

USING GIT & GITHUB VIA RSTUDIO

Marco Wähner & Johannes Breuer
Team Research Data & Methods



CENTER FOR
ADVANCED
INTERNET STUDIES

TODAYS OBJECTIVES

- Learning how to...
 - interact with Git & GitHub
 - create new version-controlled projects
 - sync local changes with GitHub
 - collaborate with others
- ... and we do all of this in/via RStudio

DISCLAIMER

- Large parts of the things we will present/discuss today are based on parts of the following two workshops:
 - Frederik Aust/Johannes Breuer: Reproducible research practices for psychologists see <https://github.com/crsh/reproducible-research-practices-workshop>
 - Johannes Breuer/Bernd Weiß/Arnim Bleier: Tools and Workflows for Reproducible Research in the Quantitative Social Sciences see <https://github.com/jobreu/reproducible-research-geis-2022>

PREPARATION

- You should already have ...
 - Git installed
 - Created a GitHub account
 - Checked that Git is connected to RStudio
 - Created a PAT (Personal Access Token) via the *GitHub* web interface
 - And checked that you can connect to GitHub (via Rstudio)

Did everything work for you?

INTERACTING WITH GIT

- There are different ways in which you can interact with Git:
 - Via a command line tool (e.g., Git Bash on Windows) – more on that later...
 - Through a graphical user interface (GUI), git-gui (simple option that comes with Git Bash), [Git Kraken](#) or [GitHub Desktop](#) (both offer quite a large range of functionalities)¹
- **Via an integrated development environment (IDE), such as Rstudio²**

¹ For an overview of different Git clients with a GUI, see: <https://git-scm.com/downloads/guis>

¹ Another popular IDE option is [VS Code](#) by Microsoft which also easily [integrates with Git](#), [GitHub](#), and also [R](#)



TYPICAL WORKFLOWS

- In your everyday work, you quite likely need different workflows, depending on the temporal order in which things are created or set up: your local project/files, version control with Git, and the remote GitHub repository.
 - New project, GitHub first
 - Existing project, GitHub first
 - Existing project, GitHub last


TYPICAL WORKFLOWS

- In this session, we will focus on the *New project, GitHub first* approach (which should be preferred as it is easy to adopt)
- GitHub first, then RStudio
 - Copy the project from GitHub to your local machine
 - It sets up the local Git repository for immediate pulling and pushing
 - Note: Under the hood, we are running the *git clone* command

CREATE A REPOSITORY ON GITHUB

- Let's start with creating a new repository
 - Log in to your GitHub account
 - Click the green  New Button or 
 - Remember: A repository (or short repo) is the place where you store your files and keep track of all changes

CREATE A REPOSITORY ON GITHUB

- Main steps for initializing a repository
 - Choose a name: descriptive but short (no whitespace, but letters, digits, -, ., or _ are allowed)
 - Write a brief description (one line/sentence)
 - Choose whether your repo is public or private
 - Initialize the repo with a README file
 - Optional: Choose a license (GitHub license options focus on software)
 - Click on 

CREATE A REPOSITORY ON GITHUB


- Choose a name for your repository and write a short but meaningful description

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner *

 marcohhu ▾

Repository name *

/ ws_git_and_rstudio ✓

Great repository names are short and memorable. Need inspiration? How about [jubilant-octo-chainsaw?](#)



Description (optional)

A test repo for the Git, GitHub and RStudio Workshop at CAIS



RESEARCH
FOR THE
DIGITAL AGE

CREATE A REPOSITORY ON 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:

Skip this step if you're importing an existing repository.

- ☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore


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

.gitignore template: None ▼

Choose a license

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

License: None ▼

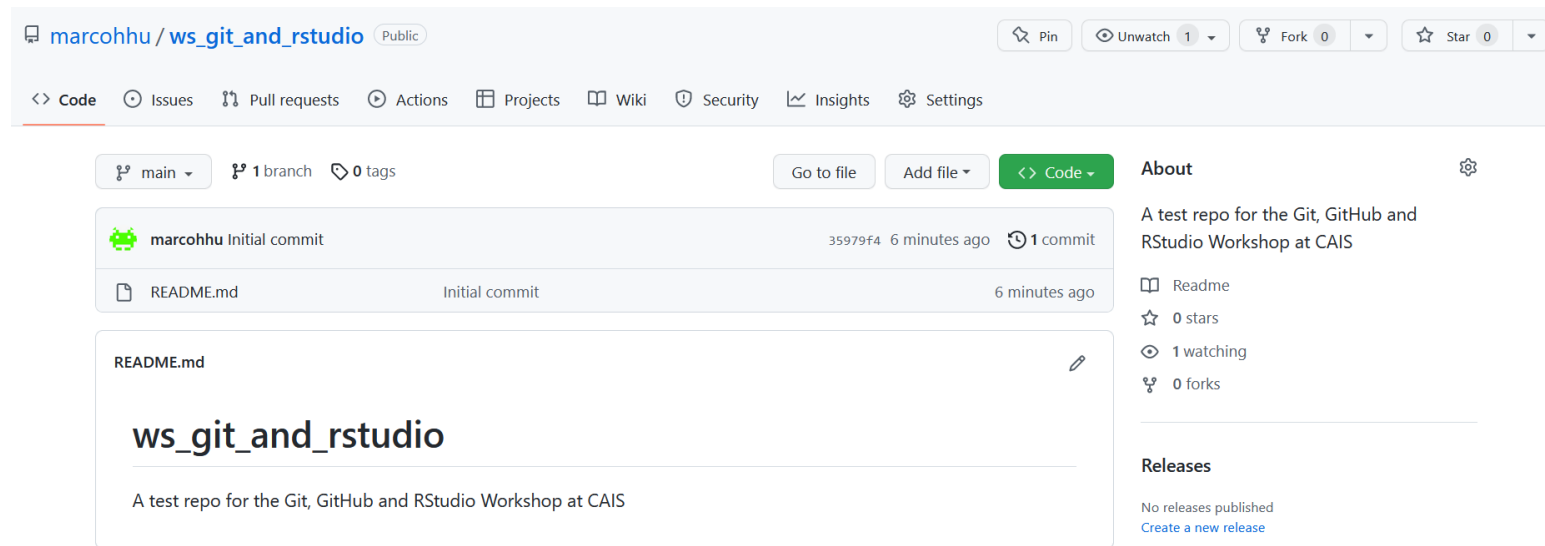
This will set  `main` as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

CREATE A REPOSITORY ON GITHUB

- We created a repository on GitHub
 - If you selected „Public“ anyone on the internet can see your repo (and copy it). Since you do not want to share your data (or other files) on GitHub we can tell Git later which files to ignore (we will come back to this later).

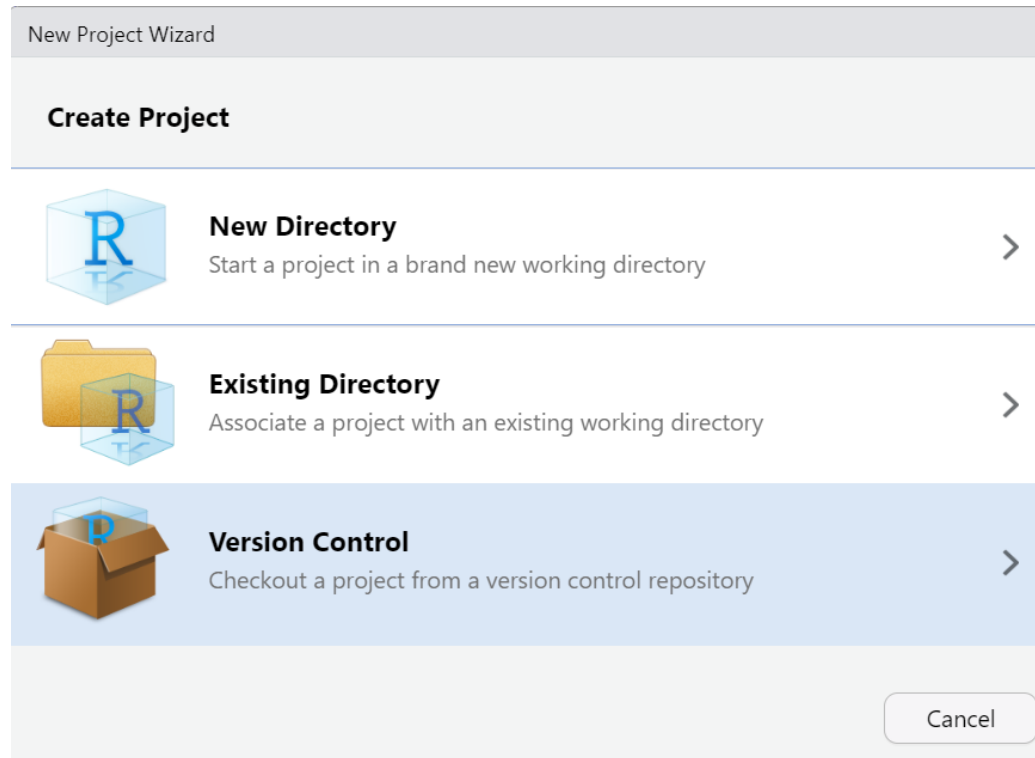


Git, GitHub and RStudio

- In order to get the best out of the combination of Git, GitHub and RStudio, it is recommendable to adopt a "project-oriented workflow"
 - RStudio projects are associated with .Rproj files that contain some specific settings for the project. If you double-click on a .Rproj file, this opens a new instance of RStudio with the working directory and file browser set to the location of that file
 - Even if you do not work with Git, you should always work with R-Projects (check the respective chapter in [What they Forgot to Teach you about R](#))

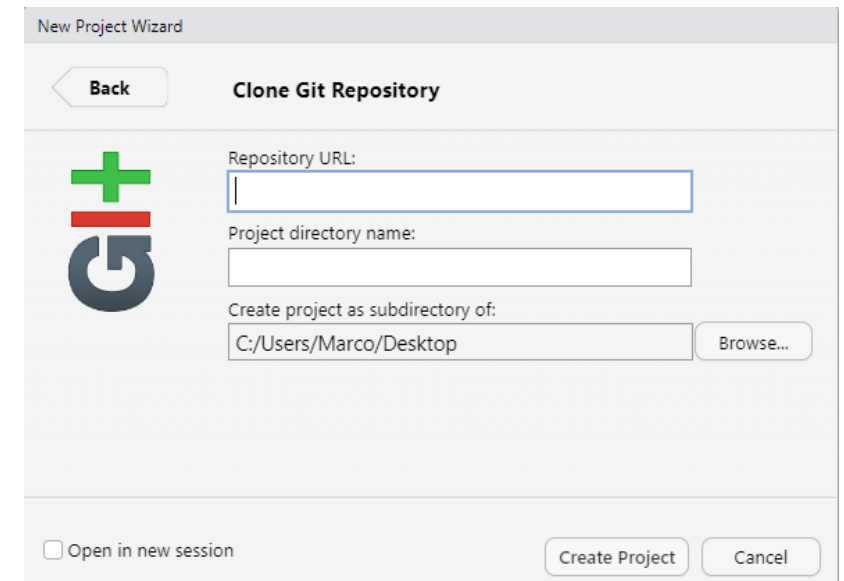
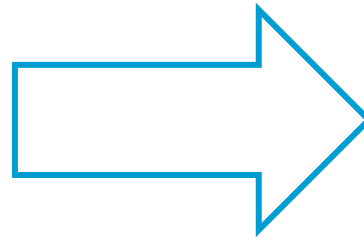
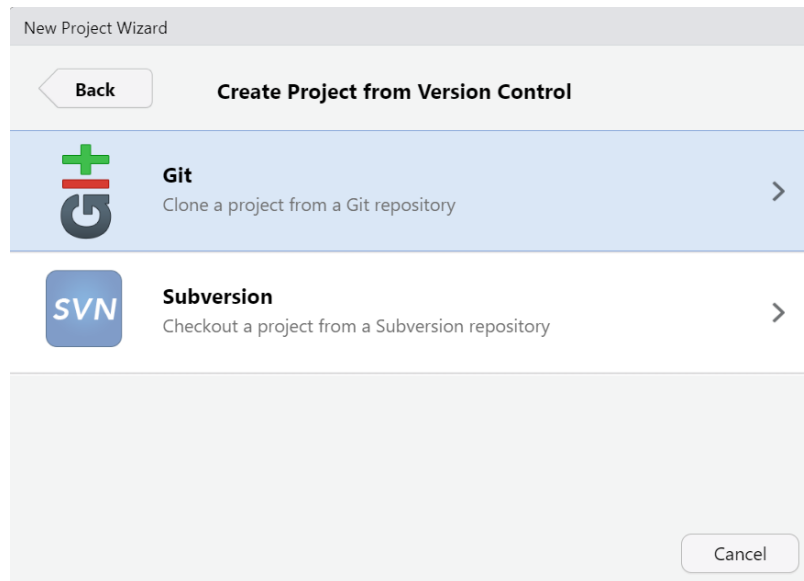
NEW PROJECT CONNECTED TO EXISTING REMOTE REPO

- You can create a new version-controlled project that is connected to a remote repository that already exists on GitHub via *File -> New Project -> Version Control* in the RStudio menu.



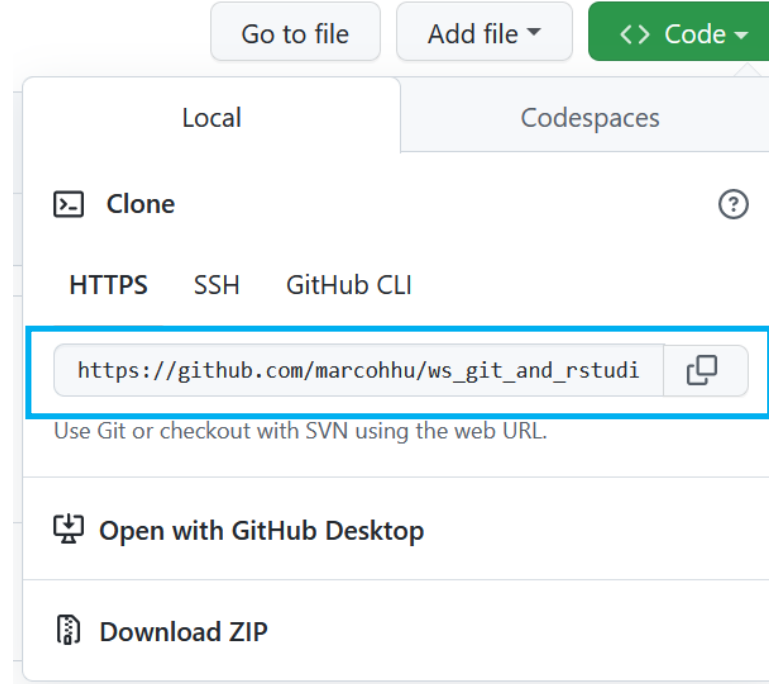
NEW PROJECT CONNECTED TO EXISTING REMOTE REPO

- In the menu that opens after that, select Git and enter the URL of the remote repository



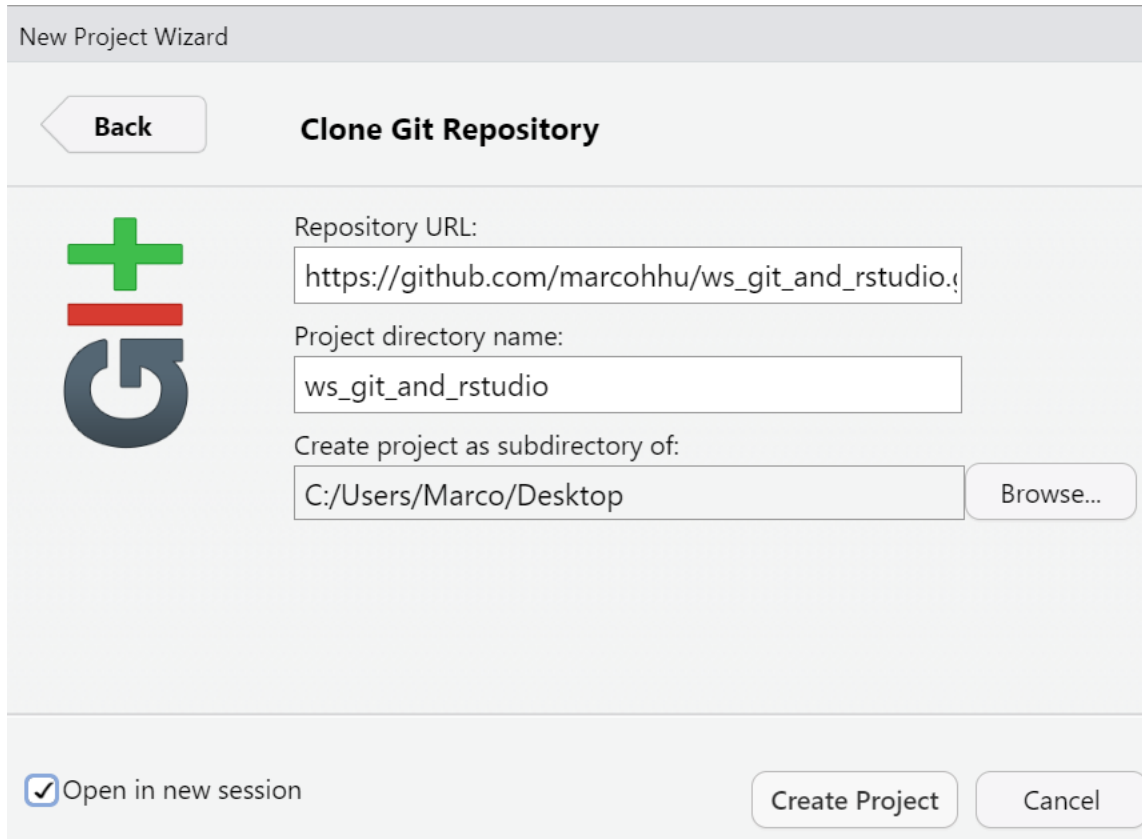
NEW PROJECT CONNECTED TO EXISTING REMOTE REPO

- Go back to your repo and copy the HTTPS link
 - Which URL you should enter depends on the authentication method you use. If you use HTTPS (with a PAT), you can simply copy the URL from the address bar of your browser.



NEW PROJECT CONNECTED TO EXISTING REMOTE REPO


- Next, give the local repository a name, and tell RStudio where it should be stored. It usually makes sense to check "Open in new session".



The screenshot shows the 'New Project Wizard' dialog box in RStudio, specifically the 'Clone Git Repository' step. The dialog has a title bar 'New Project Wizard' and a 'Back' button. On the left is a large Git logo. The main area contains three input fields: 'Repository URL:' with the value 'https://github.com/marcohhhu/ws_git_and_rstudio.', 'Project directory name:' with the value 'ws_git_and_rstudio', and 'Create project as subdirectory of:' with the value 'C:/Users/Marco/Desktop'. A 'Browse...' button is next to the last field. At the bottom, there is a checkbox labeled 'Open in new session' which is checked, and two buttons: 'Create Project' and 'Cancel'.

New Project Wizard

Back Clone Git Repository



Repository URL:
https://github.com/marcohhhu/ws_git_and_rstudio.

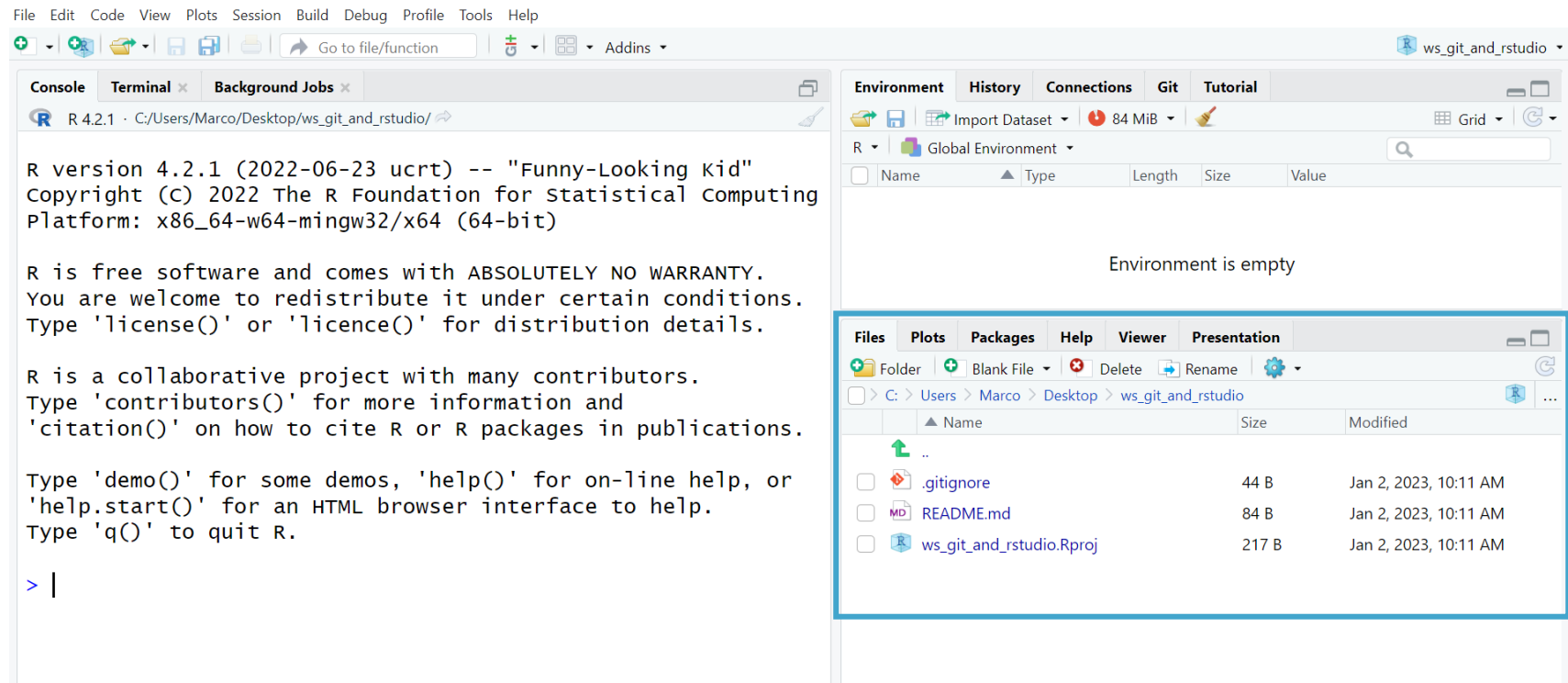
Project directory name:
ws_git_and_rstudio

Create project as subdirectory of:
C:/Users/Marco/Desktop Browse...

☒ Open in new session Create Project Cancel

NEW PROJECT CONNECTED TO EXISTING REMOTE REPO

- Congratulations! You are connected to Git and GitHub via RStudio. Now you can open and edit files locally which you have created via GitHub (see the bottom right window and the "Files" section)



README – EVERYTHING YOU NEED TO KNOW

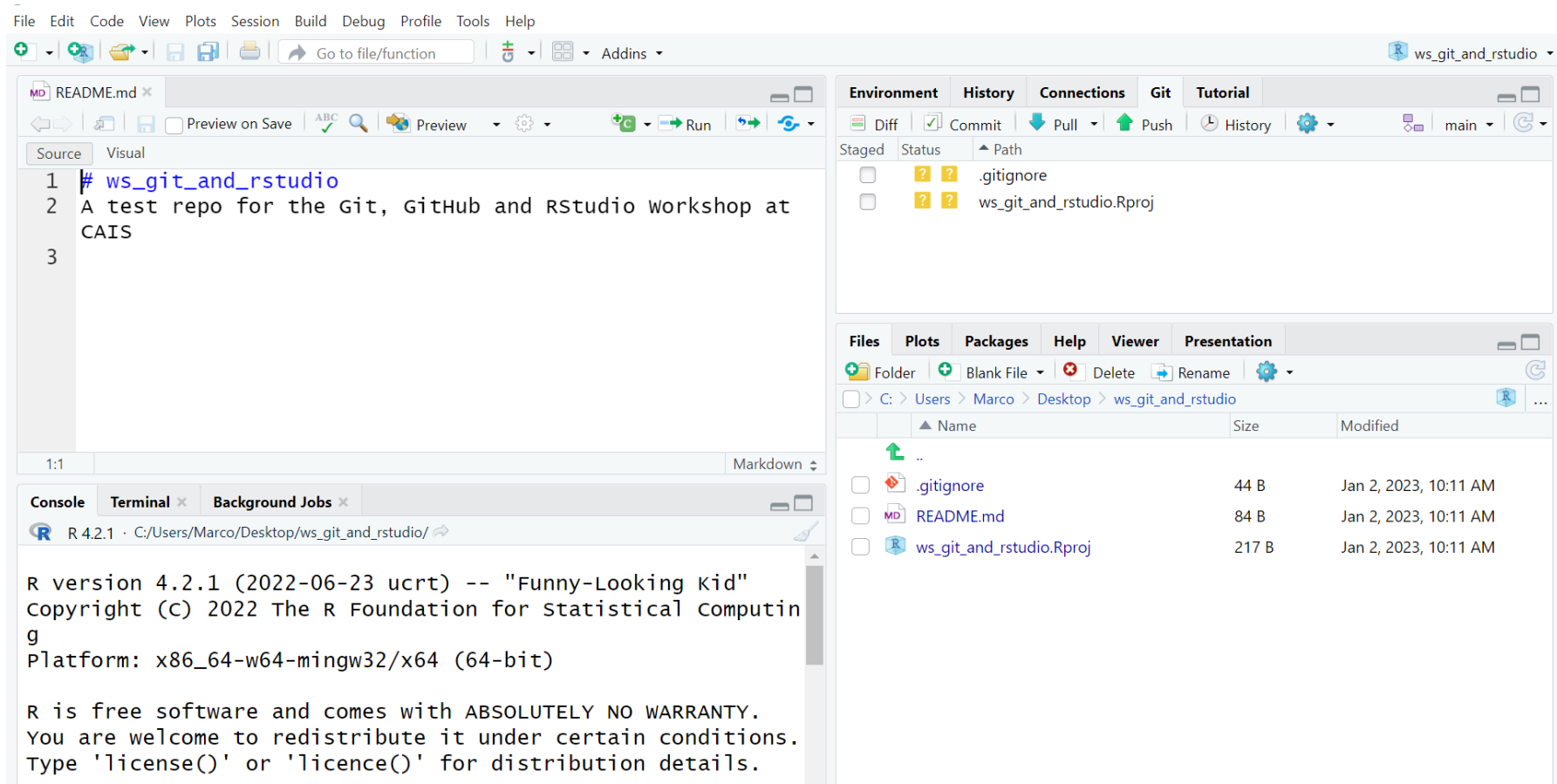
- The README file is a text file that contains information about the software, project, code, etc.
- It is good practice to initialize your repo with a README file.
- You can edit the file using [Markdown syntax](#), e.g.,
 - # Headline 1
 - ## Headline 2
 - ### Headline 3

TEST DRIVE: MODIFYING & CREATING FILES WITHIN THE PROJECT

- Once you have successfully created the project, you can start editing files or creating new ones. When you have modified existing files and/or created new ones and saved the changes, these will be displayed in the Git tab in RStudio and their status will be indicated as [M]odified, [U]ntracked, or [A]dded.

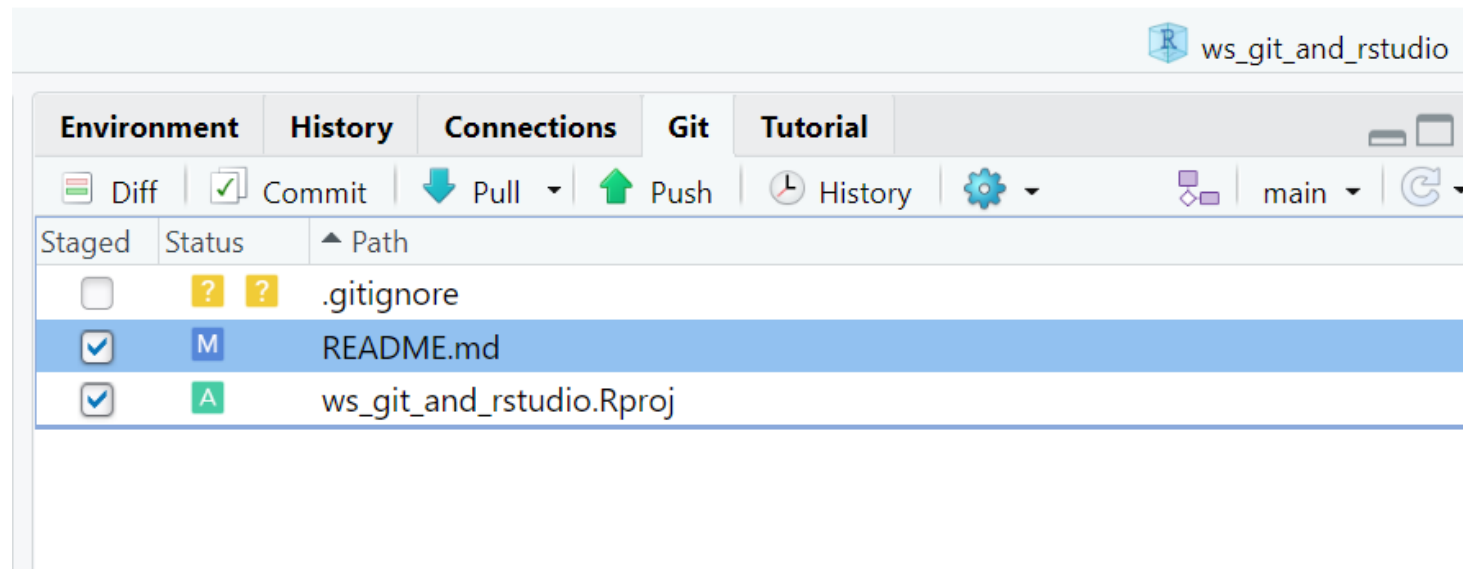
COMMITTING & PUSHING CHANGES

- You can, e.g., change the README file locally, and commit and push it to your remote repo



STAGING CHANGES

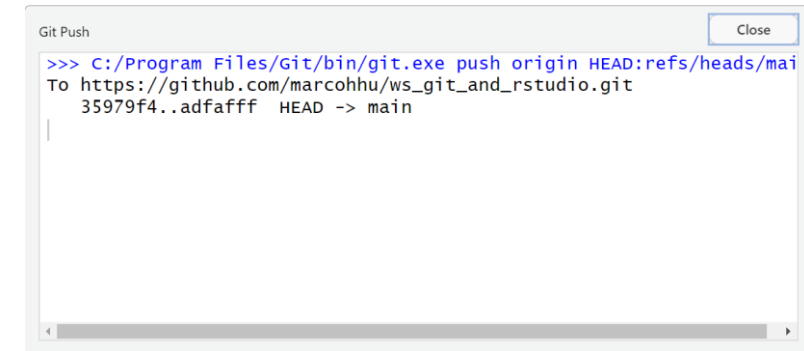
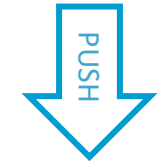
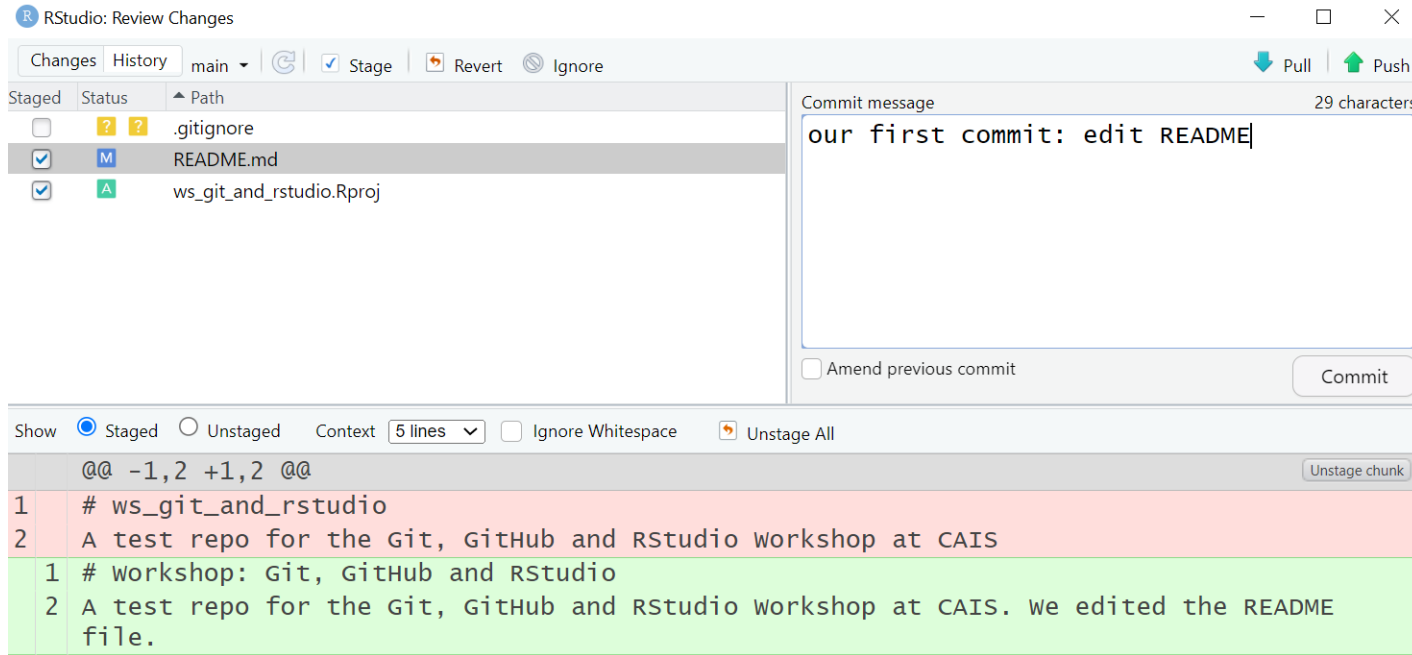
- Once you have reached a point at which you want to commit (and possibly also push) your changes, you can stage them by checking the boxes in the Staged column in the Git tab. This is the RStudio GUI equivalent of the command ``git add``. The status of previously untracked files will then change to added.



COMMITTING & PUSHING CHANGES

- After staging your changes, you can commit them via the Commit button in the Git tab. In the commit menu that opens you should enter a meaningful commit message.
 - Once that is done you can click the Commit button.
 - If you want to, you can also directly push your changes to the remote repository on GitHub via the Push button. You can, of course, also do this at a later point (directly via the Git tab).

COMMITTING & PUSHING CHANGES



COMMITTING & PUSHING CHANGES

- You can also look at the changes via the “Changes” tab

RStudio: Review Changes

Changes History main (all commits) Search Pull

Subject	Author	Date (UTC)	SHA
HEAD -> refs/heads/main our first commit: edit README	marcohu <marco.waehner@uni-duessel>	2023-01-02	adfa88f8
origin/main origin/HEAD Initial commit	Marco Waehner <39967944+marcohu@>	2023-01-02	35979f45

Commits 1-2 of 2

SHA adta88f89258d0094d0/9000b/65533da439/512

Author marcohu <marco.waehner@uni-duesseldorf.de>

Date (UTC) 2023-01-02 09:44

Subject our first commit: edit README

Parent 35979f453ff07353f36ab8c5ca0da0940215ba6d

README.md

ws_git_and_rstudio.Rproj

README.md View file @ adfa88f8

```
@@ -1,2 +1,2 @@
1 # ws_git_and_rstudio
2 A test repo for the Git, GitHub and RStudio workshop at CAIS
1 # workshop: Git, GitHub and RStudio
2 A test repo for the Git, GitHub and RStudio workshop at CAIS. we edited the README
  file.
```

QUICK NOTES ABOUT DATA PROTECTION

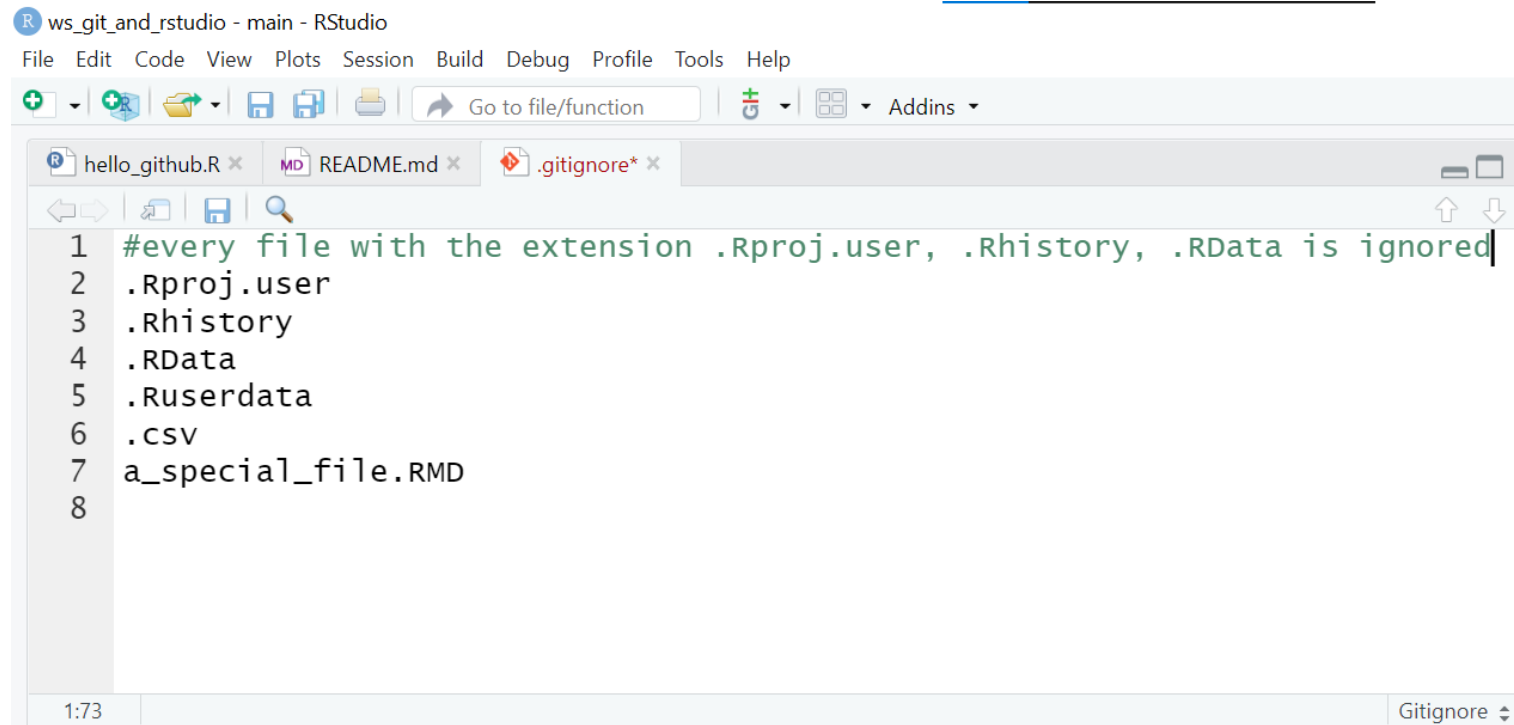
- *IMPORTANT*: Make sure that you do not (accidentally) publish anything that should not be public via *GitHub*!
- For our use cases, this will probably mostly concern a) research data (esp. if it is on the individual level and contains personal information; licenses and usage agreements also have to be taken into account here) and b) sensitive information like passwords, API keys, PAT/SSH keys, etc.

QUICK NOTES ABOUT DATA PROTECTION

- There are a few ways in which you can prevent (or at least substantially lower the risk of) accidentally sharing information (or files) that should not be made public:
 - Add files and folders you do not want to be tracked (and, thus, also not push to GitHub) to the .gitignore file for your project.
 - Keep sensitive information/files in a separate place (i.e., not in the version-controlled project directory).
 - When you create a new GitHub repository, set it to "private" and only set it to "public" once you are sure that it does not contain anything that should not be public.

WHAT IS .gitignore?

- With the .gitignore file you tell Git and GitHub which files should be ignored
 - In the standard .gitignore for R projects that you can select on GitHub (when you create a new repo), .RData files are ignored. You can also extend and customize the file (see below).

A screenshot of the RStudio interface. The title bar shows 'ws_git_and_rstudio - main - RStudio'. The menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar contains icons for file operations and a search bar labeled 'Go to file/function'. The file explorer shows three files: 'hello_github.R', 'README.md', and '.gitignore*'. The editor window displays the content of '.gitignore' with line numbers 1 through 8. The text is as follows:

```
1 #every file with the extension .Rproj.user, .Rhistory, .RData is ignored|
2 .Rproj.user
3 .Rhistory
4 .RData
5 .Ruserdata
6 .csv
7 a_special_file.RMD
8
```

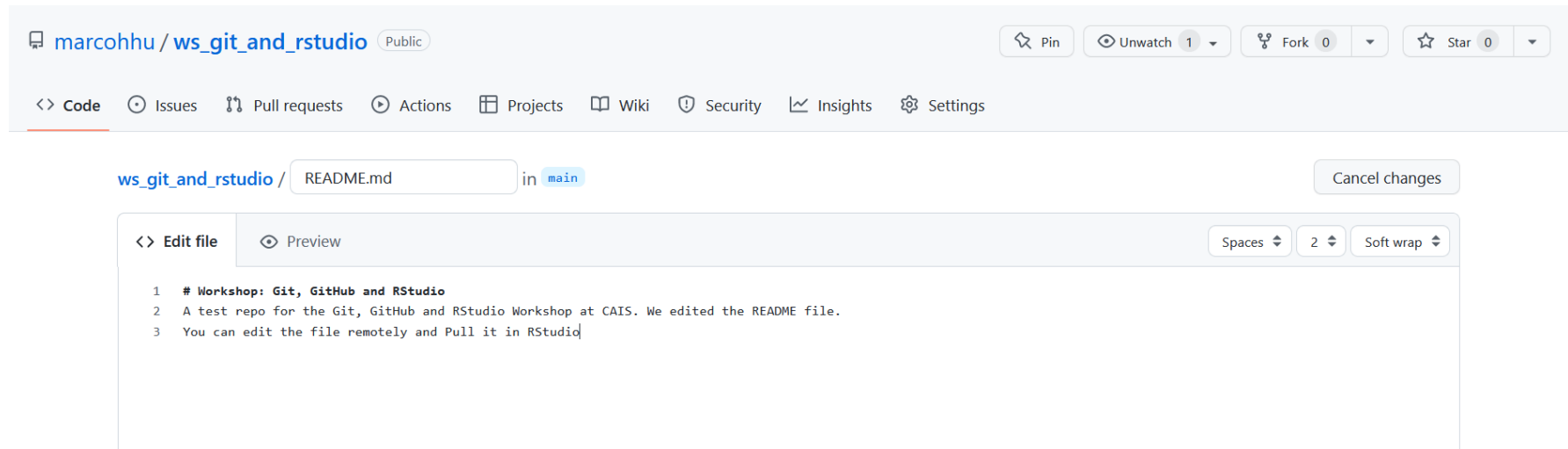
The status bar at the bottom shows '1:73' and 'Gitignore'.

PULLING CHANGES FROM THE REMOTE REPOSITORY

- You can also pull changes from the remote repository via the Pull button in the Git tab.
- As a general workflow recommendation (especially if you're just getting started with Git and GitHub), it is usually advisable to first pull from the remote repository before making (and then staging, committing, and pushing) any local changes. This is even more relevant when you collaborate with others on the same repository (more on that later).

PULLING CHANGES FROM THE REMOTE REPOSITORY

- Edit the file remotely and Pull it in RStudio



EXERCISE

1. Open/create an R script file in your project
 - Write a line of code, e.g., `print("Hello GitHub")`
 - Save the new file
 - Stage, commit, and push the R script to GitHub
2. Change the README file via the GitHub web interface
 - Pull the repo in RStudio to stay up to date

LIMITATIONS OF THE GUI

- While the RStudio GUI can be used for quite a few basic Git operations, it has a set of limitations
 - Not all git commands (and command options) are available via the GUI
 - Behind the scenes, RStudio uses certain defaults for Git operations (e.g., if you click “Pull” in the GUI, it does a “[pull with rebase](#)”)
 - It can become tedious to stage and commit (a) large (number of) changes/files via the RStudio GUI (NB: this can also cause RStudio to crash)

DESTINATION TERMINAL

- If you want to add/commit a lot of files or large files, want more control over the Git commands, or need to use more advanced Git operations, the RStudio GUI is not the best choice. Instead, you should use a command line interface (CLI).

WTF IS A CLI?!?

- **“A command-line interface (CLI) processes commands to a computer program in the form of lines of text. The program which handles the interface is called a command-line interpreter or command-line processor. Operating systems implement a command-line interface in a shell for interactive access to operating system functions or services.”**

Source: https://en.wikipedia.org/wiki/Command-line_interface

WHAT THE HELL IS A SHELL?!?

- “In computing, a **shell** is a computer program which exposes an **operating system's services** to a human user or other program. In general, **operating system shells** use either a **command-line interface (CLI)** or **graphical user interface (GUI)**, depending on a computer's role and particular operation. It is named a shell because it is the outermost layer around the operating system.”

Source: [https://en.wikipedia.org/wiki/Shell_\(computing\)](https://en.wikipedia.org/wiki/Shell_(computing))

PICKING SHELLS

- As with most types of software, there are different shell options
 - Which of those you can use, depends on your operating system (OS)
- On Windows you can typically choose between the [command interpreter cmd](#) and [PowerShell](#)
- On Unix systems like Linux and MacOS, the most popular (and typically also the default) shell is [Bash](#)¹
- If you have Git installed on your Windows machine, you can also use [GitBash](#) (this is also what I use)

¹ These days, you can also use Bash on Windows via the [Windows Subsystem for Linux](#)

WINDOWS CMD

C:\Windows\System32\cmd.exe

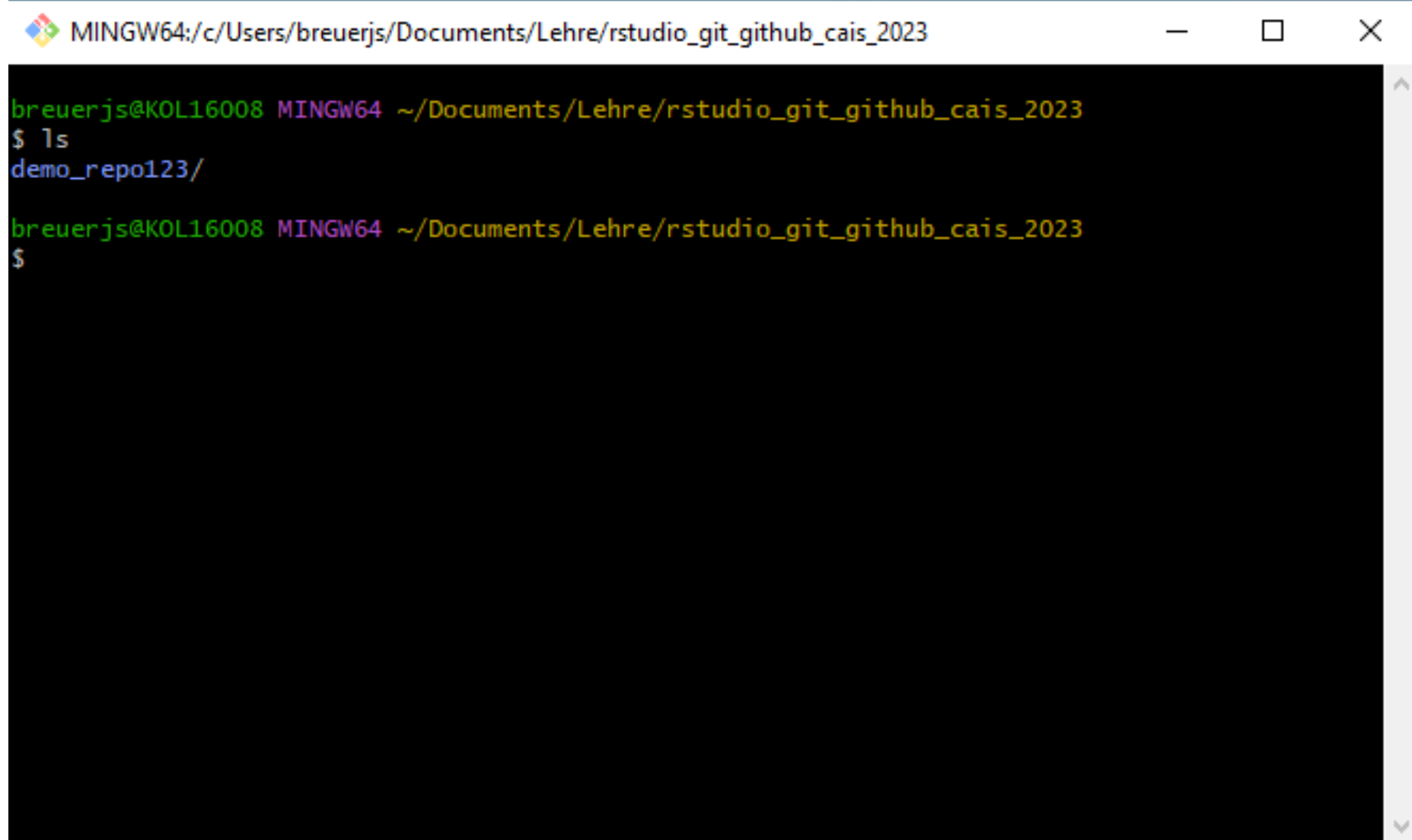
```
C:\Users\breuerjs\Documents\Lehre\rstudio_git_github_cais_2023>dir
Datenträger in Laufwerk C: ist Windows
Volumeserienummer: 9C5A-7929

Verzeichnis von C:\Users\breuerjs\Documents\Lehre\rstudio_git_github_cais_2023

20.12.2022  18:41    <DIR>          .
20.12.2022  18:41    <DIR>          ..
20.12.2022  18:41    <DIR>          demo_repo123
               0 Datei(en),               0 Bytes
               3 Verzeichnis(se), 251.330.052.096 Bytes frei

C:\Users\breuerjs\Documents\Lehre\rstudio_git_github_cais_2023>_
```

GIT BASH



```
MINGW64:/c/Users/breuerjs/Documents/Lehre/rstudio_git_github_cais_2023  
breuerjs@KOL16008 MINGW64 ~/Documents/Lehre/rstudio_git_github_cais_2023  
$ ls  
demo_repo123/  
breuerjs@KOL16008 MINGW64 ~/Documents/Lehre/rstudio_git_github_cais_2023  
$
```

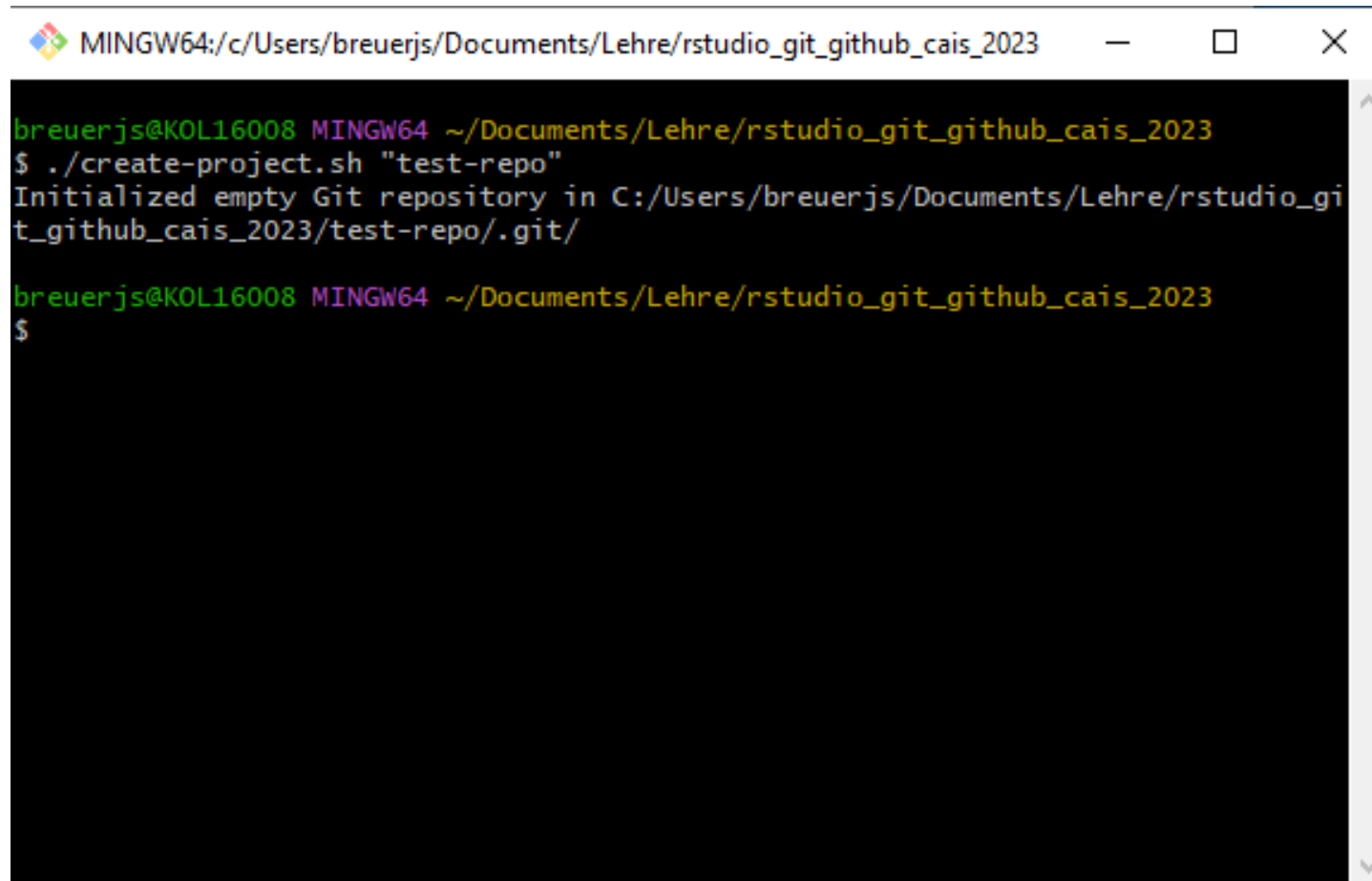
WHY A CLI?

- Fast & efficient way of interacting with your computer
- If you work with remote servers, this usually happens via a CLI
- Using a CLI makes you look like a 1337 H4X0R ;-)
- You can also use CLI tools for automation, e.g., via shell scripts

BRIEF EXCURSUS: SHELL SCRIPTS

- Shell scripts typically have the file extension `.sh`
- As the name suggests, you can execute shell scripts via a shell
- We have created a small demo script for you that does the following things when you execute it:
 - Create a new directory (folder) with a name you need to specify
 - Create three empty subfolders therein: `data`, `src`, `output`
 - Create two empty files: `.gitignore` & `README`
 - Initialize Git in the directory
- NB: You can edit `.sh` files with any text editor (such as [Notepad++](#) for Windows) as well as in RStudio (and other IDEs)

BRIEF EXCURSUS: SHELL SCRIPTS



A screenshot of a Windows command prompt window. The title bar shows the path 'MINGW64:/c/Users/breuerjs/Documents/Lehre/rstudio_git_github_cais_2023'. The prompt is 'breuerjs@KOL16008 MINGW64 ~/Documents/Lehre/rstudio_git_github_cais_2023'. The user has entered the command './create-project.sh "test-repo"'. The output is 'Initialized empty Git repository in C:/Users/breuerjs/Documents/Lehre/rstudio_git_github_cais_2023/test-repo/.git/'. The prompt is now '\$'.

```
breuerjs@KOL16008 MINGW64 ~/Documents/Lehre/rstudio_git_github_cais_2023
$ ./create-project.sh "test-repo"
Initialized empty Git repository in C:/Users/breuerjs/Documents/Lehre/rstudio_git_github_cais_2023/test-repo/.git/
breuerjs@KOL16008 MINGW64 ~/Documents/Lehre/rstudio_git_github_cais_2023
$
```

BRIEF EXCURSUS: SHELL COMMANDS

Description	Win	Linux, macOS (Bash)
Copy files, folders	copy	cp
Move files, folders	move	mv
List folder content	dir	ls
Create new folder	mkdir	mkdir
Change current folder	cd	cd
Show current path	echo %cd%	pwd
Delete file(s)	del	rm
Delete folder(s)	rmdir	rm

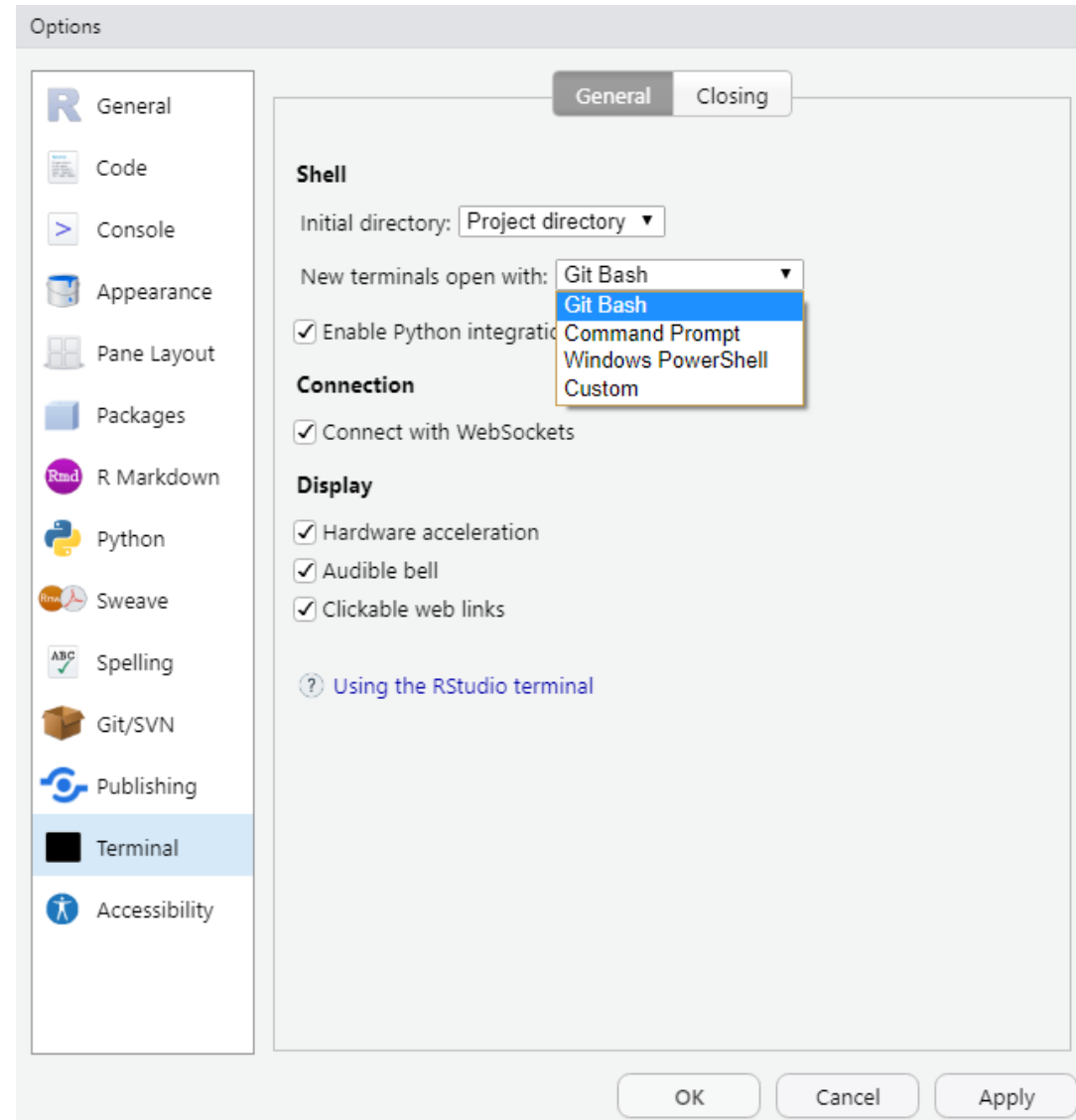
Note: Most commands also have arguments. Example:
ls -la

[Bash command cheatsheet](#)
[Windows cmd cheatsheet](#)

DESTINATION TERMINAL

- Lucky for us, if you need a CLI (e.g., to interact with Git), you don't need to leave RStudio: As of version 1.3.1056-1, RStudio provides a Terminal tab in the console pane
- Through this, RStudio provides access to a shell of your choice (if you have multiple options on your system)
- If you have properly installed Git, you can use this to execute the full range of Git commands

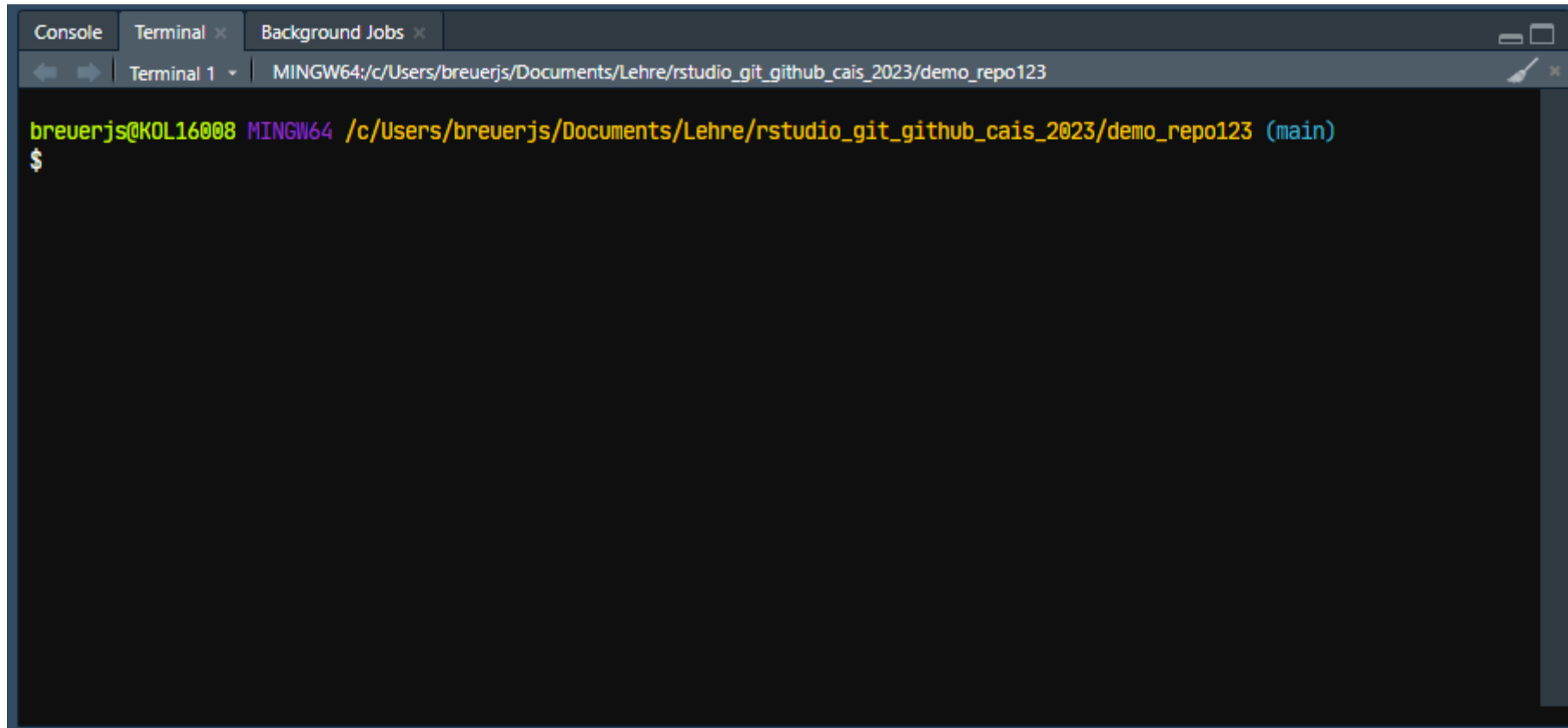
PICKING SHELLS IN RSTUDIO



PICKING SHELLS IN RSTUDIO

- If you use Windows and have installed Git for Windows, you should use Git Bash as the shell that is run in the RStudio terminal
- On MacOS and Linux, you should be able to simply use the native Bash in RStudio

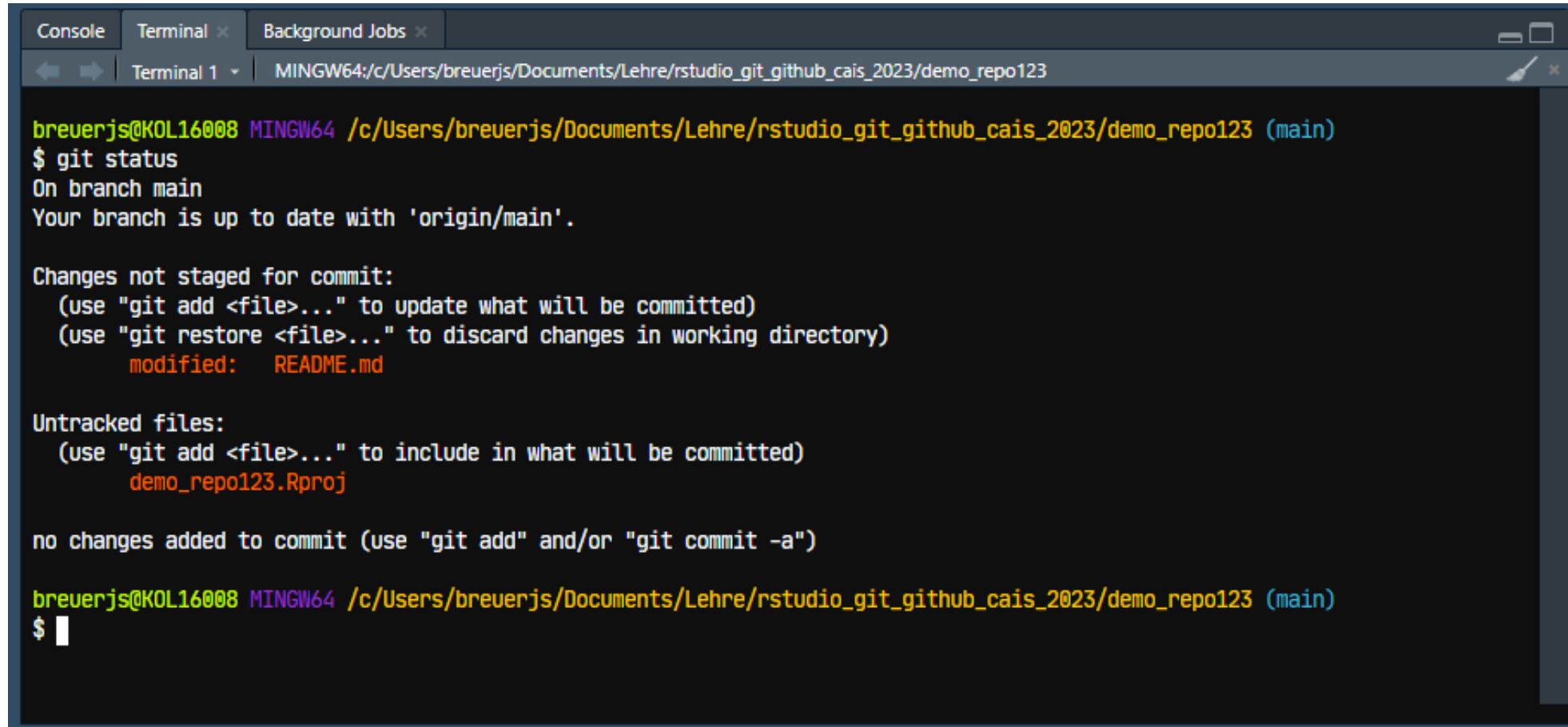
TERMINAL TAB IN RSTUDIO



The image shows a screenshot of the RStudio interface with the 'Terminal' tab selected. The terminal window displays the following text:

```
breuerjs@KOL16008 MINGW64 /c/Users/breuerjs/Documents/Lehre/rstudio_git_github_cais_2023/demo_repo123 (main)
$
```

USING THE TERMINAL IN RSTUDIO



The screenshot shows the RStudio interface with the Terminal pane active. The terminal window title is 'Terminal 1' and the path is 'MINGW64:/c/Users/breuerjs/Documents/Lehre/rstudio_git_github_cais_2023/demo_repo123'. The command executed is 'git status'. The output shows the current branch is 'main' and it is up to date with 'origin/main'. It lists changes not staged for commit: 'README.md' (modified). It also lists untracked files: 'demo_repo123.Rproj'. The prompt is '\$'.

```
breuerjs@KOL16008 MINGW64 /c/Users/breuerjs/Documents/Lehre/rstudio_git_github_cais_2023/demo_repo123 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        demo_repo123.Rproj

no changes added to commit (use "git add" and/or "git commit -a")

breuerjs@KOL16008 MINGW64 /c/Users/breuerjs/Documents/Lehre/rstudio_git_github_cais_2023/demo_repo123 (main)
$
```

USING THE TERMINAL IN RSTUDIO

- You can use the Terminal in RStudio to run all available Git commands as well as command options
- For example, to stage and commit all changes, you could run the following command in the Terminal:

git add -A && git commit -m "Your Message"

USING THE TERMINAL IN RSTUDIO

- For some more information on choosing and using the shell in RStudio, you can check out the [chapter on this in *Happy Git and GitHub for the useR*](#) or the RStudio How To Article on [Using the RStudio Terminal in the RStudio IDE](#)

OTHER OPTIONS FOR USING GIT WITH R

- In addition to using the RStudio GUI or the Terminal, there are also R packages for Git operations
 - [usethis](#) can, e.g., be used to initialize a Git repository or for managing credentials
 - [gert](#) is a simple Git client for R that can be used to perform basic Git commands, such as staging, adding, and committing files, or creating, merging, and deleting branches
 - [ghstudio](#) is a novel in-development package that provides "experimental tools to use git/github with RStudio, e.g see issues and diffs in the viewer"

COLLABORATION

- You can also use Git & GitHub (via RStudio) for collaboration
- There are different ways in which you can collaborate:
 - **Inviting or becoming a collaborator to/for a project/repository**
 - You can think of this as adding somebody with editing rights to a Google Doc
 - Initiating (or receiving) what is called a “pull request”
 - analogy: Suggesting mode in Google Docs (or Word)
 - Opening or addressing an “issue” (on GitHub)
 - analogy: adding a comment in a Google Doc (or Word document)
- Today, we will focus on the first option but at least briefly also address the other two

ADDING COLLABORATORS ON GITHUB

- GitHub provides a number of functionalities for collaboration (we will discuss some of those in the following)
- Adding collaborators works only for GitHub repositories that you own (these can be private or public)

ADDING COLLABORATORS ON GITHUB

The screenshot shows the GitHub repository settings page for 'demo_repo123'. The 'Settings' tab is selected and highlighted with a red circle and a red '1'. In the left sidebar, the 'Access' section is expanded, and the 'Collaborators' option is highlighted with a red circle and a red '2'. The main content area shows 'Who has access' with two cards: 'PRIVATE REPOSITORY' and 'DIRECT ACCESS'. Below this, the 'Manage access' section is visible, showing a message 'You haven't invited any collaborators yet' and a green 'Add people' button, which is highlighted with a red circle and a red '3'.

jobreu / demo_repo123 Private

Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights **Settings** 1

General

Access 2

Collaborators

Code and automation

Branches

Tags

Actions

Webhooks

Environments

Codespaces

Pages

Security

Code security and analysis

Deploy keys

Secrets

Who has access

PRIVATE REPOSITORY

Only those with access to this repository can view it.

Manage

DIRECT ACCESS

0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access

You haven't invited any collaborators yet

Add people 3

COLLABORATOR WORKFLOWS

- If you are a collaborator, you can...
 - a) Commit & push changes directly to the main branch
 - b) Suggest changes via a so-called “pull request”:
 - Create a new branch
 - Make changes
 - Suggest a merge via a pull request
- Sticking with the previous analogy, you can think of option a) as directly editing a document and option b) as using the suggesting mode

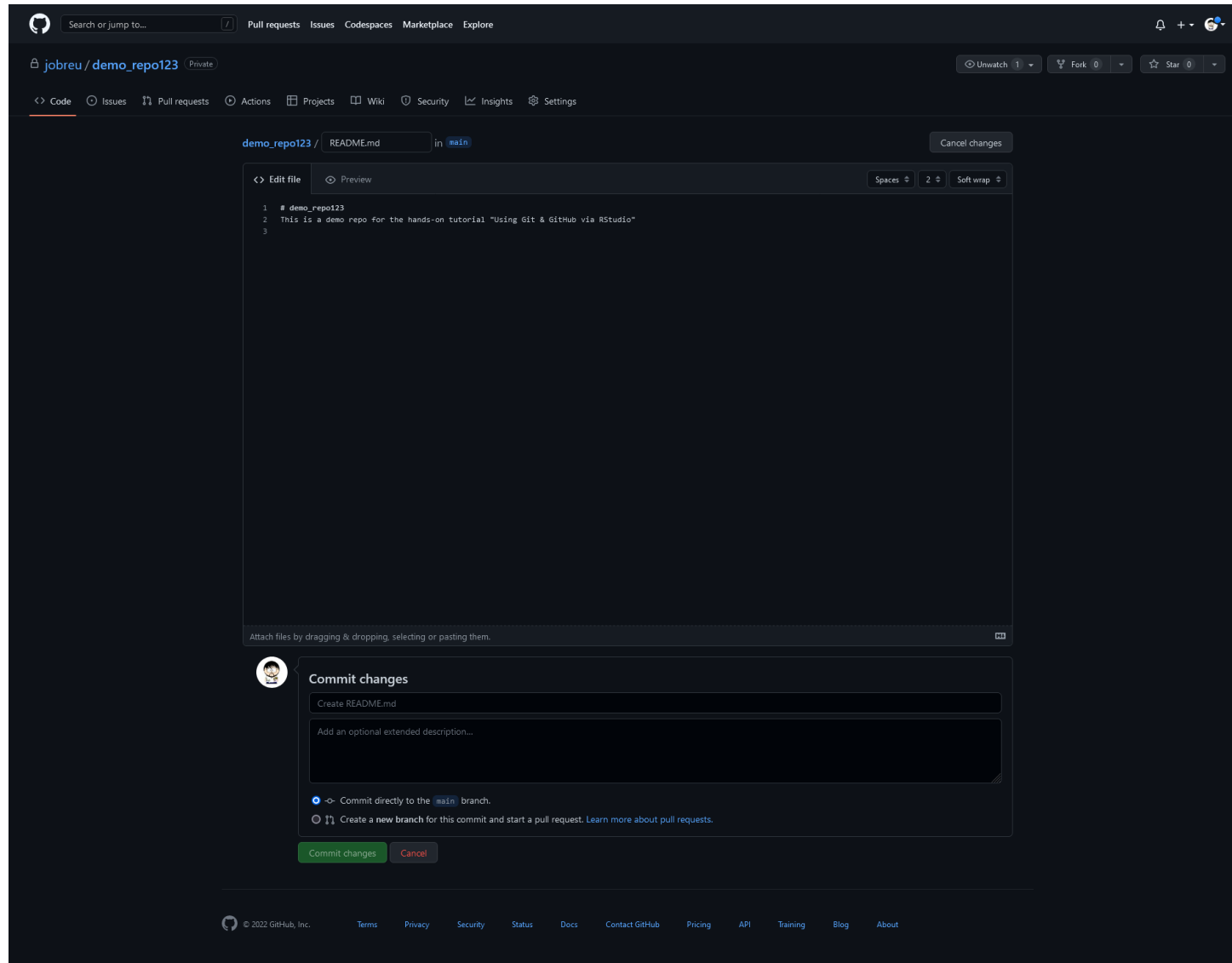
MAKING CHANGES

- As you have seen before, you can edit files...
 - locally (e.g., in RStudio)
 - this is especially advisable, if you want to work on multiple files (and/or make large changes)
 - online via the GitHub web interface
 - this works best for small changes to single files

EDITING FILES ON GITHUB

The screenshot displays the GitHub web interface for a repository named `demo_repo123` by user `jobreu`. The repository is marked as `Private`. The top navigation bar includes links for `Pull requests`, `Issues`, `Codespaces`, `Marketplace`, and `Explore`. Below this, a secondary navigation bar shows `Code`, `Issues`, `Pull requests`, `Actions`, `Projects`, `Wiki`, `Security`, `Insights`, and `Settings`. The main content area shows the `main` branch with the file `demo_repo123 / README.md`. A header for the file shows the user `jobreu` editing the file, the latest commit `a0b6219` from 29 minutes ago, and a `History` link. Below this, it indicates `1 contributor`. The file details show `2 lines (2 sloc)` and `94 Bytes`. A toolbar at the top of the file content area includes buttons for `<>`, `File icon`, `Raw`, `Blame`, `Edit this file` (highlighted with a red circle), and `Trash icon`. The file content itself shows the text `demo_repo123` followed by `This is a demo repo for the hands-on tutorial "Using Git & GitHub via RStudio"`. The footer of the page contains the GitHub logo, copyright notice `© 2022 GitHub, Inc.`, and various links: `Terms`, `Privacy`, `Security`, `Status`, `Docs`, `Contact GitHub`, `Pricing`, `API`, `Training`, `Blog`, and `About`.

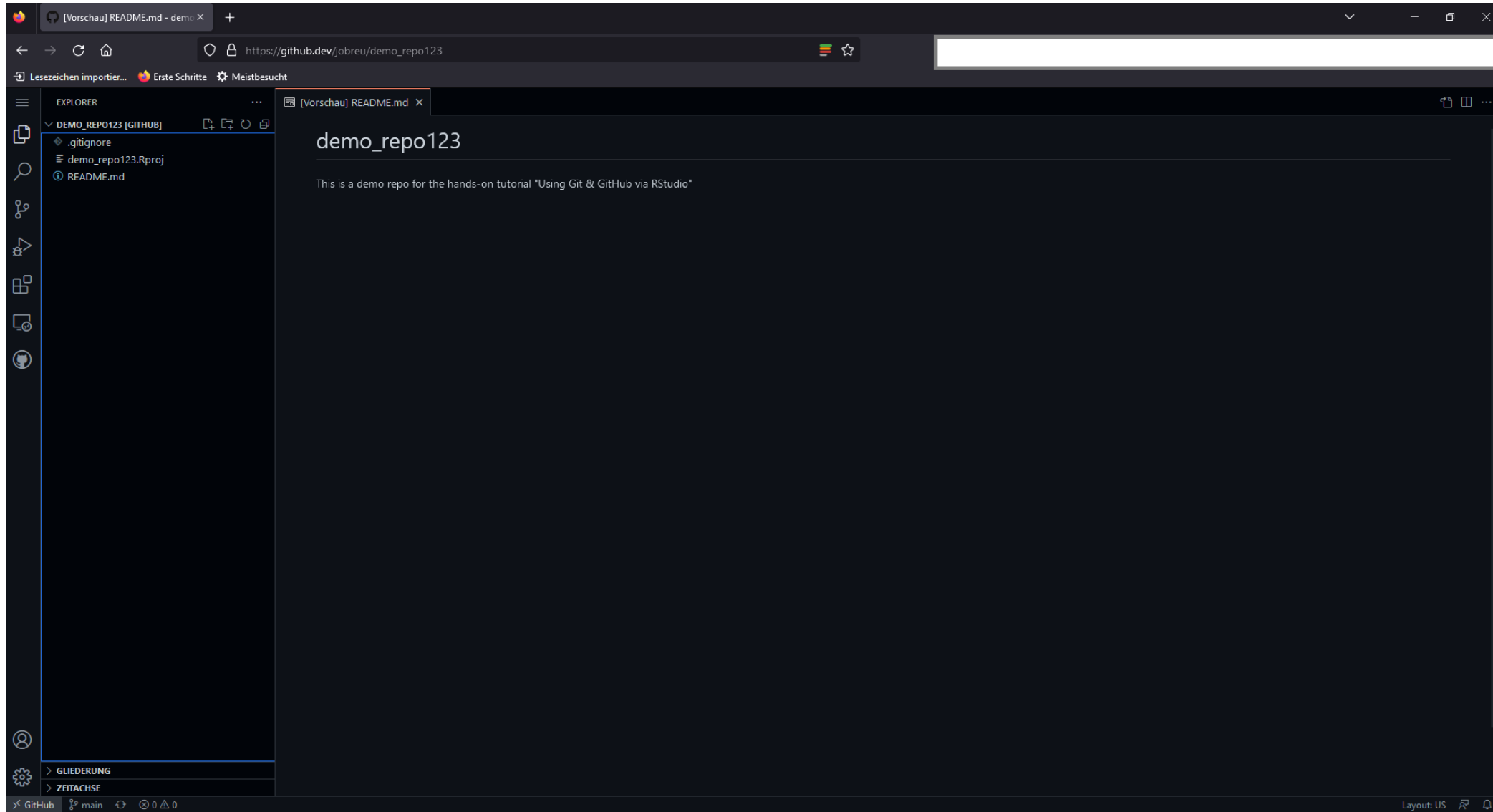
EDITING FILES ON GITHUB



SIDE NOTE: VSCODE ON GITHUB

- You can open an instance of VSCode via the GitHub web interface by simply pressing . when you are on the website of a repository you own or are a collaborator on (without needing to install anything on your computer)
- This gives you the power of a full IDE directly in your browser

SIDE NOTE: VSCODE ON GITHUB



COLLABORATOR WORKFLOW

- If you are a collaborator or have collaborators for your repo, the ideal workflow is as follows:
 1. Pull changes from the remote repository
 2. Edit files, stage, & commit changes on your computer
 3. Push new commit(s) to the remote repository
- NB: If you work on the same file(s) in parallel, this may cause so-called “merge conflicts”. If you want to avoid those, you should agree on who works (when) on which files.

ADVANCED TOPICS

- We have now covered all of the basics we wanted to include for today
- If you can apply what we have discussed so far, you have reached the learning objectives for this short workshop 😊

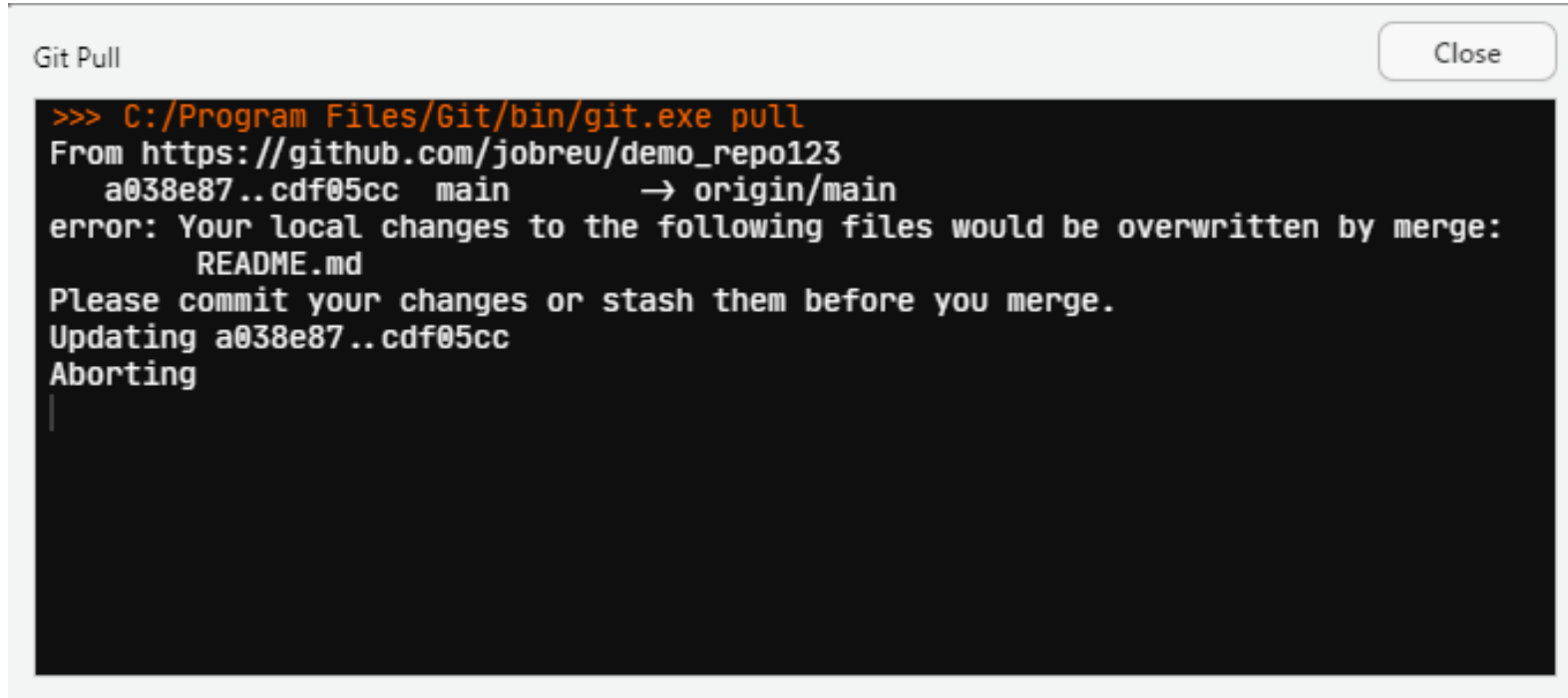
...

- However, we also wanted to cover some additional topics to make you aware that these things exist and prepare you for your future as an advanced Git & GitHub user: merge conflicts, pull requests, & issues
- We don't expect you to be able to do all of the things after this workshop, but believe it is important to have heard about them (think of this as learning vocabulary)

MERGE CONFLICTS

- Merge conflicts occur when there are competing changes made to the same lines in a file, or if one person deleted a file, while the other modified it
- Git will inform you about a merge conflict and also highlight the competing changes
- In the following, I will show an example in which I make local changes to a file, and want to pull from the remote repository (before committing and pushing my changes), but someone (in this simulated example also me but using the GitHub web interface) has edited the same lines in the file and pushed them before me

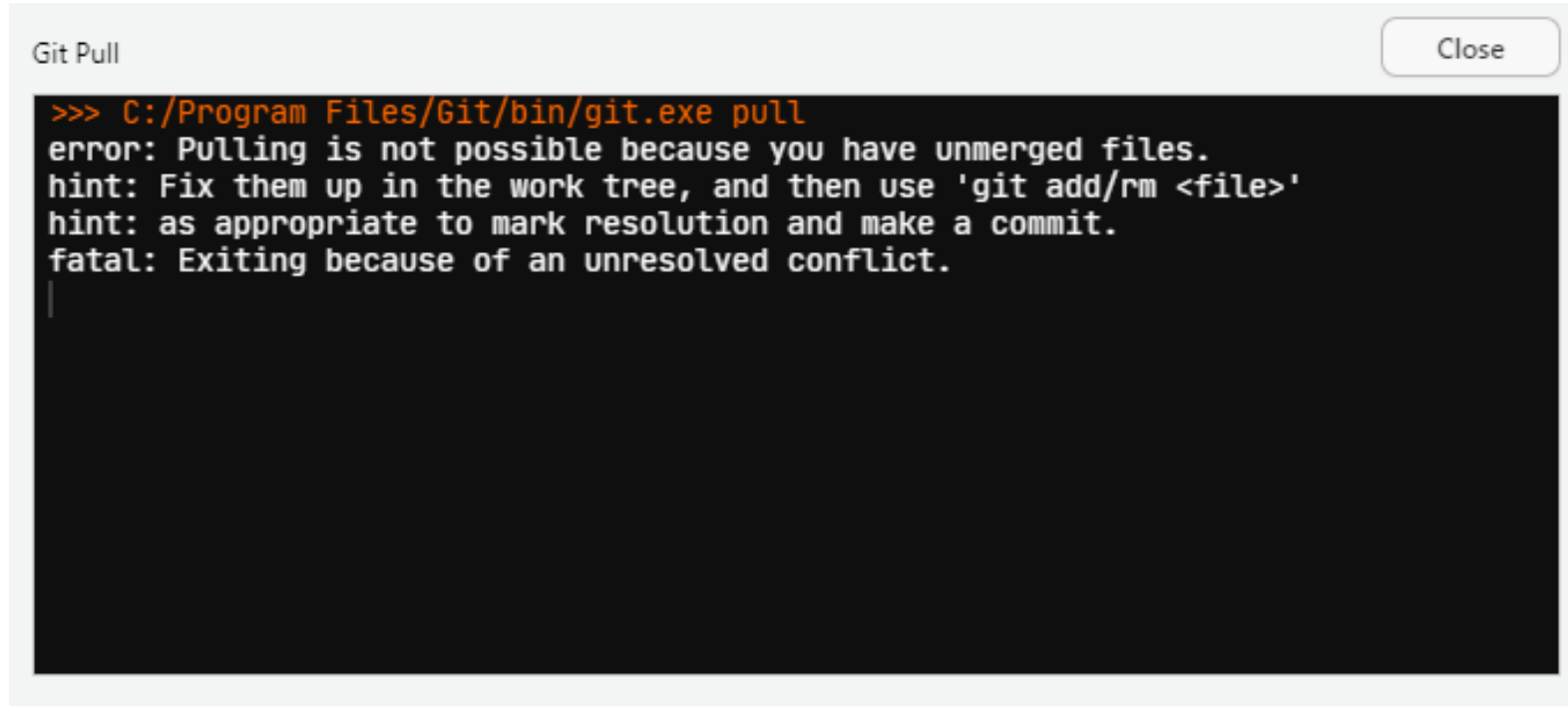
MERGE CONFLICT: UNCOMMITTED FILE



A screenshot of a terminal window titled "Git Pull" with a "Close" button in the top right corner. The terminal shows the execution of the command `git pull` from a remote repository. It indicates that the local changes to `README.md` would be overwritten by the merge. The terminal output is as follows:

```
>>> C:/Program Files/Git/bin/git.exe pull
From https://github.com/jobreu/demo_repo123
   a038e87..cdf05cc  main      → origin/main
error: Your local changes to the following files would be overwritten by merge:
    README.md
Please commit your changes or stash them before you merge.
Updating a038e87..cdf05cc
Aborting
|
```

MERGE CONFLICT: COMMITTED FILE

A screenshot of a terminal window titled "Git Pull" with a "Close" button in the top right corner. The terminal shows a command prompt where the user has entered "C:/Program Files/Git/bin/git.exe pull". The output of the command is an error message: "error: Pulling is not possible because you have unmerged files." followed by two hints: "hint: Fix them up in the work tree, and then use 'git add/rm <file>'" and "hint: as appropriate to mark resolution and make a commit." The message ends with "fatal: Exiting because of an unresolved conflict." and a cursor is visible on the line following the fatal message.

```
Git Pull
Close

>>> C:/Program Files/Git/bin/git.exe pull
error: Pulling is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.
```


HANDLING MERGE CONFLICTS

- You need to resolve the merge conflict by editing the respective file(s), then add and commit the changes
- NB: While editing the files, you also need to remove the characters Git uses to highlight the affected lines as well as the [git commit hash](#)

HANDLING MERGE CONFLICTS

The screenshot shows the RStudio IDE with a merge conflict in the `README.md` file. The conflict is highlighted with a red circle in the Source editor. The conflict arises from two different versions of the file being merged. The top version (HEAD) contains the text "This is a demo repo for the hands-on tutorial 'Using Git & GitHub via RStudio'", while the bottom version (cdcf9aeb451f88dd1ce9e26883f8e76bc10b) contains "This is a demo repo created for the hands-on tutorial 'Using Git & GitHub via RStudio' at CAIS".

Source Editor (README.md):

```
1 # demo_repo123
2 <<<<<< HEAD
3 This is a demo repo for the hands-on tutorial "Using Git & GitHub via RStudio"
4 =====
5 This is a demo repo created for the hands-on tutorial "Using Git & GitHub via RStudio" at CAIS
6 >>>>>> cdcf9aeb451f88dd1ce9e26883f8e76bc10b
7
```

Files Panel:

Name	Size	Modified
..		
.gitignore	609 B	Dec 20, 2022, 6:41 PM
demo_repo123.Rproj	218 B	Dec 22, 2022, 9:16 AM
README.md	265 B	Dec 22, 2022, 9:20 AM

Git Panel:

Environment | History | Connections | Git | Tutorial

Diff | Commit | Pull | Push | History | More

Your branch is ahead of 'origin/main' by 1 commit.

Staged	Status	Path
<input checked="" type="checkbox"/>	UU	README.md

Console:

```
R 4.1.3 - ~/Lehre/rstudio_git_github_cais_2023/demo_repo123/ ↵

R version 4.1.3 (2022-03-10) -- "One Push-Up"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

PULL REQUESTS

- Pull requests only work in the context of a web platform like GitHub
- If you are a collaborator, pull requests are the cautious way of contributing (instead of directly committing to the main branch)
- If you are not a collaborator, then pull requests are the only way of directly contributing to the repository

PULL REQUEST WORKFLOW

- If you are not a collaborator, you first need to create what is called a [“fork” of the repository](#)
- If you are a collaborator or after you have forked a repository, you need to create (and switch to) a new branch
- In the new branch, make the changes you want to make, stage, and commit them
- After that, you can [initiate a pull request](#)
- The owner of the repository can then review the pull request and accept it (by making a merge), reject it, or discuss it
 - The GitHub web interface provides useful functions for [commenting pull requests as well as changes within a pull request](#)


WHAT THE FORK?!?

- Forking means creating a personal copy of someone else's project (on GitHub)
- Forking works only for public repositories (or private repositories for which you are a collaborator)
- NB: Forking a repository is different from cloning it. You can find a good explanation and comparison here (esp. in the table at the end of the page):
<https://www.geeksforgeeks.org/difference-between-fork-and-clone-in-github/>




ISSUES


- An easier way of contributing to a repository (regardless of whether you are the owner, a collaborator or neither) is to open a so-called issue on GitHub
 - remember the analogy: creating an issue is similar to adding a comment to a document
 - typically, issues are used to make repository owners aware of things that do not work or can be improved
- Besides opening an issue (which puts the burden on the repository owner and collaborators), you can also address an issue and then close it
 - If this requires files to be changed, you either need to be the owner of the repository, a collaborator, or initiate a pull request

ISSUES



[Pull requests](#) [Issues](#) [Codespaces](#) [Marketplace](#) [Explore](#)

 [StefanJuenger / woRkshoptools](#) Public

[Watch](#) 1 [Fork](#) 0 [Starred](#) 4

[Code](#) [Issues](#) 4 [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#)

Label issues and pull requests for new contributors

Dismiss

Now, GitHub will help potential first-time contributors [discover issues](#) labeled with [good first issue](#)

Filters

[Labels](#) 9

[Milestones](#) 0

New issue

☐ 4 Open 0 Closed


Author Label Projects Milestones Assignee Sort


☐ Installation of fontawesome
#4 opened on Mar 11 by jobreu

☐ Fix paths to session config .Rmd files
#3 opened on Mar 11 by jobreu

☐ More styles [enhancement](#)
#2 opened on Jul 23, 2021 by StefanJuenger

☐ add_presenter() function [enhancement](#)
#1 opened on Jul 23, 2021 by StefanJuenger

 **ProTip!** Notify someone on an issue with a mention, like: @jobreu.

 © 2022 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

BOTTOM LINE OF ADVANCED TOPICS

- If possible, maintain a workflow that reduces the risk or minimizes merge conflicts
- If merge conflicts arise, you can tackle them by editing the conflicting files (and there are plenty of resources on this online, including the [GitHub documentation](#))
- Creating or – even better – addressing issues is a great way of getting started with contributing to open-source projects on GitHub
- Initiating a pull request is a more advanced/involved way of contributing
 - Although it may sound daunting at first, there also are many helpful online resources on this subject, including the [R-focused GitHub repo “first-contributions” by FORWARDS](#)

EXERCISE

1. Form teams of two
2. Add each other as a collaborator to your GitHub repository
3. Clone each others' repository to your local machine
4. On your local computer: Change a file in the clone of the other person's repository (e.g., the ReadMe file), then stage, commit, and push
5. After your partner has pushed their changes to your repository, pull them in your local copy of your own repository

Bonus: If you are feeling adventurous, create and resolve a merge conflict



RESEARCH
FOR THE
DIGITAL AGE

FURTHER RESOURCES

- Jennifer Bryan: Happy Git and GitHub for the useR see <https://happygitwithr.com/index.html>
- Posit/RStudio How-To-Article “Version Control with Git and SVN”: <https://support.posit.co/hc/en-us/articles/200532077>