Git: A Researcher's Guide to Collaboration and Version Control Tools

Dominika Regéciová

FIT BUT, Gen Digital
iregeciova@fit.vut.cz
dominika.regeciova@gendigital.com

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

A few words about me

- I am a Ph.D. student and researcher at FIT BUT
- Researcher for Gen Digital (formerly Avast)
- My research: Application of formal models in security
- Areas of interest:
  - Formal models and languages in security
  - Pattern matching
  - Improvement of tools for malware analysts

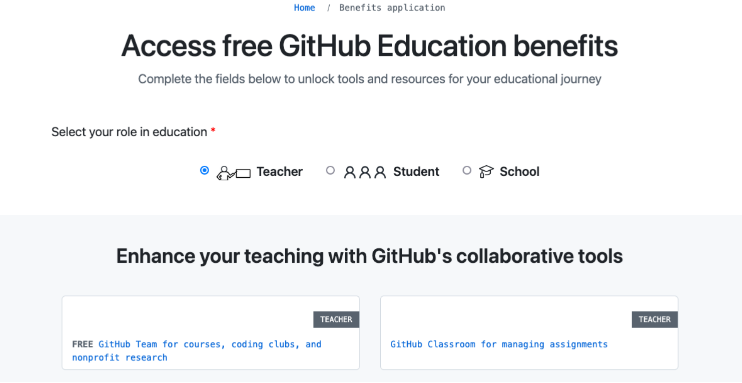
Materials

- All materials can be found at **https://github.com/regeciovad/git2025**

The First Steps

GitHub Education

- Create a free account
- Teachers and students have additional free features
- Connecting the account with the university is an easy process



Home / Benefits application

Access free GitHub Education benefits

Complete the fields below to unlock tools and resources for your educational journey

Select your role in education *

● Teacher ○ Student ○ School

Enhance your teaching with GitHub's collaborative tools

TEACHER
FREE GitHub Team for courses, coding clubs, and nonprofit research

TEACHER
GitHub Classroom for managing assignments

Installing Git

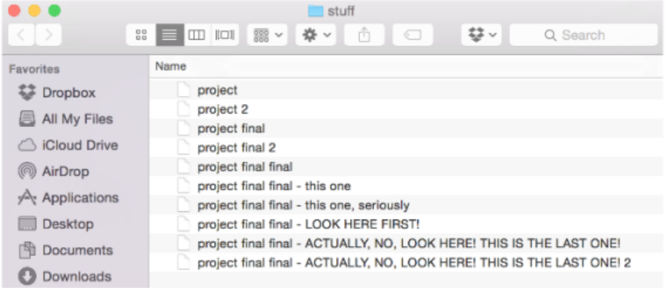
Windows installation

```
$ brew install git
```

```
$ apt-get install git
# or
$ dnf install git
```

Introduction

Motivation



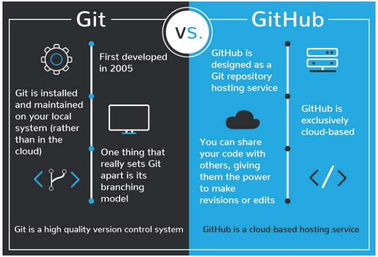
project
project 2
project final
project final 2
project final final
project final final - this one
project final final - this one, seriously
project final final - LOOK HERE FIRST!
project final final - ACTUALLY, NO, LOOK HERE! THIS IS THE LAST ONE!
project final final - ACTUALLY, NO, LOOK HERE! THIS IS THE LAST ONE! 2

Motivation

- Why use Git?
  - Version history — what changes were made and by who
  - Collaboration — many people can work on one project at once
  - Parallel development — multiple versions of files accessible at once
  - Repository — smart folder for your projects
  - Additional tools for testing, issues tracking, and much more

Story of the Git

- Git is a distributed version control system (a tool, local repository)
- Git was created by **Linus Torvalds** in 2005 as an open-source project
- Goal: managing big projects, storing the history of changes, and allowing collaboration between many users
- Git repository services (cloud service/website):
  - **GitHub**
  - SourceForge
  - Bitbucket
  - GitLab



First developed in 2005

Git is installed and maintained on your local system (rather than in the cloud)

One thing that really sets Git apart is its branching model

Git is a high quality version control system

GitHub is designed as a Git repository hosting service

GitHub is exclusively cloud-based

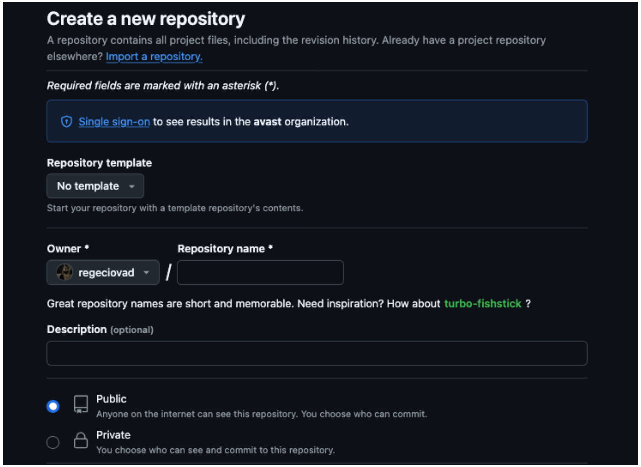
You can share your code with others, giving them the power to make revisions or edits

GitHub is a cloud-based hosting service

Basic Concepts in Git

Repositories

- The repository is a digital folder for your projects
- The easiest way to create a project is from the GitHub page



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 (*).

Single sign-on to see results in the **avast** organization.

Repository template

No template

Start your repository with a template repository's contents.

Owner * regeciovad / **Repository name ***

Great repository names are short and memorable. Need inspiration? How about **turbo-fishstick** ?

Description (optional)

● 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.

Repositories

- Public repository
  - Visible to the public (link, search, etc.)
  - Everything is visible (files, history, etc.)
  - Changes can be limited (more on that later)
- Private repository
  - Only allowed accounts can see the repository
  - The data is protected and not publicly available
  - You can manage who has access to the files

Git Settings

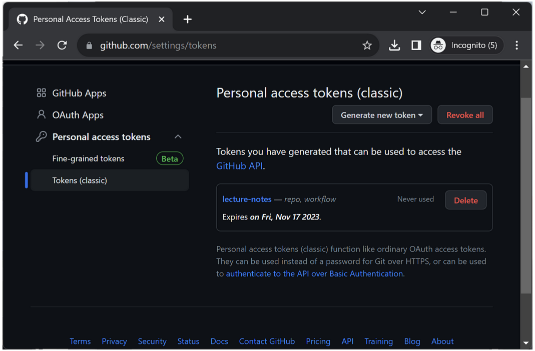
- Add this file to the home directory (C : \Users\John, /Users/John, etc.)

.gitconfig

```
[user]
        name = Dominika Regéciová
        email = regeciovad@gmail.com
[color]
        ui = true
[advice]
        ; Make git a little less verbose.
        pushNonFastForward = false
        statusHints = false
        commitBeforeMerge = false
        resolveConflict = false
        detachedHead = false
[fetch]
        ; Automatically prune during `git fetch`.
        prune = true
[rebase]
        ; Automatically stash before a rebase and unstash afterwards.
        autoStash = true
```

Git Access Token

- https://github.com/settings/tokens
- Create a new token for general use
- Save the token — it is your new password for commands



Personal access tokens (classic)

Tokens you have generated that can be used to access the GitHub API.

lecture-notes — *repo, workflow* Never used

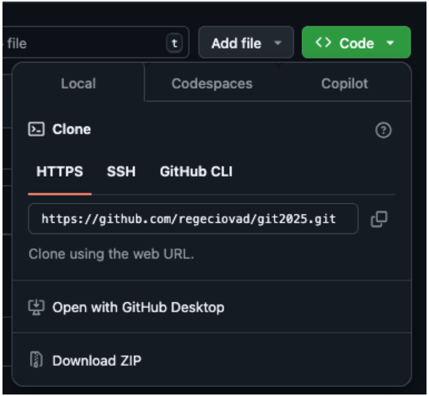
Expires *on Fri, Nov 17 2023*.

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.

Clone Repository

git clone

```
$ git clone https://github.com/regeciovad/git2025.git
```



https://github.com/regeciovad/git2025.git

Clone using the web URL.

Git Status

- Your repository is in the new folder
- The repository should be **up to date**

git status

```
$ cd git2025
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Adding New File

- When adding a new file, is it **untracked** first
- We need to **add** it for tracking

git add

```
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
        example.json

nothing added to commit but untracked files present

$ git add example.json
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
        new file:   example.json
```

Changing The File

- When changing the existing file, it becomes **modified**
- We can see changes with **git diff**
- We also need to add it to the tracking

git diff

```
Changes not staged for commit:
        modified:   README.md
$ git diff
diff --git a/README.md b/README.md
--- a/README.md
+++ b/README.md
@@ -1 +1,2 @@
-# git2025
+# git2025
+my big change
```

Creating Commits

- **Commits** — grouped changes with message, date, and author
- When committing, untracked files are ignored

git commit

```
$ git status
Changes to be committed:
        modified:   README.md
        new file:   example.json
$ git diff --staged

$ git commit -m 'Description of the change'
[main a1c8cee] Description of the change
 2 files changed, 99 insertions(+), 1 deletion(-)
 create mode 100644 example.json

$ git status
Your branch is ahead of 'origin/main' by 1 commit.
```

History Of Commits

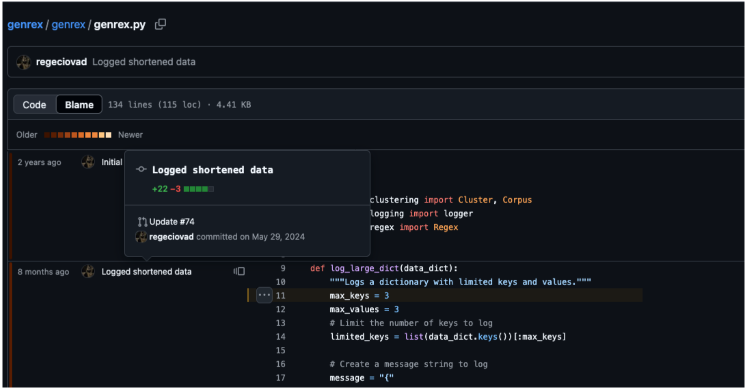
- Each commit has a hash as an ID
- In commits, we are storing just the changes

git log

```
$ git log
commit a1c8cee2f0e8f14b3fac77571004a7b632bea12b (HEAD -> main)
Author: Dominika Regéciová <iregeciova@fit.vutbr.cz>
Date:   Mon Jan 13 21:09:55 2025 +0100

    Description of the change

commit fe5f6e36a5464e9aab4b06abeba5f8193e475a43 (origin/main, origin/HEAD)
Author: Dominika Regéciová <regeciovad@gmail.com>
Date:   Fri Jan 10 14:37:43 2025 +0100

    Initial commit
```

Git Blame

- Each line and each change are logged
- With git blame, you can find who made the change
- Use website GUI for better visualization



genrex / genrex / genrex.py

regeciovad Logged shortened data

134 lines (115 loc) · 4.41 KB

2 years ago Initial

Logged shortened data

+22 −3

Update #74

regeciovad committed on May 29, 2024

8 months ago Logged shortened data

```
clustering import Cluster, Corpus
logging import logger
regex import Regex

def log_large_dict(data_dict):
    """Logs a dictionary with limited keys and values."""
    max_keys = 3
    max_values = 3
    # Limit the number of keys to log
    limited_keys = list(data_dict.keys())[:max_keys]

    # Create a message string to log
    message = "{"
```

Undoing Things

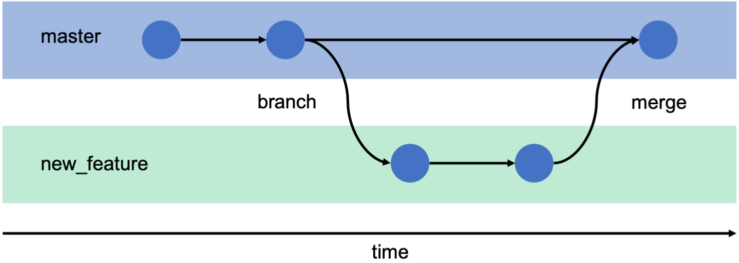
- In most cases, you can just make new changes and commit them
- In some cases, you can use special commands

git undo

```
$ git reset --soft HEAD~1 # undo the last commit
$ git reset README.md # undo add
$ git checkout README.md # undo changes
$ git restore --staged example.json # undo adding new files
```

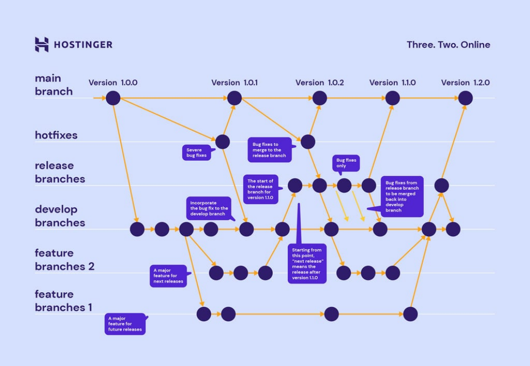
Branches

- One repository can have many versions — branches
- The main version of your repository is in a branch *origin/main* or *origin/master* (older name)
- The branches in theory:



Branches

- One repository can have many versions — branches
- The main version of your repository is in a branch *origin/main* or *origin/master* (older name)
- The branches in reality:


HOSTINGER
Three. Two. Online
main branch
Version 1.0.0
Version 1.0.1
Version 1.0.2
Version 1.1.0
Version 1.2.0
hotfixes
Severe bug fixes
Bug fixes to merge to the release branch
release branches
Bug fixes only
The start of the release branch for version 1.1.0
Bug fixes from release branch to be merged back into develop branch
develop branches
Incorporate the bug fix to the develop branch
feature branches 2
A major feature for next releases
Starting from this point, "next release" means the release after version 1.1.0
feature branches 1
A major feature for future releases

Use branches

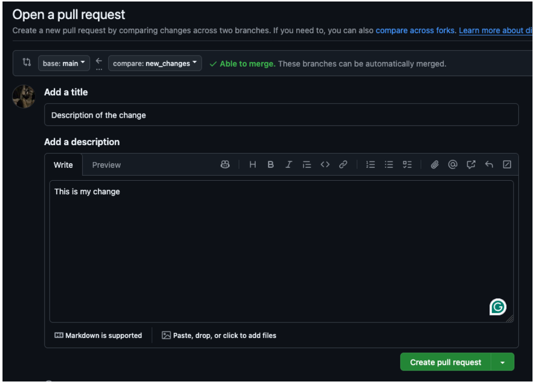
- In most cases, you want to use branches
- When publishing, you need your username and TOKEN

git push

```
$ git checkout -b new_changes # create and switch to new branch
$ git checkout main # switch back
$ git push origin HEAD:new_changes # publish your changes
```

Merge (Pull) Request

- On the website, the changes are automatically detected
- You can see every change you made
- You can add a message with more details



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks. Learn more about di

base: main ← compare: new_changes ✓ Able to merge. These branches can be automatically merged.

Add a title

Description of the change

Add a description

This is my change

Getting The Changes

- You want to keep your local repository up-to-date

git pull

```
$ git pull origin main
From https://github.com/regeciovad/git2025
 * branch              main        -> FETCH_HEAD
   fe5f6e3..0e3b31a  main        -> origin/main
Updating 3f3bdfe..0e3b31a
Fast-forward

$ git log
commit 0e3b31a7b694926768d11ac2d9a0fa1974831e3c (HEAD -> main, origin/main,
↪  origin/HEAD)
Merge: fe5f6e3 3f3bdfe
Author: Dominika Regéciová <regeciovad@gmail.com>
Date:   Mon Jan 13 22:17:39 2025 +0100

    Merge pull request #1 from regeciovad/new_changes

    Description of the change
```

Getting The Changes

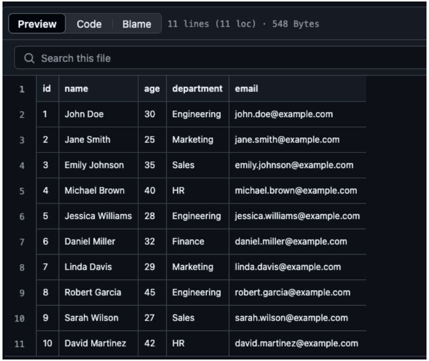
- If you have local changes, commit them and then use the **rebase** command

git rebase

```
$ git rebase origin/main
Successfully rebased and updated refs/heads/bla.

$ git log
commit bbbc585972c6affd04b9c871e4f113dfcf3d1081 (HEAD -> bla)
Author: Dominika Regéciová <iregeciova@fit.vutbr.cz>
Date:   Mon Jan 13 22:33:06 2025 +0100

    touched

commit 0e3b31a7b694926768d11ac2d9a0fa1974831e3c (origin/main, origin/HEAD, main)
Merge: fe5f6e3 3f3bdfe
Author: Dominika Regéciová <regeciovad@gmail.com>
Date:   Mon Jan 13 22:17:39 2025 +0100

    Merge pull request #1 from regeciovad/new_changes

    Description of the change
```

Supported Formats

- Text files like .txt, .tex, .html, .py, etc.
- PDF, CSV files are rendered on the website
- You can upload any format, but the tracking will not work properly

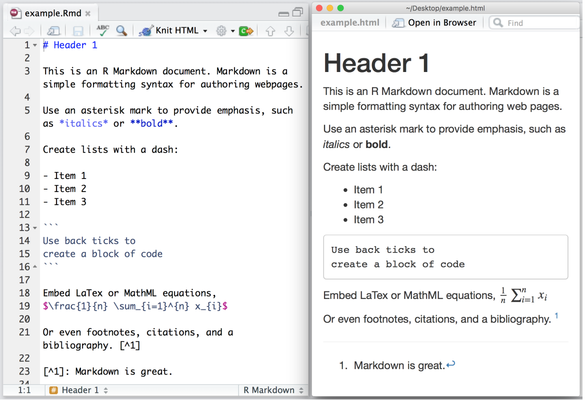


id	name	age	department	email
1	John Doe	30	Engineering	john.doe@example.com
2	Jane Smith	25	Marketing	jane.smith@example.com
3	Emily Johnson	35	Sales	emily.johnson@example.com
4	Michael Brown	40	HR	michael.brown@example.com
5	Jessica Williams	28	Engineering	jessica.williams@example.com
6	Daniel Miller	32	Finance	daniel.miller@example.com
7	Linda Davis	29	Marketing	linda.davis@example.com
8	Robert Garcia	45	Engineering	robert.garcia@example.com
9	Sarah Wilson	27	Sales	sarah.wilson@example.com
10	David Martinez	42	HR	david.martinez@example.com

Additional Tools and Tips

Markdown

- HTML-like language for your README files, comments, etc.
- Markdown is like writing in plain text but is rendered richly
- Basic syntax and examples



~~~
# Header 1

This is an R Markdown document. Markdown is a
simple formatting syntax for authoring webpages.

Use an asterisk mark to provide emphasis, such
as *italics* or **bold**.

Create lists with a dash:

- Item 1
- Item 2
- Item 3

```
Use back ticks to
create a block of code
```

Embed LaTex or MathML equations,
$\frac{1}{n} \sum_{i=1}^{n} x_{i}$

Or even footnotes, citations, and a
bibliography. [^1]

[^1]: Markdown is great.
~~~

Header 1

This is an R Markdown document. Markdown is a simple formatting syntax for authoring web pages.

Use an asterisk mark to provide emphasis, such as *italics* or **bold**.

Create lists with a dash:

- Item 1
- Item 2
- Item 3

```
Use back ticks to
create a block of code
```

Embed LaTex or MathML equations, $\frac{1}{n} \sum_{i=1}^{n} x_{i}$

Or even footnotes, citations, and a bibliography. [1]

---

1. Markdown is great.↩

Gitignore File

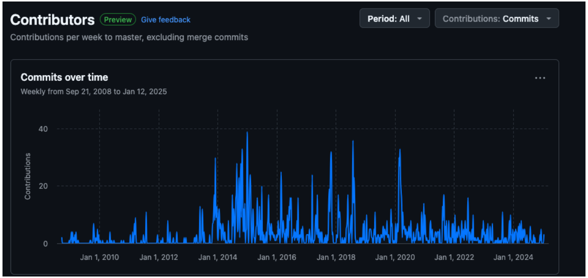
- Not every file should be added to the repository:
  - Too large files, temporary, binary files
  - Files containing private information, like passwords!
  - Gitignore file – patterns for paths that should be ignored
  - A collection of helpful examples

.gitignore file for Python

```
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class
...
# Distribution / packaging
dist/
downloads/
...
```

GitHub website

- **Issues** — tracking tasks
- **Wiki** — documentation in Markdown
- **Insights** — statistics about the repository
- **Search** — Free resource of code and information



Contributors

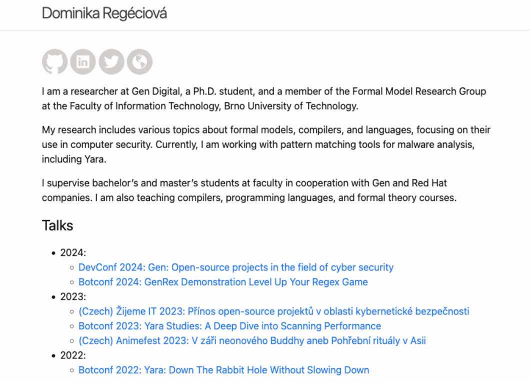
Contributions per week to master, excluding merge commits

Commits over time

Weekly from Sep 21, 2008 to Jan 12, 2025

Website

- You can build a website with GitHub
- For setup, you can follow this simple tutorial



Dominika Regéciová

I am a researcher at Gen Digital, a Ph.D. student, and a member of the Formal Model Research Group at the Faculty of Information Technology, Brno University of Technology.

My research includes various topics about formal models, compilers, and languages, focusing on their use in computer security. Currently, I am working with pattern matching tools for malware analysis, including Yara.

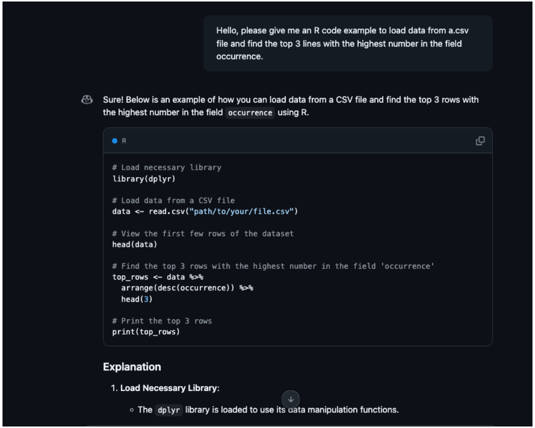
I supervise bachelor's and master's students at faculty in cooperation with Gen and Red Hat companies. I am also teaching compilers, programming languages, and formal theory courses.

Talks

- 2024:
  - DevConf 2024: Gen: Open-source projects in the field of cyber security
  - Botconf 2024: GenRex Demonstration Level Up Your Regex Game
- 2023:
  - (Czech) Žijeme IT 2023: Přínos open-source projektů v oblasti kybernetické bezpečnosti
  - Botconf 2023: Yara Studies: A Deep Dive into Scanning Performance
  - (Czech) Animefest 2023: V záři neonového Buddhy aneb Pohřební rituály v Asii
- 2022:
  - Botconf 2022: Yara: Down The Rabbit Hole Without Slowing Down

Copilot

- GitHub AI assistant
- Web interface, or as an extension in Visual Studio Code



Hello, please give me an R code example to load data from a.csv file and find the top 3 lines with the highest number in the field occurrence.

Sure! Below is an example of how you can load data from a CSV file and find the top 3 rows with the highest number in the field `occurrence` using R.

```r
# Load necessary library
library(dplyr)

# Load data from a CSV file
data <- read.csv("path/to/your/file.csv")

# View the first few rows of the dataset
head(data)

# Find the top 3 rows with the highest number in the field 'occurrence'
top_rows <- data %>%
  arrange(desc(occurrence)) %>%
  head(3)

# Print the top 3 rows
print(top_rows)
```

Explanation

1. **Load Necessary Library:**
    - The `dplyr` library is loaded to use its data manipulation functions.

Conclusion

Conclusion

- Git is a fantastic tool that can make your life a lot easier
- My recommendations:
  - Use GitHub Education license for free features like Copilot
  - Get familiar with basic commands, and try to use them on a simple project
  - Try to set up a repository for your students
  - Pro Git book