LaTeX for Theses and Dissertations

LaTeX for Theses and Dissertations

A Complete System for Writing, Formatting, and Submitting Your Thesis

Gaurav Tiwari

gauravtiwari.org

First edition, 2026.

gauravtiwari.org

To everyone whose deadline is closer
than their last backup.

Contents

Preface

A thesis deadline has a particular flavor of panic, and very little of it comes from the research. It comes from the document: the chapter that stopped compiling at midnight, the margins the graduate office rejected, the bibliography that renumbered itself, the figure that teleported four pages downstream. I've watched brilliant students lose whole weeks to their tools in the exact months they could least afford it, and I wrote my own master's thesis in LaTeX with all of those scars.

This book exists so the document is never your problem. It's a system, not a reference: we build one thesis from an empty folder to an accepted repository upload, in the order the decisions actually arrive. Structure first, university compliance second, then the content machinery (figures, tables, mathematics, code), then the scholarly apparatus (bibliography, glossary, front matter), and finally the production line: staying compiling, staying backed up, and shipping a PDF the graduate office accepts on the first try.

You need only light LaTeX experience to start, the level of a few problem sets or lab reports. If you have none at all, my *LaTeX for Students* is the on-ramp; this book begins where it ends and assumes the basics without re-teaching them.

Two promises shape every chapter. First, thesis-scale honesty: everything here is chosen for a 150-page document maintained over a year, which is a different sport from a six-page article, and I'll say so when the standard advice doesn't survive the scale-up. Second, recovery over perfection: each chapter ends with what breaks and the fastest way out, because at 11 p.m. on submission eve, nobody needs theory. They need the fix.

Start at Chapter 1 if your folder is still empty. Start at Chapter 13 if the thing just stopped compiling. That's what it's there for.

Gaurav Tiwari
gauravtiwari.org

Part I

Foundations

1

A Thesis Is a Different Animal

You already know how to make a LaTeX document. You've done problem sets, maybe a lab report, maybe a term paper: one `.tex` file, a handful of packages, compile, submit, forget. That workflow has served you for years, and here is the most useful sentence in this book: **it will not survive contact with a thesis.**

Not because a thesis is harder LaTeX. Because it's a different kind of object. Once you see the three differences clearly, every design decision in the next fourteen chapters becomes obvious, so let's take them slowly.

1.1 Scale changes the physics

A term paper is 8 pages, 2 figures, 10 references, alive for two weeks. A thesis is 100 to 250 pages, dozens of figures and tables, one to three hundred references, six chapters, and it stays alive for a year or more. At that scale, things that were free start to cost:

- **Compile time.** A one-file document compiles before your finger leaves the key. A 200-page document with 80 figures does not, and a slow feedback loop quietly ruins writing sessions.
- **Navigation.** Scrolling to find the section you're editing works at 8 pages. At 180 it's archaeology.
- **Fragility.** One unclosed brace in a term paper costs a minute. The same brace in a monolithic thesis file, three days before a draft is due, can cost an evening of binary-searching your own document.
- **Memory.** You will not remember in March what you named that figure in September. Conventions you never needed before become load-bearing.

Every one of these has a structural answer (chapters in separate files, labels with a naming scheme, a build tool, a way to compile one chapter at a time), and Chapter 3 builds them all in. The point here is simply that they must be *designed in*, at the start, not bolted on during a crisis.

1.2 The rules aren't yours

Nobody checked your problem set's margins. Your university will check your thesis's, along with line spacing, font size, page-number placement, title-page wording, the order of front-matter pages, and, increasingly, the archival PDF standard of the final upload. Somewhere on the graduate school's website is a formatting document written years ago, and a person or a script whose job is enforcing it. They can and do reject submissions the week of a deadline.

This changes your relationship with the preamble. In a term paper, formatting is taste. In a thesis, formatting is *compliance*, and the smart move is to encode the rulebook once, correctly, in one place, and then never think about it again (Chapter 4 does exactly this, rule by rule). The students who suffer are the ones who format by eye in the last week.

1.3 It outlives your attention

The third difference is time. You'll write this document across many months, in bursts, between experiments and teaching and life. You'll return after three weeks away. Your advisor will want revisions to chapter 3 while you're writing chapter 5. A collaborator or committee member will annotate a PDF. Your laptop will, statistically speaking, have one bad day.

So a thesis needs what a term paper never did: version control, real backups, a way to see what changed between drafts, and a build that works on more than one machine (Chapters 2 and 15). None of this is paranoia. Ask any finished PhD for their disaster story; everyone has one, and the punchline is always what they wish they'd set up in week one.

> *A term paper is a document. A thesis is a small software project whose output is a document. Treat it like one from the first day, and the last month of your degree becomes dramatically calmer.*

1.4 The system we'll build

Here's the whole book in one view. We build a running example thesis, file by file, and every chapter adds one subsystem:

1. **Foundations** (Chapters 2–4): the toolchain, the multi-file skeleton, and your university's rules encoded in the preamble.
2. **Content at scale** (Chapters 5–9): cross-references, figures, tables, mathematics, and code, each set up so it still works at page 200.
3. **Scholarly apparatus** (Chapters 10–12): the bibliography database, acronyms and notation, and the front and back matter your university expects.
4. **Production** (Chapters 13–15): staying compiling, versioning and advisor feedback, and the final mile to an accepted upload.

The complete skeleton appears, annotated, in Appendix A, so you can also work backwards: copy it today, then read the chapters to understand what you copied.

One reassurance before we start. You will not need to become a LaTeX expert. The set of commands a thesis actually requires is small and stable; what matters is arranging them into a system. Configuration is a one-time cost, and this book pays it with you, once, in the right order.

What breaks here

Even in a chapter with no code, there are two classic week-one mistakes worth naming:

- **Starting in the term-paper shape** (one file, ad hoc formatting) with a plan to "reorganize later." Later arrives mid-crisis. Reorganizing a 60-page monolith with a deadline near is miserable; starting from the right skeleton costs twenty minutes.

- **Copying a random thesis template** from the internet and building on it unread. Half the disasters I've debugged trace to a decade-old template stuffed with packages nobody present understood. You may absolutely start from a template, including your university's. But by the end of Chapter 4 you'll be able to read it, and you should.

2

Your Toolchain

Before a word gets written, decide where the thesis lives and what compiles it. These are twenty-minute decisions with eighteen-month consequences, so this chapter makes them explicitly, and then you never revisit them.

2.1 Local, cloud, or both

You have two good homes for a thesis and one common mistake.

Overleaf (the cloud editor) is genuinely good: nothing to install, compiles anywhere, sharing with your advisor is a link, and its history feature has saved real theses. Its weaknesses appear exactly at thesis scale: compile times on large documents (free-tier timeouts are a real risk near the end), dependence on conference wifi, and less control over the exact TeX version your university's template assumes.

A local TeX installation (TeX Live on Windows and Linux, MacTeX on macOS) compiles fast, works on trains, and is fully yours. Its weakness is that backups and sharing are now your job.

The mistake is treating this as a religious choice. The robust setup, and my recommendation, is **local-first with a synced escape hatch**: install TeX Live, work locally in a git repository (Chapter 15), and keep the project importable to Overleaf for advisor moments and emergencies. Overleaf speaks git directly on paid plans, and a plain GitHub link covers the rest. If you strongly prefer living in Overleaf instead, everything in this book still applies; just read Chapter 15's history section closely, and do a full local test compile before submission week.

Install the *full* TeX Live, not the minimal one. The full install is large, but "package not found" three days before a deadline is a trade nobody wants in reverse.

2.2 One command to build: latexmk

A thesis build is genuinely multi-pass: LaTeX, then Biber for the bibliography, then LaTeX twice more so citations and cross-references settle. Running those by hand, in order, every time, is how people end up with question marks where numbers should be, and stale bibliographies in submitted drafts.

The fix has existed for decades: `latexmk`, which runs whatever passes are needed until the document is stable.

```
latexmk -pdf main.tex        # build everything, as many passes as needed
latexmk -pdf -pvc main.tex   # rebuild automatically on every save
latexmk -c                   # sweep auxiliary files
```

Make `latexmk -pdf main.tex` the only way you ever build, from day one. Editors and Overleaf use it under the hood anyway; using it deliberately means your build is reproducible on any machine, including the one you'll borrow in a crisis. Project-specific settings go in a `.latexmkrc` file in the project root, and Appendix A ships one.

2.3 An editor that earns its place

Any serious editor works: TeXstudio (a fine default), VS Code with LaTeX Workshop (my choice, and you get git integration free), Vim or Emacs if they're already home. What the editor must give you at thesis scale:

- **Forward and inverse search**: click in the PDF, land in the source, and back. At 180 pages this is navigation, not luxury.
- **Project-wide search** across all chapter files, because "where did I define that macro" is a daily question.
- **Error-list parsing**, so a failed build jumps you to the offending line instead of a log file.

Ten minutes configuring these repays itself weekly. What you should *not* adopt in week one is a heap of clever plugins, snippets, and AI autocompletion you don't understand yet; every moving part is something to debug in April.

2.4 Where files live

Three rules, all learned the hard way by someone:

1. **One project folder, no exceptions.** Everything the thesis needs (sources, figures, bibliography, class files) lives under one directory, with relative paths inside. A thesis that references `C:/Users/you/Desktop/plot-final2.png` dies the day it meets another computer.
2. **Not inside a live-sync folder.** Dropbox, iCloud, and OneDrive corrupt active LaTeX builds just often enough to matter: they sync half-written auxiliary files and fight the compiler for locks. Work outside the synced tree; let git (plus its remote) be the sync. If cloud sync is non-negotiable, at least exclude the build outputs.
3. **Boring names, no spaces.** Folders and files in lowercase, hyphens, ASCII: `fig/ch2-setup-diagram.pdf`. Spaces and Unicode in filenames still bite various tools, and "final-v3-REALLYFINAL" as a naming scheme is what Chapter 15 exists to replace.

2.5 The five-minute proof of life

Before moving on, prove the chain works. Create the folder, then:

```
% main.tex -- proof of life
\documentclass{report}
\begin{document}
Hello, thesis.
\end{document}
```

```
latexmk -pdf main.tex
```

If a PDF appears, your toolchain is done, and honestly done: nothing in the next 200 pages requires more installation. If it doesn't, fix it now, while it costs nothing. The classic culprits: the terminal can't find `latexmk` (the TeX binaries aren't on your PATH; log out and back in, or consult your installer's note), or an antivirus quarantining freshly written PDFs.

What breaks here

- **Minimal TeX installs** fail months later with missing packages. Install full TeX Live; disk is cheaper than deadline time.
- **Cloud-sync folders** corrupt builds and eat auxiliary files. Move the project out; you'll meet this bug as unexplained "file in use" and half-updated PDFs.
- **Hand-run compile sequences** ship stale references. If you ever see `??` or an outdated citation in a PDF you just built, run `latexmk` instead of trying to remember the dance.
- **Two TeX installations** (an old MiKTeX plus a new TeX Live, say) make errors unreproducible. Keep one; uninstall the other.

3

The Skeleton

This chapter builds the structure everything else hangs on: the folder layout and the multi-file document. It's the highest-leverage twenty minutes in the book. Get the skeleton right and the thesis scales quietly for a year; get it wrong and every later chapter of this book becomes a renovation.

3.1 The folder layout

Here's the running example we'll build for the rest of the book:

```
thesis/
  main.tex            % the conductor: preamble + \include list
  preamble.tex        % every package and setting, one place
  metadata.tex        % title, name, degree, committee, date
  frontmatter/        % titlepage, abstract, acknowledgments...
  chapters/
    ch1-introduction.tex
    ch2-background.tex
    ch3-methods.tex
    ...
  appendices/
  fig/                % all graphics, prefixed by chapter: ch3-flow.pdf
  bib/references.bib  % the one bibliography database
  .latexmkrc
```

The principles behind it: content files contain *only* content (no packages, no settings), configuration lives in exactly one file, and every asset's name tells you which chapter owns it. When something breaks, and eventually something will, this layout is what makes the search space small.

3.2 The conductor file

`main.tex` stays under a page, forever. Its whole job is assembly:

```
% main.tex
\documentclass[12pt,oneside]{report}
\input{preamble}
\input{metadata}

\begin{document}
```

```
\input{frontmatter/titlepage}
\input{frontmatter/abstract}
\tableofcontents

\include{chapters/ch1-introduction}
\include{chapters/ch2-background}
\include{chapters/ch3-methods}
\include{chapters/ch4-results}
\include{chapters/ch5-conclusion}

\appendix
\include{appendices/appA-data}

\printbibliography

\end{document}
```

Two commands stitch files in, and the difference between them is the one piece of LaTeX lore this chapter insists you learn.

`\input` pastes a file in place, as if you'd typed it. Use it for the preamble, metadata, and front-matter fragments.

`\include` is for chapters, and it does three chapter-shaped things: starts a fresh page, gives the file its own auxiliary file so builds can be partial, and enables the magic switch below. Its one quirk: never nest an `\include` inside another; nesting is what `\input` is for.

The magic switch is `\includeonly`, and at thesis scale it's oxygen. Added to the preamble area of `main.tex`:

```
\includeonly{chapters/ch3-methods}
```

Now the build compiles *only* chapter 3, in seconds instead of minutes, while page numbers and cross-references from the other chapters' last build stay approximately right. You write all day inside one chapter with a fast loop, then comment the line out for full builds. This single feature is why chapters get `\include` and not copy-paste, and it's the answer to the compile-time physics of Chapter 1.

3.3 Choosing the class

The running example uses `report`: the standard class with chapters, the least surprising base for a thesis, and the easiest to bend to university rules. The other defensible choices:

- **Your university's official class**, if one exists. Use it if it's maintained; Chapter 4 covers living with one, including the unmaintained kind.
- `book`: like `report` with two-sided printing conventions baked in. Choose it only if your university wants two-sided; the `oneside`/`twoside` option matters more than the class name.
- `memoir` **or KOMA-Script (**`scrreprt`**)**: powerful, well-documented superclasses. Wonderful if you already know them; unnecessary if you don't. A thesis deadline is not the time to learn a design system.

What's not defensible: `article` (no chapters) or a conference template (compliance will be war). When in doubt, `report` with `12pt` matches more university rulebooks than anything else.

3.4 Content hygiene inside chapter files

Each chapter file starts with its `\chapter` line and label, and contains prose, sections, floats, and nothing else:

```
% chapters/ch3-methods.tex
\chapter{Methods}\label{ch:methods}

\section{Experimental setup}\label{sec:methods-setup}
...
```

Resist the temptation to define macros or load packages "just for this chapter" inside chapter files. Every setting in a content file is a landmine for your future self, who will look for it, correctly, in `preamble.tex` and not find it. The one exception worth allowing: a genuinely chapter-local figure path prefix, and even that is better handled by the naming convention.

Two small habits that pay at scale: one sentence per source line (makes git diffs and advisor comparisons readable, Chapter 15), and a comment line of dashes to mark where each section starts, so project-wide search lands cleanly.

3.5 Metadata in one place

Your name, title, degree, department, committee, and defense date will each appear in several places: title page, abstract page, declaration, PDF metadata. Type each exactly once:

```
% metadata.tex
\newcommand{\thesistitle}{Adaptive Methods for Sparse Problems}
\newcommand{\thesisauthor}{Your Name}
\newcommand{\thesisdegree}{Doctor of Philosophy}
\newcommand{\thesisdept}{Department of Mathematics}
\newcommand{\thesisdate}{May 2027}
```

Front-matter pages then use `\thesistitle` and friends. When the defense slips a semester (it happens to everyone), you change one line, and nothing is inconsistent anywhere.

What breaks here

- **The monolith relapse.** Under pressure, people paste a new section straight into `main.tex` "temporarily." The conductor stays content-free; that's what makes every later rescue possible.
- **`\include` inside `\include`** fails with confusing errors. Nested stitching is `\input`'s job.
- **A stale `\includeonly`** left active produces a mysteriously short PDF. If "half my thesis vanished," check that line first; it's the most common false alarm in the book.
- **Paths with `..` escaping the project** (like `../figures`) break on Overleaf import and other machines. Everything lives under `thesis/`.

4

Meeting Your University's Rules

Somewhere on the graduate school's site is a PDF with a name like "Thesis and Dissertation Formatting Requirements." Download it today, read it once with a highlighter, and then do what this chapter does: translate every rule into one preamble line, so compliance becomes a property of the system instead of a weekly anxiety.

We'll walk the rulebook in its usual order. All of it lands in `preamble.tex`, nowhere else.

4.1 Margins

The typical demand is one inch all around, sometimes 1.5 inches on the binding edge for printed copies. One package, one line:

```
\usepackage[margin=1in]{geometry}
% or, for a bound copy:
\usepackage[inner=1.5in, outer=1in, top=1in, bottom=1in]{geometry}
```

Resist the pre-`geometry` folklore (hand-set `\oddsidemargin` arithmetic) still circulating in old templates; if your inherited template does margin math by hand, replacing it with `geometry` is usually your first cleanup. One caution: margins interact with one-sided versus two-sided layout, so confirm which your university wants (`oneside` is the common demand for the uploaded PDF) before tuning `inner`/`outer`.

4.2 Line spacing

"Double-spaced" or "1.5-spaced" in the rulebook maps to the `setspace` package:

```
\usepackage{setspace}
\doublespacing        % or \onehalfspacing
```

`setspace` is civilized: footnotes and captions stay single-spaced, which is exactly what formatting offices expect. For long quotations they usually also expect single spacing; the `quote` environment wrapped in `\begin{singlespace}` handles the rare case. Do not fake spacing with `\baselinestretch` or blank lines; that's the kind of improvisation that fails review.

4.3 Fonts and size

Rulebooks say things like "a standard 11- or 12-point font." The class option (`12pt` on the `\documentclass` line) sets the size. For the face, three modern, safe choices, each one line:

```
\usepackage{lmodern}    % the polished default look
% or
\usepackage{newpxtext,newpxmath}   % Palatino-style, warm, great with math
% or
\usepackage{stix2}      % Times-adjacent with a full math suite
```

Pick one, once. Font-hopping mid-thesis reflows every page and resurfaces every marginal figure placement you've already tamed. If your field expects Times ("Times New Roman or equivalent" is common), `stix2` or `newtx` satisfies both the rulebook and mathematics; if the rulebook literally names fonts, any of these reads as "equivalent" in practice, but when in doubt, one email to the formatting office beats one resubmission.

4.4 Page numbering

The near-universal demand: front matter in roman numerals (i, ii, iii), main matter in arabic starting at 1, every page numbered, usually bottom center or top right. With the `report` class this is a two-command dance at the front/main boundary:

```
% after the title page, before the abstract:
\pagenumbering{roman}
...front matter...
% right before chapter 1:
\clearpage
\pagenumbering{arabic}   % resets to 1 automatically
```

Position is the `fancyhdr` package's job if the default (bottom center) isn't what's demanded. Keep headers modest: many rulebooks cap what may appear in them, and plain page numbers are never wrong.

4.5 The title page

Formatting offices are pickiest here, because it's the page they pattern-match: exact wording ("submitted in partial fulfillment of the requirements for the degree of…"), exact stacking order, sometimes exact vertical spacing. Build it by hand, from the rulebook's sample, as `frontmatter/titlepage.tex`:

```
% frontmatter/titlepage.tex
\begin{titlepage}
\centering
{\Large\bfseries \thesistitle\par}
\vspace{1in}
by\\[0.5em]
{\large \thesisauthor\par}
\vfill
A dissertation submitted in partial fulfillment\\
of the requirements for the degree of\\[0.5em]
\thesisdegree\\[0.5em]
```

```
\thesisdept\\[2em]
\thesisdate
\end{titlepage}
```

Note what it uses: the `metadata.tex` commands from Chapter 3, plain centering, and `\vfill`/`\vspace` for the gaps. Mirror your rulebook's sample line by line, top to bottom. This page is also the one to email for approval early if your office offers a format check; they'd rather bless a title page in October than reject it in May.

4.6 Living with an official class file

If your university ships an official class or template (`myuni-thesis.cls`), the calculus changes.

If it's maintained (updated in recent years, has a contact, compiles clean on current TeX Live): use it. It encodes rules you'd otherwise transcribe, and the formatting office trusts its own artifact. Your `preamble.tex` shrinks to content packages, and this chapter becomes verification instead of construction.

If it's abandoned (a 2009 file full of warnings): you have a judgment call. Small warnings you can ignore; hard errors on a modern TeX mean you're better off with `report` plus this chapter, matched against the written rulebook, which, not the class file, is the actual requirement. Keep the old class around as a specification to imitate, not code to run.

Either way, put the class file (and any university logo images) *inside* the project folder, so the build remains portable.

4.7 Encode it, then trust it

The finished compliance block in the running example's preamble is about a dozen lines, each traceable to a sentence in the rulebook. Add a comment citing the rulebook section next to each line; future you, and the student who inherits your template, will be grateful:

```
% --- University formatting (Grad School Handbook, sec. 2) ---
\usepackage[margin=1in]{geometry}       % 2.1 margins
\usepackage{setspace}\doublespacing     % 2.2 spacing
\usepackage{stix2}                      % 2.3 "Times or equivalent"
```

From here on, formatting anxiety is retired. If the office changes a rule, you change a line.

What breaks here

- **Formatting by eye in the last week.** Compliance is preamble configuration, done once. If you're nudging spacing visually in May, something upstream was skipped.
- **Old-template margin arithmetic** fighting `geometry`: loading both styles of margin control produces warnings and wrong margins. One system, and it's `geometry`.
- **Double-spaced footnotes or captions** usually mean spacing was hacked rather than set via `setspace`.
- **The re-flow cascade**: changing font or spacing late moves every float in the document. Freeze the compliance block early, and treat any late change to it as a full re-review of figure placement.

5

Your First Week: From Empty Folder to Working Thesis

The safest time to build the thesis system is before it contains a thesis. In week one, a broken build costs ten minutes. In month eleven, the same mistake is buried under 180 pages, 240 references, and a deadline.

This walkthrough starts with an empty folder and ends with a small project that compiles on command, contains one chapter, survives a clean rebuild, and can be opened on another machine. Do it once with your own university rules beside you.

5.1 Day 1: create the project, not the prose

Create this structure:

```
thesis/
|-- main.tex
|-- preamble.tex
|-- metadata.tex
|-- references.bib
|-- chapters/
|   |-- introduction.tex
|-- figures/
|-- tables/
|-- build/
|-- notes/
|-- .gitignore
```

The empty directories are not bureaucracy. They prevent the habit of dropping every file beside `main.tex` until names such as `figure-final-2-revised.pdf` become unavoidable.

Put this in `metadata.tex`:

```
\newcommand{\thesistitle}{A Precise Working Title}
\newcommand{\thesisauthor}{Your Name}
\newcommand{\thesisdegree}{Master of Science}
\newcommand{\thesisuniversity}{Your University}
\newcommand{\thesisyear}{2026}
```

The values will change. Their locations should not. One definition now prevents a different title appearing in the title page, PDF metadata, approval form, and repository record later.

5.2 Build the smallest conductor

The first `main.tex` should be boring:

```
\documentclass[12pt,oneside]{report}

\input{metadata}
\input{preamble}

\begin{document}

\begin{titlepage}
  \centering
  {\Large\bfseries \thesistitle\par}
  \vspace{2cm}
  {\large \thesisauthor\par}
  \vfill
  A thesis submitted for the degree of\\
  \thesisdegree\\[1cm]
  \thesisuniversity\\
  \thesisyear
\end{titlepage}

\tableofcontents

\include{chapters/introduction}

\bibliographystyle{plain}
\bibliography{references}

\end{document}
```

If your university supplies a class file, replace `report` with that class now. Do not copy pieces of its preamble into yours until you know why they are needed. The class owns the institutional layout; your preamble owns the packages and commands added by the project.

5.3 Add only the first packages

Start `preamble.tex` with the packages the empty skeleton already needs:

```
\usepackage[T1]{fontenc}
\usepackage{lmodern}
\usepackage{microtype}
\usepackage{amsmath,amsthm}
\usepackage{graphicx}
\usepackage{booktabs}
\usepackage[hidelinks]{hyperref}
```

This is not the final preamble. It is the first known-good state. Packages earn their way in when a chapter needs them. Adding thirty on day one creates thirty possible conflicts and no reader value.

The introduction file can also be small:

```
\chapter{Introduction}\label{ch:introduction}

This project now has one chapter and one successful build.

\section{Research question}\label{sec:research-question}

State the working question here. It can change.
```

Compile now:

```
latexmk -pdf main.tex
```

Open the PDF and check five things:

1. the title page uses the metadata values;
2. the contents lists the introduction;
3. the chapter begins on the expected side;
4. the hyperlink in the contents works;
5. a second build exits without unresolved references.

Do not write another chapter until these five pass. A successful first build is the foundation you return to whenever a later change breaks something.

5.4 Day 2: encode the university's rules

Take the official formatting document and turn each measurable rule into one preamble setting. Keep a checklist beside it:

Rule	Source	Encoded where
paper size	handbook p. 12	class option / geometry
left margin	handbook p. 12	geometry
body spacing	handbook p. 14	setspace
font and size	handbook p. 13	class option / font package
page numbers	handbook p. 16	page style
title wording	form A	title page

For a university that requires A4 paper, a 38 mm binding margin, 25 mm outer margin, and 30 mm vertical margins, the geometry is explicit:

```
\usepackage[
  a4paper,
  inner=38mm,
  outer=25mm,
  top=30mm,
  bottom=30mm
]{geometry}
```

Measure the resulting PDF. Do not adjust until it “looks right.” A rule with a number should be implemented with a number and checked with a number.

Make one page of deliberately ugly test content: a long paragraph, a displayed equation, a footnote, a section heading near a page break, and a figure placeholder. This is your formatting test, not thesis prose. The awkward page reveals spacing and margin problems while they are cheap.

5.5 Day 3: establish names that survive

Create the label scheme before labels multiply:

```
ch:introduction
sec:research-question
fig:apparatus-overview
tab:sample-properties
eq:governing-model
thm:main-result
app:calibration
```

Then add one reference to the introduction:

```
The experimental arrangement appears in
Figure~\ref{fig:apparatus-overview}.
```

It will print `??` because the figure does not exist yet. That is useful. Add a boxed placeholder rather than an imaginary finished figure:

```
\begin{figure}[tbp]
  \centering
  \fbox{\rule{0pt}{45mm}\rule{0.8\linewidth}{0pt}}
  \caption{Experimental arrangement. Replace before submission.}
  \label{fig:apparatus-overview}
\end{figure}
```

Compile twice. The `??` must disappear. You have now tested the label convention, the float style, the caption position, and the multi-pass build before any real asset depends on them.

5.6 Day 4: add one real reference

Put one complete entry in `references.bib`. Export it from your reference manager, then inspect the fields rather than trusting the export blindly.

```
@book{lamport1994latex,
  author    = {Leslie Lamport},
  title     = {LaTeX: A Document Preparation System},
  edition   = {2},
  year      = {1994},
  publisher = {Addison-Wesley}
}
```

Add a sentence that cites it and run the full build. The purpose is not to settle the bibliography style. It is to prove that the bibliography tool, database, citation command, and final list talk to one another.

If the university requires `biblatex` and Biber, switch now rather than after 200 entries:

```
\usepackage[backend=biber,style=authoryear]{biblatex}
\addbibresource{references.bib}
```

Replace the old bibliography commands with `\printbibliography`. Then let `latexmk` run the required passes.

5.7 Day 5: make failure recoverable

Create `.gitignore`:

```
build/
*.aux
*.bbl
*.bcf
*.blg
*.fdb_latexmk
*.fls
*.log
*.out
*.run.xml
*.synctex.gz
*.toc
```

Keep source figures and the bibliography. Ignore generated files that the build can recreate. If your class produces other auxiliary files, add them after confirming they are generated rather than source.

Initialize version control:

```
git init
git add .
git commit -m "Create compiling thesis skeleton"
```

Now break the project on purpose. Delete a closing brace in `introduction.tex`, compile, read the first error, repair it, and compile again. Then run a clean build:

```
latexmk -C
latexmk -pdf main.tex
```

The deliberate failure matters. A build system you have seen fail and recover is more trustworthy than one that has only succeeded once.

5.8 The borrowed-machine test

At the end of the week, copy or clone the project to another machine or a clean cloud project. Build it without adding missing files by hand.

Failures at this stage usually expose one of four things:

- an absolute path such as `/Users/name/Desktop/figure.pdf`;
- a filename whose capitalization differs between source and disk;
- a local package or font that the project never documented;
- a figure, class, or bibliography file that lived outside the project folder.

Repair the project, not the second machine. The project should contain or document every dependency it needs.

5.9 What you should have after seven days

Not a finished introduction. A system:

- one command produces the PDF;
- the institutional rules live in one place;
- one chapter, one figure, one cross-reference, and one citation have passed through the full build;
- generated files are separated from source;
- the project has a recoverable version history;
- another machine can build it.

> *Week one is successful when the project can survive change. Page count is irrelevant. A two-page thesis that builds cleanly on two machines is further ahead than a thirty-page draft trapped inside one laptop.*

What breaks here

- **The empty directory disappears from Git.** Git tracks files, not directories. Put a short `README.md` or `.gitkeep` in a directory that must exist before it receives content.
- **The clean build loses the bibliography.** The editor was running only LaTeX, not Biber or BibTeX. Build through `latexmk` and align the backend with the preamble.
- **The project builds locally but not elsewhere.** Search for absolute paths and case mismatches first. They cause more failures than missing packages.
- **The university class conflicts with a package.** Remove your copy before changing the class. The class may already load and configure that package.

Part II

Content at Scale

6

Cross-References That Never Break

A thesis is full of sentences like "as shown in Section 3.2" and "see Figure 4.7." Hard-code any of those numbers and you've planted a bug that detonates the next time a chapter moves, and chapters move. This chapter sets up referencing so numbers are always right, names are consistent, and six months of edits can't break a single pointer.

6.1 Labels and the naming convention

The mechanism you know: `\label` plants a marker, `\ref` retrieves its number. What changes at thesis scale is that you'll have hundreds of labels, and retrieval happens by memory. So the labels need a grammar. Use *type, chapter, then a name*, separated by colons and hyphens:

```
\chapter{Methods}\label{ch:methods}
\section{Data collection}\label{sec:methods-data}
\begin{figure} ... \caption{...}\label{fig:methods-pipeline}\end{figure}
\begin{table} ... \caption{...}\label{tab:methods-params}\end{table}
\begin{equation}\label{eq:methods-loss} ... \end{equation}
```

The type prefixes (`ch:`, `sec:`, `fig:`, `tab:`, `eq:`) are the standard set; the chapter-slug infix is the thesis-scale addition that keeps names unique and findable. When you type `\ref{fig:` in a decent editor, autocomplete shows every figure label; the infix means you can actually pick the right one.

Two rules of label hygiene. Label at birth: the moment a chapter, section, or float exists, its label goes in, because retrofitting labels during a citation hunt is misery. And for floats, the label goes *after or inside* the caption, never before it; a label before the caption binds to the wrong counter and quietly reports the section number instead of the figure number. That's the classic silent referencing bug, and it's worth reading that sentence twice.

6.2 Let cleveref speak

Writing "Figure `\ref{fig:...}`" by hand a few hundred times invites inconsistency: Figure here, Fig. there, a missing non-breaking space somewhere. Delegate the words:

```
% preamble.tex (load AFTER hyperref)
\usepackage[capitalize,noabbrev]{cleveref}

% in prose:
As \cref{fig:methods-pipeline} shows ...
\cref{eq:methods-loss,eq:methods-reg} together imply ...
```

```
see \crefrange{sec:methods-data}{sec:methods-eval}
```

`\cref` emits "Figure 4.7", "Equations 3.2 and 3.4", "Sections 3.1 to 3.3", with correct names, plurals, and spacing, everywhere, forever. The two options shown match formal thesis style (capitalized, unabbreviated names); set them once and the entire document is consistent by construction.

The one thing to memorize about `cleveref` is load order: it comes *after* `hyperref`, which itself comes late in the preamble. The pairing `hyperref` → `cleveref` as the final two content packages is a convention that prevents a whole genre of mysterious errors, and Appendix A's preamble encodes it.

6.3 Numbering policy

Decide numbering depth once, in the preamble, instead of arguing with it per chapter:

```
\setcounter{secnumdepth}{2}   % number down to subsections
\setcounter{tocdepth}{1}      % TOC shows chapters + sections
```

Depth 2 (numbered subsections) is the common thesis default; anything deeper than 3 reads as bureaucracy. Equations, figures, and tables number per chapter automatically under `report` (4.7 means chapter 4), which is what committees expect, so there's nothing to do. If you want an equation referenced, it gets a numbered environment and a label; if it's not worth referencing, use the starred, unnumbered form. A thesis where every displayed equation is numbered "just in case" buries the ones that matter.

6.4 Page references and the last resort

Occasionally you want a page: "the full derivation appears on page 87." That's `\pageref`, or `\cpageref` under `cleveref`, and it earns the same never-hard-code rule. If you catch yourself typing a literal section number, equation number, or page number anywhere in the prose, stop and plant a label instead. Literal numbers are correct for exactly one build.

The genuinely last-resort tool is `\nameref` ("see the chapter *Methods*"), useful in front matter where numbering may differ. Everything else is `\cref`.

6.5 The ?? diagnostic

You'll sometimes see `??` in the PDF where a number should be. It has exactly three causes, in descending likelihood:

1. **One more pass needed.** References resolve from the previous run's data. `latexmk` normally handles this; a manually interrupted build doesn't. Rebuild.
2. **The label doesn't exist**: a typo (`fig:` vs `tab:`), or the chapter containing it is excluded by `\includeonly`, so its labels are invisible this build. Both are benign; the log's "undefined references" list names the offender.
3. **The label is duplicated**: two `\label` lines with the same key; LaTeX warns "multiply defined." The naming convention makes this rare; the warning makes it findable.

A clean thesis build ends with zero undefined-reference warnings, and Chapter 14 makes "zero warnings of the kinds we track" a habit rather than a hope.

What breaks here

- **Label before caption** silently yields wrong numbers. Caption first, label second. If a figure reference reads like a section number, this is why.
- **cleveref loaded before hyperref** errors or half-works. It's the last package. Full stop.
- **Renaming labels during a cleanup** orphans references scattered across chapters. If you must rename, project-wide search-and-replace, then a full build, then check the log's undefined list.
- **Hard-coded numbers** pasted from a PDF read-through ("as we saw in Section 2.3") during late-night editing. The fix is cultural, not technical: the word "Section" in prose should never be followed by a digit you typed.

7

Figures That Land

No LaTeX topic generates more thesis-month despair than figures: they float to strange places, arrive blurry, or overflow margins the week of the format check. All three failures have systematic cures, and they're all upstream, in how figures are produced, stored, and placed. Set this chapter up once and figures become the boring part of your thesis, which is exactly what you want them to be.

7.1 Produce them right: vector first

Figure quality is decided in your plotting tool, not in LaTeX. The rule: **vector formats (PDF, or the EPS your tool exports converted to PDF) for anything made of lines and text**, which means plots, diagrams, flowcharts; **raster (PNG) only for photographs and screenshots**, at 300 dpi for print. Vector graphics scale losslessly, match your document's crispness at any zoom, and usually weigh less than a high-resolution PNG. JPEG has no place in a thesis except true photographs, and even there PNG is safer for anything with sharp edges.

Two upstream habits multiply the payoff. Export plots at roughly their final physical size (a figure meant to be 4.5 inches wide, exported 4.5 inches wide) so fonts inside the plot land at a sensible size instead of being scaled into microtype. And set your plotting tool's font size so that labels end up near your document's footnote size; a thesis whose plot axes shout louder than its prose reads like a poster.

Every figure file lives in `fig/`, named by the convention from Chapter 3: `fig/ch4-loss-curves.pdf`. Keep the script that generated each plot; Chapter 10 returns to why.

7.2 The standard figure block

```
\usepackage{graphicx}   % preamble, once

\begin{figure}[htbp]
  \centering
  \includegraphics[width=0.8\textwidth]
      {fig/ch4-loss-curves.pdf}
  \caption{Training loss under the three schedules. The adaptive
           schedule (solid) reaches the plateau in half the epochs.}
  \label{fig:results-loss}
\end{figure}
```

Four deliberate choices in ten lines. Width is expressed relative to `\textwidth`, so figures survive any margin change (recall the re-flow cascade of Chapter 4). The caption says something, a sentence a committee member skimming only floats can learn from, not just "Loss curves." The label follows the caption, per Chapter 6's bug warning. And the placement spec is `[htbp]`, which brings us to the emotional core of this chapter.

7.3 Make peace with floating

A `figure` environment *floats*: LaTeX places it where layout works, near where you asked, not exactly where you typed it. Students fight this with `[h!]`, and the fight makes placement worse, because an impossible demand falls back to worse positions and jams the float queue behind it.

The armistice that works: write `[htbp]` (here, top, bottom, or its own page, in that order of preference), reference every figure from prose with `\cref` so readers are steered to it wherever it lands, and **postpone placement polish until the document is content-stable**. Every edit reflows everything; placement tuned in March is undone by April's revisions. In the final pass, the few genuinely stubborn floats get handled with two civilized tools: the `placeins` package's `\FloatBarrier` (floats may not drift past this line; excellent at section boundaries) and, rarely, promoting a figure to `[p]`, a dedicated float page, which is where a full-page figure belonged anyway.

If a draft for your advisor needs figures pinned mid-discussion, that's the one legitimate use of the `float` package's `[H]` (exactly here). Use it as a drafting aid, then return to `[htbp]` for the final document; a thesis built on `[H]` develops ragged half-empty pages that formatting offices notice.

7.4 Multi-panel and oversized figures

For side-by-side panels, `subcaption` is the modern package:

```
\usepackage{subcaption}   % preamble

\begin{figure}[htbp]
  \centering
  \begin{subfigure}{0.48\textwidth}
    \includegraphics[width=\linewidth]{fig/ch4-acc-a.pdf}
    \caption{Baseline}\label{fig:results-acc-a}
  \end{subfigure}
  \hfill
  \begin{subfigure}{0.48\textwidth}
    \includegraphics[width=\linewidth]{fig/ch4-acc-b.pdf}
    \caption{Adaptive}\label{fig:results-acc-b}
  \end{subfigure}
  \caption{Validation accuracy for both schedules.}
  \label{fig:results-acc}
\end{figure}
```

You get Figure 4.8 with panels (a) and (b), each independently referencable. Ignore older tutorials pushing `subfigure` or `subfig`, both long deprecated and both present in exactly the inherited templates Chapter 1 warned about.

For a genuinely wide figure (a timeline, a big architecture diagram), turn the page, don't shrink the content: `rotating`'s `sidewaysfigure` environment gives it a landscape page of

its own, and readers of bound theses are entirely used to rotating the volume. Shrinking a wide diagram to portrait width produces six-point labels, and six-point labels produce committee comments.

7.5 The list of figures

Your front matter will include `\listoffigures` (Chapter 13). It's built from captions, and multi-sentence captions bloat it. The fix, when a caption runs long, is the optional short form: `\caption[Training loss under the three schedules]{Training loss under the three schedules. The adaptive schedule...}`. Short form to the list, full sentence under the figure, everyone content.

What breaks here

- **Blurry figures** are raster exports of vector content. Re-export as PDF from the source tool; no LaTeX option can add resolution that isn't there.
- **Figures overflowing the margin** are absolute widths (`width=14cm`) meeting new margins. Relative widths only.
- **A figure pileup** (everything shoved to the chapter's end) usually traces to one impossible placement demand jamming the queue, or one float physically taller than the text block. Find the oversized one; the jam clears.
- **Missing-file errors after a move** mean paths broke: figures referenced outside `fig/` or with absolute paths. The layout from Chapter 3 prevents this permanently.

8

Tables Without Tears

Thesis tables fail in two directions: visually (vertical rules and double lines everywhere, the spreadsheet look) and structurally (ten-column monsters crushed into the text width). Both cures are systematic. This chapter gives you one canonical table pattern for the 90 percent case, and the escape hatches for wide, long, and numeric tables.

8.1 The canonical table

Professional tables in print follow three rules that will sound radical if you learned tables in a word processor: no vertical rules, no double rules, and exactly three horizontal rules (top, below the header, bottom). The `booktabs` package exists to make this effortless:

```
\usepackage{booktabs}   % preamble, once

\begin{table}[htbp]
  \centering
  \caption{Datasets used in the evaluation.}
  \label{tab:methods-datasets}
  \begin{tabular}{lrr}
    \toprule
    Dataset   & Samples & Classes \\
    \midrule
    MNIST-s   &  10\,000 &  10 \\
    CIFAR-c   &  50\,000 & 100 \\
    Field-24  &   3\,214 &   7 \\
    \bottomrule
  \end{tabular}
\end{table}
```

Note three thesis-specific conventions riding along. The caption sits *above* the table (tables are read top-down; figure captions go below, table captions above, and formatting offices know the convention). The label follows the caption as always. And column types are honest: `l` for text, `r` for numbers, because numbers compare by their right edges.

That table pattern, `booktabs` rules and all, covers most tables in most theses. Copy it, change the columns, done.

8.2 Numbers that line up: siunitx

When a column holds measurements, alignment on the decimal point is what makes them readable, and `siunitx` does it with the `S` column type:

```
\usepackage{siunitx}    % preamble

\begin{tabular}{l S[table-format=2.2] S[table-format=1.3]}
  \toprule
  Method    & {Time (s)} & {Error} \\
  \midrule
  Baseline  & 12.70      & 0.041 \\
  Ours      &  3.02      & 0.008 \\
  \bottomrule
\end{tabular}
```

`table-format=2.2` reserves two integer and two decimal digits, so columns of results align like accounting. Header cells in an `S` column go in braces (they're text, not numbers). `siunitx` also gives you `\SI{3.2}{\milli\second}` for units in prose, and consistency of units across 150 pages is exactly the kind of thing committees notice.

8.3 Wide tables

A table wider than the text block has four escapes, in order of preference:

1. **Cut columns.** Half the too-wide tables I've reviewed contained a column nobody discussed (a redundant ID, a constant). The best table fix is editorial, not technical.
2. **Let text columns wrap** with `tabularx`: it stretches designated `X` columns to make the table exactly `\textwidth`, wrapping their contents. Ideal when one column is descriptions and the rest are short.
3. **Shrink the font, not the table**: wrapping the `tabular` in `\small` or `\footnotesize` is legitimate down to footnote size, and no further.
4. **Rotate**: `sidewaystable` from `rotating`, the same landscape treatment as wide figures. For the true monster, rotation plus `footnotesize` is the honorable end of the line. What is never on the list: `\resizebox` scaling a table to fit, which produces mismatched font sizes that formatting reviewers flag on sight.

8.4 Long tables

A table that outgrows one page (your 6-page hyperparameter appendix, the 300-row species list) can't float; it has to break across pages like prose. That's `longtable`:

```
\usepackage{longtable}  % preamble

\begin{longtable}{lrr}
  \caption{Complete run configurations.}
  \label{tab:appA-runs} \\
  \toprule
  Run & Steps & Seed \\
  \midrule
  \endfirsthead
  \toprule
  Run & Steps & Seed \\
  \midrule
  \endhead
  \bottomrule
  \endfoot
```

```
  r-001 & 10\,000 & 11 \\
  r-002 & 10\,000 & 12 \\
  ...
\end{longtable}
```

The block up top defines what repeats: `\endfirsthead` ends the first page's header, `\endhead` the header for every later page, `\endfoot` the rule that closes each intermediate page. It's boilerplate you write once per long table, and long tables belong almost exclusively in appendices; a main chapter interrupted by six pages of rows has an editorial problem no package fixes.

8.5 Notes under tables

Results tables often need footnotes ("averaged over 5 seeds"). Table footnotes live *with the table*, not in the page's footnote stream; the lightweight standard is the `threeparttable` package, which wraps the tabular and gives you `tablenotes` beneath, sharing the table's width. Symbols (`a`, `b`) mark cells; the notes explain. It reads exactly like the journals in your bibliography, which is the level of polish a results chapter wants.

8.6 Where table data comes from

A quiet word about workflow, expanded in Chapter 10: results tables change. Numbers get re-run; a reviewer's comment regenerates half a chapter's data. Retyping numbers into LaTeX by hand at 1 a.m. is how transcription errors reach committees. If your pipeline can *write* the table body (a script emitting the rows into `tab/ch4-results.tex`, pulled in with `\input`), the table updates when the data does, and the thesis's numbers are the pipeline's numbers, always. Even doing this for just your two or three main results tables removes the most dangerous manual copying in the whole document.

What breaks here

- **The spreadsheet look** (vertical rules everywhere) marks a table pasted from a generator or an old template. The canonical pattern above replaces it in two minutes.
- **`Misplaced \noalign` or similar errors** inside tables usually mean a rule command in the wrong place or a missing `\\` at a row's end; check the row above the reported line.
- **Columns misaligned in `S` columns** mean un-braced header text. Headers in braces, always.
- **A `longtable` whose column widths jump** at a page break needs a second compile; `longtable` measures across runs. `latexmk` handles it, interrupted builds don't.

9

Mathematics at Thesis Scale

You already typeset equations; that's problem-set LaTeX, and if you need that foundation, *LaTeX for Students* builds it properly. What a thesis adds is scale: hundreds of equations, dozens of theorems or model definitions, and notation that must stay consistent from page 12 to page 187. This chapter is about the machinery that keeps mathematics coherent across a long document, whatever your field's density of it.

9.1 The amsmath baseline

Load the suite once, and let it be the only math setup:

```
\usepackage{amsmath}
\usepackage{amssymb}   % omit if your font package provides symbols
\usepackage{mathtools} % amsmath refinements; safe to add
```

At thesis scale, three `amsmath` environments do almost all the display work, and choosing correctly among them is most of "looking professional":

- `equation`: one numbered equation.
- `align`: several equations vertically aligned on a relation sign, each line numberable; the workhorse for derivations.
- `aligned`/`split` inside `equation`: a multi-line derivation that is morally *one* statement and should carry *one* number.

What should disappear from your muscle memory: `eqnarray` (deprecated for decades, spacing is wrong, lives on in old templates) and `$$...$$` for display math (plain-TeX syntax with subtle spacing bugs; `\[... \]` is the LaTeX form).

9.2 A numbering policy you set once

Chapter 6 set the principle: number what you reference, star what you don't. The thesis-scale refinements:

- **Per-chapter numbering** ((4.7) means chapter 4) is automatic under `report` once the preamble says `\numberwithin {equation}{chapter}`. Committees expect it; page-long equation numbers like (127) expect nobody to find anything.
- **In `align`, star the lines you won't cite** with `\nonumber` (or use `align*` when citing none). A derivation where all nine lines are numbered is noise; number the destination.
- **Never renumber by hand** to match a paper you're adapting. Your thesis is its own document; its equations follow its own counters, and `\cref` does the talking.

9.3 Theorems, definitions, models

Even outside pure mathematics, theses formalize things: a Definition of the model, Assumptions 1–3, maybe a Proposition about convergence. Set the environments up once in the preamble with `amsthm`:

```
\usepackage{amsthm}
\theoremstyle{plain}
\newtheorem{theorem}{Theorem}[chapter]
\newtheorem{proposition}[theorem]{Proposition}
\newtheorem{lemma}[theorem]{Lemma}
\theoremstyle{definition}
\newtheorem{definition}[theorem]{Definition}
\newtheorem{assumption}[theorem]{Assumption}
\theoremstyle{remark}
\newtheorem*{remark}{Remark}
```

The design choice that matters is the shared counter (the `[theorem]` option): Definition 3.1 is followed by Theorem 3.2, one sequence per chapter, so a reader hunting "3.4" finds it without knowing its species. The proof environment comes free with `amsthm`, QED square included. If your field writes no proofs, you still want `definition` and `assumption`; committees quote assumptions back at you by number, and it's better to have issued the numbers yourself. (For what should go *inside* those environments, the writing itself, my *How to Write Mathematics* is the companion volume to this one.)

9.4 Notation as a contract

The silent killer in long documents is notation drift: the vector that's bold in chapter 2 and arrowed in chapter 5, the L that means loss here and length there. The cure is the same one Chapter 3 applied to metadata: define once, use everywhere. Your preamble grows a notation block:

```
% --- notation macros ---
\newcommand{\vect}[1]{\mathbf{#1}}      % vectors: \vect{x}
\newcommand{\mat}[1]{\mathbf{#1}}       % matrices
\newcommand{\R}{\mathbb{R}}
\newcommand{\E}{\operatorname{E}}       % expectation
\newcommand{\loss}{\mathcal{L}}
\DeclareMathOperator{\argmin}{arg\,min}
```

Now `\vect{x}` is the *only* way a vector is written, and if your advisor prefers arrows over bold, you change one line in March instead of two hundred instances. The same trick guarantees operator spacing: `\DeclareMathOperator` gives `\argmin` the upright face and correct spacing everywhere, where hand-typed `argmin` would be italic letter-soup.

Two rules make the macro layer work at scale. Macros are *semantic*: name them for what they mean (`\loss`), not how they look (`\calL`), so meaning survives a notation change. And the block stays small, twenty or thirty lines; a private macro language of two hundred commands makes your source unreadable to collaborators and Overleaf debugging partners. Chapter 12 adds the reader-facing twin of this block: the printed notation list.

9.5 Big objects and old friends

Assorted thesis-scale mechanics worth adopting deliberately: `cases` for piecewise definitions; `pmatrix` and friends for matrices; `\left( ... \right)` for delimiters that must stretch, but plain `(` for ones that don't (auto-stretching every bracket subtly inflates spacing); and `\text{...}` for words inside math, never bare letters, so "where c is constant" doesn't typeset as the product $constant$. Line-breaking a display that overflows is `multline` or a re-derivation into `align`; an equation running into the margin is feedback about the equation, and often the best fix is mathematical (define an intermediate quantity) rather than typographic.

What breaks here

- **Notation drift** is invisible until a committee member reads chapters out of order. The macro block is the fix, and adopting it in month one costs nothing.
- **`eqnarray` and `$$`** arriving via pasted old templates bring wrong spacing with them. Evict on sight.
- **Double superscript errors** usually mean a pasted formula with invisible characters or an unbraced exponent: `x^2n` means $x^2 n$; you wanted `x^{2n}`.
- **Missing $ inserted** at some line far from your edit almost always traces to an unclosed math delimiter above. Check the nearest preceding formula, not the reported line.

10

Code and Algorithms

If your thesis involves computation, and most now do, two kinds of code appear in it: source listings the reader should see, and pseudocode describing algorithms. LaTeX handles both well, with one setup each. The chapter closes with the workflow rule that keeps listings honest, which matters more than any styling.

10.1 Listings with the listings package

The standard tool is `listings`: pure LaTeX, no external dependencies, entirely sufficient for a thesis. Configure it once:

```
\usepackage{listings}
\usepackage{xcolor}
\lstset{
  basicstyle=\small\ttfamily,
  keywordstyle=\bfseries,
  commentstyle=\itshape\color{gray},
  numbers=left, numberstyle=\tiny\color{gray},
  frame=single, framesep=6pt,
  breaklines=true, showstringspaces=false,
  captionpos=b
}
```

Then each listing is a float-like block with a caption and label, referenced like any figure:

```
\begin{lstlisting}[language=Python, float=htbp,
    caption={The adaptive step-size update.},
    label={lst:methods-stepsize}]
def update(step, grad, m):
    m = beta * m + (1 - beta) * grad
    return step * m
\end{lstlisting}
```

Choices worth defending at a defense: grayscale styling, because theses are printed in black and white and syntax-color rainbows die in the printer; line numbers on, so your examiner can say “line 4”; and captions below, matching figures.

A note on `minted`, the alternative with richer highlighting via Python’s Pygments: it’s genuinely nicer output, and it adds an external dependency (`-shell-escape` plus a Python toolchain) that must exist on *every* machine that builds the thesis, including Overleaf (where it works) and the borrowed laptop in a crisis (where it may not). My thesis advice

is conservative: `listings` by default; `minted` only if you control every build environment and the highlighting genuinely earns its keep.

10.2 What code belongs in a thesis

An editorial rule before the mechanics continue: the thesis shows *illustrative* code, not the repository. Ten to thirty lines that carry an idea, the core update rule, the clever data structure, the query that does the work, belong as listings. Four hundred lines of plumbing belong in your code repository, cited from Chapter 11's machinery (repositories get DOIs via Zenodo, and "the implementation is available at [...]" is now standard thesis prose). A listings appendix reproducing entire files impresses no committee and guarantees the printed code is stale by the defense.

10.3 Pseudocode with algpseudocode

When the contribution is the algorithm, not its Python clothing, pseudocode communicates better and ages better. The modern pairing is `algorithm` (the float) with `algpseudocode` (the notation):

```
\usepackage{algorithm}
\usepackage{algpseudocode}

\begin{algorithm}[htbp]
  \caption{Adaptive sparse update}
  \label{alg:methods-update}
  \begin{algorithmic}[1]
    \Require step size $\eta$, decay $\beta$
    \State $m \gets 0$
    \For{$t = 1, \dots, T$}
      \State $m \gets \beta m + (1-\beta) g_t$
      \If{$\lVert m \rVert > \tau$}
        \State $x \gets x - \eta\, m$
      \EndIf
    \EndFor
  \end{algorithmic}
\end{algorithm}
```

The `[1]` numbers every line, and Algorithm floats join figures and tables in the reference system: "`\cref{alg:methods-update}`, line 5." One convention to hold: mathematics inside pseudocode uses your notation macros from Chapter 9, so the algorithm and the surrounding analysis speak one language. (If an inherited template uses the older `algorithmic` or `algorithm2e` syntax, they conflict with `algpseudocode`; pick one family per document.)

10.4 Keeping listings truthful

Now the workflow rule. A listing pasted into the thesis in February is a lie by June: the code moved on. Two habits prevent the classic defense moment where a committee member finds the bug in the printed version you fixed months ago.

First, **listings can be included from files, and should be**:

```
\lstinputlisting[language=Python, firstline=12, lastline=24,
    caption={The update rule as implemented.},
    label={lst:methods-impl}]{code/update.py}
```

Point it at a file in the project's `code/` directory, ideally a copy exported from the real repository at a tagged commit. The listing then has a provenance: it *is* the code, lines 12–24 of a file you can name.

Second, **the same rule powers generated results**: the table-emitting scripts of Chapter 8 and the plot-generating scripts of Chapter 7 live in the repository too, and a top-level "regenerate everything" script (even a ten-line shell file) is the difference between "I think these are the final numbers" and "these are the final numbers." Come the defense, you want exactly one uncertainty in the room, and it should be a science question, not a which-version-is-this question.

What breaks here

- **Tabs and Unicode in pasted code** produce ragged indentation or "invalid character" errors. Configure your editor to paste as spaces; keep listings ASCII, or set `listings`' `extendedchars` carefully.
- **Overlong lines** escape the frame. `breaklines` softens it; genuinely reformatting the snippet for print (shorter names, wrapped calls) is usually the honest fix.
- `minted` **breaks.** The cause is usually the `-shell-escape`/Pygments dependency. The fix at 11 p.m. is switching that listing to `listings`, which is why default matters.
- **Stale listings** contradicting the repository at defense time. `\lstinputlisting` from a tagged export; never paste twice.

Part III

The Scholarly Apparatus

11

The Bibliography System

A thesis bibliography is a database problem wearing a formatting costume. One to three hundred sources, cited from six chapters, formatted to a style your department dictates, and revised until the last week: managing that by hand is how bibliographies become the most error-dense pages of a thesis. This chapter builds the pipeline that makes them the most reliable ones instead.

11.1 The architecture: biblatex + Biber

Modern LaTeX bibliography is the `biblatex` package with the `biber` backend:

```
% preamble.tex
\usepackage[backend=biber, style=numeric-comp, sorting=nyt]{biblatex}
\addbibresource{bib/references.bib}

% main.tex, at the end:
\printbibliography
```

If you learned the older `natbib`/BibTeX route, or your inherited template uses it: it still works, but choose `biblatex` for anything started today. It handles Unicode names, URLs, DOIs, and modern entry types (datasets, software, preprints) natively, and its styles are options, not separate ecosystems. The one operational fact to internalize: **Biber is a separate program** that `latexmk` runs for you. If you build by hand and citations won't update, that's the missing Biber pass; Chapter 2 already made building by hand your exhabit.

Style is a department question with a one-line answer. Use `style=numeric-comp` for the bracketed-numbers world of engineering, CS, and most sciences; use `style=authoryear` for the (Name, 2024) world; use `style=apa` or `ieee` where those are named explicitly. Changing your mind in month six is changing one option, which is the entire point of the architecture.

11.2 One database, curated like code

All references live in `bib/references.bib`, and the file deserves the respect you give source code, because that's what it is. A well-kept entry:

```
@article{shannon1948mathematical,
  author  = {Shannon, Claude E.},
  title   = {A Mathematical Theory of Communication},
```

```
  journal = {Bell System Technical Journal},
  year    = {1948},
  volume  = {27},
  pages   = {379--423},
  doi     = {10.1002/j.1538-7305.1948.tb01338.x}
}
```

The craft rules that keep 200 entries healthy:

- **One key convention, forever**: `author-year-word`, so the entry above is `shannon1948mathematical`. Guessable keys mean you cite from memory without opening the file.
- **Harvest from publishers, then clean.** Every journal and arXiv page exports BibTeX. Paste, then spend the thirty seconds: fix mangled capitalization with braces (`{DNA}`, `{Bayesian}`), complete missing fields, delete the export junk fields. Auto-exported entries are drafts, not truth; garbage pasted in September is a bibliography error printed in May.
- **Prefer DOIs over URLs**, and keep exactly one of the redundant locator fields; an entry with DOI, URL, and a broken `eprint` field prints redundantly under some styles.
- **Never delete entries mid-thesis.** Uncited entries don't print (`biblatex` only outputs what you cite), so the database can be a superset of the thesis, including the papers you *might* cite. It's your reading list too.

11.3 Feeding from a reference manager

If you live in Zotero (or JabRef, which edits `.bib` natively), let it do the collecting: Zotero's Better BibTeX extension exports a continuously synchronized `.bib` with stable keys, and the browser plugin turns "paper I just found" into "entry in my database" in one click. The rule that keeps the pipeline sane: **pick one source of truth**. Either the manager owns the `.bib` (you never hand-edit the export) or the `.bib` is hand-curated (the manager is just a reading tool). Editing both is a merge conflict with your own past self, and it always ends in lost fixes.

11.4 Citing well in prose

The commands you'll actually use, in `biblatex` dialect: `\cite{key}` for the bare citation, `\textcite{key}` when the authors are the sentence's subject ("Shannon [12] showed" / "Shannon (1948) showed"), and `\cite[Thm. 3]{key}` with a postnote to point inside a source, the precision Chapter 6's philosophy demands of external pointers too. Multi-citations go in one command (`\cite{a,b,c}` gives "[3–5]" under `numeric-comp`), never as comma-chained singles.

Thesis-specific citing judgment, the part templates can't supply: cite the version of record (the journal paper, with the arXiv id as a bonus field, not instead of it); cite datasets and software you used, with the `@dataset`/`@software` entry types `biblatex` provides, since committees increasingly check; and in the background chapter, resist citation carpeting. Twelve citations after one sentence signals anxiety, not scholarship; cite the load-bearing sources and let the survey paper you cite carry its own bibliography.

11.5 Splitting and polishing

Two finishing moves. If your university wants per-chapter bibliographies (some do), `biblatex` does it with `refsection` environments and per-section `\printbibliography`; confirm the requirement before building it, because most offices want one global list. And before submission, run the bibliography's own QA pass: build, then read the printed list start to finish once. You're hunting mangled capitals, first names where initials belong (pick `giveninits=true` for consistency), duplicate entries under two keys, and the one entry whose year is `2924`. Thirty minutes, once, on the pages every committee member reads.

What breaks here

- **Citations print as bold keys or [?]**: Biber hasn't run (build with `latexmk`), or the key doesn't exist (check the log's "undefined citation" list for the typo).
- **"Sorting scheme not found" or style errors** after copying preamble lines from the internet: mismatched `biblatex` option combinations. Start from the three options shown here; add nothing without reading its documentation.
- **Title case mangled** ("dna" or "bayesian" in print): unprotected capitals; brace them in the `.bib`.
- **Biber cache corruption** (rare, memorable): mysterious Biber crashes after an interrupted run. The fix is deleting the build's auxiliary files (`latexmk -C`) and rebuilding; the database itself is never touched.

12

Acronyms, Glossaries, and Notation Lists

Most theses need some version of "define it once, expand it on first use, list it in the front matter": acronyms in every field, symbol tables in quantitative ones, occasionally a true glossary of terms. LaTeX has machinery for all three at three levels of ambition, and the honest job of this chapter is helping you pick the *lowest* level that satisfies your university, because over-engineering the glossary is a classic thesis displacement activity.

12.1 Level zero: none of it

Check the rulebook first. If your university doesn't require a list of abbreviations and your field uses a dozen acronyms total, the professional solution is prose discipline: expand each acronym at first use ("support vector machine (SVM)"), use the acronym thereafter, and skip the machinery entirely. A three-item abbreviation list is furniture nobody consults. This level is underrated and completely legitimate.

12.2 Level one: acronyms with the acro package

Required list of abbreviations, standard needs? The `acro` package is the calm middle path:

```
% preamble.tex
\usepackage{acro}
\DeclareAcronym{svm}{short=SVM,
  long=support vector machine}
\DeclareAcronym{mse}{short=MSE,
  long=mean squared error}

% in prose:
We train an \ac{svm} and report \ac{mse}.
% front matter:
\printacronyms[name=List of Abbreviations]
```

`\ac` does the thinking: first use prints "support vector machine (SVM)", every later use prints "SVM", and the printed list contains exactly the acronyms you actually used. That first-use logic is per-document, which surfaces one thesis-scale judgment call: committees often read chapters in isolation, so many advisors prefer first-use expansion *per chapter*. `acro` obliges with `\acresetall` at each chapter's start if yours does. Declarations live in the

preamble (or one `acronyms.tex` input), never scattered through chapters, per the hygiene rule of Chapter 3.

12.3 Level two: the glossaries package

When the university mandates a formal glossary or nomenclature section with specific formatting, or you need terms *and* symbols *and* acronyms as separate printed lists, you've reached the domain of `glossaries` (and its modern face, `glossaries-extra`):

```
\usepackage[acronym,symbols]{glossaries-extra}
\makeglossaries

\newglossaryentry{overfitting}{name=overfitting,
  description={fitting noise in training data at the expense
    of generalization}}
\newabbreviation{cnn}{CNN}{convolutional neural network}

% prose: \gls{overfitting}, \gls{cnn}
% front matter: \printglossaries
```

It's the most capable system and the most operationally demanding: entries are compiled by an external indexing step (`latexmk` manages it with a one-line `.latexmkrc` addition, which Appendix A includes), and the package's option surface is large. My honest guidance: adopt `glossaries-extra` only when a requirement names it or level one demonstrably can't express what you need. A thesis is not improved by the most powerful glossary system; it's improved by a finished chapter 5.

12.4 The notation list

Quantitative theses want a "List of Symbols" in the front matter: η, learning rate; $\mathcal{D}$, the training set. You could drive this through `glossaries`' symbol lists, and for a handful of symbols there's a lighter path most committees are equally happy with: a hand-maintained table.

```
% frontmatter/notation.tex
\chapter*{Notation}
\begin{tabular}{@{}ll@{}}
  $\eta$        & learning rate \\
  $\mathcal{D}$ & training dataset \\
  $\vect{x}$    & input vector (bold denotes vectors) \\
  $\loss$       & loss function \\
\end{tabular}
```

The integrity rule that makes either implementation work: the notation list is the *printed twin of the macro block* from Chapter 9. One line in the macro block, one row in the table. When you add `\loss`, you add its row in the same commit; the pairing is what keeps page 187 consistent with page 12, and the list honest. A notation list that drifts from usage is worse than none, because committees use it as the key while reading.

12.5 Order and placement

Where these lists go is a rulebook question with a common default: after the table of contents and lists of figures/tables, before the first chapter, in the order abbreviations then symbols then glossary. Front-matter assembly, including these pages, is Chapter 13's job; here, just note that every list is one `\input` line in `main.tex`, exactly like everything else in the skeleton.

What breaks here

- **An empty glossary prints** when the indexing step never ran: the `.latexmkrc` line is missing, or you built by hand. Also check you actually `\gls`-referenced entries; unused entries don't appear by default.
- **First-use expansions in odd places** (an acronym expanding in a caption or the abstract before "first use" in chapter 1) are the packages being technically correct about build order. Standard fixes: use the short-only commands (`\acs`, `\glsxtrshort`) in abstracts and captions.
- **Undefined control sequence `\ac`** in a chapter compiled standalone or by a collaborator: declarations live in the preamble the chapter didn't load. The skeleton's one-preamble rule prevents it; ad hoc partial builds resurrect it.
- **The drifting symbol table**: notation list says bold vectors, chapter 6 says arrows. The macro-block pairing above is the cure; if you're reading this item after the fact, a project-wide search for the raw notation is the audit.

13

Front and Back Matter

The front matter is the most rule-bound and least intellectually interesting part of the thesis, which makes it perfect assembly-line work: build each page as its own small file, wire them into `main.tex` in the mandated order, verify page numbering, and never think about it again. This chapter runs the line.

13.1 The mandated order

Every university publishes its sequence; the common core, in the usual order:

1. Title page (built in Chapter 4, unnumbered)
2. Copyright or declaration page ("I declare this work is my own…", often with a signature line)
3. Abstract
4. Acknowledgments (sometimes after the abstract, sometimes before; check)
5. Table of contents
6. List of figures, list of tables
7. Abbreviations / notation (Chapter 12)
8. Optional dedication and epigraph

Each is a file in `frontmatter/`, each one `\input` line in the conductor, in the rulebook's order. The whole block runs under roman page numbers with the switch to arabic at chapter 1 (Chapter 4 set this up). Two page-numbering details formatting offices actually check: whether the title page counts as page i while showing no number (usually yes; `\thispagestyle{empty}` on a counted page does exactly that), and whether every subsequent front-matter page shows its roman numeral (usually yes).

13.2 The pages with content

The abstract is prose you'll write near the end (write it after the conclusion chapter; it will be better). The mechanical side: universities often cap it (350 words is a common figure tied to historical microfilm rules) and sometimes require a specific heading format. Structure it as four moves: context, gap, what you did, what you found. No citations, no undefined acronyms, per the same isolation logic as Chapter 12's abstract rule.

The declaration is boilerplate your university supplies verbatim. Copy it exactly, including the awkward phrasing; this page is pattern-matched by readers whose job is pattern-matching. If co-authored work appears in the thesis (increasingly common with publication-based theses), the declaration or a dedicated "statement of contributions" page is where you say precisely who did what, and your university's wording requirements here

are strict; this is a page to draft early and clear with your advisor, not to improvise in submission week.

Acknowledgments are yours. The only engineering note: people check how they're thanked. Spell names right, include the funding acknowledgment your grant legally requires (granting agencies publish the required sentence), and keep the file out of any draft you circulate publicly before the defense if it contains surprises.

13.3 The generated lists

Table of contents, list of figures, list of tables: three commands, zero maintenance.

```
\tableofcontents
\listoffigures
\listoftables
```

They rebuild from your actual structure every compile, which is why hand-editing them (a thing people genuinely attempt) is both impossible and unnecessary. Control what they show at the source: TOC depth via `tocdepth` (Chapter 6), list entries via the short-caption forms (Chapter 7). If a heading is too long for the TOC, `\chapter` and friends take the same optional short form: `\section[Short TOC title]{The long expressive heading that would wrap badly}`.

One habit: skim all three lists in every full build's PDF. They're the highest-density view of the document, and they surface problems fast: the section title with a typo, the figure that lost its caption, the chapter that's mysteriously absent because of a stale `\includeonly`.

13.4 Appendices and the back

The switch is one command in the conductor:

```
\appendix
\include{appendices/appA-data}
\include{appendices/appB-proofs}
```

After `\appendix`, chapters letter themselves (A, B) and their floats number accordingly (Table A.3). What belongs there: the long tables of Chapter 8, secondary proofs, instrument details, questionnaire text, extended results. What doesn't: anything the examiner *must* read to follow the argument; appendices are for verification, not continuation. Keep each appendix as disciplined as a chapter, labels and all, because "see Appendix B" pointers need `\cref` like everything else.

The bibliography (`\printbibliography`) normally follows the appendices or precedes them per your rulebook; some universities also want an index or a CV page at the very end. All of it is more `\input` lines in the conductor, in mandated order, which by now is the least surprising sentence in this book.

13.5 The compliance dry run

The front and back matter are exactly what the graduate school's format check reads, so schedule a dry run: many offices offer a preliminary format review months before the deadline, and those that don't will still answer a title-page email (Chapter 4). Build the full PDF, check it against the rulebook's sample pages item by item (order, numbering, capitalization

of headings, the abstract's word count), and file the annotated rulebook with your project. Submission week, the whole apparatus gets checked once more by Appendix D's list, and that's the last time you look at any of it.

What breaks here

- **Page-number style wrong on one page**: usually a missing `\thispagestyle`, or the roman/arabic switch placed one page late. Fix at the conductor level, not by hacking individual pages.
- **TOC out of date** (a renamed chapter showing its old title): one more `latexmk` pass; the lists lag one build behind structure changes when a run is interrupted.
- **Abstract over the word cap** discovered at upload: the submission portal counts, you didn't. Count early; `texcount` does it from the command line.
- **Appendix floats numbered 6.x instead of A.x**: the appendix was pulled in *before* `\appendix` in the conductor, usually during a reorder. The conductor's short length is what makes this a ten-second diagnosis.

Part IV

The Production System

14

Keeping It Compiling

Somewhere around page 120, a thesis build stops being instant and starts being a system you operate. This chapter is the operator's manual: how to read errors at scale, how to keep the feedback loop fast, and the hygiene that prevents the 11 p.m. collapse, or ends it in minutes when it comes anyway.

14.1 Reading errors like a professional

LaTeX error messages are wordy but well-behaved: the log names a file and a line, and errors cascade, one real problem spawning dozens of downstream complaints. Two disciplines follow.

Fix the first error, then rebuild. Everything after the first error message is unreliable narration. Editors' error panels tempt you to triage the whole list; resist, because items 2–40 are usually item 1's shrapnel.

Trust the file, distrust the line. The named file is almost always right; the line number points to where LaTeX *noticed*, which for unclosed things (braces, environments, math delimiters) is downstream of where you *sinned*. For any "`File ended while scanning`" or "`Missing $ inserted`" class of error, search *upward* from the reported line for the unclosed thing. Chapter 9 said this for math; it's the general law.

The errors worth memorizing, because they're 80 percent of thesis-scale failures: `Undefined control sequence` (typo, or a package you use isn't loaded, often after moving text between projects); `Missing $ inserted` (math delimiters); `File ended while scanning use of ...` (unclosed brace or environment, often in a caption); `! LaTeX Error: \begin{...} ended by \end{...}` (mismatched environment, check the block you just edited); and `Undefined references` warnings (Chapter 6's `??` diagnostic).

14.2 Bisection: the debugging move

When the error is opaque and the morning's edits touched five files, don't stare, bisect. You have two binary-search axes, and they resolve almost anything in minutes:

- **Across chapters**: point `\includeonly` at half the chapters. Error persists? It's in that half (or the preamble). Gone? Other half. Repeat; three or four builds isolate the file.
- **Within a chapter**: `\endinput` anywhere in a file makes LaTeX stop reading it there. Walk it down the file; when the error disappears, you've passed it. This beats commenting out regions because it's one line you move.

Bisection works because of the skeleton: independent chapter files, one preamble, no settings hiding in content. The architecture of Chapter 3 was also, quietly, the debugging architecture.

14.3 Keeping the loop fast

Slow builds don't just waste minutes; they change how you write, discouraging the compile-often habit that catches errors while they're one edit old. The speed toolkit, in order of impact:

1. `\includeonly`, again. Editing one chapter with a two-second loop is the single biggest quality change in thesis writing. It's also why this book keeps calling it oxygen.
2. **The draft option**: `\documentclass[12pt,draft]{report}` replaces images with boxes and skips some finishing work. Good for text-editing sessions; just never send a draft-mode PDF anywhere, and yes, that has happened to real people.
3. **Externalize the expensive**: if you generate plots with TikZ/pgfplots inside the document, learn its externalization feature (figures compile once, then cache), or pre-render heavy figures to PDF via their source scripts (Chapter 7's workflow already implies this).
4. **Don't chase exotic caching** (precompiled preambles and friends) unless builds are genuinely painful; complexity in the build is risk at the deadline, the recurring theme of every tool choice in this book.

14.4 Warnings policy

A 150-page build emits a wall of warnings, most harmless, a few telling you about real defects. The sustainable policy is not "read every warning every build" (nobody does) but **zero tolerance on three classes, checked on every full build**:

- **Undefined references or citations**: always a real defect (Chapters 6 and 11).
- **Multiply defined labels**: always a real defect, usually a copy-paste.
- **Overfull \hbox beyond a few points**: text protruding into the margin, which formatting offices measure. Small ones (under ~5pt) are invisible; big ones need a rewrite, a hyphenation hint, or a resized float.

`latexmk`'s summary and your editor's log filter both surface these classes; a clean full build before every advisor draft is the habit. The final-pass sweep gets one extra tool: `\usepackage[draft]{showlabels}`? No. Simpler: the `refcheck` package flags labels nothing references, worth one run near the end to catch dead weight, then remove the package.

14.5 The auxiliary-file reset

Under the project directory, the build scatters state: `.aux`, `.toc`, `.bbl`, `.out`, Biber caches. Corrupted state from an interrupted build produces the strangest errors in LaTeX: crashes in files you didn't touch, phantom characters, a TOC from a parallel universe. Hence the first-aid rule quoted in half this book's "what breaks" sections:

```
latexmk -C     # delete all generated files, including the PDF
latexmk -pdf main.tex
```

A clean rebuild costs one full compile and resolves an outsized share of "it broke and I changed nothing" reports. It's the "turn it off and on again" of LaTeX, it works for the same reasons, and it is always the first thing to try when the error makes no sense.

What breaks here

This chapter *is* the what-breaks chapter, so instead, the triage order for any mystery, printable as Appendix C's header: (1) read the *first* error, note file and line; (2) look upward from the line for unclosed things; (3) `latexmk -C` and rebuild; (4) bisect with `\includeonly`, then `\endinput`; (5) if it's a package error after an update or on a new machine, suspect version drift and compare TeX installations. Fifteen minutes of this protocol resolves the overwhelming majority of thesis emergencies without help; the rest deserve a minimal example and a question to the TeX StackExchange, where minimal examples get answered fast.

15

Versions, Backups, and Your Advisor

Two facts about the months ahead: you will need an older version of your own thesis (the paragraph you deleted, the analysis your advisor un-deletes), and at least one machine involved will fail. Version control turns both from disasters into errands. This chapter sets up git for a thesis specifically, then builds the advisor feedback loop on top of it, including the tool that makes revision rounds civilized: `latexdiff`.

15.1 Git, thesis-shaped

You don't need git mastery; you need six commands and two files. In the project root, once:

```
git init
git add .
git commit -m "skeleton"
```

The first special file is `.gitignore`, which keeps generated files out of history:

```
# .gitignore
*.aux  *.log  *.toc  *.out  *.bbl  *.bcf  *.blg
*.fls  *.fdb_latexmk  *.run.xml  *.synctex.gz
main.pdf
```

Track sources, figures, the `.bib`, the class file; ignore everything the build regenerates. (Whether to track the PDF is a taste question; ignoring it keeps diffs clean, and Chapter 16's milestones tag the PDFs that matter.)

The second is a remote. A private GitHub or GitLab repository is the offsite backup that survives stolen laptops, and pushing is the backup act:

```
git add -A && git commit -m "ch3: rewrite data section"
git push
```

The cadence that works: **commit at every natural pause with a message naming the chapter, push at least daily**. Messages like "stuff" defeat the archaeology you'll eventually attempt ("when did I break the notation in chapter 5?"); messages like "ch5: switch to bold vectors" answer it. One-sentence-per-line sourcing (Chapter 3) is what makes those diffs readable, and this chapter is where that habit pays out.

Branches, rebasing, pull requests: skip them. A thesis is a single-author project; linear history on one branch is exactly enough, and every git feature you don't use is a git accident you can't have. The only branch-shaped move worth knowing is the tag:

```
git tag draft-2027-03-full   # the version your advisor got
```

Tag every draft you send anywhere. Revision conversations then have coordinates.

15.2 If you live in Overleaf

Overleaf's history panel covers the "older version of my own work" need, with labeled versions standing in for tags. Full project history and Git or GitHub synchronization are premium features; free accounts expose only a short history window. The GitHub route also has a real collaboration warning: synchronizing changes can displace tracked changes and comments, so don't mix active source editing in both systems without an agreed owner.

Whatever plan you use, download both the source zip and the built PDF at every draft milestone, dated, to storage outside Overleaf and outside the laptop used to write. A source export does not include the final PDF, project history, comments, or collaborator state. It is a recoverable copy of the source, not a complete archive of the collaboration.

15.3 Choose who owns the source

Most collaboration failures are ownership failures disguised as tool failures. Two people edit two copies, both assume theirs is current, and the merge happens by hand the night before a meeting.

Pick one of three models and write it at the top of `notes/collaboration.md`.

One source owner, PDF feedback. You own the source. Your supervisor comments on a dated PDF, and you apply the changes. This is the least technically demanding model and often the safest when the supervisor does not use LaTeX.

One shared project. Everyone works in the same cloud project. Use comments for questions and make source edits only in the chapter currently under review. Before a large revision, label a version and download an archive.

Git collaboration. The repository is the source. Each substantial revision happens on a branch and is merged after review. This fits a project with code and several technical collaborators, but it adds process a two-person thesis may not need.

Do not mix the models casually. A shared project plus emailed source files plus a local branch creates three candidates for "current." The tools are all good. The ambiguity is not.

15.4 The supervisor round, start to finish

A feedback round should have a visible boundary.

1. **Freeze the draft.** Commit or name the version `advisor-round-03`. Produce the PDF from that exact source.
2. **State the request.** "Please check the argument in Sections 3.2–3.4 and whether Figure 3.6 supports the conclusion." A specific reading request gets better feedback than "thoughts?"
3. **Collect comments in one place.** One annotated PDF, one shared comment thread, or one review branch.

4. **Triage before editing.** Mark each comment accept, discuss, or defer. A question from the supervisor may reveal reader confusion even when the proposed wording is not the right fix.
5. **Apply and record.** Make the changes, compile, and write a five-line response listing what changed and what still needs discussion.
6. **Close the round.** Commit the resolved state and archive the annotated PDF beside it.

The archive matters because comments often refer to page numbers. After the thesis grows, "p. 47" points somewhere else. Keeping the commented PDF beside the source version preserves the conversation.

15.5 Avoid conflicts before they exist

If more than one person edits source, divide work by file, not by line. One person owns `chapters/methods.tex` during a revision while another works on `chapters/results.tex`. Shared edits to `preamble.tex`, `metadata.tex`, and the bibliography need announcement because they affect the whole project.

Before beginning:

- pull or sync the current project;
- compile it unchanged;
- state which files you will edit;
- commit one purpose at a time;
- do not reformat an entire file while changing two sentences.

The last rule keeps diffs readable. A reviewer should see the argument change, not 600 lines whose wrapping changed because an editor applied a formatter.

15.6 A clean handoff

At submission, graduation, or a collaborator's departure, hand over more than a folder.

The receiving person needs:

- the exact source state and its final PDF;
- one build command and the engine it expects;
- the university class and local style files, where redistribution is allowed;
- the bibliography database and source figures;
- a short README naming the main file, build steps, and known warnings;
- the tag or commit that produced the submitted artifact.

Run the borrowed-machine test on the handoff bundle. If the recipient has to ask where `logo-final.pdf` came from, the bundle is not done.

15.7 The advisor loop

Advisors return feedback in three forms, and each has a clean workflow:

PDF annotations (the most common): they mark up the PDF in a reader; you work through comments against the source. The one upgrade worth making: send PDFs with line numbers during drafting (the `lineno` package, `\usepackage[pagewise]{lineno}` plus `\linenumbers`, removed for final builds) so their "the claim on page 34" becomes "page 34, line 12." Whole email threads of ambiguity disappear.

Inline notes in the source, for advisors who'll touch LaTeX: the `todonotes` package puts their `\todo{is this the right test?}` in the margin, and `\listoftodos` gives you the work queue. Also your own best drafting tool: every "fix this later" becomes a visible, listable margin note instead of a mental one, and the pre-submission check that the todo list is empty (Appendix D) closes the loop.

"What changed since last time?", the question every revision round starts with, has a dedicated answer:

```
latexdiff old/main.tex main.tex > diff.tex
latexmk -pdf diff.tex
```

`latexdiff` produces a compilable document with deletions struck through and additions underlined, the change-marked PDF committees ask for after a defense with revisions ("resubmit indicating changes"). For a multi-file thesis, use its `-flatten` option against the two tagged versions, and generate it from the tags, which is the payoff of tagging every sent draft. Practice the incantation once mid-thesis, not for the first time the week revisions are due; flattening plus heavy custom macros occasionally needs a workaround, and you want to know yours early.

15.8 Backup honesty

Version control is not backup theater, but it only protects what it holds. The audit, once: every input to the build is either in the repository or regenerable from something that is. Instrument data, the spreadsheet behind Table 4.2, the survey export: in the repo (small), or in the lab's archived storage with a pointer in the repo's README (large). The test worth actually performing, one afternoon, months before the deadline: **clone the repository onto a different machine, or a clean Overleaf import, and build.** If `latexmk -pdf main.tex` produces your thesis on borrowed hardware, your disaster story will be a paragraph, not a chapter. If it doesn't, you've just found every absolute path and untracked file at the cheapest possible moment.

What breaks here

- **Auxiliary files committed** make every diff a war zone. The `.gitignore` above, from day one; if they're already tracked, `git rm -cached` them once.
- **Sync conflicts from working on two machines** without pushing: pull before you start, push when you stop, and the problem never exists. If a conflict lands anyway, the one-sentence-per-line habit makes the conflict markers legible.
- **latexdiff choking on custom macros or floats**: diff from flattened tagged versions, and where it still trips, its `-exclude-textcmd` options skip the offender. Discover this in month six, not in revision week.
- **The untested backup**: a remote you've never cloned-and-built from is a hypothesis. Run the borrowed-machine test once; it converts hope into knowledge.

16

The Final Mile

The thesis is written, the committee satisfied, and one task remains: producing the exact artifacts the university accepts. This is the least documented part of the entire process, the part no template README covers, and the part where a Friday deadline meets a Thursday-night surprise. This chapter walks the mile in order: metadata, the archival PDF, the print build, and the upload itself.

16.1 PDF metadata and hyperlinks

Your PDF has invisible properties (title, author) that repositories index and rulebooks increasingly specify. You've had `hyperref` loaded since Chapter 6; finish its configuration:

```
\usepackage{hyperref}
\hypersetup{
  pdftitle={\thesistitle},
  pdfauthor={\thesisauthor},
  pdfsubject={\thesisdegree\ thesis},
  hidelinks
}
```

The metadata pulls from `metadata.tex` (Chapter 3, still paying dividends). `hidelinks` deserves a sentence: on screen, colored link boxes help; in an archival document, most universities want links functional but visually quiet, and `hidelinks` (or discreet coloring via `colorlinks` with dark colors) is the convention. Check your rulebook; some explicitly require black text throughout, which is `hidelinks` exactly.

16.2 PDF/A: the archival demand

Increasingly, repositories demand **PDF/A**, the archival profile: fonts embedded, no external dependencies, standardized metadata. Two facts defuse the panic this requirement produces.

First, modern LaTeX is most of the way there by default: your fonts are embedded (verify with your PDF reader's properties panel, or `pdffonts main.pdf`, where every row should say "emb: yes"; the classic violator is a figure PDF from another tool carrying an unembedded font, which you fix by re-exporting that figure).

Second, the last step has a standard tool. The `pdfx` package (`\usepackage[a-2b]{pdfx}` before `hyperref`, plus a small `main.xmpdata` metadata file) produces conformant output directly for most documents. Where a stubborn figure or font blocks it, the pragmatic fallback is post-conversion: LibreOffice, Acrobat, or Ghostscript can convert a clean PDF

to PDF/A, and repositories accept the result. Validate with the free veraPDF checker *before* upload night, because the repository's own validator is the worst place to learn. Budget one afternoon for the whole dance, once, weeks early, with a current draft; the final build then inherits a solved problem.

16.3 The print build

If your university still wants bound copies (many do, for the library or the committee), print and screen want different builds, and the differences are exactly three preamble switches:

- **Two-sided layout**: `twoside` class option (chapters open on odd pages, margins mirror). Screen/upload copies are almost always `oneside`.
- **Binding offset**: the `inner` margin bump from Chapter 4, sized per the binder's spec.
- **Links**: irrelevant on paper; `hidelinks` covers both worlds.

Keep both configurations honest with a single flag near the top of the preamble (a `\newif` or a commented pair of lines) rather than editing settings by hand per build; hand-edited builds drift. Print from the print build's PDF at 100 percent scale, never "fit to page," which silently shrinks your compliant margins into noncompliant ones; the copy shop's default is the enemy here, and one test page with a ruler settles it.

16.4 The last visual pass

Before the build you'll submit, one final sweep, in this order (each item earlier in the book, assembled here):

1. Full clean build: `latexmk -C`, then `latexmk -pdf` (Chapter 14).
2. Zero tolerance check: no undefined references or citations, no multiply defined labels, no serious overfull boxes.
3. The todo list prints empty (Chapter 15), and `lineno` is off.
4. Front matter against the rulebook, page by page (Chapter 13); bibliography read once end to end (Chapter 11).
5. Flip every page at speed looking only at shapes: stray float pages, a figure over a margin, a landscape page rotated the wrong way. Twenty minutes at two seconds a page, and it catches what readers of content miss.

Then tag it: `git tag submitted-2027-05-12`. Whatever happens next, you can rebuild this exact artifact forever, which is the whole system keeping its final promise.

16.5 Upload day

Mechanics for the repository portal, learned from many students' worst evenings: do it days early, not hours (portals reject, validators disagree, IT closes at five); have the PDF/A already validated locally; have the abstract as plain text ready to paste (portals want it separately, with their own word cap); know your embargo answer if the work touches patents or pending papers, a decision to make with your advisor *before* the form asks; and after acceptance, download what the portal actually serves and open it. You are the last quality gate, and the version of record deserves one final look.

Then close the laptop. The document part of your degree is, finally and provably, done.

What breaks here

- **PDF/A validation failures** trace overwhelmingly to one figure with unembedded fonts or exotic color spaces. `pdffonts` finds it; re-exporting the figure fixes it.
- **The portal rejects the file size**: usually raster figures at absurd resolution. Downsample photographs; vector content never causes this.
- **"Fit to page" shrinkage** at the copy shop turns compliant margins noncompliant. Print at 100 percent; measure one page.
- **The abstract word cap** enforced by the portal, not your document: trim the plain-text version to their counter, and update the in-document abstract to match before the final build, so the record is consistent.

Part V

Toolkit

A

The Template, File by File

The complete running example, assembled. Type it in (or rebuild it from the chapters cited), and you have the thesis skeleton this whole book operates: compile-ready, compliant, and boring in all the right ways.

The layout

```
thesis/
  main.tex  preamble.tex  metadata.tex  .latexmkrc  .gitignore
  frontmatter/  titlepage.tex  abstract.tex  acknowledgments.tex
                notation.tex
  chapters/     ch1-introduction.tex ... ch5-conclusion.tex
  appendices/   appA-data.tex
  fig/          (all graphics, chapter-prefixed)
  bib/          references.bib
  code/         (exported listings sources)
```

main.tex

```
\documentclass[12pt,oneside]{report}
\input{preamble}
\input{metadata}

% \includeonly{chapters/ch3-methods}   % fast loop: uncomment to work

\begin{document}

\input{frontmatter/titlepage}
\pagenumbering{roman}
\input{frontmatter/abstract}
\input{frontmatter/acknowledgments}
\tableofcontents
\listoffigures
\listoftables
\printacronyms[name=List of Abbreviations]
\input{frontmatter/notation}

\clearpage
\pagenumbering{arabic}
```

```
\include{chapters/ch1-introduction}
\include{chapters/ch2-background}
\include{chapters/ch3-methods}
\include{chapters/ch4-results}
\include{chapters/ch5-conclusion}

\appendix
\include{appendices/appA-data}

\printbibliography

\end{document}
```

Chapter 3 explains every line; the commented `\includeonly` is the daily driver.

preamble.tex

```
% ---- University compliance (Handbook sec. 2) -- ch. 4 ----
\usepackage[margin=1in]{geometry}
\usepackage{setspace}
\doublespacing
\usepackage{stix2}                  % "Times or equivalent"

% ---- Content machinery -- chs. 5-9 ----
\usepackage{graphicx}               % figures
\usepackage{subcaption}             % multi-panel figures
\usepackage{rotating}               % sideways floats
\usepackage{placeins}               % \FloatBarrier
\usepackage{booktabs}               % civilized tables
\usepackage{tabularx}
\usepackage{longtable}
\usepackage{threeparttable}
\usepackage{siunitx}                % aligned numbers, units
\usepackage{amsmath,amssymb,mathtools,amsthm}
\usepackage{listings}\usepackage{xcolor}   % code
\usepackage{algorithm}\usepackage{algpseudocode}

% ---- Apparatus -- chs. 10-11 ----
\usepackage[backend=biber, style=numeric-comp]{biblatex}
\addbibresource{bib/references.bib}
\usepackage{acro}
\input{acronyms}                    % \DeclareAcronym lines

% ---- Theorems (ch. 8) ----
\numberwithin{equation}{chapter}
\theoremstyle{definition}
\newtheorem{definition}{Definition}[chapter]
\newtheorem{assumption}[definition]{Assumption}

% ---- Notation macros (ch. 8): the printed twin is
%      frontmatter/notation.tex -- keep them in sync ----
\newcommand{\vect}[1]{\mathbf{#1}}
\newcommand{\R}{\mathbb{R}}
```

```
\newcommand{\loss}{\mathcal{L}}
\DeclareMathOperator{\argmin}{arg\,min}

% ---- Drafting aids: OFF for final builds (chs. 14-15) ----
\usepackage{todonotes}
% \usepackage[pagewise]{lineno}\linenumbers

% ---- Always last (chs. 5, 15) ----
\usepackage{hyperref}
\hypersetup{pdftitle={\thesistitle}, pdfauthor={\thesisauthor},
            hidelinks}
\usepackage[capitalize,noabbrev]{cleveref}
```

The order is load-bearing at exactly one point: `hyperref` then `cleveref`, last. Everything else is grouped by the chapter that explains it. (One note for `todonotes` users: it wants to be loaded before `hyperref`, as here, and its notes must be gone before final builds.)

.latexmkrc

```
$pdf_mode = 1;          # build PDF via pdflatex
$bibtex_use = 2;        # run biber as needed
@default_files = ('main.tex');
```

With this file present, the entire build, on any machine and in any editor, is the one command from Chapter 2: `latexmk`.

.gitignore

The block from Chapter 15, verbatim. Initialize the repository before you write a word of chapter 1; the first commit should be this skeleton.

metadata.tex and the front matter

`metadata.tex` carries the six `\newcommand` lines from Chapter 3; the title page consumes them per Chapter 4's rulebook-mirroring pattern; abstract, acknowledgments, and notation are ordinary fragments per Chapters 13 and 12. None of them contains a package or a setting, and that discipline, more than any individual line above, is the template's real content.

B

The Package Toolbox

Every package this book installed into the running example, one honest paragraph each: what it does, when you need it, the gotcha. Packages outside this list should enter your thesis the way Chapter 1 suggested: read about, understood, and then admitted.

geometry Margins as declarations (`margin=1in`, `inner=1.5in`). Everyone needs it. Gotcha: don't combine with hand-set margin lengths from old templates; one system only.

setspace Line spacing (`\doublespacing`) that correctly leaves captions and footnotes single-spaced. Gotcha: don't touch `\baselinestretch` alongside it.

stix2 / newpxtext+newpxmath / lmodern Font packages; pick exactly one. Gotcha: symbol packages can clash with font-provided symbols (this book's own build omits `amssymb` under `stix2`); if you get "already defined" errors on symbols, the font package probably supplies them.

graphicx `\includegraphics`. Always. Gotcha: widths relative to `\textwidth`, never absolute.

subcaption Panels (a), (b) within one figure. Gotcha: its predecessors `subfigure`/`subfig` are deprecated; don't let a template load them alongside.

rotating `sidewaysfigure`/`sidewaystable` for landscape floats. Gotcha: sideways floats always occupy their own page; that's the design, not a bug.

placeins `\FloatBarrier`, the polite fence for drifting floats. Gotcha: a barrier per section is structure; a barrier per figure is fighting the float system again.

booktabs `\toprule`, `\midrule`, `\bottomrule`. The difference between spreadsheet and publication. Gotcha: it deliberately provides no vertical rules; that's the point.

tabularx Tables stretched to text width with wrapping `X` columns. Gotcha: `X` columns are for prose cells; numbers stay in `r` or `S`.

longtable Page-breaking tables for appendices. Gotcha: needs two builds to settle widths; interrupted builds show it mid-thought.

threeparttable Notes attached beneath a table at its width. Gotcha: it wraps the `tabular`, so the `table` float stays outside.

siunitx Decimal-aligned `S` columns and `\SI` units. Gotcha: header cells in `S` columns need braces.

amsmath + mathtools + amsthm Displayed math, refinements, theorem environments. Gotcha: `eqnarray` and `$$` are the tells of an old template; replace on sight.

listings Code display with no external dependencies. Gotcha: pasted tabs and Unicode; configure your editor, keep listings ASCII.

algorithm + algpseudocode Pseudocode floats. Gotcha: incompatible with the older `algorithmic` and `algorithm2e` syntaxes; one family per document.

biblatex (backend=biber) The bibliography system. Gotcha: Biber is a program, not a package; hand-built compile sequences skip it and citations freeze.

acro Acronyms with first-use expansion and a printed list. Gotcha: reset per chapter (`\acresetall`) if your committee reads chapters standalone.

glossaries-extra The heavy glossary machinery, when mandated. Gotcha: needs its indexing step wired into `latexmk`; adopt only on requirement.

todonotes Margin todos and `\listoftodos`. Gotcha: loads before `hyperref`; must be empty (and ideally unloaded) in final builds.

lineno Line numbers for advisor drafts. Gotcha: off for final builds; check twice, it's embarrassing in a repository copy.

latexdiff (tool) Not a package: the change-marked PDF generator for revision rounds. Gotcha: run it on flattened, tagged versions; practice before revision week.

hyperref Links, PDF metadata, `hidelinks`. Gotcha: loads late, nearly last, and `cleveref` after it, always.

cleveref References with correct names, plurals, ranges. Gotcha: it is the last package. This book will not stop saying so, because the error when it isn't is baffling.

pdfx PDF/A conformance for the repository upload. Gotcha: loads *before* `hyperref`'s slot (it loads hyperref itself); test the full pipeline weeks early with veraPDF.

C

Errors and Fixes

The emergency page. Protocol first (from Chapter 14): read the *first* error only; suspect the lines *above* the reported one for unclosed things; `latexmk -C` and rebuild; then bisect with `\includeonly` and `\endinput`. Below, the specific messages, most common first.

Undefined control sequence A typo (`\inculde`), or the command's package isn't loaded, common after pasting text from another project or an old template. The log prints the offending token; search for where it was defined, or should have been.

Missing $ inserted Math outside math mode: an underscore or caret in prose (`run_2` in a filename), or an unclosed formula upstream. Escape the character (`run\_2`) or close the math; look above the reported line.

File ended while scanning use of ... An unclosed brace, usually in a `\caption`, `\textbf`, or a macro argument. The named file is right; the brace is above the line. Balance-check the paragraph you last edited.

\begin{...} ended by \end{...} Mismatched environment, classically from deleting half a float or half an `align`. Fix the pair you just edited, not the one the message names second.

Undefined references (warning) / ?? in PDF One more build pass; then a label typo; then a chapter excluded by `\includeonly`; then a label placed before its caption. In that order. (Chapter 6.)

Citation ... undefined / citations print as keys Biber hasn't run (use `latexmk`), or the key isn't in the `.bib` (typo), or the `.bib` itself has a syntax error, in which case Biber's own log (`.blg`) names the broken entry. (Chapter 11.)

Multiply defined label A copy-pasted `\label`. The warning names the key; the naming convention makes the duplicate obvious.

Overfull \hbox (badness / pt) Text or a float wider than the line. Under ~5pt, ignore during drafting. Bigger: an unhyphenatable word (give LaTeX a hint: `data\-set`), an absolute-width figure, or an unbreakable `\texttt` path; rewrap or resize. Zero serious ones by final build. (Chapter 14.)

Float(s) lost Usually a float declared inside a box or a footnote, or a `sideways` float in a hostile position. Move the float to top level of the section.

Too many unprocessed floats The float queue jammed: one unplaceable float blocking the rest. Find the oversized or `[h!]`-constrained one; add a `\FloatBarrier` or loosen its spec. (Chapter 7.)

Option clash for package ... The package was loaded twice with different options, typically by you *and* by the class or another package. Load it once, early, with the union of options, or pass options via `\PassOptionsToPackage`.

Command ... already defined Two packages fighting (fonts and symbol packages, commonly), or your macro colliding with a package's. Rename yours, or for the font case, drop the redundant symbol package. (Appendix B.)

Dimension too large / TeX capacity exceeded Almost always a corrupted auxiliary state or a pathological figure. `latexmk -C`; if it persists, bisect to the figure and re-export it.

Package biblatex Error: ... backend The document asks for Biber but BibTeX ran (or vice versa), the signature of a mixed-era template or editor misconfiguration. Align on `backend=biber` and build with `latexmk`.

It compiles but the PDF looks stale The viewer is showing a cached file, or the build failed and you're reading yesterday's PDF. Check the build actually succeeded (the log's tail), then the PDF's timestamp. Editors with integrated viewers make this near-impossible; standalone viewers cause it weekly.

Works on my machine, fails on Overleaf or a colleague's Version drift, an absolute path, a file outside the project, or a case-sensitivity mismatch (`Fig/` vs `fig/`, invisible on macOS, fatal on Linux). The borrowed-machine test of Chapter 15 finds all four early.

Runaway argument? TeX reached the next paragraph or end of file while still reading a command's argument. The cause is usually an unclosed brace in a long caption, title, or section heading above the reported line. Collapse the command to one line temporarily and count braces from the command's opening.

Emergency stop / cannot find file A missing chapter, figure, class, or bibliography database. Check the exact capitalization and extension, then confirm the file is inside the project. Do not repair an absent source file by adding a machine-specific absolute path.

Misplaced alignment tab character & An ampersand appeared in prose without escaping, often in a department name, or the number of alignment markers differs between rows of a table. Use `\&` in prose; in a table, count columns in the preamble and every row.

There's no line here to end A forced line break `\\` was used where TeX was not building a line, commonly after a heading or blank paragraph. Remove the forced break and control spacing structurally.

Missing number, treated as zero A length command received text or a unitless value, such as `width=0.8` instead of `width=0.8\linewidth`. Inspect the nearest dimension, scale, or spacing command.

PDF string warning from hyperref A bookmark contains math or a formatting command that cannot be represented as plain PDF metadata. Give the heading a plain-text alternative with `\texorpdfstring` rather than disabling bookmarks.

Font shape ... undefined The requested font family, weight, or shape has no matching face. TeX substituted another. In final production, inspect the substitution because bold small caps and specialty math fonts can silently lose a distinction.

The table of contents is one build behind Generated lists are auxiliary data. Let `latexmk` run until stable; if the stale state persists, clean and rebuild. Editing the `.toc` file repairs nothing because the next build overwrites it.

When none of these fits: reduce to a minimal example (copy the project, delete halves until the error is a screenful), which either reveals the cause outright or produces the exact artifact that gets fast, accurate answers on TeX StackExchange.

D

The Submission Checklist

Run this in one sitting, days before the deadline, against a full clean build. Every item points at the chapter that set it up.

The build (Chapters 2, 13)

1. `latexmk -C`, then `latexmk -pdf main.tex` completes with no errors.
2. Zero undefined references or citations; zero multiply defined labels; no serious overfull boxes.
3. `\includeonly` commented out; `draft` option off.
4. The todo list prints empty; `lineno` disabled.

Compliance (Chapters 4, 12)

5. Margins, spacing, font, and page-number placement match the rulebook, verified against its sample pages.
6. Title page wording and stacking mirror the official sample exactly; metadata (name, degree, date) current in `metadata.tex`.
7. Front matter order matches the mandate; roman-to-arabic switch lands on chapter 1, page 1.
8. Abstract within the word cap, no citations or unexpanded acronyms; declaration text verbatim from the university; funding acknowledgment included.

Content integrity (Chapters 5–11)

9. TOC, list of figures, list of tables skimmed in full: no typos, orphans, or missing entries.
10. Every figure and table referenced from prose; captions say something; labels follow captions.
11. No literal section, figure, or page numbers typed in prose (spot-check with a search for "Section " followed by a digit).
12. Notation list matches the macro block; acronym list contains only used entries.
13. Bibliography read once end to end: capitalization, initials, years, duplicates.
14. Listings match the tagged code export; results tables regenerate from the pipeline.

The artifact (Chapter 15)

15. PDF metadata (title, author) correct; links quiet (`hidelinks`).

16. All fonts embedded (`pdffonts`: every row “emb: yes”); PDF/A validated with veraPDF if required.
17. Page-flip pass at speed for visual defects.
18. Print build (if required): `twoside`, binding offset, printed at 100 percent, one page measured.
19. `git tag submitted-YYYY-MM-DD`; pushed; the borrowed-machine rebuild has been proven at least once.
20. Portal upload done days early; served PDF downloaded and opened; the record checked by the last quality gate there is: you.

E

Further Resources

Short and honest, like the book. Each entry says when it's worth your time.

The package documentation you already have. Every package in Appendix B ships a manual: `texdoc geometry` (or `texdoc` anything) opens it locally, and CTAN hosts the same PDFs. When a package surprises you, its manual beats any blog post, because it describes the version you actually have.

TeX StackExchange. The field's collective memory. Search first; your error has been asked. When you ask, bring the minimal example from Appendix C's closing advice and you'll often have an answer within the hour. Etiquette in one line: show code that reproduces, name versions, accept what worked.

The Overleaf documentation. Uncommonly good tutorial articles on exactly the topics of this book's Part II and III, useful even for local-first authors. Their gallery of university templates is also where to check whether your institution maintains an official one before adopting a stranger's.

The Not So Short Introduction to LaTeX (Oetiker et al.). The free classic general introduction. If this book's assumed baseline felt fast, an evening there fills it; if you want the full on-ramp built for students, my *LaTeX for Students* is that book, and the two of them disagree about nothing.

The LaTeX Companion (Mittelbach et al.). The encyclopedia, third edition. Not a thesis-month purchase; the right book when LaTeX becomes a career tool, which for academics it quietly does.

Your graduate school's formatting office. Listed last, consulted first. A short email with a specific question ("does `stix2` satisfy rule 2.3?") settles in a day what forums debate for pages, and the office that approved your title page in October is the office that processes your upload in May. They are, in the end, the only reference that's authoritative about *your* thesis.

F

Repair Lab: Eight Thesis Failures

Each exercise starts with a project that looks plausible and fails for a specific reason. Diagnose before reading the repair. In a long document, the habit of naming the failure class is faster than trying random fixes.

F.1 1. The reference that stays undefined

```
\begin{figure}
  \label{fig:setup}
  \includegraphics{figures/setup.pdf}
  \caption{Experimental setup.}
\end{figure}
```

Symptom. `\ref{fig:setup}` prints the wrong number or remains unresolved after repeated builds.

Repair. Put the label after the caption:

```
\caption{Experimental setup.}
\label{fig:setup}
```

The caption steps the figure counter. A label placed before it records the previous counter value.

F.2 2. One chapter compiles, the thesis does not

```
% chapters/results.tex
\documentclass{report}
\begin{document}
\chapter{Results}
...
\end{document}
```

Symptom. The chapter works alone but fails when included from `main.tex`, often with “Can be used only in preamble” or a second-document error.

Repair. Chapter files contain content only:

```
\chapter{Results}\label{ch:results}
...
```

The conductor owns `\documentclass`, `\begin{document}`, and the preamble. If you need standalone chapter builds, use `\includeonly` or a package made for subdocuments rather than nesting whole documents.

F.3 3. The figure works on one machine

```
\includegraphics{
  /Users/me/Desktop/thesis figures/final plot.pdf
}
```

Symptom. Local success, immediate failure in Overleaf, Linux, or a colleague's clone.
Repair. Move the asset inside the project and use a relative, case-correct path:

```
\includegraphics{figures/final-plot.pdf}
```

Spaces can work, but predictable filenames reduce quoting and tooling problems. The project should not depend on your desktop layout.

F.4 4. The bibliography vanishes after a clean build

Symptom. Citations print as keys, the bibliography is empty, and running LaTeX again changes nothing.

Diagnosis. The project uses `biblatex` with `backend=biber`, but the editor's build button runs only the TeX engine.

Repair.

```
latexmk -C
latexmk -pdf main.tex
```

Then inspect the Biber log if the problem remains. Repeated TeX passes cannot replace the missing bibliography backend.

F.5 5. A table runs through the margin

```
\begin{tabular}{llllllll}
...
\end{tabular}
```

Symptom. An overfull box of 80 pt and a clipped right edge.
Repair order.

1. remove columns the argument does not use;
2. shorten headings and put units in them once;
3. allow wrapping with `p{...}` or `tabularx`;
4. use a landscape page only if the table is genuinely wide.

Shrinking the entire table to unreadable text is not the first fix. A thesis table serves a question; if half its columns do not serve that question, the width problem is editorial.

F.6 6. A package update changes the output

Symptom. The same source produces different line breaks or a new error on a second machine months later.

Repair. Record the engine and important package versions with the submitted source. Keep the log from the final clean build. If the project depends on a university class or a local style file, archive that file in the submission bundle when licensing and university rules allow.

Do not respond by copying random old packages into the project. A frozen dependency without its companion files can create a harder failure than the update.

F.7 7. Supervisor edits overwrite each other

Symptom. Two returned copies, `chapter3-supervisor-final.tex` and `chapter3-my-revision.tex`, both contain changes you need.

Repair. Stop exchanging editable source snapshots without an owner. Choose one workflow:

- one shared project, with comments and tracked history;
- PDF comments returned to one source owner;
- Git branches merged by the person responsible for the source.

File suffixes are not version control. Recover the two copies with a diff tool, merge once, commit the result, and agree on the owner before the next round.

F.8 8. The final PDF passes locally and fails the repository

Symptom. The portal reports a PDF/A or font-embedding error.

Repair path.

1. run `pdffonts` and find any row whose embedding column says `no`;
2. inspect imported PDF figures, which often carry the offending font;
3. re-export that figure with fonts embedded or text converted appropriately;
4. run the same PDF/A validator the repository recommends;
5. upload early enough to repeat the cycle.

Do not flatten the whole thesis to images. That destroys search, selection, accessibility, and usually file size. Fix the one asset that violates the archival requirement.

F.9 The repair record

For any failure that costs more than fifteen minutes, add one line to `notes/build-log.md`:

```
2026-10-14 | undefined citations after clean build
Cause: editor ran TeX without Biber
Fix: switched project build command to latexmk -pdf
```

This is not a diary. It prevents the same expensive problem from becoming new again six months later.

A Note from the Author

Thank you for reading, and congratulations in advance on the defense. Books like this one live on word of mouth from students, so if it saved your submission week, a short honest review where you bought it, or a loan to the labmate whose thesis folder is still one giant file, genuinely matters.

The checklists and the template are meant to be working tools. Printable companions and more of my writing for students live at **gauravtiwari.org**, and corrections are welcome there too; future editions will owe their sharpest fixes to readers who wrote in.

Also by Gaurav Tiwari

LaTeX for Students

A Practical Guide to Beautiful Documents and Mathematical Typesetting

The on-ramp this book builds on: documents, equations, figures, and templates, taught step by step for students with no prior LaTeX experience.

How to Write Mathematics

Proofs, Papers, and Problem Sets for Students

What goes *inside* the document: proofs graders can follow, full-marks problem sets, and your first paper or thesis, taught through before-and-after rewrites of real student work.

Other titles

- *Indian Polity: Complete Notes for UPSC Prelims & Mains*
- *Website and Blog Copywriting*
- *Marketing Essentials for Bloggers*
- *The Affiliate Bloggers System*
- *Ultimate Guide to Backlink Building Strategies*

All titles are available on Amazon. Find the full, current catalog at **gauravtiwari.org**.