

Organize your research and teaching material with Git (and , or whatever floats your boat)

Waldir Leôncio Netto

Scientific programmer

Oslo Centre for Biostatistics and Epidemiology (OCBE)

 [ocbe-uio](#)  [wleoncio](#)



First, let's get *sooo* meta

About the slides

- 🌐 Will be made available on  [ocbe-uios/public-slides](#)
- 📜 Licensed under 

About this presentation

!❓ Feel free to interrupt me with questions or comments at any time!

Remember Zoom's Chat is bad at letting the presenter know if someone asked them something.

- 💬 I might post some stuff on Chat (mainly links)
- 💻 If you haven't installed Git on your computer yet, this might be your last chance:
<https://git-scm.com/download/>



What Git is and what it isn't

"The Merriam-Webster dictionary defines 'git' as..."

The screenshot shows the Merriam-Webster dictionary website. At the top, there's a logo with 'Merriam-Webster' and 'SINCE 1828'. Below the logo, the word 'git' is searched, and the results are displayed. The first result is for the noun definition, which is described as 'a foolish or worthless person'. The second result is for the verb definition, which is described as a 'dialectal variant of GET'.

git noun

Save Word

\ 'git' ⓘ \

Definition of *git* (Entry 1 of 2)

British

: a foolish or worthless person

git

Definition of *git* (Entry 2 of 2)

dialectal variant of [GET](#)

The screenshot shows the Git Manual page for the command `git`. The title is `GIT(1)`. The manual page is titled "Git Manual". It starts with the **NAME** section, which describes `git` as "the stupid content tracker". The **SYNOPSIS** section shows the command-line syntax for `git`, including various options like `--version`, `--help`, and various path specifications. The **DESCRIPTION** section provides a brief overview of Git as a distributed revision control system. It also suggests reading `gittutorial(7)` and `giteveryday(7)` for beginners, and the `Git User's Manual[1]` for more depth. The **OPTIONS** section lists additional command-line options, such as `--version`, which prints the Git suite version. A note at the bottom indicates that the manual page is for the `git(1)` command.

GIT(1)

Git Manual

GIT(1)

NAME

`git - the stupid content tracker`

SYNOPSIS

`git [--version] [--help] [-C <path>] [-c <name>=<value>] [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path] [-p|--paginate|-P|--no-pager] [--no-replace-objects] [--bare] [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>] [--super-prefix=<path>] <command> [<args>]`

DESCRIPTION

Git is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to internals.

See `gittutorial(7)` to get started, then see `giteveryday(7)` for a useful minimum set of commands. The `Git User's Manual[1]` has a more in-depth introduction.

After you mastered the basic concepts, you can come back to this page to learn what commands Git offers. You can learn more about individual Git commands with "`git help command`". `gitcli(7)` manual page gives you an overview of the command-line command syntax.

A formatted and hyperlinked copy of the latest Git documentation can be viewed at
<https://git.github.io/htmldocs/git.html> or
<https://git-scm.com/docs>.

OPTIONS

`--version`
Prints the Git suite version that the `git` program came from.

Manual page `git(1)` line 1 (press h for help or q to quit)

Git is

- A version-control system
- Created in 2005 by Linus Torvalds  (of Linux fame)
- Maintained by Junio Hamano and others
- At the base of a reproducible workflow (Peikert and Brandmaier, 2020)
 - Rmarkdown → Git → Make → Docker
- Super good at handling pure-text files
 - R, Python, MATLAB, Stata files (data analysis scripts, packages, etc.)
 - Whatever you use to write documents (especially L^AT_EX and Markdown)
 - notes
 - papers
 - presentations



Git is *not*

- Just for software developers
- A good place to dump
 - Large files (e.g. data)
 - Files that are generated from other files (e.g. a .pdf from a .tex or .rmd file)
 - GitHub may be an exception when used as a file-hosting page
- GitHub
 - Distributed workflows are a Git feature
 - ...albeit a secondary one. Git's main job is version control



Installing Git (last chance, for real this time)

- General instructions: <https://git-scm.com/download/>
- 🐧 Linux
 - It usually comes pre-installed (verify by running `git --version`)
 - Otherwise, use your distro's package manager (apt, dnf, pacman...)
- 🍏 macOS
 - Open a terminal and type `git --version`
 - Plan B: install Xcode Command Line Tools
- 💻 Windows
 - Download and execute the file on <https://git-scm.com/download/win>



Git is primarily text-based software (just like R!)

- 😊 Perfect if you're looking for speed and/or use your keyboard a lot
- 😢 Not great if you like to use your mouse and/or don't care about speed

...but even if you identify with latter category, you may love using Git!

Graphical interfaces to Git

- RStudio's built-in GUI
 - 😊 Convenient if you only use RStudio
 - 😡 Very limited and freeze-prone
- VSCode's built-in GUI + Git Graph extension (+ R)
 - 😊 Great if you work in many languages (R, C++, Markdown, T_EX) and/or *Github Codespaces*
 - 😡 Bad handling of pulls and merge conflicts (without further extending)
- GitKraken
 - 😊 Very robust and featureful
 - 😡 Adds another app to your workflow
 - Chosen tool for this talk (+ terminal)!



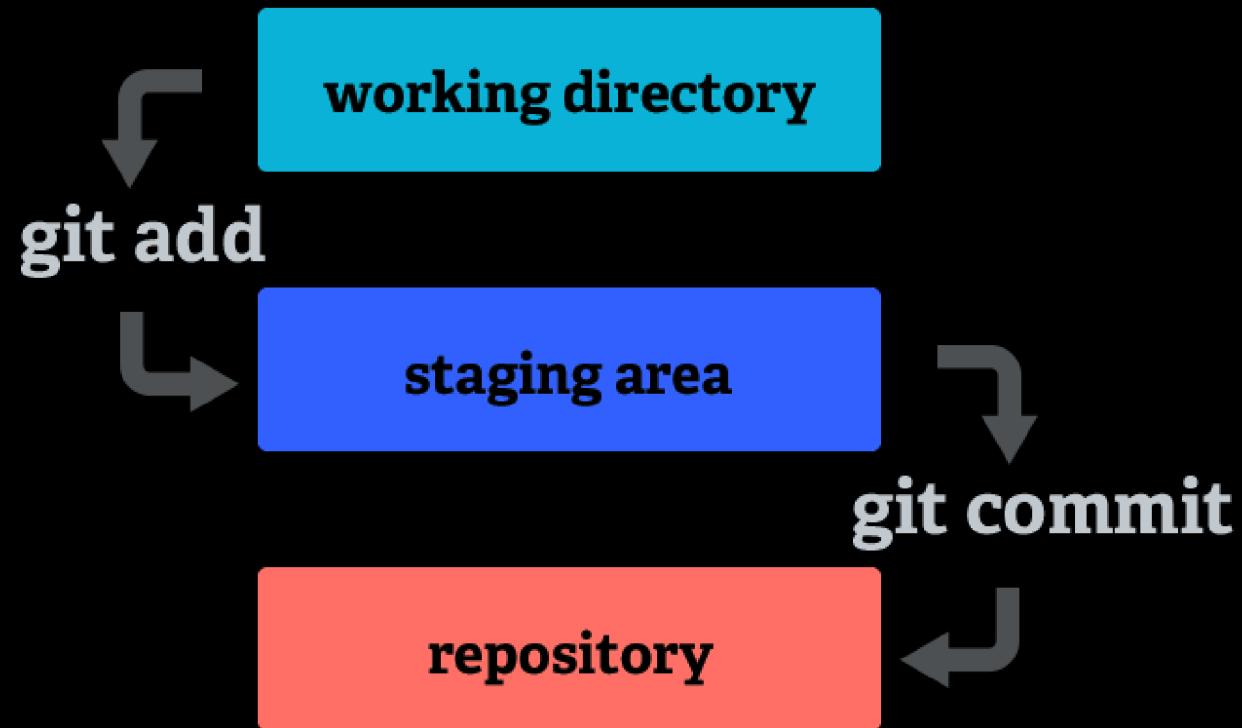
Git lingo, simplified

1. **Repo(sitory)**: A collection of past and present versions of your working directory
 - Locally, a hidden `.git` folder in your working dir
 - May also be hosted online (e.g. GitHub, GitLab, Bitbucket)
2. **Commit**: One particular version/snapshot of your working directory
 - A repo is, at the top level, a collection of commits
 - Basic workflow behavior
 - Save (`ctrl+s`) as much as you want between commits
 - Saves are ignored, only commits matter (unlike Dropbox)
3. **Hash**: a 40-character unique alphanumeric identifier of a commit
 - `a4508a3d857eg282aced33ba75d18ede34fde99f`
 - `a4508a3d857eg282aced33ba75d18ede34fde99f`
 - Don't worry, you won't need to know them by heart



Where on Git are your files?

1. **Working directory**: you see the files in your file explorer
2. **Staging area**: a temporary area where files are gathered and prepared for a commit
3. **Repository**: after the commit, the files are moved from the staging into the repository for permanent storage

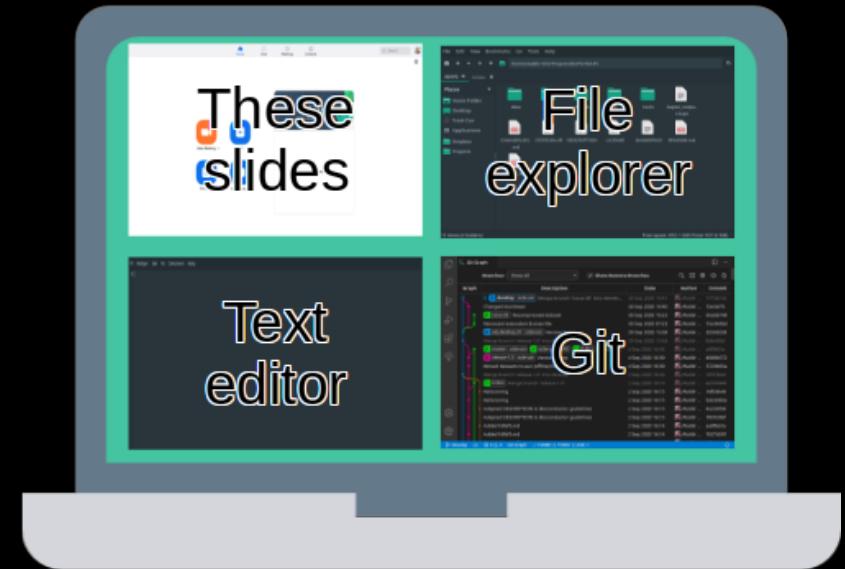


One last thing before we start working

Optimize your workspace to follow this workshop

- **Slides** to follow the workshop instructions
- **Chat/notepad** to socialize and take notes
- **File explorer** to see what's going on with your files
- **Git**, which is the whole point of the workshop

👍 No overlapping window mess
👍 Front-row seats to all the action



Basic Git Commands

(yes, we're *finally* starting for good 🎉)

Configuring your user info

```
git config --global user.name "(name)"  
git config --global user.email "(e-mail)" # GitHub user? use your GH e-mail
```

Initializing a repository

```
git init # within working directory
```



Doing some actual work

1. First do some science



2. Then commit—i.e. register on Git—a version of your science

```
git status  
git add (file)  
git commit -m "(message)"  
git log
```

3. Go make more science



4. Realize what you have done

```
git status  
git diff (file)
```

Taking it up a notch

Undoing changes

```
git reset (file)    # unstages file  
git reset (commit) # undoes all commits thereafter, but keep the changes  
git reset --hard (commit) # go nuclear (i.e., lose changes after commit)
```

Peeking at a file's history of changes (i.e., patches)

```
git log -p (file)  
git show (commit):(file) # commit or tag!  
git show (commit):(file) > (file).copy # retrieve the copy from hash
```

Tagging

```
git tag (tagname) (commit)
```



Git's greatest strength

(in my opinion)

Creating parallel versions of your files

- Think of your files as leaves on the branch of a tree (your repo)
- You can have different versions of the same file on different branches! 😊
- Expect your branches to merge back with the trunk
- Parallel versions: different branches
 - i. Files start the same
 - ii. Differ after some commits
 - iii. Goal: merge back with trunk



Create a branch

```
git branch <branch name>
```

Check out—i.e., switch to—a branch

```
git checkout <branch name>
```

Merging back with main (master) branch

```
git checkout master  
git merge (branch name) # LPT: --no-ff  
git branch -d (branch name) # recommended cleanup
```



Ignoring files

- Datasets
- References/Literature
- Files generated from other files
- Files that will never change

Create a file called `.gitignore` and chuck all that in it.

When the moon hits your eye
like a big pizza pie,
that's amore



When a file you don't need
exceeds 50 mb,
`gitignore`



Example:

```
~/UiO/Projects/contingencytables cat .gitignore
# History files
.Rhistory
.Rapp.history

# Session Data files
.RData

# User-specific files
.Ruserdata

# Example code in package build process
*-Ex.R

# Output files from R CMD build
/* .tar.gz

# Output files from R CMD check
/* .Rcheck/

# RStudio files
.Rproj.user/

# produced vignettes
vignettes/*.html
vignettes/*.pdf

# OAuth2 token, see https://github.com/hadley/httr/releases/tag/v0.3
.httr-oauth

# knitr and R markdown default cache directories
*_cache/
/cache/

# Temporary files created by R markdown
*.utf8.md
*.knit.md

# R Environment Variables
.Renvironment
```



Lightning round?

Stashing and popping

(one "O", two "P"s)

```
git stash  # 1. Sweep a temporary version under the rug  
git pop    # 2. Retrieve it when you switch back
```

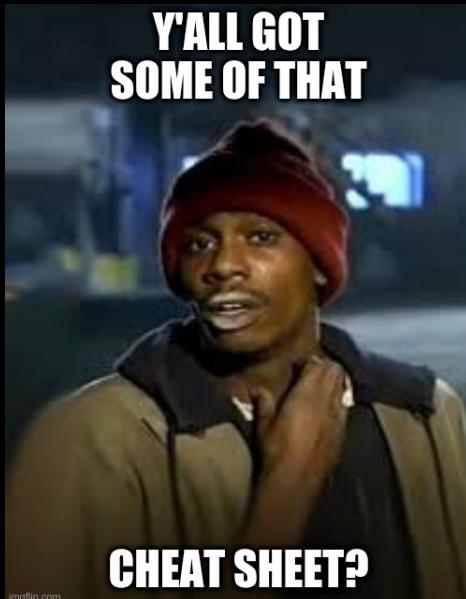
Useful if Git refuses to change branches!

Working with remote repositories (e.g. GitHub, GitLab)

```
git clone  # 1. create a local copy of the remote repo  
git pull   # 1. (if previously cloned, copy latest version of remote)  
           # 2. do your sciency thing  
git push   # 3. Copy your local branch to remote
```

Supplementary material

- Your terminal
 - `git --help`
 - `man git`
- (Very well-curated) official documentation: <https://git-scm.com/doc>



- Yes.



Software used in this presentation

- Marp: Markdown Presentation Ecosystem
<https://marp.app/>
- VSCodium: compiled, telemetry-free VSCode
<https://vscodium.com/>
- GitKraken: Free Git GUI for Windows, Mac, Linux
<https://www.gitkraken.com/>



Reference

Peikert, A., & Brandmaier, A. M. (2019). A Reproducible Data Analysis Workflow with R Markdown, Git, Make, and Docker (p. 45). p. 45. DOI:[10.31234/osf.io/8xzqy](https://doi.org/10.31234/osf.io/8xzqy)

Credits

OS icons made by [Freepik](#) from www.flaticon.com

Lightning photo by Frank Cone from Pexels

