Scientific Writing, Documentation, and Presenting

Markdown, Latex, and Git(Hub)

Alexander Heimann

NCKU - Department of Geomatics

February 7, 2022

Table of Contents

Organisation of this presentation I

Syllabus:

- What is LaTeX
- What is JabRef
- What are Git and GitHub
- Basic commands of Git and LaTeX
- Quick demonstration of JabRef
- How to integrate Git into a project
- How upload and share your project via GitHub
- How to use LaTeXtemplates
- Tex editors, Git visualisation

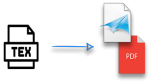
At the end of this session you will be able to write notes in the Markdown markup language, create a LaTeX file + PDF and track changes & share it with others via Git(Hub).

Software Explained

- "LaTeX, which is pronounced "Lah-tech" or "Lay-tech", is a document preparation system for high-quality typesetting. It is most often used for medium-to-large technical or scientific documents but it can be used for almost any form of publishing" [1]
  Simplified: it is a kind of programming language, that needs a compiler (TexLive or MikTex), and produces PDFs.
- Git it a version control software. It lets you create snapshots of your projects (coding, documentation, thesis, etc.)
- JabRef is a reference management software that creates .bib files, which we use in LaTeX. It can also create XML files that can be used in Microsoft Office Word.
- GitHub/BitBucket are provider of Internet hosting for software development and version control using Git.

LaTeX Details

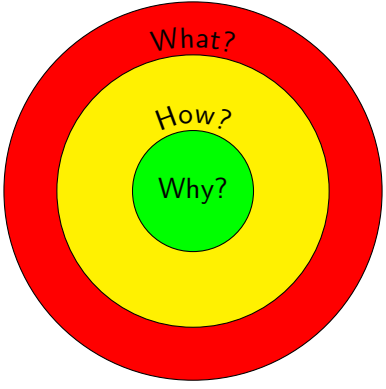
LaTeX is a markup language. It means that the text includes commands that are then interpreted by the compiler. For example, if I want to write a word in **bold** I would use the command `\textbf{text in bold}`. It gives me maximum flexibility, however it creates slightly more writing. Latex files typically use the ending .tex. Tex files are text files (such as txt) and can be opened and edited as such. For libraries, latex usually uses .bib files, standing for bibliography. An example will be shown later. In order to use latex we need to install a compiler, either MikTex or TexLive (more information under installations)



Git Details

Git creates snapshots of the current files. That means that every time I want, I can save the current version including a message, to describe what has been changed in this/these file/files in this version. That is good when working in parallel with other people, when working from different locations, and when I want to have 'backups' of my files. I can create branches, meaning that I can make copies of the current files and work on them independently, without having dozens of different folders and files, including the mess of naming them all different by version number. Git needs to be installed in order for us to use the commands. GUI software exists that allow us to use git without command line interface/terminal, but with a press of a button. A link to a list of GUIs are at the end of this presentation.
GitHub is a hosting service for sharing documents and work files.

Overview Latex



- **Why: Purpose/Cause/Believe**
  - You want to communicate your research in the most beautiful and simple way.
  - You want to focus on writing and less on editing.
  - The text is the most important thing.
- **How:**
  - Separate content from how the end-product will look like.
  - Write the text independent of the document format.
  - Change format after text written.
- **What:**
  - Latex lets you keep your text/content untouched and design the format around it.

Personal Reasons

Why do I personally use LaTeX?

1. It keeps me closer to coding (less mouse, more keyboard)
2. I like to re-use and standardize, meaning: once I have a good looking document, I will just copy and fill with new text.
3. I use Git because I experienced file loss before and it helps me have copies of different versions.
4. Back in Germany we had to learn LaTeX, as we were told: all scientist use LaTeX
5. Because I use VSCode for programming, I checked and it was easy to also use it with Git and LaTeX, so I can stay within the same eco-system.

Personal side-note: While I see it as an advantage to have the option and flexibility to choose my own software and framework, for others it's complex and they prefer one standardized software such as WORD, ArcGIS, etc.
LaTeX has a steep learning curve and Git(Hub), when used without care, only creates a mess. It is not difficult, but both need time AND practice, which not everyone can put into. If you have the time to learn it and you see some benefit in it for yourself: learn LaTeX and use Git(Hub).

What is LaTeX I

LaTeX

- Software System for document preparation
- Using plain text as opposed to WYSIWYG (MS Word, LibreOffice Writer.)
- Typeset, such as markdown

Word	LaTeX
Combine content and formatting	Separate content from formatting
Implicit semantic	Explicit semantics
Pro: WYSIWYG	Full control
Con: Difficult to change	Pro: Change content or formatting independently
Con: Limited Control	Con: Slightly more typing

When is it Useful to Use LaTeX and When Not I

Reasons why you must, should, and should not use LaTeX:

When you MUST use LaTeX?

- When you are in academia, especially in any STEM discipline.
- You work with abundant bibliography.
- You are using formulas.
- You expect figures using the best quality possible.
- You want a free solution.
- Forget about document layout.

When is it Useful to Use LaTeX and When Not II

Should

- You want your documents to stand out.
- You are considering writing a book, article, or manuscript.

Should not

- Your document is already written in another format.
- You are doing collaborative work and you are the only LaTeX practitioner.
- The layout of your document means everything to you.

When to use LaTeX or Word?

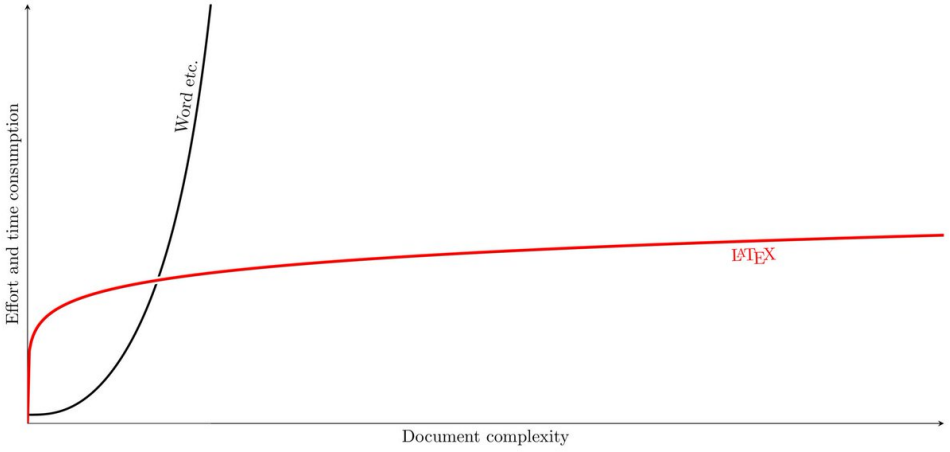


Figure 1: LaTeX vs. Word

Git(Hub): Why, How, What

Why

If you are updating your LaTeX code, code-base, and/or working with others then it is crucial to be able to track changes and roll back.
You want to be able to **merge** files of different authors.
You want to have the files online, accessible from everywhere for everyone involved.

How

Every update of the code/text gets a name (commit message) and identifier. In a history one can look up who changed what and when. Each version can be switched to.

What

Source control with Git, and an online storage ("hub") with GitHub.

What is Git(Hub) I

Git(Hub)

- Git is a version control system.
- GitHub is a hub: a place to store and share online

Think of Git as a programming language you need to install first before you can call any command or function of it. GitHub is like a webstorage where we can upload our repository. GitHub, Bitbucket, ... offer limited amount of storage (usually 2GB). This is not much, but plenty when you consider what Git(Hub) is for: it is for sharing projects and code. It is not for sharing files and data. You usually store your programming files, such as your Matlab, Python, C++, ..., and LaTeX, but you wouldn't store las files, too many images there.

Git vs. GitHub

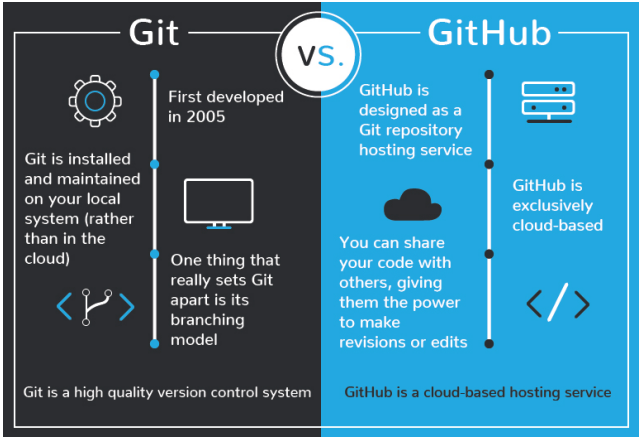


Figure 2: Difference between Git and GitHub, [2]

Compatibility

"But Alexander, what about all my old documents? Are they compatible? Can I convert them to LaTeX??? "

Fear not!

There is software where you can (partially) convert it to LaTeX and LaTeX system files. It is a plug in for Word, it will save your images too, but doesn't deal with libraries. Link to the Word plug-in: GrindEQ

"But Alexander, why would I use PDF over Word?"

PDF has many features, you might not have heard of.

Since PDF supports vector graphics[VG] and LaTeX can not only use but also generate VG diagrams, it is perfect for best visualisation. Besides, PDF can also use effects/animations, although limited.

"But Alexander, why I don't just use Word with any source control?" **Because it only works with 'text' based files such as .txt, .cpp, .py, ..., and of course .tex and .bib**

LaTeX/JabRef: Sofware Needed, Options

Software

LaTeX files are with the ending .tex. Tex files are text files, you can create, open, edit them with any text editor for example notepad. But we need a compiler, to create PDFs out of the .tex files. After we installed the LaTeX compiler, we can also install a Tex-Editor such as TexMaker (I recommend it) that will help us editing, including snippets, auto formatting, helping us with compiling and viewing. **JabRef** is a good reference manager. It is a good way to store your bibliography.

Installation

- There are two compilers (called distributions). Install one: Miktex and Tex Live (I recommend TexLive)
- Install the editor of your choice, see a list of best latex editors or 12 Best LaTeX Editiors You Should Use
- JabRef can be downloaded from the JabRef homepage

Git/GitHub: Software Needed, Options

Software

Git is a version control software. We need to install it to use its commands. We can then install a GUI if we want to have a more convenient way of using Git.
GitHub and BitBucket are code repositories. They don't need to be installed but you need to create a user account in order to access them and use their online storage.

Installation

Install git for your system from here (includes information about how and what to install for git and github, GUI, etc.): Github webpage Two popular online repositories are GitHub and Bitbucket. Both are free.
Download whichever you want, both work the same way.

MarkDown: Software Needed, Options

Software

Markdown (often just called MD) is a text-to-HTML conversion tool. It is used to write articles and blog posts but can also be used for scientific writing and note taking. Just like LaTeX, Markdown is a text-based language. You can already create, open, or edit any .md file. You can also use Markdown to create .pdf files.

Installation

If you want to convert MD files to PDF you need to install Pandoc
A good Webpage explaining how to do scientific writing with Markdown including editors for Win/macOS/Linux, check out Jaan Tollander De Balsch's Post.

Other Options

What do I personally use?
I installed Git and TexLive.

Which GUI do I use?
I use **VSCode**.
VSCode offers language support for LaTeX, LaTeX extensions that have shortcuts and snippets. I also use Git with VSCode for the same reason: extensions that allow me to use and visualise my repo without using command line interface (terminal).

If you already use VSCode or VisualStudio, you only need to install git and TexLive and you are ready to go. I will demonstrate Git and LaTeX later on using VSCode, both: command line as well as the extensions.

Feel free to ask me about how to set up LaTeX and Git on your system.

Tipps, Tricks, and Good Habbits

Remember: LaTeX and Git work together, even better if we get sure to follow some basics. LaTeX ignores leading and trailing line breaks and spaces, therefore:

- Only one sentence per line (also to help with Git)
- Focus on producing text first
- Add figures, but care about labels and captures first
- Keep your library updated

Terminology I

LaTeX (fig. 3):

- package : packages are installed already with TexLive but we need to 'call' them in order to use their functions
- TeX or tex file: when talking about tex or tex-file (tech or tech-file) we talk about files with the ending tex, which we need to open with a text editor
- Preamble: document layout setting
- body: content of the file
- top/front matter: beginning of a document (title page, abstract, toc, ...)
- main matter: main content part
- back matter: ending part of a document (references, etc.)

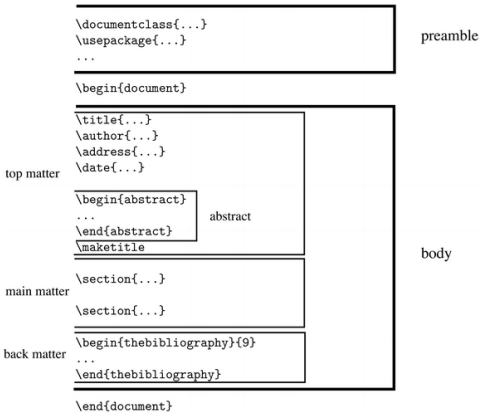


Figure 3: Parts of a TeX file

Terminology II

Git:

- snapshot: every time you commit, or save the state of your project, Git basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot.
- modified: means that you have changed the file but have not committed it yet.
- stage: means that you have marked a modified file in its current version to go into your next commit snapshot.
- commit: means that the data is safely stored in your local database.
- checkout: means that you go to a snapshot and see it in that state
- repository: or "repo" for short, is a storage location. Local repo is on your pc, remote repo means for example GitHub.
- remote: storage not on the computer.

Terminology III

- local: means on my computer
- origin: where the project is originally from. Usually the online version (on GitHub) is the origin
- branch: can be a new version of a repository, experimental changes, or personal forks of a repository for users to alter and test changes.
- fork: a branch can have some files of the main branch. A fork is a copy of another branch and has all the files of it.
- main: primary branch of the repository
- head: the most current commit of the repository in which you are working. When you add a new commit, HEAD will then become that new commit.
- push: updates a remote branch with the commits made to the current branch. You are literally "pushing" your changes onto the remote.

Terminology IV

- pull: reverse to push. Updates local branch by downloading from the remote branch.

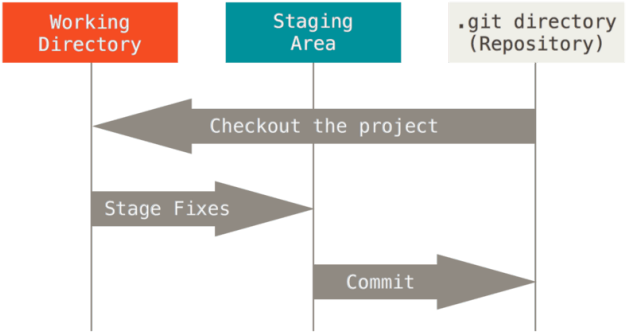


Figure 4: Three Stage Architecture: Working Directory, Staging Area, Repository

Common GIT Commands I

Following are git commands that can be used in a terminal (command line) but GUIs use the same terminology (Note: the $< >$ are not to be typed but what's between them. **Important**: Branch names longer than one word can use the following separators: `_`, `-`, `.` for example: `angela.merkel_feature_new-experimental-changes`):

Git Command	**Explanation**

Common GIT Commands II

git init	initialize git for that folder and sub-folder. Starts the tracking
it config –global user.name <name>	Set your username so the commit messages show who did them. Here no need to use separator.
it config –global user.email <email>	Set your email address
git status	shows the status of the work directory such as which files have been modified
git branch	shows all local branches (the one with the is the one that is active)

Common GIT Commands III

git branch –all	shows all branches, local and remote
git branch <name>	creates branch with name "name"
git checkout <name>	switch to branch name
git branch -d <name	deletes branch with name "name"
git branch -m <new name>	rename the current branch to new name
git push origin -u <new name>	push the <new name> local branch and reset the upstream branch
git push origin –delete <old name>	delete the <old name> remote branch

Common GIT Commands IV

git add .	adds all files in current repo to stage (staging)
git reset	unstage all staged files, but doesn't delete your current files
git reset –hard/soft HEAD~1	reset with/without keeping changes to last commit
git commit -m "message"	commits all staged files with message "message"
git revert	Create new commit that undoes all of the changes done in the previous commit
git log -5	show log of last 5 commits

Common GIT Commands V

git log –oneline –graph	show only ID and commit message and graphs
git diff	Show difference between working directory and last commit.
git pull	download and integrate changes to repo and local files
git push	upload and integrate changes to remote
git remote add origin <repo url>	(connect repo to remote (ex. GitHub))
git clone <repo-url>	Clone repo located at <repo> onto local machine.

Common GIT Commands VI

git merge <branch name>	merge <branch name> into current branch

Common GIT Commands VII

Git Reset

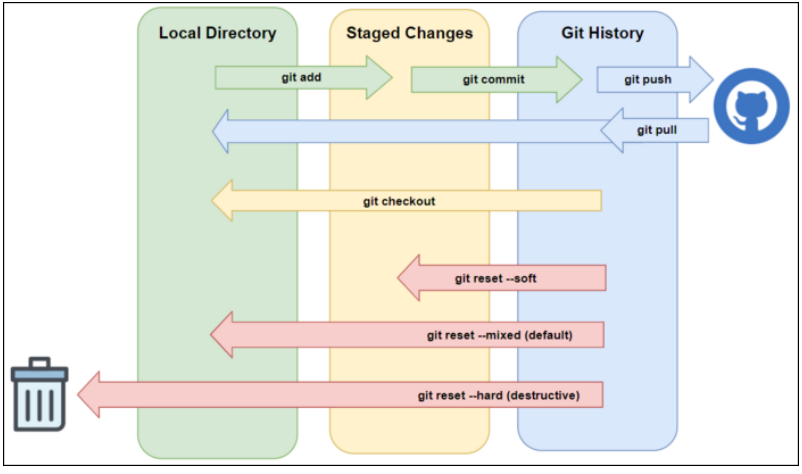


Figure 5: Basic Git creset visualisation

Git Stages - Remote

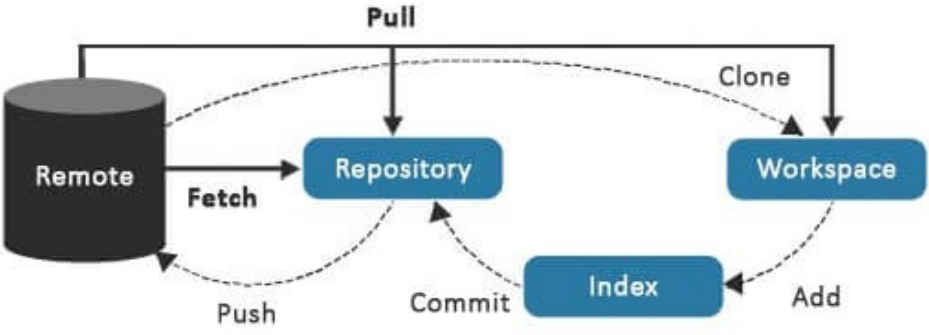


Figure 6: Basic Git commands visualisation

Commit Messages I

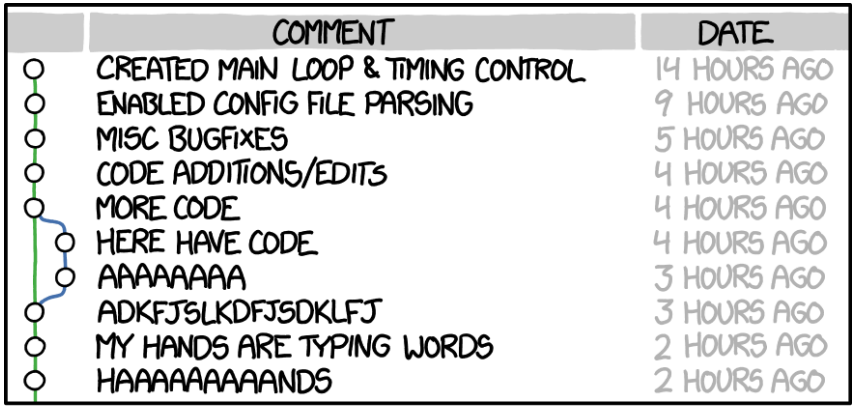


Figure 7: Commit Messages Disaster

Commit Messages II

- feat: (new feature for the user, not a new feature for build script)
- fix: (bug fix for the user, not a fix to a build script)
- docs: (changes to the documentation)
- style: (formatting, missing semi colons, etc; no production code change)
- refactor: (refactoring production code, eg. renaming a variable)
- test: (adding missing tests, refactoring tests; no production code change)
- chore: (updating grunt tasks etc; no production code change)
- add, create, remove, update, ... just stay consistent

It is a convention that the first line is 50 characters or less. Then a blank line. Remaining text should be wrapped at 72 characters. I usually don't use more than one line (less than 50 characters).

LaTeX Basics

The simplest document has at least 3 parts:

```
\documentclass[a4paper]{article}
\begin{document}
\end{document}
```

We start with the class, such as article, book, beamer (classes examples, illustrations, etc.) If we wanted to add additional features, we would put them below the document class, before the '`\begin{document}`'
Then we start the document and end the document.
Everything between begin and end is the actual document.
A LaTeX Cheat Sheet can be found here

LaTeX Example

Example, showing headers and loading of a package:

```
\documentclass[a4paper]{article} % article class, paper size A4
\usepackage{lipsum}              % add package for dummy texts

\author{Alexander}               % author of the document
\title{Thesis}                   % title/Name of the document

\begin{document}                 % start document

\maketitle                         % Create title page
\pagebreak                         % whatever comes next: be on the next page
\tableofcontents                   % create a table of contents
\pagebreak                         % see above

\section*{Epilogue}              % unnumbered header (*) lvl 1
\lipsum[1]                         % create some dummy text

\section{Introduction}           % numbered header lvl 1
\lipsum[2-4]

\section{Methodology}            % numbered header lvl 1
\lipsum[1-3]

\subsection{First subsection}    % numbered header under previous header, lvl 1.1
\lipsum[1]

\end{document}                   % end of document
```

Demonstration

Now it is time to start.
I assume you have installed the software already and created accounts if necessary.
Now I will demonstrate the following steps:

1. Create a repository locally
2. Link local repo to remote (remote means in this case GitHub)
3. add files, edit files, etc. by using tex, bib, and md

Git Basics

I will perform the following steps:

- Create a local folder to test git
- create a txt file and a docx file
- terminal: initialize git, add files, commit, status, log,
- GUI: use Git GUI
- change files: diff, add files, reset, commit, soft reset(HEAD-1), commit, hard reset, revert
- make branches: branch (display all branches), checkout (checkout branch), change files in other branch, merge, change file in both branches, merge
- Now for the folder of the demo: initialize git and add files
- Explain .gitignore and README.md
- how to start github repository and how to link them
- git push, pull, clone, etc.

LaTeX Basics

I have already created two very basic example LaTeX files. First one is called 'article_temp.tex', which is a template for a LaTeX article document. The second one is called 'beamer_temp.tex', which is a template for a LaTeX beamer document.

- I will open them and explain the basics of LaTeX
- I will show you how to use JabRef and how to cite a paper in LaTex.
- If time allows, I will also use git to go along with it

Downloading Templates

One of the best parts of LaTeX is that you can write your text independent from the final layout. You can download a template and just include your text as input file.
Example: Download the IEEE template from here

Detaching Text From Format

Based on the previous slide, we will now try to add our own text, but this time without writing into the file, but by loading out text from another file:

- Create chpt1.tex and chpt2.tex file into the main folder.
- Add some test text to these files
- Create or open the main.tex file (i.e. the IEEE template).
- Add the two lines '`\input{ch1.tex}`' and '`\input{ch2.tex}`'
- Safe and compile.
- Be amazed!
- Edit as you like.

Workflow

How to integrate existing projects into git:

- initialize git in your working directory
- add files you want to track, exclude the ones you don't want to be tracked
- whenever you update/change/add/delete a file, commit the changes
- keep your library updated

Demonstration: Step by Step I

These are the steps in the video:

I. Explain the tutorial, explained software and commands
II. Explain files in the folder ('example' files will be under templates)
III. Initialize git, add all files, show status, log, and branch
IV. Explain .gitignore file
V. Start editing example article (in VSCode), explain functions
VI. Show how to use JabRef to search for papers or files, rename files (/PDFs/sensors-19-00172.pdf), copy bibtex from online, and citation keys and other functions of JabRef
VII. Start editing example beamer class (in VSCode)
VIII. Use TexMaker to edit beamer class.
IX. Create repository on GitHub

Demonstration: Step by Step II

X. Link local repo to remote (on GitHub)

XI. Edit the README.md file

XII. Demonstrate how to use IEEE or Nature templates

Test: Prerequisits

Test

A few tasks to test your knowledge.

Purpose: Demonstrate that you have learned the basics of Git, LaTeX, JabRef, GitHub and Markdown

Prerequisite: You have installed LiveTex, Git, JabRef, and have made a user account on GitHub (or BitBucket). It also works if you have no user account and don't want to have one, but then I can't check.

Knowledge Required: 1. Clone a Repository, 2. Create and edit LaTeX files, 3. Compile LaTeX files, 4. Create and edit a library with JabRef, 5. Use Git for version control, 6. (Optional) Upload the result to my GitHub, your GitHub (great, show me your results when you are done)

Test: Steps I

Note: Read everything before you do anything!

1. Clone the repository at https://github.com/A-Heimann/ArsemTest.git
2. Change to branch: homework
3. Copy the 3 files (papers.bib, `test_article.tex`, `test_presentation.tex`) into a folder `submit/yourname`, example: `submit/alexander/`
4. Open the 'papers.bib' file with JabRef, search for the following paper within JabRef: Hone-Jau Chu, Chi-Kuei Wang "Integration of full-waveform LiDAR and hyperspectral data to enhance tea and areca classification"
5. Change the citation key to Wang2016
6. Open the `test_article.tex` file, create a basic *article* file, A4, with: you as author, a title of your choice, a title page, a separated table of contents, a dummy text, a citation to the paper mentioned above, a separated reference list

Test: Steps II

7. Open the `test_presentation.tex` and create a *beamer* presentation including author, title, titlepage, citation, references
8. In the beamer presentation add: one numbered AND one unnumbered list, in one of the lists add pauses, one image that is upside down (the original image should be normal, you should make it upside down in latex), change the theme and color to whatever you like
9. When you are done with it, create a simple README.md file in your folder containing a title and another header, describe which color and theme you have chosen for your beamer presentation.

Test: Steps III

10. Did you notice I didn't mention anymore Git? You should have made several commits during the progress with meaningful but short commit messages. After you are done and you don't have GitHub, show me your result. The same for if you choose to push your changes to your own GitHub. If you don't want to use GitHub (or BitBucket) show me your result and the git log on your computer.

Resources I

Git:

- List of Git GUI Clients
- List of MOST COMMON Git commands, meaning, example
- More git commands, also well explained
- More complete list of commands, simple explanation
- .gitignore templates
- PDF: Git cheat sheet with description
- PDF: Excellent git cheat sheet, shorter
- GIT terminology explanation.
- Very well explained git cheat sheet
- How to merge Word files with GIT (with doc2txt)
- How to merge Word files with GIT (using Pandoc)

Resources II

LaTeX:

- List of 14 LaTeX editors
- A good LaTeX tutorial page with examples (Including Beamer). This page is like the Wiki for LaTeX. Includes infos about packages, beamer, etc.
- Youtube Video Writing your Thesis in LaTeX in 30 minutes
- Latex Templates such as resumes, posters, etc.
- Beamer Themes and Colors
- Non-standard Beamer Templates
- VSCode Extension: LaTeX Workshop
- VSCode Extension: LaTeX Workshop - Keyboard Shortcuts
- An Online Latex Table Generator

JabRef:

- JabRef Homepage

Resources III

Markdown:

- Basic Syntac including Dos and Don'ts
- Markdown Cheat Sheet

References

[1] T. L. Project, "Introduction to latex," (2022), [Online]. Available: https://www.latex-project.org/about/.

[2] D. Mountain, "Git vs. github: What's the difference?" (2022), [Online]. Available: https://blog.devmountain.com/git-vs-github-whats-the-difference/.