Advances in Biology Laboratory Education
Publication of the Association for Biology Laboratory Education
Volume 43, Article 10, 2023

Using Markdown, R Studio, and free tools to write, publish, and share an open-source scientific writing guide

A. Daniel Johnson

Wake Forest University, Department of Biology, 1834 Wake Forest Road, Winston-Salem, NC 27106, USA
(**johnsoad@wfu.edu**)

One of our essential tools for teaching scientific writing in multi-section introductory biology courses is a standardized Scientific Writing Resource Guide that students and instructors can use across multiple courses. We have published our Writing Guide as an open-source book that others can modify to meet their individual needs. To simplify conversion between different formats, we wrote our Writing Guide using Markdown. This lightweight markup language lets authors write text once, then output it in a variety of formats such as HTML5 for web pages, or Word/PDF documents for handouts. Groups of Markdown files can be combined to create interactive online books. Markdown takes ~20 minutes to learn, and marked text remains easily readable. These features make it ideal for writing lab materials. Participants in this workshop do not need any prior technical knowledge beyond basic computer skills. They will learn to write with Markdown by editing existing pages from our Guide and creating new pages. They also will convert Markdown files into formatted Word and HTML5 documents. We will demonstrate how to use R Studio to assemble collections of Markdown documents into books, and how to use GitHub to manage and share writing project files. Participants will leave with a complete copy of our Scientific Writing Resource Guide that they can revise to match their course requirements, plus the tools for converting Markdown files to their preferred format. They will have a starter book template for R Studio that they can use to write and compile new book projects of their own.

Keywords: scientific writing, writing guide, Markdown, lab development tools, R Studio, web publishing

Link To Supplemental Materials: https://doi.org/10.37590/able.v43.sup10

Introduction

Scientific writing helps students learn to state problems and present claims precisely, summarize evidence to support those claims, and explain their reasoning. For many years, our *Scientific Writing Resource Guide* has been one of our main tools for teaching scientific writing in multiple courses. The Guide focuses on writing a lab report that models a journal article because the same components are used in most other forms of scientific communication too. Our general format conforms to the *Council of Science Editors* (8e) standards, with some modifications to make writing easier for students just starting out. The Resource Guide includes sections on data visualization, basic biostatistics, and how to cite literature.

In 2021 we published our Guide under terms of a Creative Commons BY-NC-SA 4.0 license with a goal of supporting two audiences: undergraduates learning the craft of scientific writing, and biology instructors who either teach scientific writing to undergraduates or supervise teaching assistants who do. An interactive edition of the

Guide is available online. The source files and full Guide as an MS Word document are available from our GitHub repository.

Rather than try to make one guide that can meet the needs of every possible audience, we designed it to be an evolving collaborative resource. This workshop explains why we used Markdown to create the Guide, and how users can access and modify the Resource Guide to fit their local needs and requirements.

Why Use Markdown to Write Lab Documents?

Markdown is a lightweight markup language that provides a simple solution to the challenge of writing lab documents that may have to go to multiple destinations. With Markdown, authors can write a text once, then convert it into multiple file formats as needed. It takes less than 20 minutes to learn most of the Markdown syntax needed for routine writing tasks, and marked up text is still easy to read. Markdown text can be converted to clean HTML5 that can be posted directly to most LMSs. Texts also can be converted directly into MS Word or PDF documents. Collections of Markdown documents can be compiled into interactive online books using R Studio, then published multiple ways.

About This Workshop

The main goals of the workshop are:

- Share our Writing Resource Guide in multiple formats;
- Show potential users how to modify the Guide to meet their program needs; and
- Introduce participants to using Markdown in a more flexible “write once & reuse approach” to writing workflow.

Users do not need any technical skills beyond basic computer skills. The exercises assume no prior knowledge of Markdown, the other tools, or computer coding.

During the workshop, participants will do the following:

1. Use a browser to download the Resource Guide files and supplemental materials for this workshop from GitHub.
2. Edit some existing and create new Markdown documents, using two different tools:
    a. A plain text editor
    b. R Studio
3. Convert Markdown files into MS Word documents using:
    a. R Studio
    b. Pandoc via Terminal commands
4. Convert Markdown files into HTML code for use in an LMS or web site, using:
    a. A text editor’s built-in functions
    b. A freestanding web app
5. Install the *bookdown* package for R Studio. This package converts a set of individual Markdown files into a book that can be hosted online or printed.
6. Download a starter book template and explore how individual Markdown files can be converted to rich books for the web or to print.

Participants will leave this workshop with a copy of our Science Writing Resource Guide that they can revise to match their local needs. They also will have experience writing and editing Markdown and converting it to other formats.

Student Outline
Pre-Workshop Exercise: Install the Required Software

If you have participated in one of ABLE's prior workshops on R or R Studio, you may have these programs installed already. If not, download and install R (the engine) **before** R Studio Desktop (the user interface). Usually both installations run smoothly, but give yourself enough time to install the software. Installing R Studio Desktop typically takes 10-15 minutes.

1. Download and install R.
   - If you are a Windows user: Click on "Download R for Windows", then click on "base", then click on the Download link.
   - If you are macOS user: Click on "Download R for (Mac) OS X", then under "Latest release:" click on R-X.X.X.pkg, where R-X.X.X is the version number. For example, the latest version of R as of November 25, 2019 was R-3.6.1.
2. Download and install R Studio.
   - Scroll down to "Installers for Supported Platforms" near the bottom of the page.
   - Click on the download link corresponding to your computer's operating system.

Here are illustrated instructions for installing R & RStudio. For even more detailed instructions, consult Installing R and R Studio, Step by Step.

Exercise 1: Copying the Workshop Repository From GitHub

Background

GitHub is a low cost, secure collaboration and code sharing site that is widely used by data scientists and programmers. GitHub is becoming popular for hosting blogs, e-books (for example, Using Markdown inside R) and static project websites (see the slides from the workshop (in Supplemental Files) for examples of biology education-specific project sites hosted on GitHub.)

GitHub stores files and data for a project in an annotated folder called a **repository ("repo")**. GitHub users create separate repositories for different projects. The annotations control how the folder and files it contains behave, and who has access to them. Private repositories can only be seen by their owner and other GitHub users that the project owner specifies. Public repositories are available for anyone to view or copy.

You do not have to be subscribed to GitHub to **clone** (download a copy of) a public repository or use the files in it. The clone is downloaded as a ZIP file and stored on your personal computer like any folder of files. This is the simplest way to copy files stored on GitHub to your computer, and the kind of copy we will use initially in the workshop.

When you clone someone else's repository via ZIP file, the cloned copy is **completely separated from its original source**. You do not see changes that the original owner makes after you make your copy, and they cannot see changes you make. You also lose access to GitHub's powerful file backup and archiving tools. We will look at these a little later in the workshop.

In this exercise you will be cloning the **ABLE 2022 Workshop** repository. After cloning you can edit any page or add new pages without damaging the source files. If you make a mistake in a document and cannot fix it, just unzip the repo again and grab a new copy. The cloned repository contains all of the files needed for the **first** part of the workshop.

Procedure

Before diving into the starting files, let's look at the **end product** to see what can be created from Markdown's plain text files.

1. Open the current published edition of the Resource Guide. This is the active, publicly shared e-book.

2. Look at the Table of Contents briefly and see the range of topics it covers. Look at how the book itself is arranged. Then look at how 1-2 individual pages to see how the contents are formatted. This will give you an idea of what the **final product** looks like (and what you are trying to make.)

Now let's go back to the beginning and look at the **starting documents** we used to build the Resource Guide. Remember, the entire final published Guide was built from simple plain text files combined in a specific way.

3. Use a web browser to open the URL for the ABLE workshop repository. Alternatively, go to GitHub, look for the Search window at the top left, and enter "**adanieljohnson/ ABLE_2022_Workshop**." This is a shortcut to the project repository.
4. What you see will be something similar to Figure 1. Click on "Code" to open the window with the dialog to clone this repository (see figure 2). Click on "Download ZIP" (the BLUE arrow.)
5. The entire repo is over 70MB, so takes a minute or so to download. When it is done, go to your Downloads folder and click to unpack the ZIP archive.
6. Drag the unpacked folder to your desktop. Now you have a personal copy of the files to work with. Do not worry about damaging any files. If you need to replace a corrupted file, you can go back to the original ZIP archive and unpack it again, or clone it again from GitHub.

Now that you have the text files for the Guide, you can start exploring how thir Markdown text is formatted.

7. Open R Studio, choose **File > Open** and navigate to your local copy of the **ABLE 2022 Workshop** repository. Now open the folder named **Abridged Writing Resource Guide.** It contains the Markdown files (with the ".md" extension) for the full Writing Resource Guide.
8. Open the file named **Table of Contents.md.** This is the full list of pages in the Guide. Find 1-2 EXISTING pages you think look interesting, or that you might want to edit or contribute to.
9. Locate the .md file for a page you wanted to edit and double-click to open it. It should open with R Studio. If it opens in a plain text editor, that is fine for right now; you are just reading the file, not editing it.

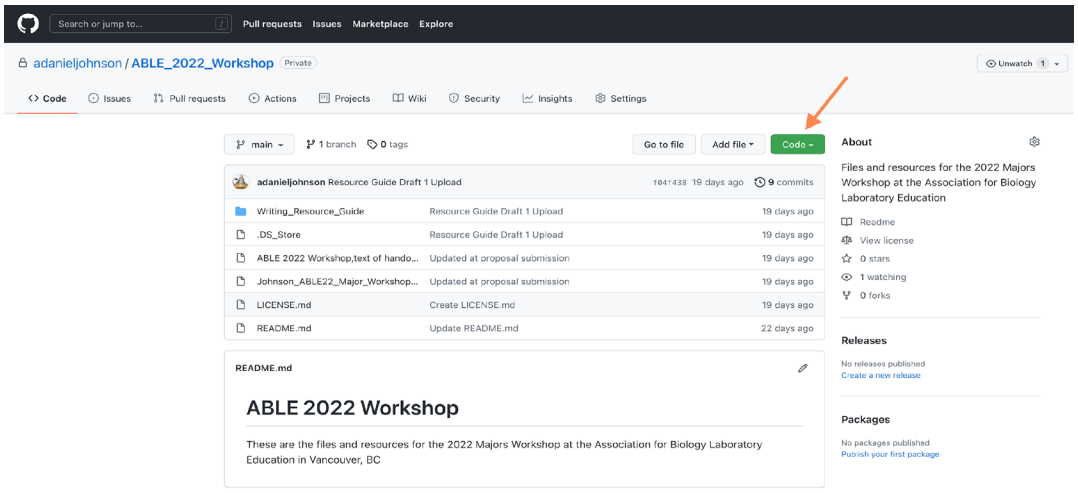


Figure 1. Main page of ABLE Workshop repository

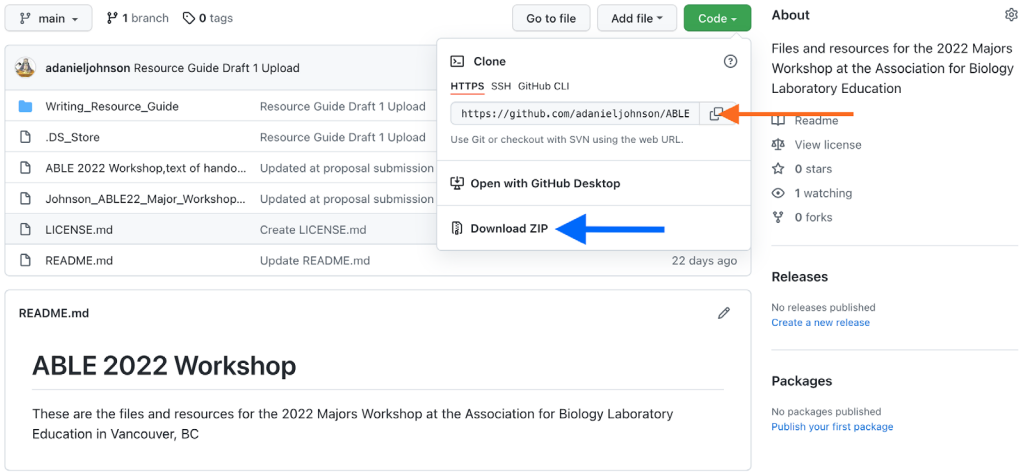


Figure 2. Blue arrow marks where to download a ZIP file.

10. **Handout for Exercise 1** is a short guide to Markdown formatting. Think of it as your ".md to .html/.docx dictionary." Looking at the .md file you opened and Handout 1, try to translate how the markup added to the text in the .md file produces different formats.

    - Do you see how the structure of the rendered file is encoded in the Markdown tags? (If needed, you can click the **Preview** tab at the top of the window to switch from Code to Preview modes.)
    - As you work, think about these questions:
      – How hard or easy is it to read the **original text** in the .md file? Could you still understand what the document says, even in the unconverted format?
      – How do you predict the markup text will look when converted to a Word document?
      – How do you predict the markup will look when converted into a web page?

Handout for Exercise 1: Short Guide to Markdown

Markdown was created as a way for writers to annotate text quickly to show formatting without having to embed full HTML tags. The goal with Markdown is to separate the content (words of the text) of a document from the format (headers, paragraphs, bullets, etc.) from the. By using one well-defined set of marking conventions (called the **syntax**), we can use converters to read a marked-up text and render it to many different formats. For example, the converter we will be using can read Markdown and export it in over 55 text formats.

What Does Markdown Syntax Look Like?

Here is some text that has been formatted using Markdown:

```
#### Level 4 Header - Formatting

There are two main kinds of text formatting:

* Inline formatting (_italics_, subscripts like H~2~SO~4~, etc.)
* Block- or paragraph-level formatting (headings, sub-headings, etc.)
```

Here is the same text after it has been converted to MS Word format:

<u>Level 4 Header - Formatting</u>

There are two main kinds of text formatting:
1. Inline formatting (*italics*, subscripts like H_2SO_4, etc.)
2. Block- or paragraph-level formatting (headings, sub-headings, lists, etc.)

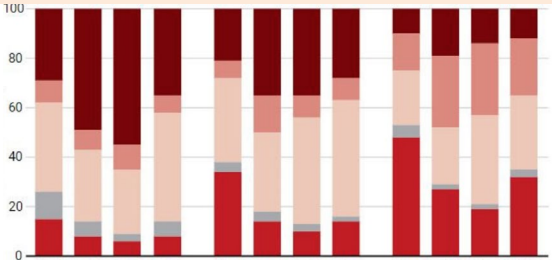
There are 3 main rules that help you remember most of the syntax of Markdown.

1. The number of hash (#) marks starting a line indicates the level of the header.
2. Words or blocks of text with specific formatting are surrounded by a **<u>pair</u>** of symbols indicating the type of formatting to apply.
3. Blocks of text that should not run together are separated by an extra space.

The rest of this handout explains the markup codes that you need most often. Keep a copy handy until you can remember the specific codes reliably. Better yet, copy the text of the original .md file (in the Supplements for this article) and edit it to suit your own needs.

Markdown Codes for Inline Formatting

The table shows how to mark text, and what it will look like when rendered.

Inline Formats	How to Mark Them	How They Appear
Italics	`_italicized_ word`	*italicized* word
	`*italicized* word`	*italicized* word
Bold	`__bolded__ word`	**bolded** word
	`**bolded** word`	**bolded** word
Bold Italics	`___marked___ word`	***marked*** word
	`***marked*** word`	***marked*** word
Superscripts	`Super^script^ed letters`	Super^{script}ed letters
Subscripts	`Sub~script~ letters`	Sub$_{script}$ed letters
Horizontal rule	`***`	Draws a line across page _____
Inline code (not rendered)	`` `code block` ``	`code block`
Escape a special character	`\*code block\*`	*code block*
Links to web pages	`[text](link)`	RStudio
Links with URL	`[link](link)`	https://www.rstudio.com
Embed local images	`![Image description](path/to/local_image).`	
Embed images from web	`![Image description](URL for image).`	

Codes for Block Level Formatting

Headers

Header levels are indicated with 1-6 hash marks followed by a space. The table below shows the raw codes, and how they render using the ABLE chapter template formats.

Markdown Code	How It Looks When Rendered
`# Level 1 Header`	**Level 1 Header**
`## Level 2 Header`	**Level 2 Header**
`### Level 3 Header`	*Level 3 Header*
`#### Level 4 Header`	Level 4 Header
`##### Level 5 Header`	*Level 5 Header*
`###### Level 6 Header`	Level 6 Header

Paragraphs

Regular paragraphs need to be separated by a blank line, or they will run together. For example, a text written like this, without spacing lines:

```
There are several kinds of text formatting to explore.
Inline formatting text.
Block- or paragraph-level formatting.
Lists and quotes.
```

Looks like this when rendered:

There are several kinds of text formatting to explore.Inline formatting text.Block- or paragraph-level formatting.Lists and quotes.

Compare the above to text written WITH spacing lines like this:

```
There are several kinds of text formatting to explore.

Inline formatting text.

Block- or paragraph-level formatting.

Lists and quotes.
```

Which looks like this when rendered:

There are several kinds of text formatting to explore.
Inline formatting text.
Block- or paragraph-level formatting.
Lists and quotes.

Block Quotes

Block quotes are written after the “>” symbol. A new “>” symbol is needed after each line break. This text:

```
> "I thoroughly disapprove of duels. If a man should challenge me, I would take him kindly
and forgivingly by the hand and lead him to a quiet place and kill him."
>
> --- Mark Twain
```

Renders like this:

> “I thoroughly disapprove of duels. If a man should challenge me, I would take him kindly and forgivingly by the hand and lead him to a quiet place and kill him.”
> — Mark Twain

Preventing Text From Rendering

Blocks of text that should be displayed as written and not be rendered are enclosed between three backticks. Blocks typed like this:

```
```Block of text or code that should NOT be rendered, like this __bold__ word. ```
```

are rendered like this:

```
Block of text or code that should NOT be rendered, like this __bold__ word.
```

Hidden Comments

Hidden comments let you add information that will not go in the final document. There are two ways to add comments to Markdown that will not display in the rendered text:
```

```
[//]: # (This comment in Markdown will be removed from both .html, and .docx files)

<!--- This comment will be retained in .html files, but removed from .docx files-->
```

Lists

Bulleted or unordered lists need to be separated from the preceding paragraph by a blank line. I items start with *, +, or -, followed by a space before the first letter or numeral. Lists are nested by indenting each sub-list by four spaces. For consistency, it is best to make it a habit of using one character for main bullets, and the other two for sub-bullets.

An unordered list encoded like this:

```
* First item
* Second item
    + First sub-item
        - First sub-sub-item
        - Second sub-sub-item
    + Second sub-item
* Third item
```

Renders as:

- First item
- Second item
  - First sub-item
    - First sub-sub-item
    - Second sub-sub-item
  - Second sub-item
- Third item

Ordered list items start with numbers, but the rules for nesting are the same as for unordered lists. So this code:

```
1. First item
2. Second item
    1. First sub-item
        1. First sub-sub-item
        2. Second sub-sub-item
    2. Second sub-item
3. Third item
```

Renders as:

1. First item
2. Second item
    1. First sub-item
        1. First sub-sub-item
        2. Second sub-sub-item
    2. Second sub-item
3. Third item

The two formats can be mixed together, so this code:

```
1. First item
    + Sub-item
    + Sub-item
2. Second item
3. Third item
```

Renders as:

1. First item
    – Sub-item
    – Sub-item
2. Second item
3. Third item

Numbered lists can be a bit fussy in Markdown, because each new list defaults to re-starting with #1. There is no simple and consistent way to force a list to start with a particular number. I usually fix this either by editing the HTML or Word document directly, or just pre-number lists myself.

Math Expressions

Inline equations are written using standard LaTeX syntax and enclosed by $ signs. This code:

```
$f(k) = {n \choose k} p^{k} (1-p)^{n-k}$
```

Renders an inline version of the equation as: $f(k) = \binom{n}{k} p^k (1-p)^{n-k}$

Equations in display style are written enclosed in a pair of double dollar signs. This code:

```
$$f(k) = {n \choose k} p^{k} (1-p)^{n-k}$$
```

Renders as:

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Other Tips for Equations:

- This link goes to a tutorial on **writing LaTeX equations**.
- **Online equation builder** lets you write LaTeX equations using a visual editor.
- Markdown equations sometimes fail to render correctly. It is good practice to always check any formatted equations you have in a .md file the first time you try out a new rendering tool.
- Unlike mathematicians, we usually do not need to edit equations after typing them the first time. This is a simple workaround that avoids the rendering problem:
  - Render each equation once
  - Take a screen shot of the equation
  - Embed the equations as images.
- As with everything, whatever simplifies your workflow is what you should do, as long as you do it the same way every time.

Handy HTML Bits

There are some formats that Markdown does not handle well. For example, subscript and superscript marks may not work on the first character or number in a word (like ^{3}H) or on a whole ^{word}. Some other useful items are not in the GHFM syntax at all. You can fill in these gaps with a few basic HTML codes.

If you need to…	Insert this HTML snippet	For this result
Force a superscript for a whole word or first character	`<sup>super</sup>script`	^{super}script
Force a subscript for a whole word or first character	`<sub>sub</sub>script`	_{sub}script
Strike through text	`<strike>This</strike> word`	~~This~~ word
Add an extra space between items	` `	
Add an extra line between items	` `	
Add a horizontal rule between lines	`<hr/>`	———————
Add Greek letters	`&alpha;, &eta;`	α, η

Learning More

These web resources can help you expand your Markdown writing skills. More are listed in the Notes for Instructors.

- GitHub's Introduction to GHFM
- Making Tables in Markdown
- Web-based Markdown Table Maker
- This page is a good source for other HTML shortcuts.
- You can find more information about special symbols here.

Exercise 2: Creating and Editing Reusable Markdown Pages

Background

What Exactly IS Reusable Text?

When we write documents for courses and labs, we tend to think of them as going out to students in one format (a Word document, or a web site for example.) Yet **many texts we write get reused across multiple formats.** The same piece of text might be posted on the campus LMS, go out to students as a printed handout, and be incorporated into a lab manual or course pack. We must spend time ensuring that multiple separate texts still match, or the texts can drift and become increasingly different over time. The goal of **"reusable text"** is to assume it will be used in multiple formats. The text is formatted so reusing the text is as easy as possible. Most reusable text contains 3-4 elements:

- The content or information you want to convey,
- Tags that indicate the document's structure,
- Instructions for processing the text. These instructions (which may be stored in a separate document) include:
    - How the final document will be generated,
    - What file format will be used, and
    - What it will look like.

Reusable texts have these elements in a certain place and in a certain format. This organizational structure lets us automate conversion between formats.

Reusable texts might seem like extra work, but following a FEW rules & constraints can save HOURS of time. It helps to think of texts as having another feature that we often do not consider: the time invested in them. Your time is a limited resource, and it should be spent where it provides value. The value of a text comes from the time spent creating and organizing its informative content; time spent converting text from one file format to another does not add to its value.

Why Use Markdown Instead of MS Word or Google Docs?

Markdown was intended for the **start** of the authoring workflow. **It assumes that text needs to be reused across multiple formats.** MS Word & Google Docs are designed to produce polished documents at the end of the authoring and publication workflow. That is why their document files include a lot of hidden styling information that **limits reusability.**

For example, if you have ever converted a long Word document into a web page, or combined several documents into one, you probably have seen unpredictable changes in format. This happens because of the embedded, hidden styling information. Markdown does not hide the formatting information, making it much easier to combine and convert documents.

What other advantages does Markdown offer?

1. Markdown files are **MUCH** smaller than Word .docx files. For example, the Markdown file containing the handouts for this workshop is 58KB; the corresponding Word file is 5.9 MB (100x larger.)
2. Markdown files "call in" images when the document is rendered. One master set of images can be stored in a single location and called into any Markdown document. When an image is updated, every document or web site that uses that image will use the new version the next time the file is generated.

3. When a Markdown file is converted to another format, headers, bold words, etc., get tagged with standardized styling codes for that document type. This greatly reduces time spent reformatting converted documents.
    - Say you create a 55-page document, convert it to Word, and decide to change all 155 section headers from blue to red text in a different font. You do not have to convert each header one at a time; change the master style settings for headers once, and every header changes.
    - If you copy/paste Markdown HTML into an existing site (like an LMS page), the text adopts the styles of its new location.
4. Markdown is truly platform-agnostic. The same file can be edited on a Mac, a PC, and a Linux machine, and never have any compatibility problems.
5. Markdown and the tools needed to convert it are available for free.

How Do We Create Markdown Documents?

Markdown files are plain text files with **.md** as the extension. Unlike HTML, Markdown does not require any additional special code or starting text at the beginning of a new document. The file extension is the only requirement. You can write .md formatted files in any plain text editor then convert them with another program to their final format. Alternatively, you can both write and convert them to other formats using R Studio.

The goal of this exercise is to introduce you to each method, so you can find which workflow feels comfortable to you. In this exercise you will try writing and editing Markdown texts using R Studio as a text editor, then using a standalone text editor. Your goal is to experience for yourself how Markdown behaves in different editing environments. One probably will feel more comfortable than the other; that is the way you should start out building your editing workflow. The documents you create will be the files you try converting in the next exercise.

Procedure

Editing Markdown Using R Studio

1. Open R Studio, choose **File > Open** and navigate to your local copy of the **ABLE 2022 Workshop** repository. Open to the file containing the **Table of Contents** page for the Writing Resource Guide.
2. Find 1-2 EXISTING pages you think you might want to edit or contribute to. Also find 1-2 topics that you do not see in the current pages that you think should be in the Guide.
3. Now open the folder named **Writing Resource Guide.** It contains the Markdown files (with the ".md" extension) for the full Writing Resource Guide. Locate one of the pre-existing .md files you wanted to edit and open it.
    - First look at the current structure. If needed, compare it to the markup guide in the handout from earlier. Do you see how the structure of the rendered file is encoded in the Markdown tags? (If needed, you can click the **Preview** tab at the top of the window to switch from Code to Preview modes.)
    - Look for a section you want to edit. Make some changes, save, then see how your changes look using **Preview** mode.
4. Next you will start a new file. Choose **File > New File > Markdown file** (Figure 3, orange arrow). Do NOT use R Markdown (the blue arrow).

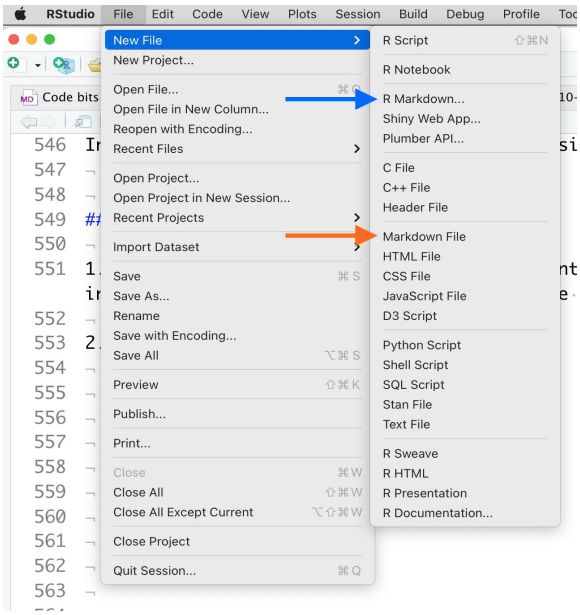


Figure 3. Start a new .md file

Enter some starting text. One or two words is enough.

5. Save the file in your **ABLE 2022 Workshop** folder. Give your file a name that reflects the topic you did not see. Include the .md extension. **TIP:** do not include spaces in file names. Write all file names in CamelCase (FileName1.md) or Snake_Case (File_Name_1.md)
6. Now you have a starting file that you can edit any way you wish.
   - If you want to see how changes to the Markdown code affect it, click the **Preview** button at the top of the window.
   - If the .md file will not display, you probably have an error in the format. Look at the bottom of the page for a tab called **R Markdown**. There usually is an error message telling you what went wrong.
7. When you are finished editing your 1-2 pages, save them both in the **ABLE 2022 Workshop** folder.
8. Close R Studio for now.

Editing Markdown Using a Plain Text Editor

1. Navigate to your **ABLE 2022 Workshop** folder on your desktop.
2. Click on one of the .md files in the **Writing Resource Guide** folder to select it. This time you want to open the file in a plain text editor.
   - For Windows, the default editor is **Notepad++** (Link: https://notepad-plus-plus.org/). If you do not have it, you can install it in ~1 minute by clicking on the link above.
   - For MacOS, **TextEdit** is the default pre-installed editor.
   - You can open and edit .md files with any text editor you have installed, such as:
     - Visual Studio Code
     - Atom
     - Brackets
     - Bluefish
3. After you have edited an existing .md file, try creating one from scratch. Once again, create a new file for a topic you think is missing from the Table of Contents of the Resource Guide.
4. When you have finished, save your edited .md files in the **ABLE 2022 Workshop** folder.

Writing Tips and Tricks

- Start all Markdown pages with a Level 1 header that contains the title of the document. This does not seem very useful at first but becomes essential when you start compiling multiple .md files into longer documents or books.
- Do not remove spaces between lines of text in .md files. Markdown relies on blank lines to keep track of blocks of text. They are removed when Markdown is rendered.

Exercise 3: Generating Output Documents

Background

So far you have learned how to create, edit, and read Markdown files. Now we are going to create output documents with those files. There are several text file processors that can inter-convert .md files. Many programs use the **Pandoc** "universal document converter" to inter-convert files. In this exercise you will:

- use a web-based converter to create clean. minimal HTML,
- use R Studio to create MS Word documents, and
- (Optional) convert single files using Pandoc from the command line.

In a later exercise you will see how R Studio's *bookdown* package can automate the processes of converting and compiling single files into an e-book.

Procedure 1: Converting Text.md to Text.html

Many plain text editors can convert Markdown to very compact plain HTML. If your text editor does not have this feature, this web-based Javascript converter works well.

1. Go to the Ashkanph MD to HTML Converter.
2. In the upper right corner, use **Browse** to find your local folder, and choose one of your recently edited .md files.
3. The .md file will be converted to .html and displayed.
4. Use the **Save as HTML** button to write a copy in .html back to your local folder.
5. If you open the version of the file with the .html extension in your plain text editor, you will see it has been rewritten using HTML markup. You can copy and paste the HTML code into most LMS pages. Alternatively, you can share the file itself with others; they can open it in any web browser.

Why not use R Studio to convert .md to .html? When R Studio generates standalone HTML files, it embeds the raw code for every image plus a LOT of extra styling code. For example, an HTML file containing the handouts for this workshop generated using the Ashkanph converter is 63 Kb. The same HTML file generated using R Studio is 6 Mb. This problem goes away when R Studio creates e-books; the HTML files are very compact, because they reference separate styling and image files rather than embedding them.

Other options for HTML. Two good free MD to HTML converters are Markdown It and Markdown to HTML; both have very user-friendly interfaces, but require copy/pasting the text rather than choosing a file. Dillinger is a web-based editor that shows the Markdown code in a window next to either the HTML code or the styled text. Rendered Markdown can be exported as HTML, but like R Studio, Dillinger includes some extra styling information. If you prefer to use a standalone Markdown editor, HarooPad produces both clean and styled forms of HTML. The trade-off is that HarooPad has some trouble with equations.

Procedure 2: Converting Text.md to Text.docx

R Studio works very well for converting .md files to well-formatted Word documents.

1. Open R Studio and go to the workshop files. Open one of the files you edited or created.
2. At the top of the screen, open the menu next to **Preview** and choose **Word document** (Figure 4).

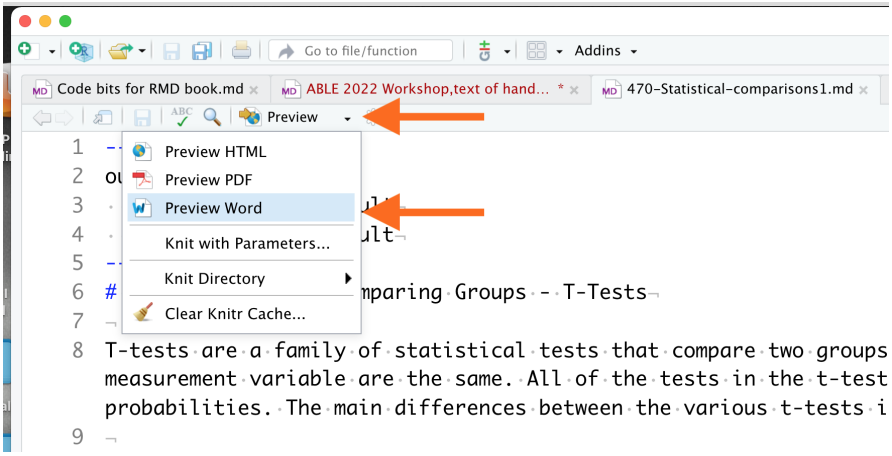


Figure 4. Converting Markdown to Word format

3. R Studio will call Pandoc, generate a pre-formatted MS Word .docx file, then save it to the same directory where the .md file is located.
4. Click on the .docx file to open it in MS Word.

MS Word files are formatted using a standard template. To reformat documents, create a template file with the header and text styles that you want. When newly generated documents are copy/pasted into the template file, the text will change to the styles you set. **An example template file is provided in this chapter's supplemental files.**

Optional Procedure 3/Demo 1: Running Pandoc From the Command Line

It is very easy to install and run Pandoc from your computer's terminal command line. It can generate .docx, .html, and many other file types. If you would like to try it yourself:

1. Go to the Pandoc installation page for instructions to download and install to your computer.
2. Use EITHER of the following Terminal commands to convert from .md to a different format. The commands are the same, except for the extension on the output file. The extension tells Pandoc what format it should create.

```
pandoc -s /filepath/Input_Filename.md  -o /filepath/Output_Filename.html

pandoc -s /filepath/Input_Filename.md  -o /filepath/Output_Filename.docx
```

You are not limited to HTML and DOCX files. Pandoc can convert Markdown files into slides, bibliographic formats, Jupyter notebooks, CSV data files, and multiple wiki languages too. This list shows all of the possible conversions you can do.

Optional Procedure 4/Demo 2: Using R's Bookdown to Compile Documents

R Studio has powerful libraries for creating online materials. The *bookdown* package uses Pandoc to combine individual .md files into full length technical books, complete with a linked Table of Contents. These books can be hosted online or stored and accessed in your own computer.

Bookdown is the package that was used to create the published version of the Resource Guide that you examined briefly at the start of Exercise 1. The Home page for the R Bookdown package has other examples of books that were created using this package.

1. Open R Studio.
2. Add the *bookdown* package to your R Studio setup using the command:

```
install.packages("bookdown")
```

3. The easiest way to start a new book project is from within RStudio IDE. Go to **File > New Project > New Directory > Book project using bookdown**. This creates a new folder and directory with an example book as a starter template.
4. Look at the template book files. Note that *bookdown* uses text files in R Markdown (.Rmd) format to build chapters (see Figure 5).
    - Each Rmd file contains one and only one chapter, and the chapter is defined by the first-level heading `#`.
    - Other than representing single chapters, the main difference from a plain .md file is that the ".Rmd" extension specifies the file should be opened in R Studio. Otherwise the file contents of .md and .Rmd files can be identical, and most text editors can open .Rmd and .md files both.

5. BEFORE making any modifications check that your template works by building the HTML version of the template book. Go into the Build pane, then click on **Build Book > bookdown::gitbook** (Figure 5, orange arrow).
6. Building the book may take a minute. Once it is finished, R Studio will display the book's landing page in a new window.
6. If your template builds correctly, you are ready to try creating new pages. Open an existing .Rmd file from the book, add some text of your own, then save the file **under a new name**.
7. Now try converting one of the .md files you made in Exercise 2 to a chapter. Open the .md document you created, then choose "Save As" and save it with the .Rmd extension. Be sure to save the new file to the folder where your bookdown demo book is located.
8. Build the book again. If you have formatted them correctly, the two new .Rmd files that you just created should now be incorporated into your demo book.

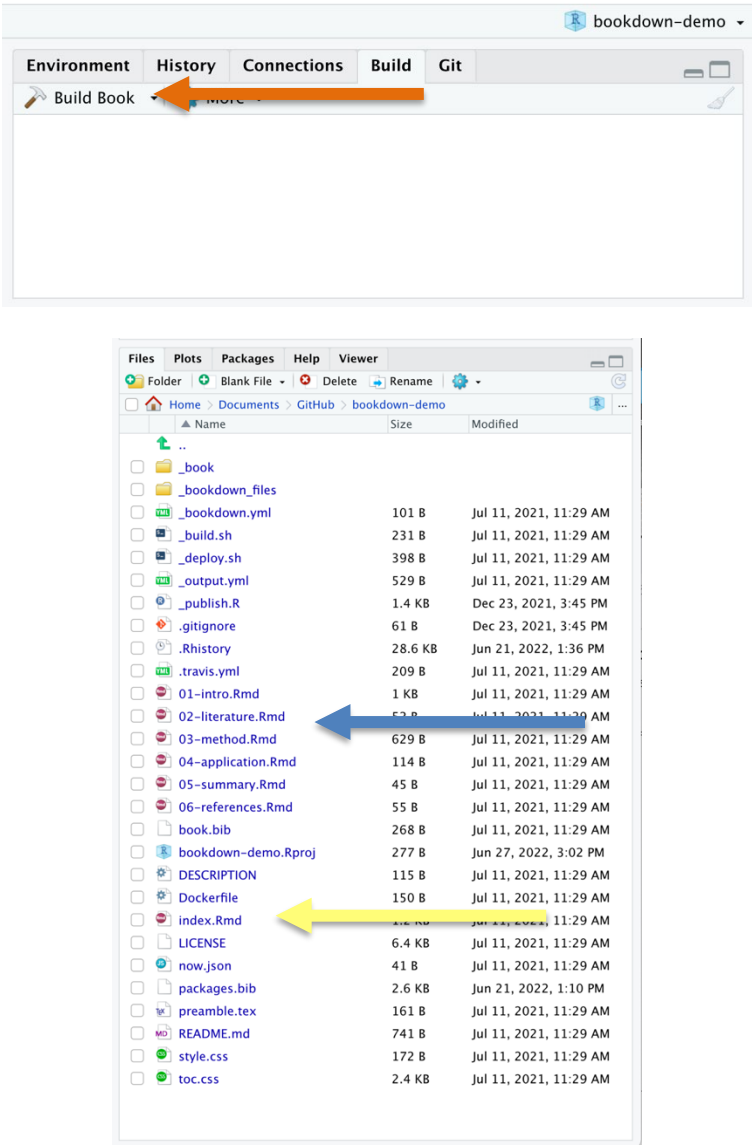


Figure 5. **bookdown** *uses text files in R Markdown (.Rmd) format to build chapters (blue arrows). The number in the file name determines the order in which content files are used in the book. The "Build" function (orange arrow) is part of the Environment panel. The index.Rmd file (yellow arrow) contains most of the book's structural information.*

How Does *bookdown* Determine the Structure of a Book?

Exercise 2 explained that reusable texts have instructions that control structure, how the output document will be generated, and what it will look like. In *bookdown*, book structure information is provided in the file named "index.Rmd" (see yellow arrow in Figure 5, and file in Figure 6.) *Index.Rmd* is the first file processed when *bookdown* builds a book. The instructions in the first few lines of the index.Rmd file control how other documents associated with the book are brought together. The layout and HTML format for the individual pages of the book is controlled by a separate .CSS file.

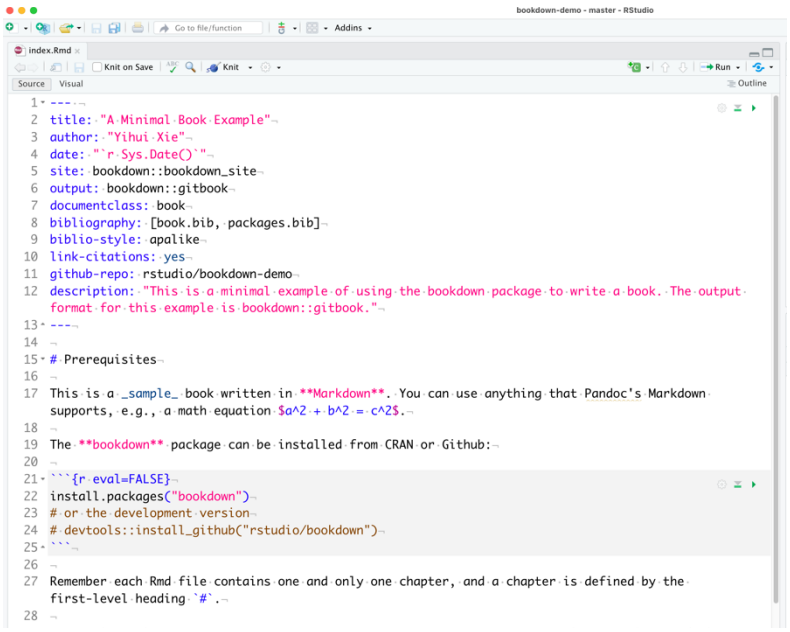


Figure 6. The index.Rmd file for the template bookdown project. Instructions for how to build the book and generate the final output are contained in the first 13 lines of code.

Getting Help With *bookdown*

This is a VERY brief introduction/demo. The simplest way to learn how to make books with bookdown is to build one. A practical first book project is a 7-10 page resource journal that documents how to make your book and pages behave a certain way.

Make a list of the 7-10 features you expect to use most often in future books. For example:

- How to add illustrations and figure
- How to insert equations
- Making tables
- Templates for specific pages
- Etc.

Create a new template book then make a separate .Rmd chapter file for each feature in your list. Use each chapter file to document how to accomplish that specific task. After creating 7-10 individual pages, build a book from them. By the time you have produced a journal that you like, you should have all of the skills you need (plus backup notes) to start more ambitious book projects.

If you cannot solve a particular problem, the RStudio community forum is a friendly place to ask any questions about bookdown. Be sure to use the bookdown tag on your posts to draw attention.

Exercise 4 (optional): Using GitHub to Back Up, Manage and Host Files

Up to this point you used GitHub as a source for files only, editing and saving your .md files locally. Connecting R Studio with a personal GitHub account lets you do much more.

- GitHub provides powerful tools for writing and sharing reusable texts, and for writing collaboratively.
- GitHub can store your text and data files securely. If you work across multiple computers (home and work for example), GitHub helps ensure you always are working on the most recently updated files. It also helps manage conflicting versions of files.
- Storing text files in GitHub repositories makes it easier to share and track use of your reusable texts and images.
- You can use .md files to set up and host a personal web site for each repository you have on your account. These sites make it easy to share resources with colleagues or students outside of your organization.
- If you are interested in using R or data science tools in your lab courses, then you definitely should become comfortable using GitHub.

Setting up Git, GitHub, and R Studio so they communicate properly can be difficult. Fortunately, there is Happy Git and GitHub for the useR, a very clear and easy-to-understand, step by step guide that takes you through the process of setting up everything from basic connections to advanced tools. It also introduces you to graphical interfaces like GitKraken and GitHub Desktop that make using Git and GitHub **much** easier.

Written by Jenny Bryan, former professor of statistics at the University of British Columbia and now a software engineer at RStudio, this book began as a guide for her STAT 545 course at UBC and has been extensively tested with students. You will not need every part of this book at first, but as your skills and needs grow, this book can help you take more advanced steps.

Materials

Appendix A lists the supplemental files for this workshop that are available on the ABLE publication site. One of the available files is current edition of the full Resource Guide in .docx format. Other files are available for download from GitHub repositories (links provided.) Appendix B is a compiled list of all links to tools and apps in the handout texts. Appendix C summarizes the overall strategy and specific practices used to deliver this workshop in hybrid format.

Notes for the Instructor

This workshop had three goals:

- Introduce and share our open-source Writing Resource Guide;
- Show potential users how to modify the Resource Guide (if needed) to meet their program needs; and
- Introduce participants to a more flexible writing workflow that we used when developing the Guide for publication.

Options for Using Our Resource Guide

There are several ways to access and/or revise the Guide to fit local needs.

MS Word Format

Users who do not need to translate their Guide between file types can edit the Resource Guide directly in Word. A complete version of the Guide in .docx format is included in the Supplemental Materials for the workshop. It also can be downloaded from the workshop repository.

Several chapters contain **Instructors Only** recommendations for revising the Guide to fit local requirements. This information should be deleted before posting the revised Guide for students.

The digital .docx file can be shared with students freely within the terms of the Creative Commons license. Note that printed copies of the document may be given to students but ***cannot be sold for profit, either directly or indirectly by a third party such as a for-profit campus bookstore.***

HTML Format

Follow the instructions in Exercise 1 to download a fully formatted and cross-linked version of the Guide in HTML format. This version of the Guide can be opened with any web browser. Students will need both the folder of HMTL pages and images subfolder.

To share the HTML version of the Guide with students:

1. Copy the file folder (and sub-folders) named 3_Writing_Resource_Guide.
2. Upload the folder (along with the Images sub-folder) to a web site or page that students can access.
3. Test the copy by opening the **Index.html** page. The index uses relative HTML links, which means all links in the Table of Contents should connect to their appropriate pages.
4. If the index page opens others correctly, check that images are being displayed correctly.
5. Assuming the previous two steps worked, the web version is ready. Students can access it by opening the **Index.html** page.

HTML is a good format for sharing the Guide with students but editing raw HTML files can be difficult. To make changes beyond short minor edits, we recommend modifying the .md files and regenerating the HTML instead.

Editing the .md Master Files

Editing the .md master files is the most reliable and flexible way to make large changes to the Guide. Users can replace or revise the order of chapters and modify the linked Table of Contents. Also, we routinely update and add new content to the Resource Guide; new content is written and shared as .md formatted files.

1. Clone the full Resource Guide repository from GitHub. This repo has additional files not provided in the ABLE 2022 workshop repository.
2. In R Studio, open the individual .Rmd chapter files and:
    a. Review our recommendations for localizing the Guide.
    b. Revise the text or remove any .Rmd files that do not match local needs and requirements.
    c. Add new .Rmd files as needed to incorporate local resources.
3. To compile the Guide into a book:
    a. Download and install the *bookdown* library. The installation process will notify you of any missing tools or R libraries needed.
    b. Follow instructions in *Demo 2* of the workshop handout for downloading and testing a simple demo e-book. If the demo book compiles properly, the *bookdown* installation is configured correctly.
    c. In R Studio, switch to the folder containing your revised .Rmd files.
    d. Open the index.Rmd file and run the **Build** command. This converts the .Rmd files to .html files.
4. After the revised Guide is converted to .html, share it with students either by:
    a. Posting the HTML files plus image folder on the local LMS, or
    b. Publishing the localized guide to a dedicated GitHub web site. This link is an example of the Guide on a publicly accessible GitHub web site.

These are two useful references for localizing and compiling your Guide in R Studio.

- Authoring Books and Technical Documents With R Markdown has detailed instructions for using and customizing a *bookdown* book.
- Using Markdown inside R provides additional information on writing Markdown specific to R and R Studio.

Anyone interested in writing new materials to expand the master version of the Writing Guide should contact the author directly or consult the STEM Writing Project web site.

Suggestions From the Workshop

Participants were rightly concerned with the steep learning curve. These tools **can** seem very challenging at first, partly because Markdown requires users to think about writing content independently from its structure and output elements. With a few hours of practice though, writing in Markdown can become almost second nature. It also can save many hours of time by making repetitive writing tasks easier.

Participants also wanted to have a working copy of the Resource Guide that they could begin using immediately. We have added that as an MS Word document to the Supplemental Files.

Acknowledgments

Thank you very much to all of the WFU BioCore students and teaching assistants who have helped improve our Writing Guide over the last ten years.

About the Author

Dan Johnson has been a teaching professor of biology at Wake Forest University since 1998. He teaches in the first-year sequence for majors, plus comparative physiology, cell and molecular biology, and cancer biology. From 1998 to 2021 he coordinated the undergraduate biology laboratory program. Currently his research focuses on using computational linguistics and data science methods to track and assess students' development as scientific writers.

Appendix A: List of Supplemental Files for the Workshop

If you plan to recreate the Guide using ***bookdown***, do not use the .md files in the ABLE 2022 Workshop GitHub repository. The ABLE workshop repository was simplified make it easier for participants to use. Instead download the current version of the official repository at https://github.com/adanieljohnson/SWP_student_writing_guide. It contains the full and most up-to-date file set for recreating the Scientific Writing Resource Guide in R Studio/*bookdown*

Writing Resource Guide Files	
1_SWP_Writing_Guide_6_30_22.docx	The June 2022 edition of the Writing Resource Guide, in MS Word .docx format. This is a standalone version of the Guide that can be edited without using R Studio. Sections highlighted in yellow should be modified, revised, or removed before sharing the Guide with students.
images.zip	A compressed archive containing all images using in the June 2022 edition of the Guide

Workshop Slides and Supplemental Files	
2_ABLE_2022_Workshop.pptx	Powerpoint slide show from the workshop
3_ABLE_2022_Workshop_Markdown.md	Handouts from the workshop in their original Markdown (.md) format
4_ABLE_2022_Workshop_Raw_ conversion.docx	Workshop handouts (File #3) immediately after conversion using Pandoc/R Studio
5_ABLE_2022_Word_Template.dotx	Sample MS Word template used to modify headings and text formats for a Pandoc output file
6_ABLE_2022_Workshop_Cleaned_ conversion.docx	Workshop handouts after reformatting. This file was produced by copy/pasting text from File #4 (the raw Pandoc-formatted output file) into File #5 (a formatting template)
7_ABLE_2022_Workshop_Pandoc_to_ HTML.html	HTML formatted version of workshop handouts produced by processing File #3 using Pandoc via Terminal command
8_ABLE_2022_Workshop_Ashkanph_to_ HTML.html	HTML formatted version of text produced by processing File #3 using Ashkanph online converter

Appendix B: Resource Links

Links to resources described in the workshop handouts are provided again here for convenience, along with additional related resources.

Getting Started

- WHY and HOW to write well-structured, reusable texts
- Workshop Repository and Writing Resource Guide

Free Plain Text Editors

Many Markdown users prefer to keep their workflow consolidated, and use R Studio for both writing content and converting it to other formats. Others (including the workshop presenter) prefer to use a separate text editor. Either works; the decision should be based on which way is most comfortable for the user. Those interested in comparing R Studio vs. a plain text editor should try these out.

- Microsoft's Visual Code Studio is a very powerful open-source text and code editor. It probably is one of the most popular code editing platforms in the world. It comes with support for Markdown files right out of the box. The main drawback is that some commands are not where you think they should be.
- Google Text Editor is available to anyone with a Gmail account. It is not as fully featured as many editors but does recognize Markdown natively.
- Brackets is popular with many code and text wranglers. It displays Markdown but does not export it easily.
- Notepad++ is a good text editor for Windows. MarkdownViewer++ plugin adds the ability to render and export Markdown contents to HTML. Detailed instructions for installing the MDV++ plugin are available here and here.

Markdown-Native Editors

- HarooPad produces very clean HTML.
- StackEdit integrates with Dropbox, Google Drive, and other online services. A good choice if you mostly work with HTML and need tight integration with Google Drive.

Using GitHub

- Install GitHub Desktop
- GitHub home page
- Getting Started With GitHub Desktop

Markdown Syntax and Add-Ons

- GitHub's Introduction to GHFM
- Quick Guide to Mastering Markdown
- Extended Syntax for Markdown
- Emoji cheat sheet. There are two ways to add emojis to Markdown: copy and paste, or type in emoji shortcodes. GHFM does not support emoji shortcodes, so copy/pasting from this cheat sheet is your best option.
- Making Tables in Markdown
- Web-based Markdown Table Maker
- HTML shortcuts
- HTML special symbols
- Writing LaTeX equations
- Online equation builder
- Official Specification for GHFM is the complete documentation.

File Format Conversion

- To HTML5 for the web:
  - MD to HTML Converter
  - Dillinger
  - Markdown It
  - Markdown to HTML

- To MS Word documents:
  - Pandoc Converter

R and R Studio; Bookdown Package

- Install R.
- Install RStudio.
- Installing R and R Studio, Step by Step
- Using Markdown inside R
- Home page for the R Bookdown package
- Guide to using Bookdown

Commercial Products

- Marked 2 is an inexpensive desktop application for previewing Markdown. Mac only.
- Markdown Pad is similar to Marked 2, but for Windows.
- Typora lets you write and view Markdown immediately. Costs $15. Works well for Word documents but adds extra code to HTML.
- Caret is a newer cross-platform tool.

Appendix C: Improving Zoom Hybrid Workshops

The strategy I describe here stems from my experience when I had to cancel my travel plans for the ABLE 2022 Conference. I documented what I learned assuming other hosts and presenters might run into a similar situation in the future. I also included some things I **wish** I had done that would have made the workshop run more smoothly. If you have questions, please contact me directly.

Overview

If you cannot travel to the ABLE meeting to lead your workshop, do not automatically assume you must cancel it. Some workshops can be modified and presented via Zoom with help from onsite assistants or other ABLE members. This will not work for extensive wet lab activities but is feasible if most of the work is done on paper or computers.

Before You Decide to Go Hybrid

1. Contact the Major Workshop Committee first. They can help you decide whether offering your workshop in hybrid format is a realistic option. They also can help you decide when to tell the host, who already has a lot on their plate.
2. Be **brutally** honest with yourself about whether you have the time and energy to make the switch. Making a last-minute change requires considerable work. Consider too how many participants you have signed up. You may not have anyone drop out, but 50+% of registered participants may opt to go to another workshop. Your group could become very small if you already had a small group. **It is nothing against you**; people are simply tired of Zoom and want live interactions.
3. When you talk with the local host:
    – Ask what **their** preference is. You may be ready to go hybrid, but the host may not have the resources you would need, or time or sufficient help to adjust their existing plans.
    – If they want to go forward, ask how many participants have signed up for your workshop. How you should structure the hybrid session will depend partly on whether you have two full workshops of 20-24 participants, or smaller groups.
4. Assuming the joint decision is to go forward, keep the Workshop Committee and the host informed as you make changes and plan your hybrid session.

Planning For the Workshop

1. Get a list of the registered workshop attendees with their email addresses as early as you can. There may be ABLE members in the workshop that you can ask to be local assistants for you. Also, you likely will need to contact your attendees and send them information in advance.
2. Ask the host to put you in contact with a student or staff volunteer that can work as your onsite assistant before AND DURING the workshop.
    – This is IN ADDITION TO any ABLE member you recruit. The onsite assistant knows their local department's practices and resources. They can do things a non-local ABLE member cannot.
    – This is unlikely to be possible, but if you have a choice, pick an onsite assistant who has some basic knowledge of the topic of the workshop, or experience using the primary software you plan to use.

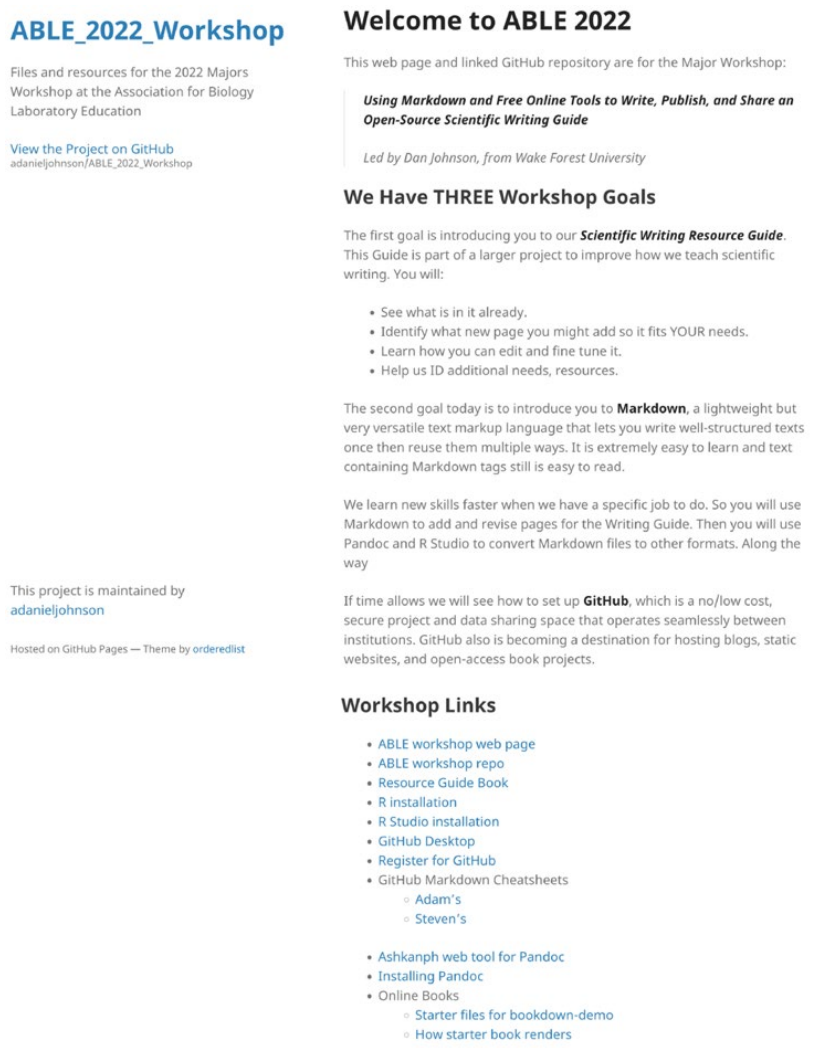
Revise and Share Workshop Handouts & Online Information

1. Look CLOSELY at your submitted workshop materials.
    – What parts **require** you to be physically present to demonstrate? Can you revise those so that participants do not need to do them, or those parts can be optional?
    – Which of your learning goals for the workshop is **most** important? For example:
        • In my face-to-face workshop I had planned to teach participants how to use two tools (Markdown and GitHub), but I knew this would not work unless I was there to troubleshoot GitHub installations.
        • For the hybrid workshop I modified the workflow so participants did not need to register for GitHub.

        - While I had to drop one of my original workshop goals (introducing ABLE members to GitHub), doing so let me spend more time teaching participants about Markdown, and achieve one of my original goals fully.
    - What software or apps do participants need to use?
        - Can they download them in advance?
        - Could the host provide computers with the software pre-loaded?
        - Could you provide the software installed on a flash drive, so it is "plug and play"?
2. After you revise your workshop materials have someone try them out.
    - Try to pre-test your revised workshop at least once using Zoom. Set up two computers in adjacent rooms so you can troubleshoot if needed.
    - Have someone test your set-up who does NOT do the procedure. Remember that you know the procedure, but workshop attendees do not.
    - **Pay attention to how long activities take to complete.** This is something I did not do, and as a result I greatly under-estimated the time participants would need to complete certain steps of the procedure.
3. You likely are switching at the last minute, so assume that the workshop binder materials have already been formatted and sent out by the host. Plan to send any revised workshop materials directly to registered attendees in advance. Include the date of the update in the header or footer of the file.

4. It is **very helpful** if you can set up a simple static web page (see Figure C.1.) that:
    - Summarizes how you have revised the workshop (a few sentences is enough);
    - Outlines the work plan for the workshop; and
    - Provides an organized list of all the links that your participants will use;
        - Having all of the web links in one place makes it easier for participants to access the resources.
        - It is very likely that you will have to adjust the order of activities during your workshop. Having your links in one place means participants do not need to scroll up and down through the workshop handouts.

5. Meet with your onsite assistant via Zoom in advance. Have a technical rehearsal where you walk them through what you plan to do.
    - Ideally, have them try out the room and equipment that the host plans for you to use during your workshop.
    - The technical rehearsal can identify potential problems while there still is time to respond.
    - This is something I did not do, and we had sound/video problems as a result.
    - Identify specific tasks or questions your local assistant will manage for you.



ABLE_2022_Workshop

Files and resources for the 2022 Majors Workshop at the Association for Biology Laboratory Education

View the Project on GitHub
adanieljohnson/ABLE_2022_Workshop

This project is maintained by adanieljohnson

Hosted on GitHub Pages — Theme by orderedlist

Welcome to ABLE 2022

This web page and linked GitHub repository are for the Major Workshop:

> ***Using Markdown and Free Online Tools to Write, Publish, and Share an Open-Source Scientific Writing Guide***
>
> *Led by Dan Johnson, from Wake Forest University*

We Have THREE Workshop Goals

The first goal is introducing you to our ***Scientific Writing Resource Guide***. This Guide is part of a larger project to improve how we teach scientific writing. You will:

- See what is in it already.
- Identify what new page you might add so it fits YOUR needs.
- Learn how you can edit and fine tune it.
- Help us ID additional needs, resources.

The second goal today is to introduce you to **Markdown**, a lightweight but very versatile text markup language that lets you write well-structured texts once then reuse them multiple ways. It is extremely easy to learn and text containing Markdown tags still is easy to read.

We learn new skills faster when we have a specific job to do. So you will use Markdown to add and revise pages for the Writing Guide. Then you will use Pandoc and R Studio to convert Markdown files to other formats. Along the way

If time allows we will see how to set up **GitHub**, which is a no/low cost, secure project and data sharing space that operates seamlessly between institutions. GitHub also is becoming a destination for hosting blogs, static websites, and open-access book projects.

Workshop Links

- ABLE workshop web page
- ABLE workshop repo
- Resource Guide Book
- R installation
- R Studio installation
- GitHub Desktop
- Register for GitHub
- GitHub Markdown Cheatsheets
  - Adam's
  - Steven's
- Ashkanph web tool for Pandoc
- Installing Pandoc
- Online Books
  - Starter files for bookdown-demo
  - How starter book renders

Figure C.1. This is the static GitHub web page for this workshop. It outlines the goals and workflow. A list of workshop links connects participants to every resource they need during the workshop, as well as supporting resources. We did not have time to get to some workshop activities, but participants still had the resource links and instructions for future use.

If I were doing this workshop again, I would recommend numbering the workshop links. This would make them easier to reference.

Equipment

Equipment choices affect the overall workshop experience on **both sides**.

On the Presenter's Side

Buy, borrow, or beg:

1. The largest monitor your presentation computer can support.
   – You are likely to be monitoring several documents at once, along with the Zoom participants. This is easier with a larger monitor.
   – I used a 27-inch UHD monitor and had relatively few problems.
   – Consider installing a window-management program like **Divvy** to divide up your screen into working panels.

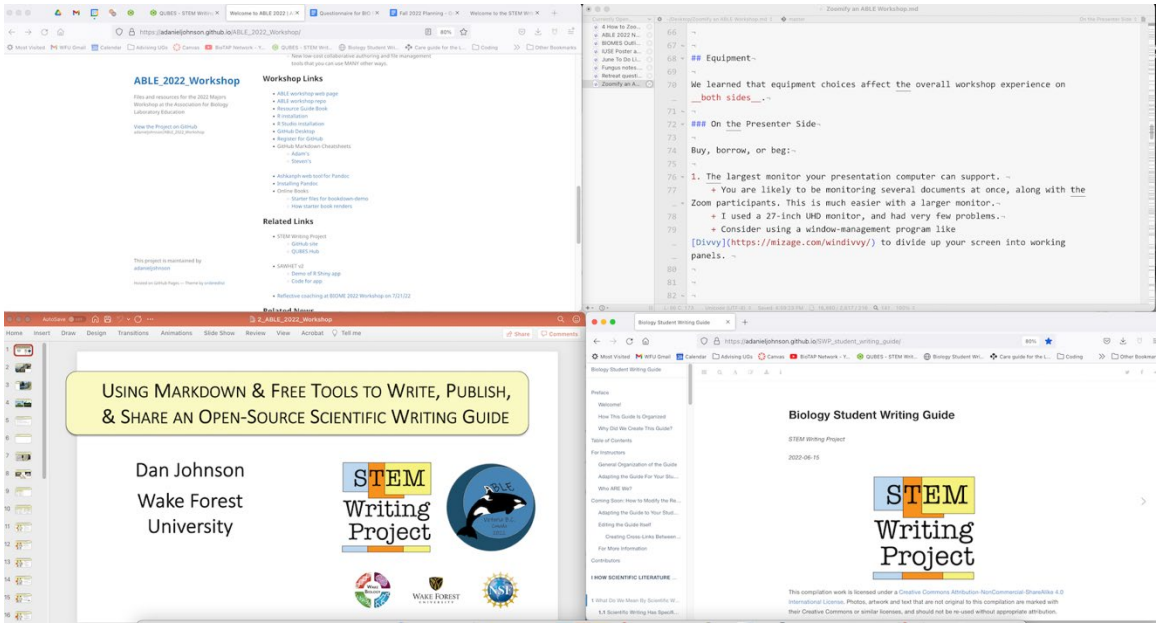


Figure C.2. This screenshot shows how a windows manager like Divvy can automatically resize program windows and place them in specific screen positions.

2. An HD-quality web camera, preferably separate from the fixed laptop camera. We found that moving AWAY from the camera made the workshop feel more natural. If you put the camera further from the speaker, how they look on Zoom better approximates how they look in real life.
3. A wireless Bluetooth headset with microphone. Even low-end headsets have better microphones than a laptop, which will make it easier for participants to understand the speaker. A headset also makes it easier for the presenter to hear questions from participants.

On the Host's Side

1. An HD quality computer projector. This is the only non-negotiable item. Low quality video display is tiring for participants to watch, and they cannot read text documents the presenter shows.
2. External speakers. Laptop speakers are designed for listeners within 3-6 feet of the computer. If the laptop will not send audio to built-in room speakers, you can buy decent speakers for under $100. The bass booster box is not needed.
3. An external microphone. Laptop microphones are designed to capture sound within 3 feet of the computer. Sounds from further away are degraded and hard for the presenter to make out. An external omni-directional microphone designed for conference rooms costs less than $100 and makes it **much** easier for a presenter to hear questions.

Before the Workshop

1. Send participants a clear agenda, and any pre-workshop instructions several days in advance.
2. Have participants download essential software in advance, OR have the onsite assistant download it to host-side computers.
3. Even though participants have the workshop handouts in electronic form, ask the onsite assistant to print out several hard copies. This makes it easier when participants need to refer to their computer screen at the same time that they are reading directions from the handout.

During the Workshop

1. Start the Zoom session 30 minutes in advance of the workshop's start time. Chat with the participants as they come in. This lets you:
   - Check that you can see where participants are sitting;
   - Check both sound and video on both ends;
   - Set a tone for the workshop that it will be an interactive experience.

2. Take a screenshot of the first slide of your workshop and use it as your background image in Zoom. Turn it off once the workshop starts.
3. If possible, ask participants to log into Zoom themselves so they can follow along individually, not just from projected screen. This is easier if:
   - Participants have their own headphones (to prevent feedback and echoes);
   - The local network is stable and reliable for visitors.

4. When you start the workshop, ask participants to use Zoom's chat as an open conversation space. They can:
   - Ask questions of you;
   - Ask the local participants for help;
   - Point out when they are lost;
   - Etc.

5. Don't spend a lot of time introducing the workshop. Get participants working on something within the first 10-15 minutes.
6. Break up the material you present by lecturing or presenting slides. Ideally, you should not talk more than 10 minutes at a time. Organize content into bite-sized chunks. Attention strays after 15-20 minutes, so plan to change up what participants are doing accordingly
7. Have your onsite assistant and/or ABLE helpers circulate as the group works. Assistants can also watch the Zoom chat and forward any questions from the group to you.
8. Be careful when showing text. If you have a lot of text or the projector is lower resolution, participants may not be able to see what you are showing. This was a problem that I had in my 2022 workshop.
9. Plan how you will budget your time **in advance**.
   - A rule of thumb for group training sessions is that workshop group will need 25-50% more time to complete an activity than your pre-tester did.
   - If one activity is taking too long, use the time when teams are working (next step below) to decide what you can adjust or drop. When I was revising my hybrid session, I pre-selected steps and demonstrations ahead of time that I could shorten or drop entirely. This turned out to be a good strategy, because I greatly under-estimated how long the primary activities would take.

10. **Assume** that you will need to modify your plans as the workshop unfolds. For example:
    - In my workshop I wanted to collect data from participants, but we did not reach an appropriate end-point. I adjusted by asking participants to provide it by email.
    - I had a simple form prepared that I planned to use to collect the information. I adjusted by asking them to complete the form as post-workshop feedback.
    - One participant made a suggestion to give everyone 1-2 weeks to digest the workshop before asking them for their feedback. This might not work for your workshop, but makes sense given what my goals were.

Managing the Group

1. Have participants work in teams of 2-4.
2. DON'T use Zoom breakout rooms. ABLE workshops are loud and chaotic, with people sharing ideas freely. Teams can and should be sharing information and helping each other. Let people talk between teams and help each other; just ask them to write down questions and answers in the free-flowing chat.
3. Document the questions and trouble points.
    - Provide a shared document where participants can post questions.
    - Have the onsite assistant keep notes of common questions.
    - Save a copy of the free-form chat, and pull out questions and answers.
    - Compile this information and include it as part of your post-workshop write-up.

4. If you do not have a web page set up for this, have a paste-ready list of links for workshop materials. Copy/paste links into the chat window as needed when questions arise.

Closing the Session

1. Help the participants see how the pieces of the workshop connect by:
    - Summarizing what they did, and/or what skills or tools they learned.
    - Talking briefly about what they can do to expand on those skills and tools.
    - Providing suggestions for where they might learn more.

2. If you plan to follow up (share more materials, request feedback), ask your participants what a reasonable timeline is.

Tips for Making Zoom Feel Less Artificial

I tested these methods during my 2022 hybrid workshop and participants said they liked them.

1. Move back from the camera. A problem with Zoom is that we see each other from a much closer point of view than we do in everyday life. Moving back from the camera is more natural, and lets others see your hand gestures and body language.
2. Set up and present in a lab, not your office. We are talking about lab activities after all.
3. Don't try to switch between multiple documents rapidly using single screen-shares. Keep the workshop less formal by sharing your entire screen.
    - Put the windows you want to focus on in quadrants on the screen, and **describe** where they are (right side, left side, top half, bottom half).
    - In your computer preferences, enlarge the size of your pointer so it is easy to see.

Suggestions From the ABLE Workshop Participants

1. Put page numbers on handouts so you can refer people to specific pages. If you have multiple handouts, give each one a letter name, so you can refer to Handout A p.4, Handout B p.2, etc.
2. Include the Zoom workshop recordings as part of Supplemental Materials in the workshop write-up. Trim the recordings down as much as you can; unedited HD recordings of 3-hour workshops can be 1-3 GB files.
3. Make full use of your onsite assistant. Give them specific tasks to do, data to collect, or a checklist to complete with the participants. Their presence reinforces that the participants have someone looking out for their needs.

Published Tips

While preparing my own workshop, I learned of other tips for making Zoom sessions more engaging that might be useful for others, even if I did not use them.

1. Start with music or a short video! The video does not need to be directly associated with your workshop, just entertaining.
2. If it is appropriate for how your workshop runs, assign roles to individuals.
3. Provide as many opportunities for collaboration as make sense. For example:
    - Open the Zoom whiteboard.
    - Enable the slides annotation features.
    - Power up the chat function. Create a live discussion by having participants share their thoughts, comments and reactions.
    - Use collaborative cloud-based apps to improve interactivity.
    - Use interactive Zoom polls to vote on topics or get feedback. These are a great way to find insights and valuable information about your audience that will inform the rest of the presentation. They don't require verbal participation, and they're accessible on all devices.
4. Try thinking of new ways to use virtual backgrounds. For example, have users choose a solid green or red background to quickly show if they agree or disagree with a topic (hint: use Grid view for this).
5. Keep it conversational. Bringing in a new voice, face, or background makes it more of a conversation, and regains the social elements of workshops.
    - Find a partner with whom you can interact with online and create a dialogue that helps make the experience feel informal.
    - If you can't find another person to interact with, refer to something physical in the participants' environment to create a sense of change.

In Summary

I hope your workshop goes as planned and that you never have to make a last-minute change like this. If it DOES happen though, advance planning and staying flexible during the workshop will go a long way towards making it a good experience for your participants.

Mission, Review Process & Disclaimer

The Association for Biology Laboratory Education (ABLE) was founded in 1979 to promote information exchange among university and college educators actively concerned with teaching biology in a laboratory setting. The focus of ABLE is to improve the undergraduate biology laboratory experience by promoting the development and dissemination of interesting, innovative, and reliable laboratory exercises. For more information about ABLE, please visit http://www.ableweb.org/.

Papers published in *Advances in Biology Laboratory Education: Peer-Reviewed Publication of the Conference of the Association for Biology Laboratory Education* are evaluated and selected by a committee prior to presentation at the conference, peer-reviewed by participants at the conference, and edited by members of the ABLE Editorial Board.

Citing This Article

Johnson AD. 2023. Using Markdown, R Studio, and free tools to write, publish, and share an open-source scientific writing guide. Article 10 In: Boone E and Thuecks S, eds. *Advances in biology laboratory education*. Volume 43. Publication of the 43rd Conference of the Association for Biology Laboratory Education (ABLE). https://doi.org/10.37590/able.v43.art10