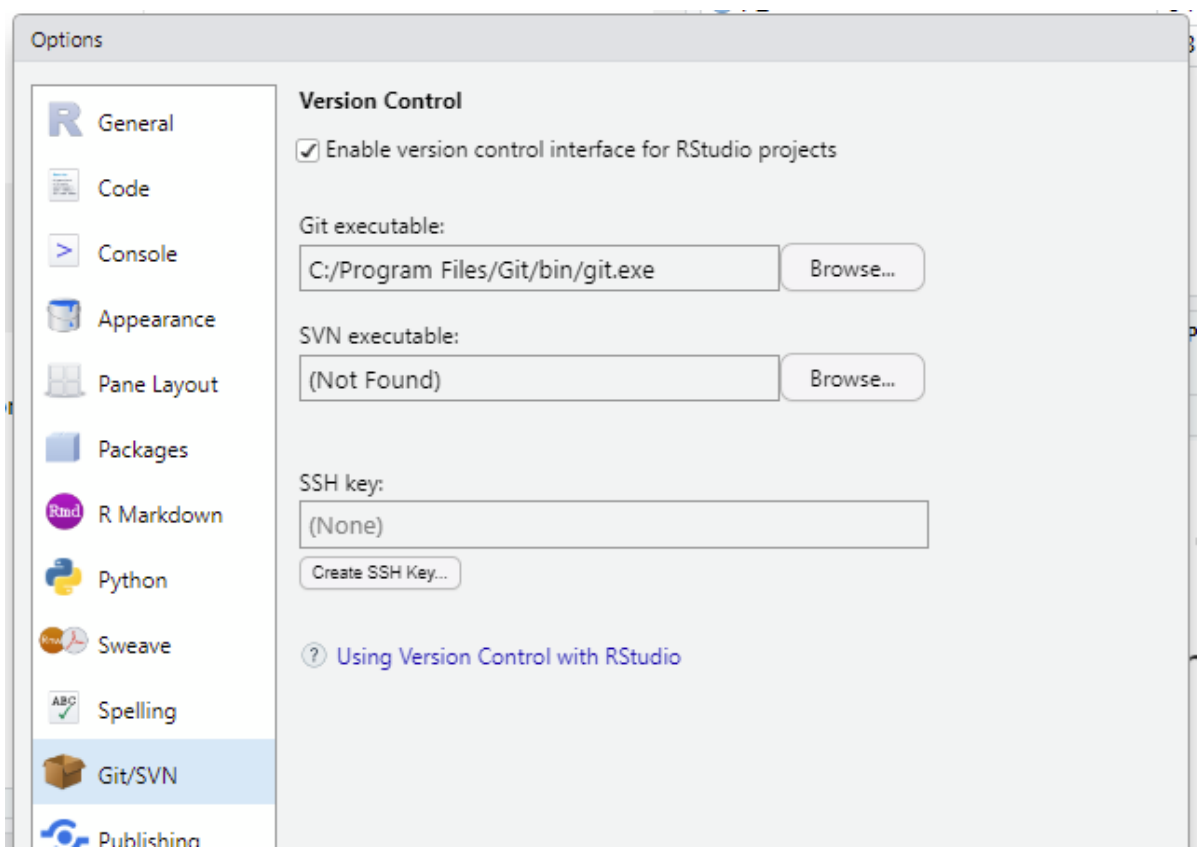
or let R take care of it with

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

If your PAT goes missing, you need to re-generate your token at https://github.com/settings/tokens using the same description as before (or generate a new token).

Detect Git from RStudio

In RStudio go to menu `Tools > Global Options... > Git/SVN`



Do you see a path displayed for *Git executable*?

Path to Git …

… on **Mac/Linux** should be `/usr/bin/git`

… on **Windows** should be `C:/Program Files/Git/bin/git.exe`

Note for Windows users: Do **not** use `git-bash.exe` or `C:/Program Files/Git/cmd/git.exe`!

Git basics

Ok, now we are set, finally! Let's start using Git…

Every time you want Git to store a *version/snapshot* of a file you need to **commit** it.

Before you can do so, you need to **stage** the file.

A file is only *staged* and *committed* if it was *modified*.

Git basic states

- **unmodified** file: nothing will happen to the file
- **modified** file: the has been changed compared to a previous version
- **staged** file: the modified file will be included in the next snapshot
- **committed** file: the file is stored in your local Git database

Let's create some files

- In **RStudio** create a new project (`File > New Project... > New Directory > New Project`). Let's call this ***test***.
- Then, go to the **terminal** window.

Start an RScript and create some code and data.

```
ds <- data.frame(name = letters[1:10],
                 number = 1:10)
write.csv(ds, "data.csv")
```

Then save the RScript as `code.R`.

Basic Git commands using the terminal

`git status`	checks status of Git
`git init`	creates a Git database for current folder
`git add <file name>`	adds file to staging pool
`git add .`	adds all files in folder to staging pool
`git commit -m "My commit message"`	adds staged files to Git database

Basic Git commands – step by step

First, let's list the content of the directory with `ls`:

```
te@288-GIC-DIR-NB MINGW64 ~/Desktop/test (main)
$ ls
code.R  data.csv  test.Rproj
```

Now, check the status of Git:

```
$ git status
fatal: not a git repository (or any of the parent directories): .git
```

So, we have to initiate the Git database first:

```
$ git init
Initialized empty Git repository in C:/Users/te/Desktop/test/.git/
```

Basic Git commands – step by step

When we now check the status of Git:

```
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        code.R
        data.csv
        test.Rproj

nothing added to commit but untracked files present (use "git add" to track)
```

Ok, we need to *stage* our files now with:

```
$ git add .
```

Basic Git commands – step by step

Check the Git status again:

```
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   code.R
        new file:   data.csv
        new file:   test.Rproj
```

With `commit` we now add these *staged files* to the Git database. Note, you need to provide a description:

```
$ git commit -m "First commit"
[main (root-commit) e419468] First commit
 3 files changed, 27 insertions(+)
 create mode 100644 code.R
 create mode 100644 data.csv
 create mode 100644 test.Rproj
```

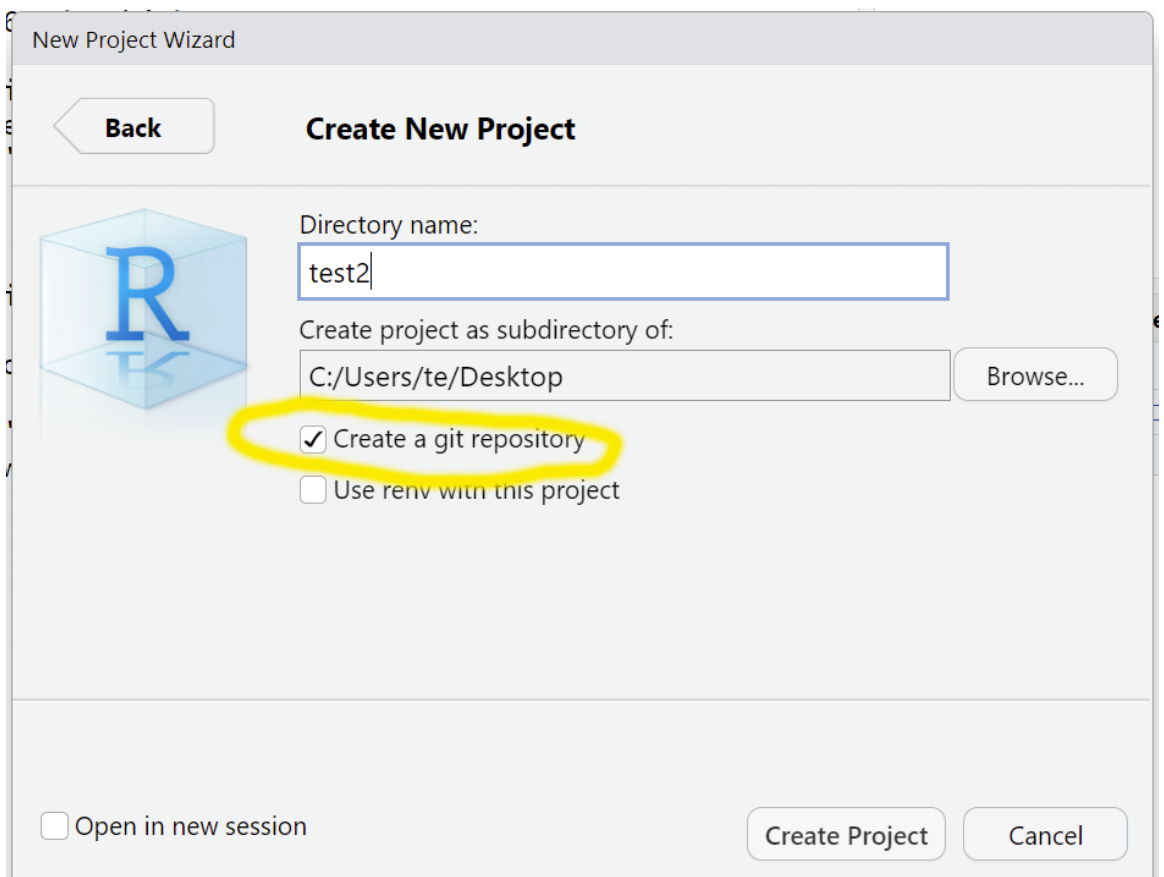
Basic Git commands – step by step

Check the Git status again:

```
$ git status
On branch main
nothing to commit, working tree clean
```

Let’s create some more files

In **RStudio** create a new project (`File > New Project...`) called ***test2***. But now toggle on `Create a git repository`:



Create the files now

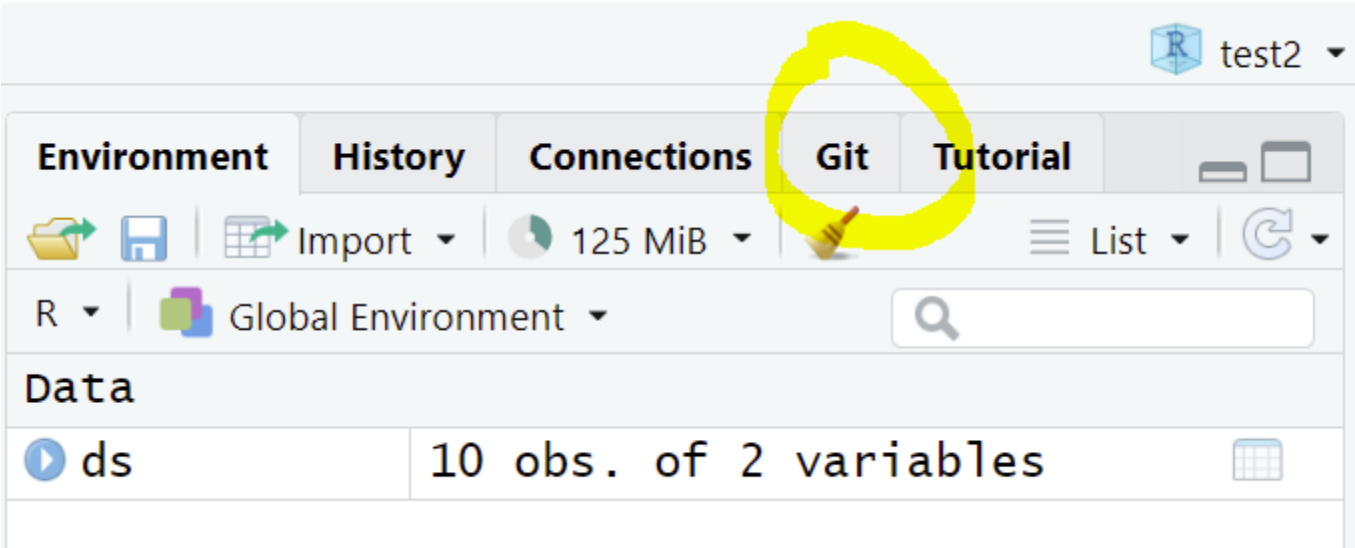
Start an RScript and create some code and data.

```
ds <- data.frame(name = letters[11:20],
                 number = 11:20)
write.csv(ds, "more_data.csv")
```

Then save the RScript as `more_code.R`.

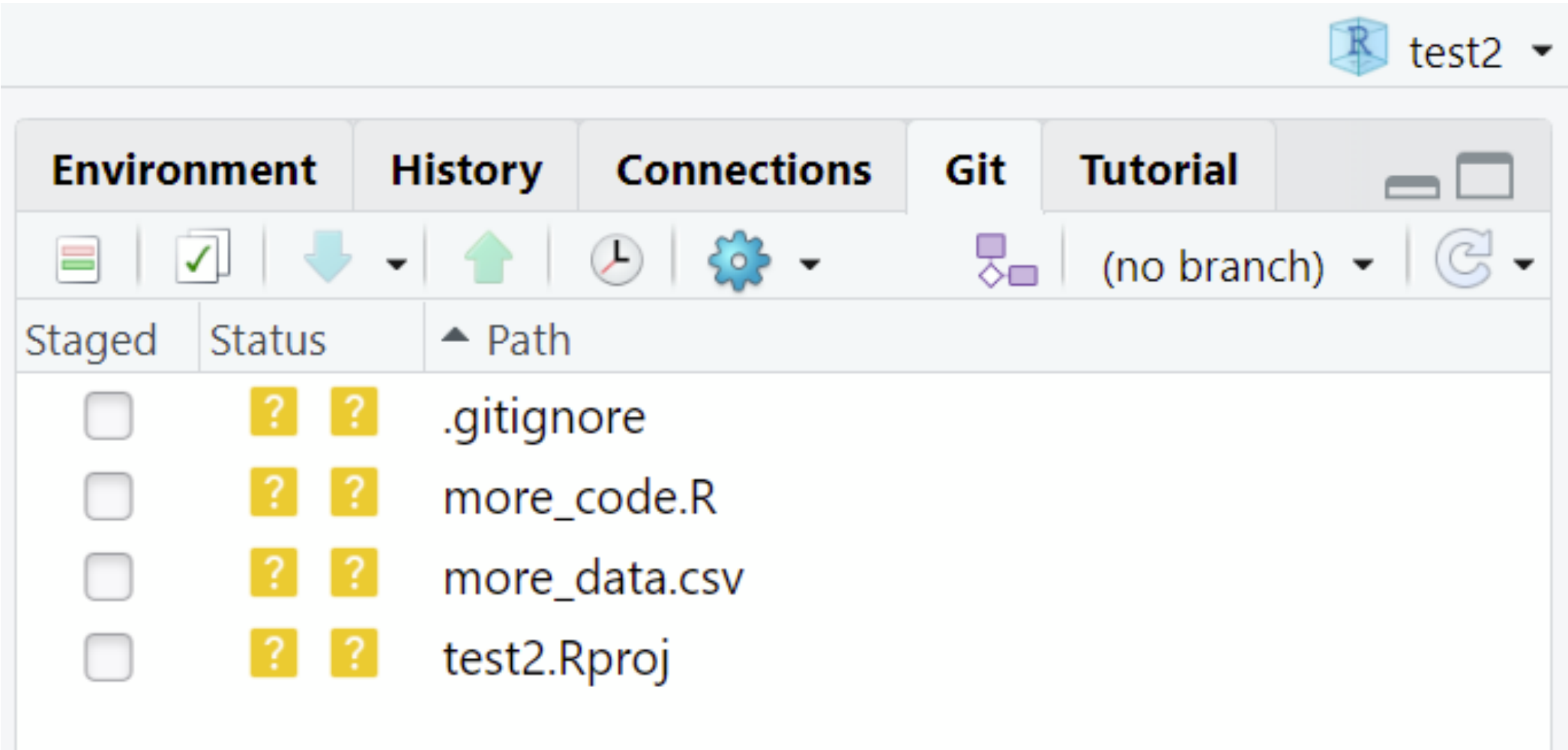
Using Git in R

So, what is different?



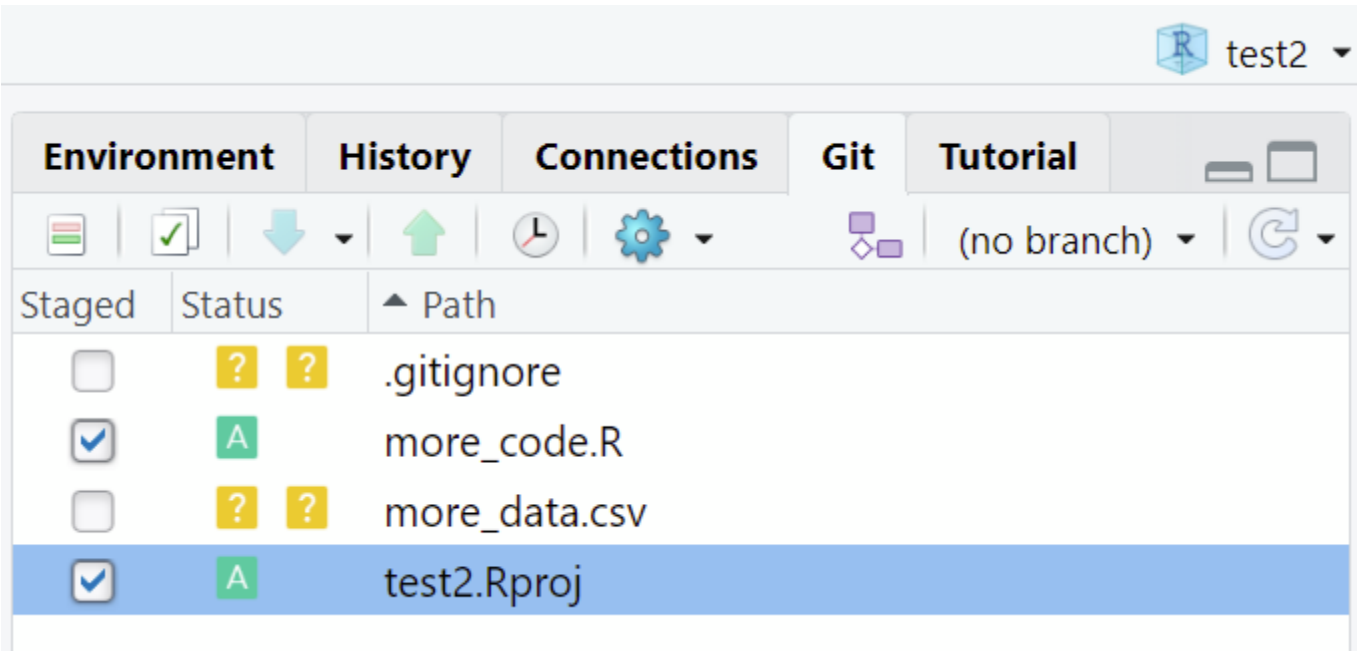
The `Git` tab also appears on the project ***test*** where we initiated Git from the command line. However, it will only appear after you re-opened the project (as we initiated it *manually*).

The *Git* tab in R



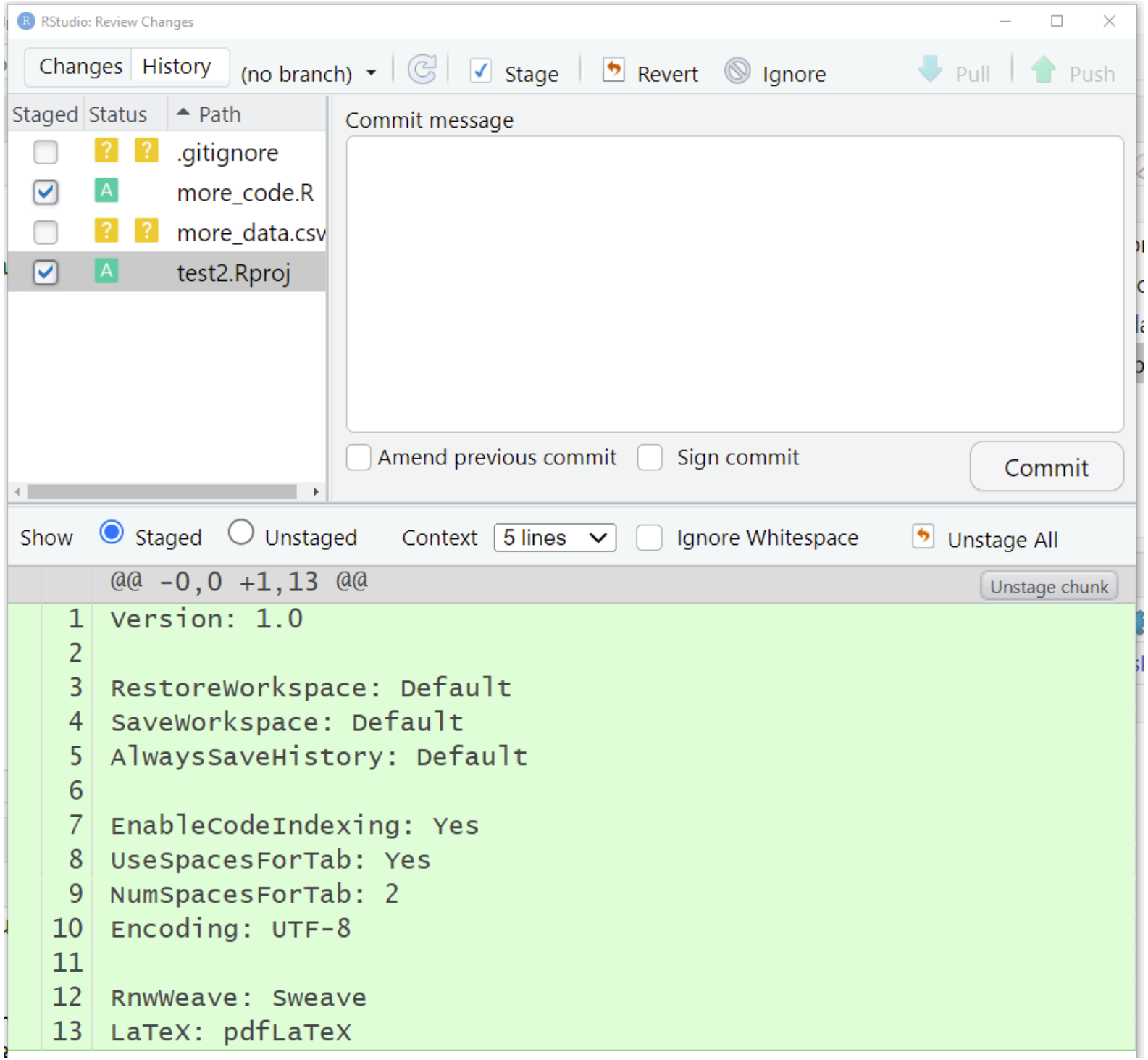
Basic Git steps in R - *Staging*

By toggling the checkbox for a file, we can add it to the staging pool:



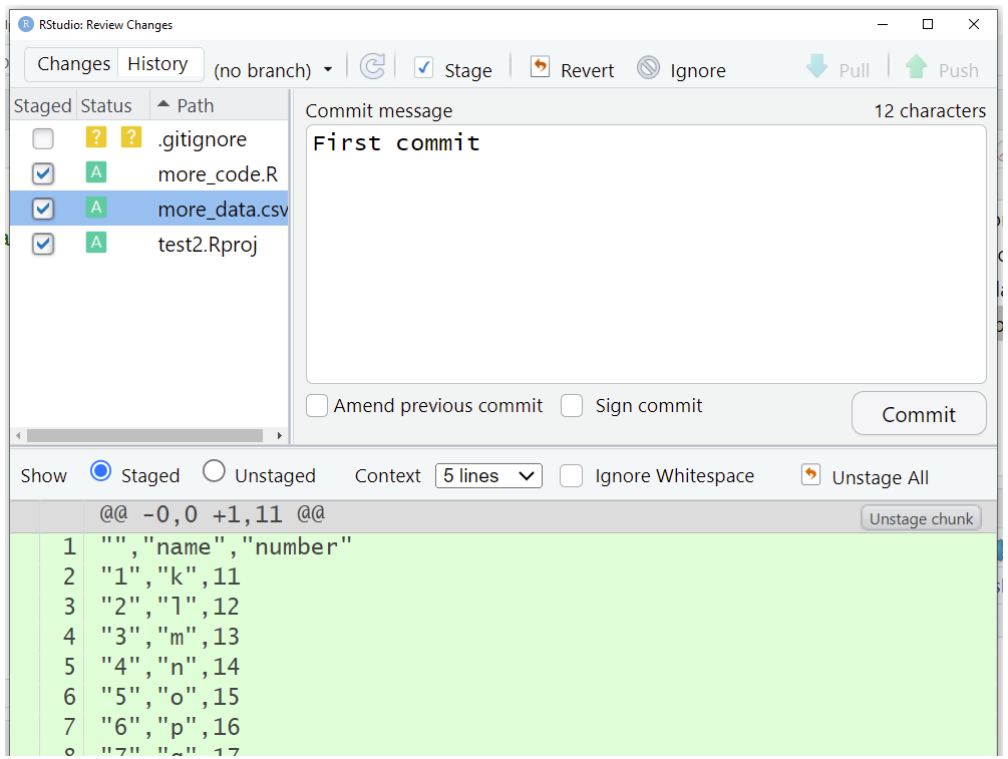
Basic Git steps in R - *Committing*

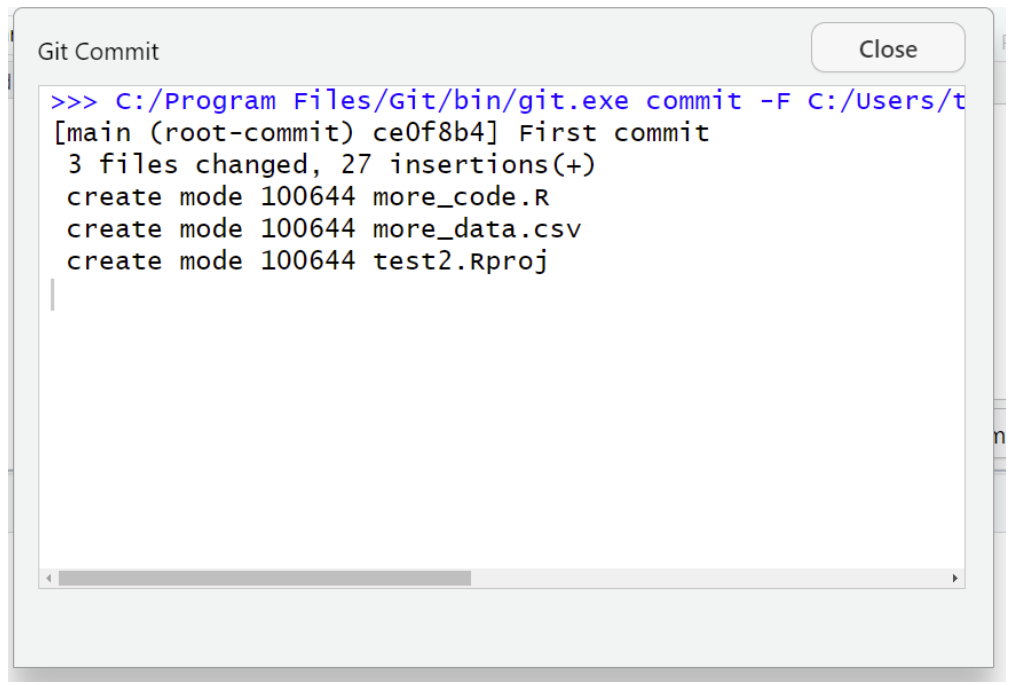
In the `Git` tab, click on *Commit pending changes*:  or use shortcut `Ctrl+Alt+M`. The *Commit window* will open:


RStudio: Review Changes
Changes
History
(no branch)
Stage
Revert
Ignore
Pull
Push
Staged
Status
Path
.gitignore
more_code.R
more_data.csv
test2.Rproj
Commit message
Amend previous commit
Sign commit
Commit
Show
Staged
Unstaged
Context
5 lines
Ignore Whitespace
Unstage All
@@ -0,0 +1,13 @@
Unstage chunk
Version: 1.0
RestoreWorkspace: Default
SaveWorkspace: Default
AlwaysSaveHistory: Default
EnableCodeIndexing: Yes
UseSpacesForTab: Yes
NumSpacesForTab: 2
Encoding: UTF-8
RnwWeave: Sweave
LaTeX: pdfLaTeX

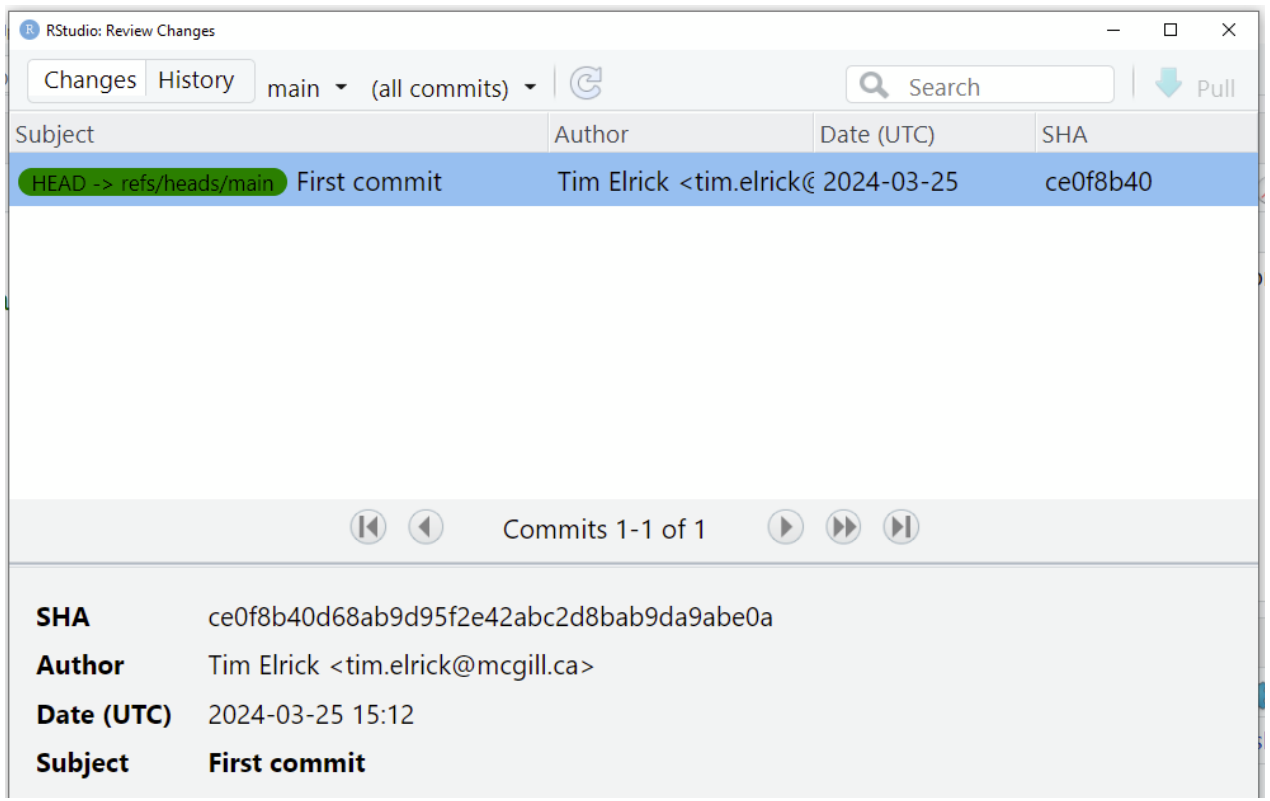
Basic Git steps in R - *Committing*

Enter your *commit message* (description) and click *Commit*.





Tracking changes in Git



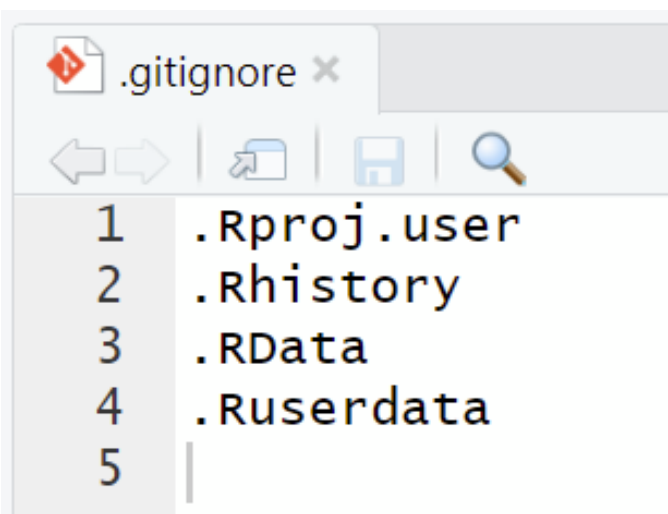
or using the **terminal** with `git log`:

```
$ git log
commit ce0f8b40d68ab9d95f2e42abc2d8bab9da9abe0a (HEAD -> main)
Author: Tim Elrick <tim.elrick@mcgill.ca>
Date:   Mon Mar 25 11:12:50 2024 -0400

    First commit
```

What is this *.gitignore* about?

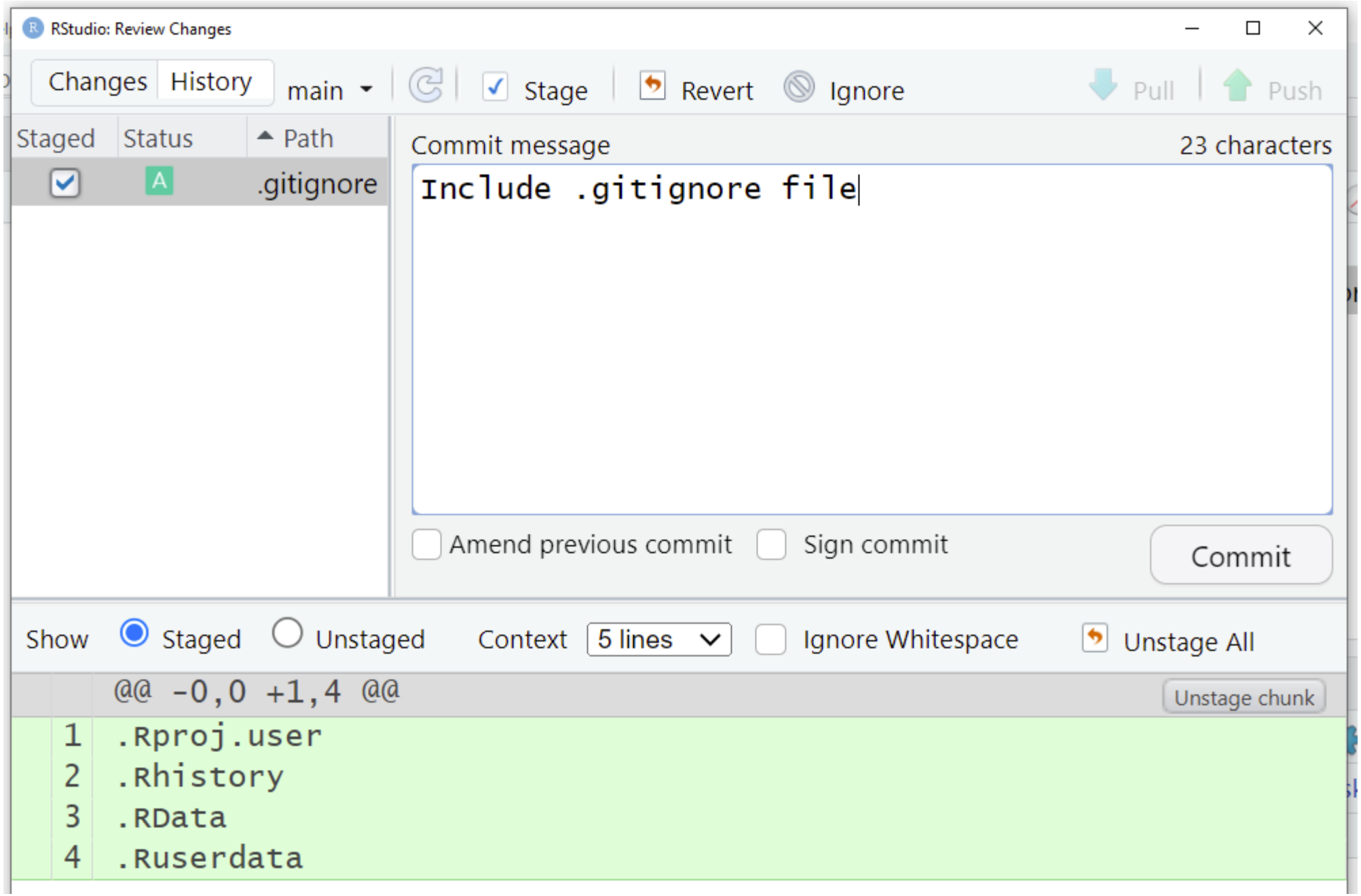
The `.gitignore` file stores file names that should **not** be included in your Git database, i.e. files you do not want to track changes, like temporary files or *.Rhistory* and *.RData* which allow you to speed up work on your computer, but could be replicated anytime.



You want `.gitignore` to be included in your Git datab

Second commit

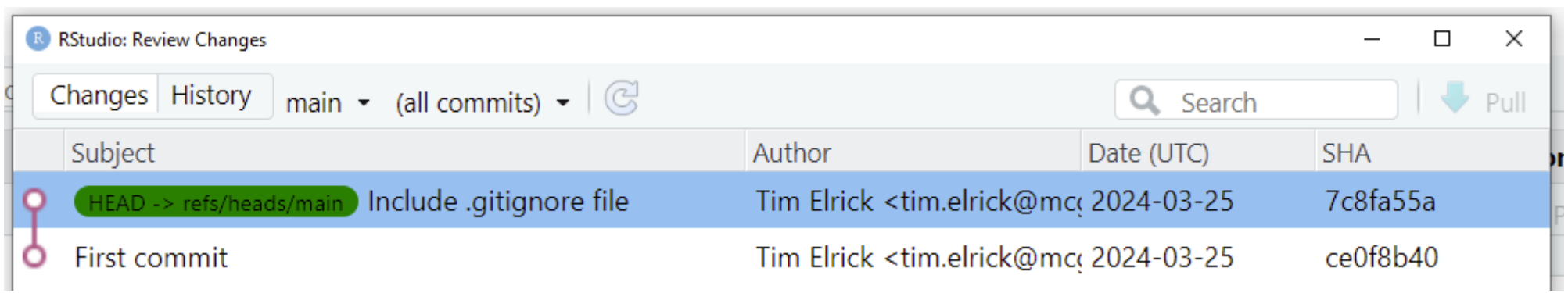
Now add your `.gitignore` file to *staging* and *commit*. Don't forget to enter a description why you committed the file(s):


RStudio: Review Changes
Changes
History
main
Stage
Revert
Ignore
Pull
Push
Staged
Status
Path
.gitignore
Commit message
23 characters
Include .gitignore file
Amend previous commit
Sign commit
Commit
Show
Staged
Unstaged
Context
5 lines
Ignore Whitespace
Unstage All
@@ -0,0 +1,4 @@
Unstage chunk
1 .Rproj.user
2 .Rhistory
3 .RData
4 .Ruserdata

Understanding Git change history

Let's look at the *Git history* again, either by

- typing `git log` into the **terminal**
- or by using the *History*  in the `Git` tab in **RStudio**



Let's make changes

In our ***test2*** project, let's change the code in *more_code.R*:

```
# ds <- data.frame(name = letters[11:20],
#                  number = 11:20)
# write.csv(ds, "more_data.csv")

sum(ds$number)
```

Then save the RScript.

Work with a new file version

In the `Git` tab you will see the modified file showing up.

Now, you can - stage it - commit it to the Git database

Or undo your changes…

Undo changes

To go back to an older version of a file, it depends on whether

1. it has only been modified
   a. in the **Git tab**, right-click on the file and choose `Revert...`
   b. in the **terminal** type `git restore yourFileName`
2. it has been staged
   a. in the **Git tab**, right-click on the file and choose `Revert...`
   b. in the **terminal** type `git restore --staged yourFileName`

Undo changes

3. it has been committed
   a. in the **terminal** type `git checkout yourCommitHash yourFileNameOptional`

Note, it can get complicated when you work with a previous commit. We will cover more in “Version control with Git in R”.

Add Git to your existing R project

Either

- go to the menu `Tools > Project options > Git/SVN` and choose Git
- or use `usethis::use_git()`

If the `Git` tab doesn't show, you need to restart *RStudio*.

Work with GitHub - RProject first

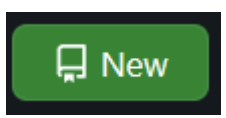
To upload your Git database to GitHub, either use

- in the **R console**: `usethis::use_github()`
- in the `Git` tab click on  , then click `Add Remote` and enter *https://github.com/yourGitHubName/yourProjectName.git*
- in the **terminal**:
  - `git remote add origin https://github.com/yourGitHubName/ yourProjectName.git`
  - `git push -u origin main`

Note: You only do this once, then your Git is connected to GitHub!

Work with GitHub - GitHub first

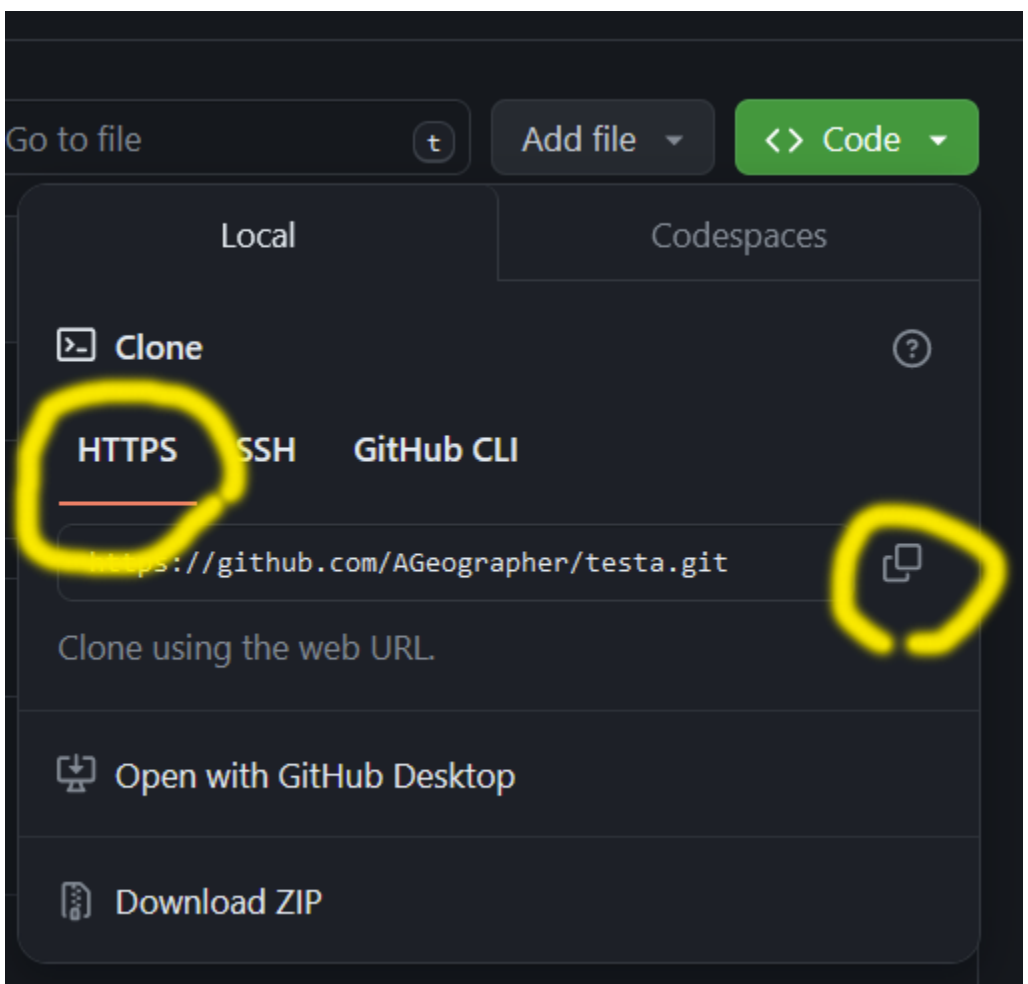
Alternatively, you can set up your project first on **GitHub**:

- in your **browser** navigate to `https://github.com/yourGitHubName?tab=repositories`
- click on 

and
  - enter a *project name*
  - decide if your project should be *private* or *public*
  - add a *README file* and *.gitignore* for *R*

Work with GitHub - GitHub first

- after the repo is created click on <> Code and then, copy the HTTPS URL



Work with GitHub - GitHub first

- then in **RStudio** you
  - either type `usethis::create_from_github('URLtoRepo', 'path/where/you/want/your/local/repo/ to/be/created')`
  - or you go to menu `File > New Project > Version Control > Git` and paste your repo URL

Up- and download files to/from GitHub

To *upload* files from your computer we need to `push` them GitHub.

To *download* files from GitHub to your computer we need to `pull` them.

- in **RStudio** on the `Git` tab you use 

- in the **terminal** you type `git push` or `git pull`

Use someone else's GitHub repository

- in your **browser** go to https://github.com/AGeographer/mock_project
- copy the HTTPS URL and change back to **RStudio** and use
  - either `usethis::create_from_github()`
  - or `File > New Project > Version Control > Git`
  - or in the **terminal** type `git clone yourRepo.git`

Resources

- Posit developer Jenny Bryan's https://happygitwithr.com
- Aengus Bridgman, Poli Sci, McGill GitHub bootcamp: https://abridgman.ca/github_bootcamp
- Alex Douglas, et al. 2024 An Introduction to R, chapter on version control: https://intro2r.com/github_r.html