Introduction to Computational Social Science

(Week 3: Open Science and Reproducibility)

Aliakbar Akbaritabar[1]

[1]Max Planck Institute for Demographic Research, Rostock, Germany;
akbaritabar@demogr.mpg.de

Intro to CSS course outline and syllabus

Week 1	Introduction	What is "Computational Social Science"? Inductive and deductive research; Big data revolution
Week 2	Digital Trace Data	Observational data; Available vs. designed data; APIs and web scrapping; Representativeness
Week 3	Open Science and Reproducibility	Reproducibility crisis; Pre-registration; Version control, Git and GitHub
Week 4	Mobility and Migration	Computational approaches to migration research
Week 5	Science of Science	Robert K. Merton; Sociology of scientific knowledge vs Bibliometrics/ Scientometrics
Week 6	Network Analysis	Tie formation mechanisms in social networks; Violence of independence of observations
Week 7	Ethics in Computational Social Science	Informed consent; Personal data; GDPR
Week 8	Mid-semester evaluation	Short essay and multiple choice questions
Week 9	Social Simulation	Agent-based modelling for social scientists; Micro-Macro link and Coleman's boat
Week 10	Text as Data	Natural Language Processing; Topic modelling; Structural Topic Models
Week 11	Machine Learning	Supervised and unsupervised use of observational data; Feature learning
Week 12	Other CSS Skills	Parallelization; Functional vs Object-Oriented; Graph database; DuckDB; SnakeMake Workflow
Week 13	Limitations of Computational Social Science	Pitfalls of digital trace data and computational approaches; Representativeness
Week 14	Conclusions	Thick vs Big data, Survey experiments; Linked data; Future of CSS
Week 15	Final semester evaluation + Student presentations	Short essay and multiple choice questions OR presentation

Who am I?

- Computational Social Scientist with a background in sociology.
- Interest in science of science, scholarly migration, social and collaboration networks.
- Expertise in network analysis and modeling for large-scale and complex social media, bibliometric and textual data.
- Thanks, Sha Jiang, for generously offering to help today with the Git installation step

Practicing open science since 2011: akbaritabar.github.io

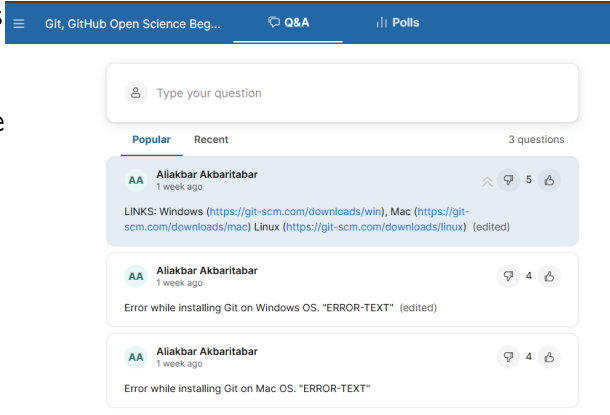
Housekeeping rules

I will not monitor Zoom chat (today's complex setup with two in-person rooms + online + hands-on).

Instead, please use the Q&A (link on the next slide) throughout the course to post your questions.

Check if your question has already been asked by someone else, and upvote it.

Remember to mention your Operating System (Windows/Mac/etc) and the error texts.



Slido Poll and Q&A!

To post your questions in Q&A, share error messages, and answer polls + quiz:

Scan the QR code on the right ⇒

Or visit www.slido.com and enter the number: "1011941".

Or visit: https://app.sli.do/event/thfeKFo975LGRnm1pQpvHR.

Why use plain text and version control?

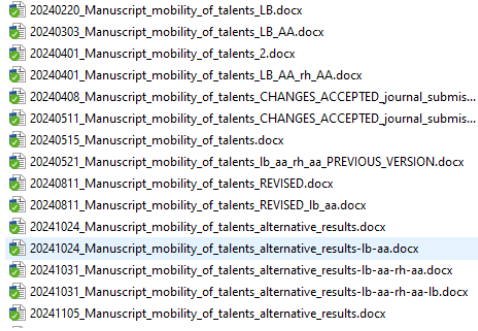
What is wrong with MS Word and binary files, review tab, and tracked changes?

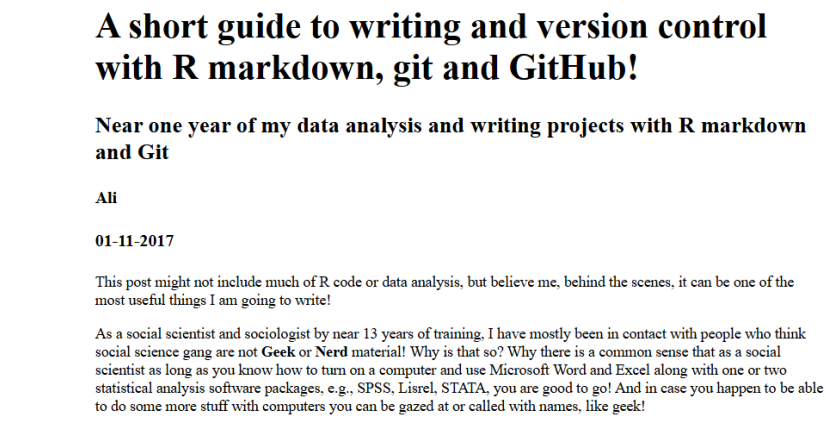
Improve it using name initials and dates

Or use Google Docs or something with history.

You should not be versioning files. Git can do that.

You should focus on the content!





A short guide to writing and version control with R markdown, git and GitHub!

Near one year of my data analysis and writing projects with R markdown and Git

Ali

01-11-2017

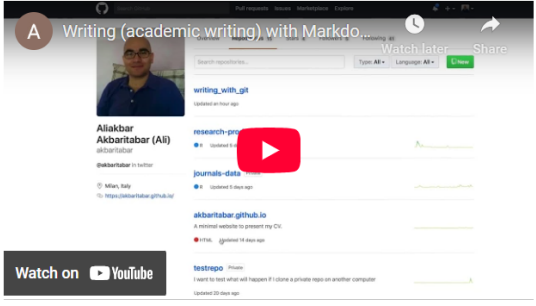
This post might not include much of R code or data analysis, but believe me, behind the scenes, it can be one of the most useful things I am going to write!

As a social scientist and sociologist by near 13 years of training, I have mostly been in contact with people who think social science gang are not **Geek** or **Nerd** material! Why is that so? Why there is a common sense that as a social scientist as long as you know how to turn on a computer and use Microsoft Word and Excel along with one or two statistical analysis software packages, e.g., SPSS, Lisrel, STATA, you are good to go! And in case you happen to be able to do some more stuff with computers you can be gazed at or called with names, like geek!

Videos and blog post:
`https://akbaritabar.github.io/CV_MD/git_github_for_academic_writing.html`.

Preaching this since 2017

OK! If the above description was enough, watch this **20 min** video on simple steps where to start with these tools to live an easier academic life!

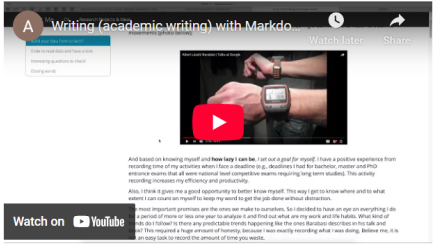


Briefly, for me "Git" is like a huge "Ctrl + Z" that allows me to go back in time to whatever extent that I want and correct my mistakes, isn't that amazing?

Videos and blog post:
`https://akbaritabar.github.io/CV_MD/git_github_for_academic_writing.html`.

Preaching this since 2017

- 1st video topics reviewed: **1:05**
- 2nd vide topics: **2:14**
  - Clone a git repository: **4:00**
  - A real example of R Markdown with R code chunks: **10:42**
  - How to go from RMD report to presentation file: **20:28**
  - Using git branches: **28:24**
  - Using git stashing: **44:47**
  - Review and links to further learning: **56:40**



Links mentioned in 2nd video (advanced topics)

- Learn R markdown formats
- Presentation file of topics in 2nd video
- Github repository of 2nd video

Videos and blog post:
`https://akbaritabar.github.io/CV_MD/git_github_for_academic_writing.html`.

What is Git?

DropBox, Google Drive, NextCloud, etc.

Usually, no history of files.

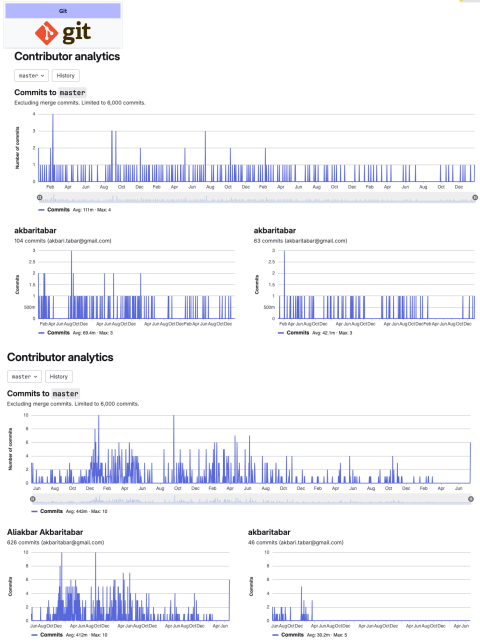
What is Git?

DropBox, Google Drive, NextCloud, etc.

Usually, no history of files.

Git is a tool for keeping a history of changes in your files and project.

Think of it as an endless "Ctrl + Z."



What is Git?

DropBox, Google Drive, NextCloud, etc.

Usually, no history of files.

Git is a tool for keeping a history of changes in your files and project.

Think of it as an endless "Ctrl + Z."

All is backed up in Git if you stage and commit it.

GitLab

Bitbucket

What is Git?

DropBox, Google Drive, NextCloud, etc.

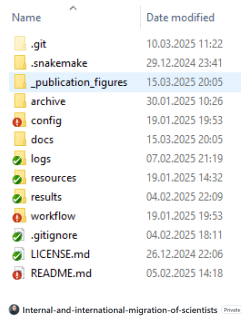
Usually, no history of files.

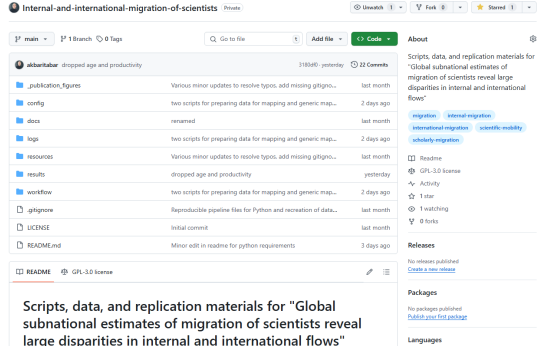
Git is a tool for keeping a history of changes in your files and project.

Think of it as an endless "Ctrl + Z."

All is backed up in Git if you stage and commit it.

Web hosting of Git in GitHub, GitLab, BitBucket, etc., like file sharing with history!



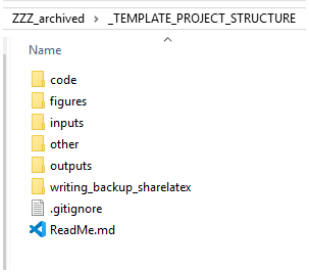


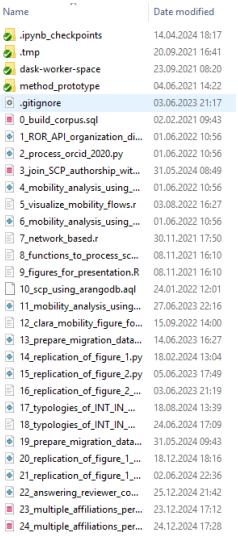
A clear and consistent folder structure

Create a clear structure and stick with it.

This will be (through time) clear to you and your collaborators.

A habit of using numbered scripts or workflow managers.





Hands-on part

Slido Poll and Q&A!

To post your questions in Q&A, share error messages, and answer polls + quiz:

Scan the QR code on the right ⇒

Or visit www.slido.com and enter the number: "1011941".

Or visit: https://app.sli.do/event/thfeKFo975LGRnm1pQpvHR.

Installing Git

See course instructions; Links also in Q&A platform

Install git (No admin rights needed)

OR, download, unzip, and use git portable for Windows

Check by writing "git - -version" (two dashes, without space) and pressing enter.

GitHub (Beginner): Using Git and Github for Open Science & connection with OS repositories for sharing data and codes (OSF and Zenodo)

(Aliakbar Akbaritabar)

Prior knowledge required: None (just being enthusiastic about doing research suffices!)

Technical equipment required: Personal laptop with permission to install software (if you cannot install software, you can follow along on the projector)

ToDo before the workshop: Please download and install Git for Windows (https://git-scm.com/downloads/win), Mac (https://git-scm.com/downloads/mac) or Linux (https://git-scm.com/downloads/linux). Please note: no GUI installation is needed. Only follow the instructions in one of those links, depending on your operating system, and install the Git software. If you were asked for Windows installation to add "context menu shortcut" and "add Git to PATH" please select "yes" for both (selected by default).

Description:

This tutorial assumes no prior knowledge about version control and using Git, GitHub, and similar tools for research. At the end of the tutorial, participants will know what they will gain by using version control and how they can use different version control tools in writing scientific text. The tutorial covers academic writing and analysis scripts, how version control helps, and the benefits of switching to Latex, RMarkdown, and other plain text formats for academic writing. Participants will learn to use Git and GitHub in Terminal (CLI) and popular IDEs such as RStudio and Visual Studio Code. Participants will learn how to deposit replication materials on Open Science Framework (OSF) and Zenodo and share a view-only link for peer review.

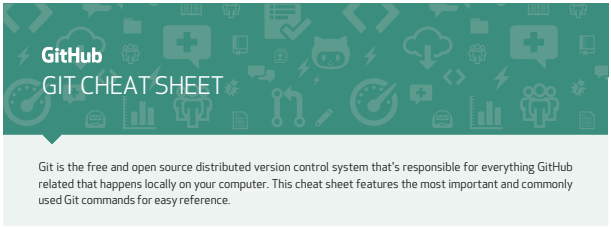
Command line, Terminal, CLI, CMD, Bash, etc.

- Command line is your friend (for me, I make less mistakes than GUIs)
- "pwd" (print working directory, only for Mac/Linux/Bash)
- "dir" (directory, for Windows/Bash)
- "ls" (list, only for Mac/Linux/Bash)
- "cd" (change directory)
- Or easier "open git bash here" in right-click context menu
- "- -argument" and "-arg"
- help COMMAND-NAME
- "git - -version"

Essential Git Commands

Get the git cheat-sheet

GitHub education:
https://education.github.com/git-cheat-sheet-education.pdf.



GitHub

GIT CHEAT SHEET

Git is the free and open source distributed version control system that's responsible for everything GitHub related that happens locally on your computer. This cheat sheet features the most important and commonly used Git commands for easy reference.

INSTALLATION & GUIS

With platform specific installers for Git, GitHub also provides the ease of staying up-to-date with the latest releases of the command line tool while providing a graphical user interface for day-to-day interaction, review, and repository synchronization.

GitHub for Windows
https://windows.github.com

GitHub for Mac
https://mac.github.com

For Linux and Solaris platforms, the latest release is available on the official Git web site.

Git for All Platforms
http://git-scm.com

SETUP

Configuring user information used across all local repositories

`git config --global user.name "[firstname lastname]"`
set a name that is identifiable for credit when review version history

`git config --global user.email "[valid-email]"`
set an email address that will be associated with each history marker

`git config --global color.ui auto`
set automatic command line coloring for Git for easy reviewing

STAGE & SNAPSHOT

Working with snapshots and the Git staging area

`git status`
show modified files in working directory, staged for your next commit

`git add [file]`
add a file as it looks now to your next commit (stage)

`git reset [file]`
unstage a file while retaining the changes in working directory

`git diff`
diff of what is changed but not staged

`git diff --staged`
diff of what is staged but not yet committed

`git commit -m "[descriptive message]"`
commit your staged content as a new commit snapshot

BRANCH & MERGE

Isolating work in branches, changing context, and integrating changes

`git branch`
list your branches. a * will appear next to the currently active branch

`git branch [branch-name]`

git config - -list

git config

git config - -global user.name “[firstname lastname]”

git config

git config - -global user.email “[valid-email]”

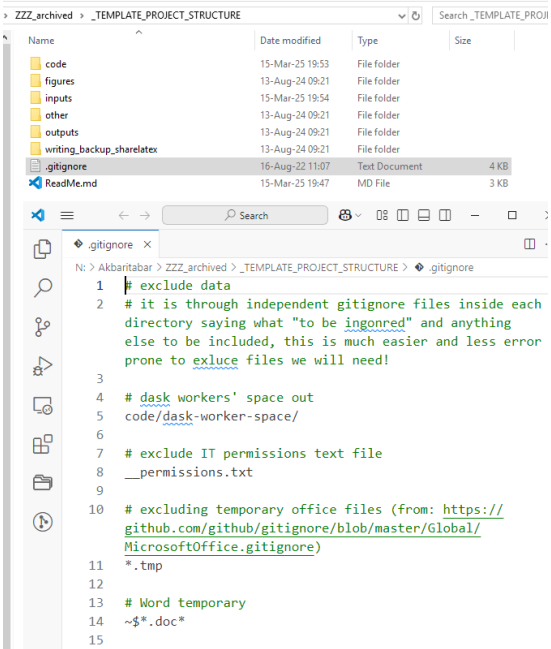
git init

Initiates a new repository in the local directory and tells git to start watching this directory.

Git .ignore file

Defines things to be ignored.
Can have one in the template folder structure and copy/paste and reuse and include a customized one in sub-folder

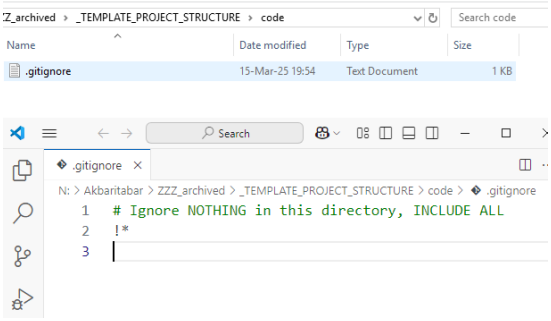
"logs/" ; "*.notes" ; "pattern*/"



Git .ignore file

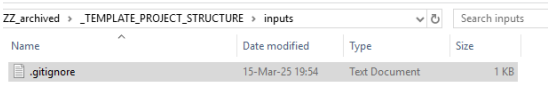
Defines things to be ignored.
Can have one in the template folder structure and copy/paste and reuse and include a customized one in sub-folder

"logs/" ; "*.notes" ; "pattern*/"



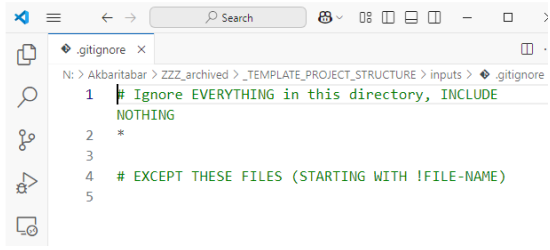
Codes folder, include everything.

Git .ignore file



Defines things to be ignored.
Can have one in the template folder structure and copy/paste and reuse and include a customized one in sub-folder

"logs/" ; "*.notes" ; "pattern*/"



Inputs folder, exclude everything EXCEPT some.

git status

Reports the status of the working directory/files.

git add [file]

Adds the selected file (or pattern of files) to the staging area to commit to history (git, watch these files).

git commit -m “[descriptive message]”

Writes the staged files to history.

git log

Shows the history.

git diff - -color-words [file(s)]

Similar to MS Word's review tab, it shows word-level changes in the staging area (not yet committed; for committed, add "- -cached").

git branch

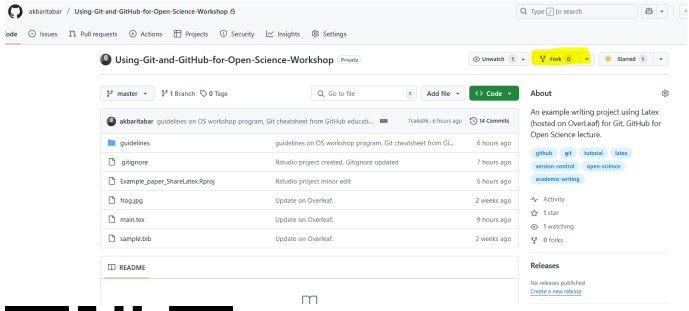
For experiments that might be thrown away. Allows branching off of the main trunk of history (that can later be merged back into main history, more advanced for now).

Fork a repository on GitHub

GitHub, bring a copy of this repository and its history, and store it "on my GitHub profile."

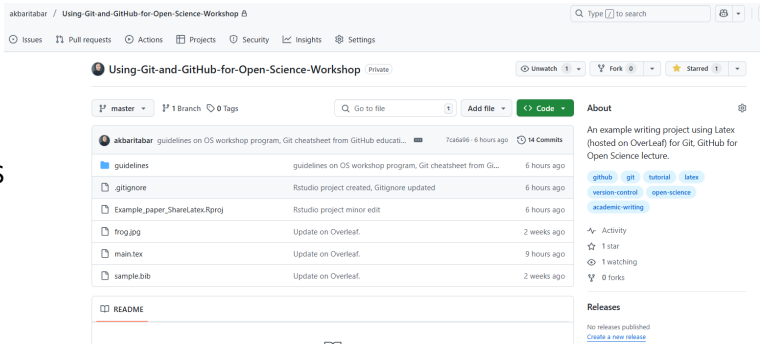
BTW, keep it linked to the original repository!

GitHub repository:
tinyurl.com/
osgithub25



git clone [url]

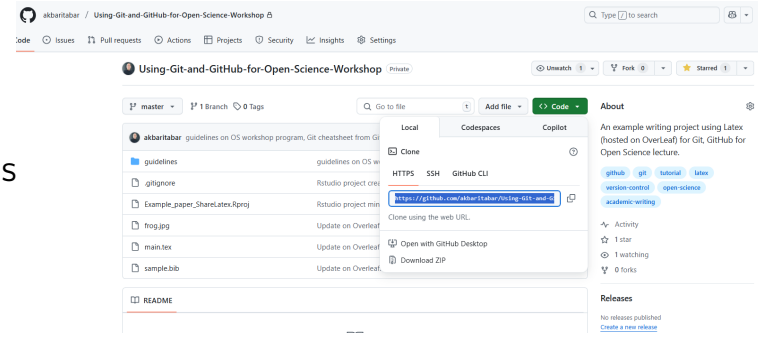
Git, bring a copy of this repository and its history, and store it here "on my computer".



Link: `tinyurl.com/osgithub25`

git clone [url]

Git, bring a copy of this repository and its history, and store it here "on my computer".



Link: `tinyurl.com/osgithub25`

git remote -v

Git, tell me where I have cloned this repository from.

git push

Git, I have made some changes in files locally, I added and committed them. My git status is clean. Send these changes to the remote, where I cloned it from (if I have write/push rights).

git pull

Git, I want to work on this local repository, but before that, I want to have the most updated version of it, in case a collaborator has made changes, "bring (pull) those changes here".

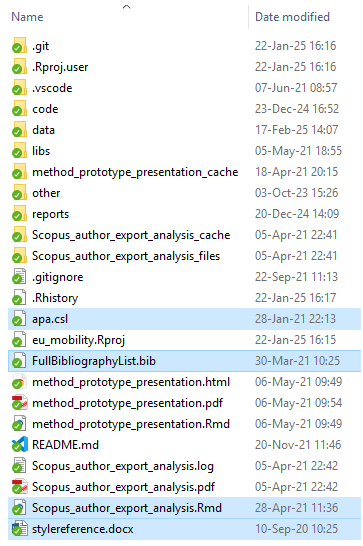
Git and plain text for writing/coding

Writing using plain text

RMarkdown (R code in the same file)

Issues with collaborators' wishes to compile

Latex, OverLeaf, and ShareLatex



Writing using plain text

RMarkdown (R code in the same file)

Issues with collaborators' wishes to compile

Latex, OverLeaf, and ShareLatex

Our example repository + anonymous editing)
tinyurl.com/latexex25

Git remote -v and multiple push remotes

Backup of ShareLatex/OverLeaf on private GitHub repository (credit: https://jigarius.com/blog/multiple-git-remote-repositories)

1. git remote -v (to use later)
2. git remote set-url - -add - -push origin https://github.com/REPO-URL-HERE.git
3. (add sharelatex push URL again) git remote set-url - -add - -push origin https://git@sharelatex.gwdg.de/git/code-to-take
4. git remote -v (to check all is well? should show 3 URLs)
5. git pull - -all
6. git push - -all (wait, why did it give 2 outputs?! ;))
7. Everything up-to-date
8. Everything up-to-date

Git in popular IDEs

Git in RStudio

Note: choose your IDE as if you are choosing a lifetime companion

My advice: Use Git and GitHub in Terminal (CLI)

For RStudio:

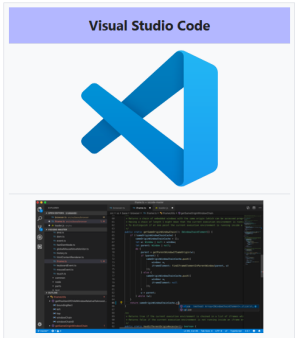
1. Add git executable path in global options under version control
2. Create a project
3. Open terminal
4. Or, open git tab/pane

RStudio auf Ubuntu 12.10

Git in Visual Studio Code

Highly recommended IDE with a lot of flexibility and multilingual!

1. Install VS Code
2. It should already be added to the user's PATH
3. Open Git Bash
4. "cd" to the project folder
5. Or right-click and select "open git bash here"
6. Write "code ." and press enter
7. The VS Code terminal will already inherit git
8. Check it with "git - -version"
9. It works even with the portable git
10. Check out extensions such as "Git Graph"



A word on Jupyter project, Jupyter notebooks, Jupyter Lab, version control, and using them with plain scripts

Reproducibility, OSF and Zenodo

Open Science Framework (OSF) + Zenodo

Deposit replication materials on Open Science Framework (OSF)

Share a view-only link for peer review (Credit: `https://help.osf.io/article/201-create-a-view-only-link-for-a-project`).

How to create a release in GitHub and publish it on Zenodo.

Check also: `https://github.com/dgraziotin/disclose-data-dbr-first-then-opendata`

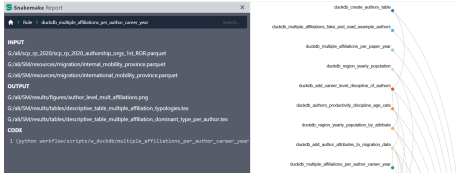
More?

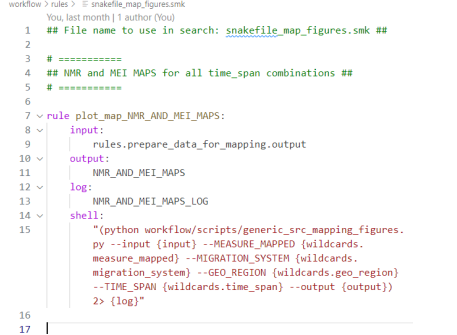
Where to go next?

- https://github.blog/developer-skills/programming-languages-and-frameworks/what-is-git-our-beginners-guide-to-version-control/
- https://skills.github.com/
- https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

More advanced topics

- Usable for "academic writing using plain text"
- "Coding for analysis etc"
- "Workflow management using SnakeMake" + Targets in R
- GitHub pages, shell script to automate a personal website? paper introduction?
- Example: `https://akbaritabar.github.io` that uses a shell script and markdown plain text files
- GitHub CodeSpace (push "." on a repository)
- GitHub actions to automate recurring works
- Beamer and presentations in plain text and LaTeX (even these slides are version-controlled)
- Latexdiff Perl script for tracked changes in the Revise and Resubmit stage of publication





Quiz time!

Slido Poll and Q&A!

To post your questions in Q&A, share error messages, and answer polls + quiz:

Scan the QR code on the right ⇒

Or visit www.slido.com and enter the number: "1011941".

Or visit: https://app.sli.do/event/thfeKFo975LGRnm1pQpvHR.

Thanks for your attention!

Questions and comments are welcome!

Aliakbar Akbaritabar

Research Scientist

Max Planck Institute for Demographic Research (MPIDR), Rostock, Germany.

Email: akbaritabar@demogr.mpg.de

LinkedIn/BlueSky/Twitter/Mastodon: @akbaritabar

Credit and URLs for logos and photos

- https://de.m.wikipedia.org/wiki/Git
- https://de.m.wikipedia.org/wiki/Datei:Dropbox_Logo_01.svg
- https://de.m.wikipedia.org/wiki/Google_Drive
- https://de.m.wikipedia.org/wiki/Bitbucket
- https://de.m.wikipedia.org/wiki/GitLab
- https://de.m.wikipedia.org/wiki/GitHub
- https://de.m.wikipedia.org/wiki/RStudio
- https://de.m.wikipedia.org/wiki/Visual_Studio_Code
- https://de.m.wikipedia.org/wiki/Overleaf