# Adapting the Resource Guide For Your Local Students

Every group of students is different; what our students want or need to know may be different from what your students need. Rather than try to make a Guide that covers every possibility, we published the Guide [under the terms of a Creative Commons BY-NC-SA 4.0 license](http://creativecommons.org/licenses/by-nc-sa/4.0/) so that instructors can edit, extend or modify it to fit their particular needs and requirements.

All source files and instructions for converting the Guide to print or digital formats are available from the [Stem Writing Project’s GitHub repository](https://github.com/adanieljohnson/SWP_student_writing_guide).

## General Strategy

Adapting the Guide typically involves:

* Identifying what the most common writing errors are locally.
* Deciding which general recommendations in the current Guide need to be revised to match local expectations.
* Replacing any generic pages in the shared Guide that duplicate local resources which you already know work well.
* Shortening the explanations of our rationale for certain activities.

Not all of this work needs to be done at once. Generally the revision process will take a few semesters, and students are very good at finding contradictory information. They also can say what they need that is missing.

## Suggested Workflow

1. Identify the **largest, most serious writing skills gaps your students have first.** Look at writing products of your students systematically, and try to quantify how many students make particular mistakes (not just which mistakes you or your instructional team find the most irritating.)
	* What essential features or requirements of writing in your particular discipline or field are most often missing from students’ written work?
	* What features are present in texts from A/B-level writers that are consistently absent from texts by C/D-level writers?
2. Rank the mistakes so you are confident that you have identified the 5-10 most serious shortcomings in their scientific writing. These are your priority goals that you should focus on first, and what you should keep in mind as you revise this Guide.
3. Look at the topics listed in the Preface and Table of Contents. If you have resources you are using to teach these elements of scientific writing successfully, flag the overlapping units or pages, move them to an archive folder, and use your own resources. Pages in the archive folder will still be available in the future, or you can download new copies from our repository.
4. The Preface is written assuming students are the main audience. That said, the text may be too long for your students, or a particular section may go too deeply into our rationale. Delete those sections before generated the individual documents or book.
5. **Part 3** is the main deep dive into scientific articles and lab reports for biology. We show students what goes in each section, point out where our own students struggle with each section, and suggest ways to avoid common pitfalls. We strongly recommend instructors incorporate their own observations and suggestions so that the Resource Guide addresses their specific local needs.
6. Ideally the Appendices should be replaced with examples of reports that your own local students have written.
7. While not part of the shared Guide, we do recommend selecting 4-5 primary articles that demonstrate the desired features for local writing, and making them available either as part of the text or as linked files. Open-access journals such as the PLOS series are good sources for articles. Include examples of secondary and lay literature as well, so students can see the differences.

## Editing the Guide Itself

Each chapter is a plain text file with an .Rmd file. This is the standard format for R Markdown files. If you change the extension of the file name from .Rmd to .md, the file will switch to a plain Markdown file, and vice versa. The two file types are both plain text files. The extensions simply indicate which programs the computer should use to open and edit the files. In practice, you can write new chapters entirely as plain text files using any text editor. When you are ready to use them, simply change the extension on the filename.

Each .Rmd file in the shared Guide begins with a 3-digit number that defines the order of the files when compiled into a book. To change the order of the chapters, change the order of the numbering. There are other ways to tell R Studio how to arrange pages, but this way is obvious and fairly painless.

A good practice when renumbering files is to number them sequentially by 10s (10, 20, 30, etc.) This leaves room to add new pages without renumbering all existing ones.

Chapter files begin with the chapter title as a level-one header, e.g., # Chapter Title. Each chapter is divided into sections using lower-level headers, e.g., ## A Section Within a Chapter.

### Creating Cross-Links Between Pages

When an .Rmd file is converted to HTML for a book the chapter and section headings are formatted and sequentially numbered automatically. Chapter and section headers include a cross-reference tag; the auto-generated tags can be replaced by adding an explicit {#label} after the chapter. For example, the H1 chapter header for the document describing scientific writing in general has this header line:

# What Do We Mean By Scientific Writing? {#goals100}

Adding an explicit {#label} to the end of all chapter headers is a good practice if you know you’re going to cross-reference a topic repeatedly. We have already attached explicit {#labels} to the top level header of most pages (only pages in the Preface were excluded.)

In-book crosslinks are formatted this way:

[link text](#goals100)

The first part in square brackets is the linked text that will be displayed. The hashed text in parentheses is the label from the page or section that is the target.

To refer to the NUMBER of a particular chapter or section, use \@ref(label). For example:

In chapter \@ref(install-git) we explain how to install Git.

renders this way:

In Chapter 6 we explain how to install Git.

See the project page “Using Markdown” for more information about Markdown syntax and conventions.