Project template

August 4, 2023

Abstract

This is a short guide to help you setup your latex projects. The guide also tries to provide reasons for why it is good to follow these guidelines, in form of Why-notes. If you're not interested in the reasons just skip over them.

Contents

1 Project structure

Project structure Figure 1 shows an overview of the project structure.

Why?

If everyone in the group follows the same structure it is easier to collaborate and you don't need to learn a new system all the time.

main.tex: This is the main file of your document. Keep it as clean as possible. Conferences generally provide a template use this template to create the main file. Ideally only add the following in front of `\begin{document}`:

```
%Set to 1 to compile Figures
\def\compileFigures{0}
\newcommand{\filename}{main}
\newcounter{figureNumber}

\input{packages_and_commands}
\input{macros}
```

Don't write the sections in this file just use `\input{sections/sec_<section_name>}`.

```
paper
|       main.tex
|       references.bib
|       abb.bib % Optional
|       packages_and_commands.tex
|       macros.tex
|___sections
|       |       sec_introduction.tex
|       |       sec_conclusion.tex
|       |       sec_<meaningful_name>.tex
|       |       ...
|___fig
|       |       %Leave this folder alone
|       |       ...
|___figure_scripts
|       |       fig_<meaningful_name>.tex
|       |       ...
|___data
|       |       %data to create figures and/or tables
|       |       ...
|___images
|       |       %images that really cannot be created with latex
|       |       ...
```

Figure 1: The structure of the project.

Why?

There will be multiple versions of your paper (submissions to different workshops, conferences, Arxiv, university events, etc.). Each requires a different template and you have to merge this template with your main.tex file. The simpler the main.tex file the better.

package_and_commands.tex: In this file you put all the `\usepackage{}` imports and code for more complicated commands, which you might use in multiple projects.

Why?

If you found a useful package or spent time creating a complicated command you want to reuse it.

macros.tex: This file contains all your project-specific macros. Use macros as much as possible. For example, instead of writing `\mathbb{R}` every time for the real numbers create a macro `\newcommand{\R}{\mathbb{R}}`. Create macros when referring to existing algorithms, data sets, and names of things. When you go through your files with Ctrl + f trying to change every instance of a declaration you decided to change the notation for, you should have created a macro for that.

Why?

Macros are often project specific having all of them in a small file makes it easier for collaborators to find them.

references.bib: The file containing all your citation references. See in the Citations section on how to create the citations.

Why?

Keep the main.tex file clean.

sections: Each section of your paper should be in a separate .tex-file in this folder, give them meaningful names that start with `sec_`.

Why?

This simplifies the main file and makes it easier for people to work on different sections simultaneously.

fig: The figures generated by your scripts will be saved in this folder. Add everything but the .pdf-files to .gitignore.

Why?

Compiling figures can get time-consuming and sometimes requires specific settings. People not working on the figures can just input them as .pdf See Figures for more information.

figure_scripts: Each figure you create with TikZ or PGFPlots should be in a separate file with a meaningful name in this folder (starting with `fig_`).

Why?

The code to generate figures can get long and make it harder to work on the text. Outsourcing the scripts makes the files cleaner.

data: The data used in your graphics.

Why?

When you rerun some experiment you only need to replace the data.

images: This folder contains images (e.g. .png, .jpg.) that are used in the paper. It should not contain graphics you generated somewhere else, but really only pictures.

Why?

Keep the main folder clean.

2 Naming conventions and references

Give everything meaningful names and create labels `\label{<name>}` for sections, figures, and tables as soon as you create them.

Use `sec:<meaningfulname>` , `fig:<meaningfulname>` , `tab:<meaningfulname>`, `eq:<meaningfulname>` when creating labels for sections, figures, tables and equations respectively.

A meaningful name is a name that tells the reader (i.e. future you or a collaborator) what the label refers to. For sections the section title is a good start. Avoid using numbers and cryptic letter combinations.

Why?

You want others and future you to be able to understand what the label means.

3 Citations

When you want to create a new reference. Use the bibtex entries provided by Google Scholar.

1. Go to Google Scholar.
2. Search for the paper.
3. Go on All ¡versions¿ and find the right version (i.e. where possible the final conference/journal version)
4. Click on cite ()
5. Click on BibTeX and copy the entry
6. Double-check and fix the entry if something is wrong.

If you have to create a bibtex entry from scratch (because it doesn't exist on Google Scholar), follow the reference format `<lastname_first_author><yyyy><first_proper_word_title>`.

Why?

Everyone following one convention avoids duplicate citations. GoogleScholar has an easy format and is a good start for almost any bibtex entry.

Optional: If you want consistent references to a venue you can use the abb.bib file. Change the booktitle to `booktitle=proc # {<xx>th} # <venue_name>`

Tip: There is a Google Scholar browser extension that creates a button to directly search.

4 Figures

There are two wonderful packages in LaTeX to create figures TikZ and PGFPlots. The former can be used to create any kind of vector graphic, the latter focuses on plots. Together they can create almost any kind of figure you need for a paper. Use them.

Why?

- Creating figures inside LaTeX ensures that they use the correct font. You can scale the figure without ending up with tiny fonts.
- They just look better.
- Makes it easier to switch between different templates, often a simple change to the scale of the figure is enough. Externally created figures are harder to scale when desiring consistent font(size).

When using figures, put the following statement into your main.tex file before `\input{packages_and_commands}`:

```
%Set to 1 to compile Figures
\def\compileFigures{0}
\newcommand{\filename}{main}
\newcounter{figureNumber}
```

Your `packages_and_commands.tex\verb` file should contain:

```
\usepackage{tikz}
\if\compileFigures1
\usetikzlibrary{external}
\tikzexternalize[prefix=fig/] % activate!
\fi
```

Important: To use the external library you need to add -shell-escape to your compile command. How to do this depends on your IDE.

For every figure you create use the following template:

```
\begin{figure}
 \centering
 \if\compileFigures1
 \input{figure_scripts/fig_<meaningful_filename>}
 \else
 \includegraphics[]{fig/\filename-figure\thefigureNumber.pdf}
 \stepcounter{figureNumber}
 \fi
 \caption{<meaningful_caption>}
 \label{fig:<meaningful_label>}
\end{figure}
```

Why?

- With this setup, by changing `\def\compileFigures{0}` you can switch from creating the figures to just including them as pre-generated pdfs.
- For the camera ready version of your paper you submit your code. Here you want to ensure that everything compiles correctly, this is easier when just submitting the .pdf of each figure.
- Allows a collaborator to compile the file without compiling the figures.

Note: If your figure consists of ¡x¿ Tikz pictures, the two lines in the `\else` block need to be repeated ¡x¿ times.

Tip: If you have multiple figures with a common legend you can use the following code to create the legend:

```
\begin{tikzpicture}
\begin{axis}[scale=0.01,
hide axis,
xmin=0, xmax=1,
ymin=0, ymax=1,
legend columns=<number>,
]
\addlegendimage{<configuration_of_entry>}
\addlegendentry{<name>};
%Example
\addlegendimage{blue,mark=square*}
\addlegendentry{QII};
\end{axis}
\end{tikzpicture}
```

5 Tables

Use the booktabs package. Ideally only have three horizontal lines. Don't have vertical lines! Generally follow the advise here.

Tip

If your table gets so big that you really think you need vertical lines. You don't. But color might help.

Why?

Why You spend so much time on creating the data for your table. Having an easily comprehensible table increases the chances that someone actually reads it.

6 Generating diff files

1. If you want to generate diff file for two folders (e.g., one folder has updated version and one folder has the old version), you can use the diff.sh in this folder to do so. It requires three inputs: old folder, new folder and output folder. The script loops all tex files in the sections folder and root folder to compute the diff between the file in the older folder and the one in the new folder with the same file name and output the file into the output folder. Besides, the script copies files in other folders (e.g., image, figure_scripts) to the output folder. After the execution, you can compile files in the output folder to get diff file. Example: `bash diff.sh submitted_version new_version diff`

2. Instead, if you want to generate diff file between two commits on Github, things become easier.

- Install latexdiff first. Check instructions for Ubuntu, Windows and MacOS.
- Install git-latexdiff. Check instructions here.
- Run the following commands based on your needs: Diff the previous revision with the latest commit:

`git latexdiff HEAD^ --main main.tex`
Diff the latest commit with the working tree:
`git latexdiff HEAD -- --main main.tex`
Diff latest commit with branch master:
`gt latexdiff master HEAD --main main.tex`
Pass –type=CHANGEBAR to latexdiff to get changebars in the margins instead of red+trike/blue+underline diff:
`git latexdiff --type=CHANGEBAR HEAD^ --main main.tex`
Use a specific latexdiff configuration file:
`git latexdiff --config /path/to/file HEAD^ --main main.tex`
Note that, for those two versions, you should set compileFigures to 0. The diff will not highlight the changes of figures inside the tikzpicture but will highlight changes in the captions. In addition, you need to generate bbl manually (by runing bibtex main.tex) for these two versions and `run git latexdiff HEAD^ --main main.tex` if you want to see the changes in the reference.bib.

7 Useful Tools

- MathCha editor: GUI for creating LaTeX diagrams and equations. You can export LaTeX code e.g. tikzpicture for diagrams.