



Git(Hub)

A Gentle Introduction for R Users

Dr. Edgar J. Treischl

Last update: 2025-11-05

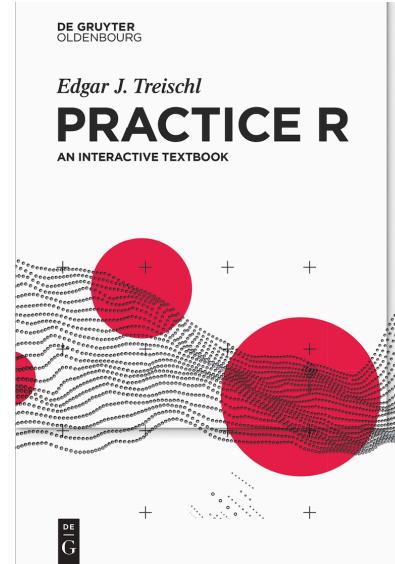
Press **O** or

Agenda

1 Intro Git/GitHub

2 Getting Started with R Studio

3 Basics of Git and Workflow



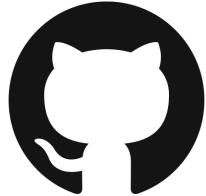
This workshop is based on Practice R.

1 Git/GitHub ... What?

(01) Git/GitHub ... What?



- Git is a version control system to track changes over time
- Continuous integration (CI) and continuous deployment (CD)
- GitHub (GitLab) is an host (website) for Git-based projects
- Each project is stored in a repository (folder with all files)
- A repository can be *public* or *private*



Why Hosts like GitHub?

- Easier to track code changes
- Share work
- Automated workflows
- Websites for projects
- Work together
- ...

(01) Git/GitHub ... What?

Track Code

Share Code

Automate workflows

		@@ -1,6 +1,6 @@	Stage chunk	Discard chunk
1		#' Shows Links from the Practice R Book.		
	1	#' Shows the links from the Practice R Book.		
2	2	#'		
3	3	#' @description The function copies the URL and		
4	4	#' opens the link in the browser.		
5	5	#'		
6	6	#' @param name URL name		



2

Getting Started

HBOmax

6 / 35

(02) Getting Started

Push and pull 💪 on a regular basis:

1. Install Git:

- Install on your local machine
- Introduce *yourself*
- Learn some *Git*

2. Connect to GitHub

- Create a *Personal Access Token* (PAT)
- Create a *Repository*

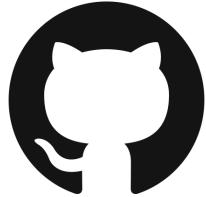
3. Connect GitHub and R Studio

- *Clone* the newly created repository
- *Setup R Studio* and connect to GitHub
- Get in touch with the R Studio *Git pane*

(02) Install Git on your local machine



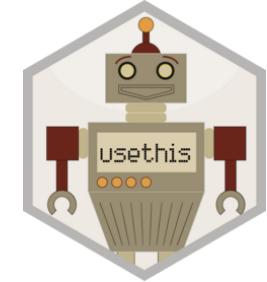
- Windows: <https://git-scm.com/download/win>
- OS X: <https://git-scm.com/download/mac>
- Debian/Ubuntu: sudo apt-get install git-core



```
# I am git code to check if git is installed  
  
# Go to the console and type:  
git --version  
  
## git version 2.50.1 (Apple Git-155)
```

- Next, create a free GitHub account

(02) Install Git: Introduce yourself



Via the shell

```
#In the console  
git config --global user.name "User Name"  
git config --global user.email "email@adress.com"
```

Or with the 🚀 usethis package

```
library(usethis)  
use_git_config(user.name = "Jane Doe",  
               user.email = "jane@example.org")
```

(02) Install Git: Introduce yourself

Check global configuration settings

```
git config --global --list

## filter.lfs.clean=git-lfs clean -- %f
## filter.lfs.smudge=git-lfs smudge -- %f
## filter.lfs.process=git-lfs filter-process
## filter.lfs.required=true
## user.name=edgar.treischl
## user.email=edgar.treischl@isb.bayern.de
## core.excludesfile=~/ .gitignore
## color.ui=auto
## credential.helper=osxkeychain
```



Git can be scary, but ...

(02) Install Git: New Vocabulary



Wait, Git is scary because it's full of technical jargon¹:

- **Repository:** A folder with all project files
- **Push:** Send code (files) to Git
- **Pull:** Get code (files) from a Git repository
- **Commit:** A snapshot of a repository at a certain point in time
- **Branch:** A parallel version within the repository to test new features

...



[1] As often, R Studio helps us to work with Git.

'Artwork by @allison_horst'

(02) Install Git: New Vocabulary



- **Merge:** Combine two branches
- **Merge conflicts:** When two branches have changed the same part of a file, Git will not be able to automatically determine what is correct
- **Forks:** Forks are interconnected repositories, for example, to examine a repository from someone else
- **Pull requests:** A way to suggest changes to a repository



'Artwork by @allison_horst'

(02) Connect to GitHub: Create a PAT

Create a PAT in R Studio:

```
# The create_github_token() function will open a browser window to create a token
usethis::create_github_token()

#The set_github_pat() function will store the token in the .Renviron file
credentials::set_github_pat("token")
```

Or manually via:

On the GitHub website, go to:

- Settings > Developer settings > Personal access tokens:
- <https://github.com/settings/tokens>

(02) Connect to GitHub

Check if R Studio is connected to GitHub:

```
# Actually, whoami will show you who you are on GitHub ...
gh::gh_whoami()
```

```
## {
##   "name": "Edgar Treischl",
##   "login": "edgar-treischl",
##   "html_url": "https://github.com/edgar-treischl",
##   "scopes": "gist, read:org, repo, workflow",
##   "token": "gho_...P1X2"
## }
```

 ... you're connected!

(02) Connect to GitHub: Create a repository

Create a new repository

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

[Import a repository.](#)

Repository template

Start your repository with a template repository's contents.

No template ▾

Owner *



Repository name *

Add repository name

Great repository names are short and memorable. Need inspiration? How about [urban-giggle](#)?

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.

Public or private repository

(02) Connect to GitHub: Create a repository

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/edgar-treischl/Test_Repro.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

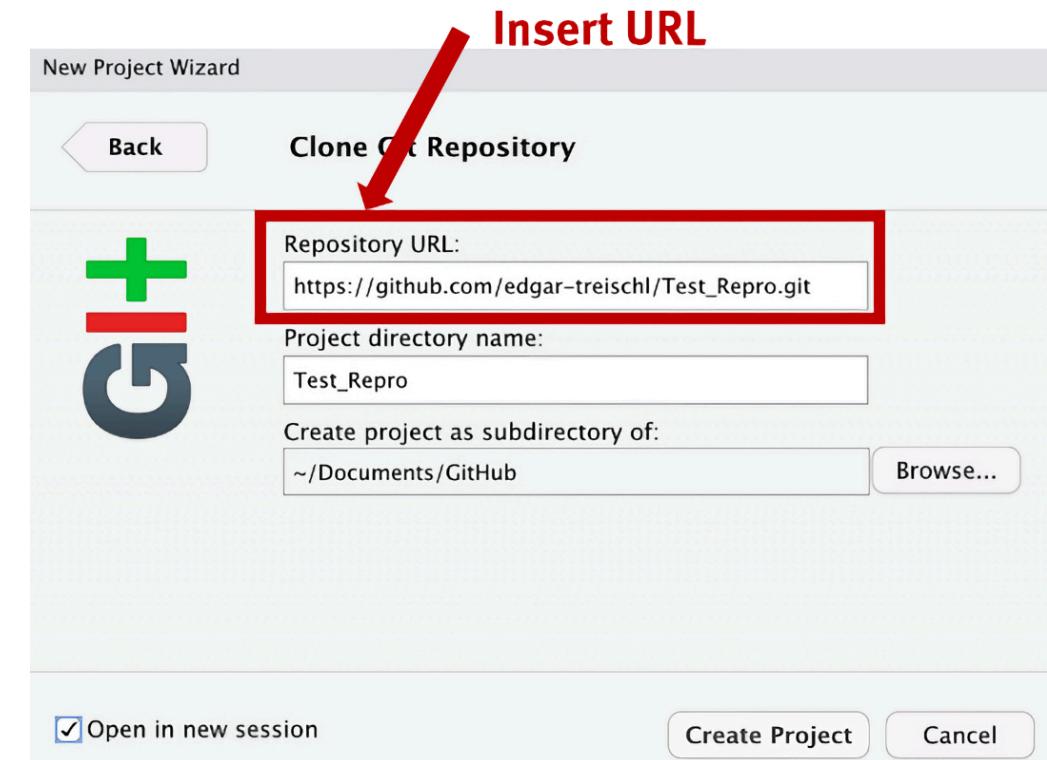
```
echo "# Test_Repro" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/edgar-treischl/Test_Repro.git
git push -u origin main
```

Copy URL

A red arrow points from the text "Copy URL" to the copy icon (a clipboard with a plus sign) located next to the repository URL.

(02) Connect GitHub and R Studio

1. Create a new project in R Studio and click next.
2. Pick Version Control and click next.
3. Pick Git and click next.
4. Paste the repository URL: 



(02) Connect GitHub and R Studio: Git Pane

Once R Studio is connected, search for the Git pane:

Files:

Modified

Deleted

Added

Untracked

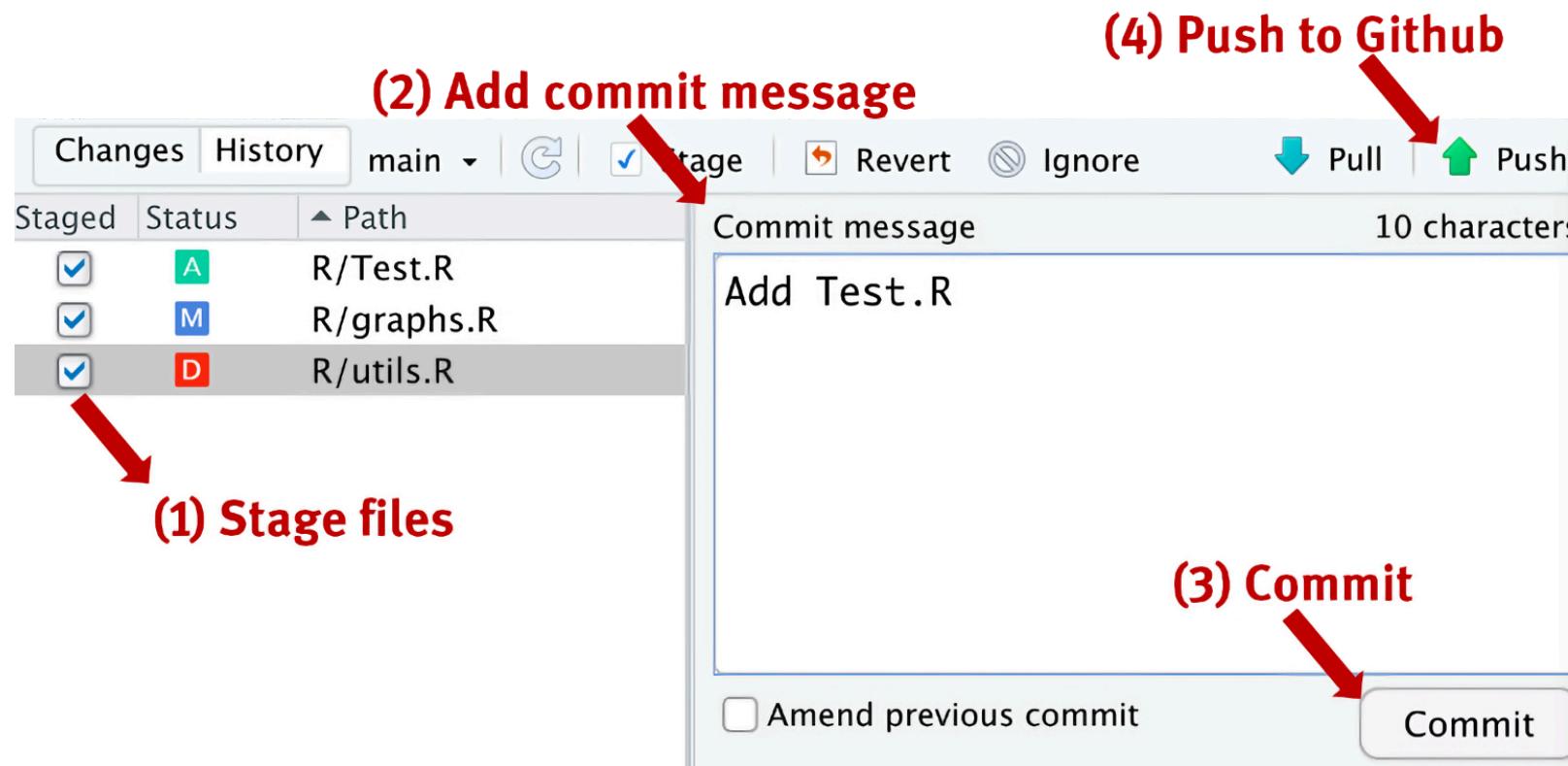
Renamed

The screenshot shows the R Studio interface with the 'Git' tab selected in the top navigation bar. Below the tabs, there are buttons for Diff, Commit, Pull, Push, History, and More. A message indicates that the branch is ahead of 'origin/main' by 4 commits. The main area is titled 'Files:' and contains a table with columns for Staged, Status, and Path. The table lists the following files:

Staged	Status	Path
<input type="checkbox"/>	M	R/graphs.R
<input type="checkbox"/>	D	R/model.R
<input checked="" type="checkbox"/>	A	R/new_model.R
<input type="checkbox"/>	?	R/test_file.R
<input checked="" type="checkbox"/>	R	R/analysis_v77.R -> R/analysis_final.R

(02) Connect GitHub and R Studio: Git Pane

You gonna stage, commit, push, and let it all out 🎉





Practice Makes Perfect

Practice Makes Perfect

+ Create a new repository on GitHub

 Clone the repository on your local machine

 Add a new repository file

 Commit the changes

 Push the changes to the remote repository

A grayscale photograph of a person with long hair, wearing a dark hoodie, sitting at a desk and looking down at a laptop screen. The background is slightly blurred.

3 Basics of Git

(03) Git basics

```
# Initialize a new repository  
git init <your repository name>  
  
# Clone a repository  
git clone <git-repo-url>  
  
# Create a new branch  
git branch <branch-name>  
  
# List all branches  
git branch --list  
  
# Delete a branch  
git branch -d <branch-name>  
  
# Switch to a branch  
git checkout <branch-name>
```

(03) Git for the daily workflow

```
# Add a file to the staging area
```

```
git add <file-name-1>
```

```
# Add all files to the staging area
```

```
git add -A
```

```
# Commit changes
```

```
git commit -am <commit-message>
```

```
# Push changes to the remote rep
```

```
git push <remote> <branch-name>
```

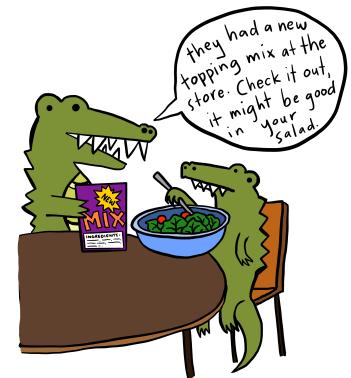
```
# Pull changes from the remote rep
```

```
git pull
```

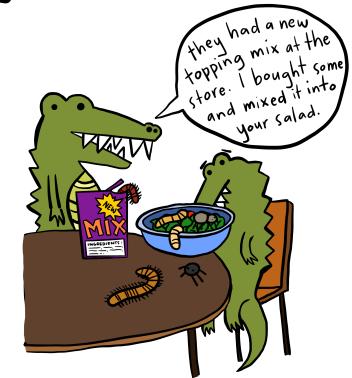
```
# Merge two branches
```

```
git merge <branch-name>
```

git fetch: bring stuff home



git pull: bring stuff home AND merge it in



check what you're getting before merging it in with git fetch
@allison_horst

'Artwork by @allison_horst'

(03) Feature Branches

Work on new ideas and experiment without worrying about breaking the code or interrupting your teammates' flow

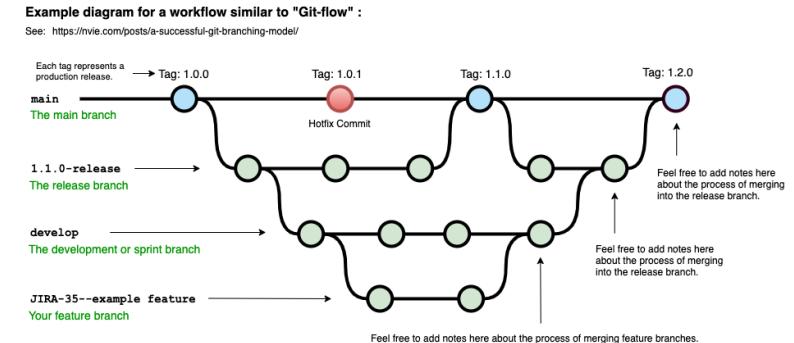
```
# Create a new branch and switch to it  
git checkout -b feature/shiny_map
```

```
> Switched to a new branch  
'feature/shiny_map'
```

- Push your branch to the shared repository:

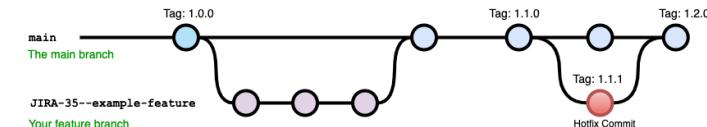
```
# Push your branch to the remote repository  
git push origin feature/shiny_map
```

Example Git Branching Diagrams



Example diagram for a workflow with a simpler branching model:

See: <https://gist.github.com/jbenet/ee6c9ac48068889b0912> or <https://www.endoflineblog.com/oneflow-a-git-branching-model-and-workflow>



'Source: bryanbraun'

=> Next step: creating a pull (GitHub) or merge (GitLab) request

(03) Merge Request

- MR are key checkpoint for collaboration, feedback, and maintaining high-quality code (Code Review)
- MR description:
 - A summary of what you changed and why
 - Mentions of any relevant people using @username
 - Links to related *issues*, tickets, or discussions

```
# Create a new branch for the MR and push it
git checkout -b 42-add-shiny-map
git push origin 42-add-shiny-map
```



Closing Keyword for *Issues*:

- Closes #42
- Fixes #42
- Resolves #42

(03) Even more Git? 😱

Do I need all the code if I am happy with the Git Pane?

- The Pane simplifies common Git operations, but Git Code gives you more control, flexibility, and the ability to handle complex operations
- **Common example:** Merge conflicts can occur when two branches have changed the same part of a file and Git will not be able to automatically determine what is correct
- Git commands give you a more detailed understanding of where the conflicts are happening. This is particularly useful when the conflict isn't straightforward to resolve.
- Of course, we don't have to remember in detail 😊

Merge Conflict:

```
# <<<<< HEAD
# print("This is my feature branch.")
# =====
# print("This is my master branch.")
# >>>>> master
```

Typical workflow to solve the merge conflict:

- Manually resolve the conflict in the file(s) and run:

```
git add script.R
git commit -m "Resolved merge conflict"
```

(03) Commit history in R Studio

Commits

Subject	Author	Date (UTC)	SHA
origin/HEAD@{1} origin/master origin/HEAD del emoji 1	edgar-treischl <edgar.treischl@fau.de>	2022-09-09	56c57ba2
V1	edgar-treischl <edgar.treischl@de>	2022-04-24	0f021914
add description	edgar-treischl <edgar.treischl@de>	2022-04-23	2fd9ac3f
add description	edgar-treischl <edgar.treischl@de>	2022-04-23	723c89fe
add description fun	edgar-treischl <edgar.treischl@de>	2022-04-14	f1421857
addin readme text adjusted	edgar-treischl <edgar.treischl@de>	2022-03-28	fafaf714
...			

SHA 2fd9ac3f494da64d30224f1dead3d3f78810562b
Author edgar-treischl <edgar.treischl@de>
Date (UTC) 2022-04-23 08:01
Subject add description
Parent 723c89fef03d9bc4a889b37f74adf9815b5f1d57

R/copycat_addin.R

R/copycat_addin.R
@@ -1,7 +1,7 @@
1 1 #' Manage code snippets via the `copycat_addin()`.
2 2 #'
3 3 #' @description The `copycat_addin()` starts a small app in the viewer
4 4 #' that shows the copycat or any data frame for Copycat. Just select an R package
5 5 #' @description The `copycat_addin()` opens a gadget shiny app
6 6 #' that shows the copycat or your data for Copycat. Just select an R package
7 7 #' and the corresponding code snippet. Then press
8 8 #' the button and the code will be insert into the current document at the
9 9 #' location of the cursor. The `copycat_addin()` is inspired by parsnip addin,

View file

View file @ 2fd9ac3f

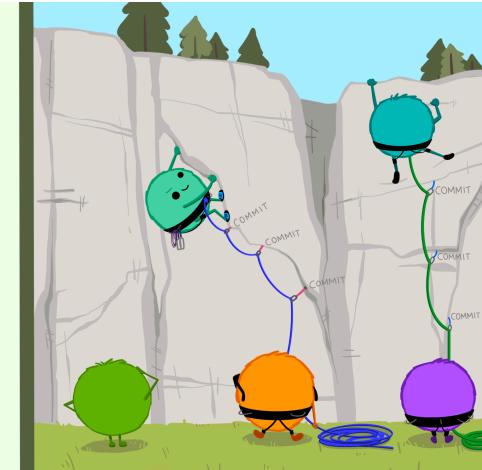
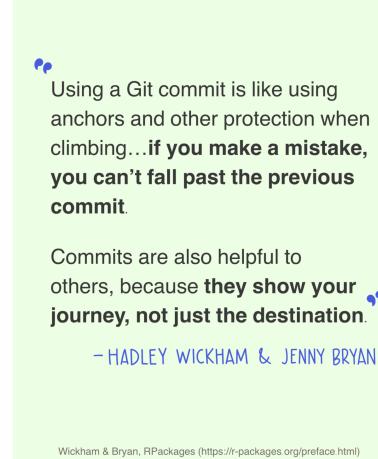
Commit Messages

Commit messages help to:

- Understand the evolution of the code
- Debug effectively
- Collaborate with others
- Use GitLab issues to track development
- Improve CI/CD troubleshooting
- Start with a verb in the imperative mood: "Add", "Fix", "Modify", "Delete"
- Keep the subject line short and focused (ideally under 50 characters)
- Optional: Include a short body if the change is complex and needs context

An Example:

```
#> Fix plot alignment in create_pdf  
#> It prevents overflow when long axis labels are used.
```



'Allison Horst'

Commit Messages: Pitfalls

Vague commit messages: "Fix bug"

Omitting context for complex changes: "Refactor function"

Committing unrelated changes together in one commit: "Update complex_function and fix bug several data processing steps"

It's helpful to integrate changes (1) early, (2) often, and (3) safely.

- (1): Push code as soon as it becomes functional, even if it's not polished yet. Holding onto work increases the risk of merge conflicts.
- (2): Commit changes regularly and keep them small. Frequent, incremental updates are easier to understand and quicker to review.
- (3): Every change that lands on the main branch must be ready for deployment. (pre-commit hooks, linting, and secret detection, integration tests).

(03) Git for troubleshooting



```
# Show who changed what and when
git blame file.txt

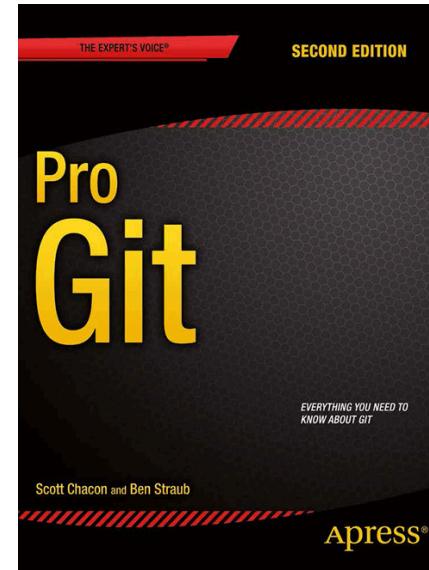
# Shows the working tree status
git status

# Shows changes between commits
git diff

#💀💀💀 DANGER
# Revert some existing commits
git revert SHA

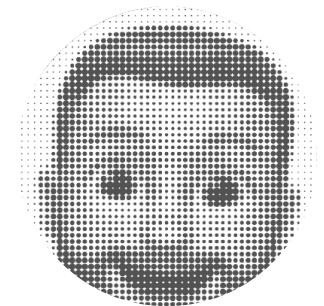
# Reset the repository to a previous commit
# Take care: This will discard any local
# changes
git reset

#Ultima ratio ;-)
# A "hard reset": forcefully discard any
# local changes
```



'Pro Git by Chacon and Straub'

Thank you for your attention!



 www.edgar-treischl.de

 [edgar-treischl](#)

 edgar.treischl@isb.bayern.de

Licence

This presentation is licensed under a CC-BY-NC 4.0 license. You may copy, distribute, and use the slides in your own work, as long as you give attribution to the original author on each slide that you use. Commercial use of the contents of these slides is not allowed.



References

- Chacon, S. and B. Straub (2014). *Pro Git*. Englisch. 2nd ed. New York, NY: Apress. ISBN: 978-1-4842-0077-3.
- Treischl, E. J. (2023). *Practice R: An interactive textbook*. Publication Title: *Practice R*. De Gruyter Oldenbourg.
- Wickham, H., J. Bryan, and M. Barrett (2022). *usethis: Automate Package and Project Setup*. R package version 2.1.6. URL: <https://CRAN.R-project.org/package=usethis>.
- Wickham, H., J. Hester, W. Chang, et al. (2022). *devtools: Tools to Make Developing R Packages Easier*. R package version 2.4.5. URL: <https://CRAN.R-project.org/package=devtools>.