A HOW TO USE THIS TEMPLATE

This LaTeX template was developed to (1) streamline collaborative manuscript editing and (2) automate many processes that manuscripts undergo throughout the publication process of academic papers. Many ideas, features, and aspects in this template were adapted from my previous collaborators and mentors such as Qi Sun, Li-Yi Wei and Jane Hoffswell.

A.1 Usage

The LaTeX compilation is managed by a `Makefile`. The `Makefile` features multiple build targets meant for various versions of the document, such as `internal` for sharing between collaborators, and `submission` for anonymous submissions. These targets are built by running `make <target_name>` (e.g., `make internal`). See Section A.4 for details on the various build targets.

Pre-requisites. The `Makefile` assumes the usage of `pdflatex`, so please make sure to install the `texlive`, `texlive-latex-extra`, `texlive-fonts-recommended`, `texlive-science`, and `texlive-xstring` packages. Additionally, `ghostscript` is used for post-compilation PDF document processing such as to separate the document into separate files for the manuscript and supplementary materials. See more details on this in Section A.4.

A.2 LaTeX macros for collaborative editing

This template features a number of very convenient macros designed to make collaborative writing much easier and more streamlined. All of these macro definitions can be found in `conf/macros.tex` and edited as necessary. Below, we outline what the features are and how to use them.

Tracking Progress: Status Badges and To-do Items. When collaborating on a large document, it's often helpful to keep track of what sections have been reviewed, what sections are complete etc. This template enables status tracking of paper sections via *Status Badges* as well as *To-do Item* annotations which can be viewed in summary at the top of the document.

Status Badges indicate what state a particular section of the manuscript is in and allows collaborators to claim, and release sections as necessary. To add a badge to a section, simply add the corresponding badge command at the end of the section title (e.g., `\section{Section Title \incomplete}`). This badge will not only annotate the section title with its corresponding status, but also include the annotation to a summary "table-of-contents" which appears at the top of the document when rendered in the `internal` version. For the full list of possible status badges, see Table 2.

To-do Items are also a helpful tool for annotating important to-do items that need to be tracked. Including an actionable in the source document as `\todo{revise this sentence}` will render the to-do item within the manuscript in a highlighted red color, and also include a copy of this to-do item in the "table-of-contents" at the top of the document just like the *Status Badges*.

The combination of these two macros can help collaborators keep track of everything that still needs to be done for a paper submission prior to deadlines.

Table 2. Status badges

Command	Created Badge
`\incomplete`	incomplete
`\underRevision`	under revision
`\feedbackNeeded`	feedback needed
`\feedbackGiven`	feedback given
`\complete`	complete
`\locked`	locked

Exchanging feedback via in-document comments. In addition to important to-do items, sometimes it is helpful to simply have asynchronous discussions that aren't necessarily mission critical within the document draft itself. To facilitate this workflow, the template also supports macros for in-document comments by multiple collaborators, each highlighted in a different color.

Monde: For example, I might have some thoughts on the organization of this section.

Qi: And I might respond to Monde's concerns in response.

The current template has pre-defined macros for four collaborators: `monde`, `qi`, `anjul`, and `rachel`. To use the macro call the macro as `\monde{comment contents}` for a simple comment. To annotate existing text to reference by the comment, **Monde:** like this, put the existing text into square brackets. For example, `\monde[annotated text]{comment contents}`. There is also a generic `guest` command to allow a guest to input their name into their comment without having to define a new macro. To use this version, call the macro as `\guest{name}{comment contents}` for a simple comment and `\guest[annotated text]{name}{comment contents}` for annotated comments.

Revision annotation. During the revision stage of a document for publication, reviewers often ask for specific edits to the manuscript. To enable an easy back and forth between the reviewer and the authors, highlighting all the changed text can be very helpful. To this end, this template features a `\revise{text to remove}{text to add}` macro which annotates both removed and added text. By inputting the to-be-removed text in the first argument, and the to-be-added text in the second argument, the annotation ~~won't~~will look like this. By default, the removed text will only be shown in the `internal` version of the document, while the `revision` version of the document only shows the added text. But you can edit this behavior in the `conf/macros.tex` file. The automated build pipeline (cf. Section A.4) allows seamless changes between the different versions of how a revised text should be rendered.

A.3 File Organization

This template is organized into a granular structure and logical separation of various types of source files that make up a LaTeX document. Table 3 explains what types of files are at the various paths.

Sections. All numbered sections are combined in `sections/document.tex`, which in turn are inputted in the compilation process

Table 3. File paths and their functions

Path	Function
`sections`	numbered section LaTeX source files
`sections/abstract.tex`	document abstract
`sections/acks.tex`	acknowledgements section
`sections/document.tex`	manuscript sections entry point
`sections/readme.tex`	how to use this template
`sections/supp.tex`	supp. material entry point
`tables`	table LaTeX source files
`figures`	figure LaTeX source files
`assets`	illustrations and plots
`conf`	LaTeX configurations
`metadata`	metadata of the document
`targets`	LaTeX compilation entry points
`paper.tex`	combination of source files
`paper.bib`	bibliography
`Makefile`	build pipeline definitions
`build`	build outputs

in `paper.tex`. Each numbered section name is prepended by its number for visual clarity, and each section is given its LaTeX label in the form `\label{sec:<section_name>}`. Further subsection labels are nested as `sec:<section_name>:<subsection_name>`.

Tables and Figures. All table and figure source files are separated from *Section* source files to enable easy edits to where a figure should be added in the source. Each figure is given its LaTeX label in the form `\label{tab:<table_name>}` or `\label{fig:figure_name}`

Packages, Aliases, and Macros. LaTeX package imports are consolidated in `conf/packages.tex`. To avoid complications when some packages need to be deprecated in the future, it's advised that the specific usage of each package is documented as a comment. For example, we use the `soul` package to enable underlining and strike-through functionality for annotating collaborator comments, and text revisions. If in the future, either this feature of the template is changed or removed, removing this package dependency will be much easier to identify and execute.

All symbol aliases used during writing are consolidated within `conf/symbols.tex`, while more complex macros are in `conf/macros.tex`. More details on what kind of macros are used in this template are in Section A.2.

Paper metadata. All source files pertaining to paper metadata, such as the title, authors, journal number, publication year etc. are included in `metadata/` and are self-explanatory. These metadata fields are inserted into the compilation in `paper.tex`. If you're changing the document class, include any document class related metadata fields in the `metadata/` directory, and change the organization of source files within `paper.tex`.

Build targets. All build targets have their associated LaTeX entry point inside `targets/` as well as a `Makefile` target. See more details about the build pipeline in Section A.4.

A.4 Build Pipeline

The build process of the manuscript is fully automated via a `Makefile` and allows the user to seamlessly produce different versions of the manuscript and supplementary materials documents. i.e., the document can be produced at various stages of "readiness for publication", with each different stage revealing or hiding various comments and annotations. The supported document stages are described in Table 4.

Table 4. Document stages and their usage and features.

Build target	Usage and features
`internal`	internal collaborative use with comments shown
`submission`	anonymous submission with no annotations
`revision`	annotated anonymous revision for reviewers
`camready`	final camera-ready with author names shown
`camauthor`	author-version for sharing post-publication

You may add additional document stages by

- adding a new LaTeXcompilation entry file inside `targets/`,
- updating the `Makefile` targets, and
- adding your custom logic for what annotations to enable in `conf/macros.tex`.

Automated manuscript and supp. material separation. Often a paper publication requires an associated supplementary materials section for any exposition that does not central to the publication and are omitted from the main manuscript. In order to take advantage of LaTeX cross-referencing features, such as `\Cref`, it's best to compile all the source files in one batch and generate a single PDF output. However, doing so requires post-processing on the outputted file to separate them into multiple PDFs via an external application such as Adobe Acrobat etc. Additionally, many paper templates also print the number of pages of the document which include the supplementary material pages if this method is used to generate two separate files for the main manuscript and the supplementary material.

This template takes care of these manual steps and also solves the page count issue and keeps all the complexity of the compilation and post-processing logic tucked away in the `Makefile`. In order to take advantage of this feature, make sure to use the Makefile to compile your document and have its dependent `ghostscript` package installed on your machine.

A.5 Future Features

I'm constantly updating this template to help make the paper authoring and collaboration process more streamlined. Below is a short list of features that I'm looking to add in the future:

- add easy to use LaTeX linting and formatting scripts;
- add helper scripts which "commit/revert" the revisions by stripping the LaTeX tags and incorporating the existing revisions into the native text;
- add a `Makefile` target for preparing the publication for Arxiv upload;
- streamline the process of swapping out different style files;

- split the package dependencies so that unnecessary development / internal dependencies are omitted;
- add a script which "dumbs down" the LaTeX source files into a single source file to satisfy strict publisher requirements.