Published in final edited form as:
Math Biosci. 2022 July ; 349: 108844. doi:10.1016/j.mbs.2022.108844.

R Markdown as a dynamic interface for teaching: Modules from math and biology classrooms

Kristine L. Grayson[a,1], **Angela K. Hilliker**[a,1], **Joanna R. Wares**[b,1,*]

[a]Department of Biology, University of Richmond, United States of America

[b]Department of Mathematics and Computer Science, University of Richmond, United States of America

Abstract

Advancing technologies, including interactive tools, are changing classroom pedagogy across academia. Here, we discuss the R Markdown interface, which allows for the creation of partial or complete interactive classroom modules for courses using the R programming language. R Markdown files mix sections of R code with formatted text, including LaTeX, which are rendered together to form complete reports and documents. These features allow instructors to create classroom modules that guide students through concepts, while providing areas for coding and text response by students. Students can also learn to create their own reports for more independent assignments. After presenting the features and uses of R Markdown to enhance teaching and learning, we present examples of materials from two courses. In a *Computational Modeling* course for math students, we used R Markdown to guide students through exploring mathematical models to understand the principle of herd immunity. In a *Data Visualization and Communication* course for biology students, we used R Markdown for teaching the fundamentals of R programming and graphing, and for students to learn to create reproducible data investigations. Through these examples, we demonstrate the benefits of R Markdown as a dynamic teaching and learning tool.

Keywords

R markdown; Data visualization; Pedagogy; Herd immunity; Teaching programming

1. Introduction

Interactive tools can engage students in the classroom and make computational research and programming more accessible and exciting. R is an open-source computing language that is rapidly increasing in prevalence as a tool for analysis and research [1,2]. RStudio

[*]Corresponding author. jwares@richmond.edu (J.R. Wares).
[1]All authors contributed equally to this work.

Declaration of competing interest
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data
Supplementary material related to this article can be found online at https://doi.org/10.1016/j.mbs.2022.108844.

is a popular and free integrated development environment (IDE) for scripting and project management. Incorporating R in the classroom requires a commitment to teaching coding skills along with course content [3,4]. While the learning curve can be intimidating for instructors and students, there are extensive online resources for teaching and learning R and RStudio. More recently, integrated tools in RStudio have improved the interface for documenting code and can be leveraged to guide students through hands-on experience working with data and coding [5]. In this article, we describe using RStudio and R Markdown as a dynamic authoring framework in two courses at the University of Richmond, one in mathematics and one in biology.

R Markdown provides an interface in RStudio that allows authors to mix formatted text with code, which can be rendered into a complete final document [6]. R Markdown provides a significant advance for easily-formatted reproducible workflows, and its benefits for research translate into substantial advantages for teaching [7,8]. For an instructor, the interface can be particularly useful for building structured examples and assignments. For the students, explanations of the material can be mixed with open code blocks (termed “code chunks”) that students fill and execute, with as little or as much scaffolding and guidance through the code as desired. Students can also respond in text to questions included in the module or assignment, or include graphics. R Markdown has benefits for mathematicians and others familiar with LaTeX, as they can use LaTeX tags inside of the document to create professionally formatted equations.

R Markdown files have a variety of final output options. Upon completion of the assignment, students can easily compile, or “knit”, the instructor-generated R Markdown file to create a formatted, polished report. The uniformity of the final reports makes grading easier than open format projects. The format of the final document can be static (such as a PDF) or dynamic (such as HTML and dashboards) formats, allowing the distribution of the final product through a variety of media. R Markdown also demonstrates the value of reproducible research to students, where code for models or data analyses can be interspersed with exposition, and others can easily re-run the code and understand the results.

There are a range of environments to run R, depending on the course needs and budget. As the desktop version of RStudio is free, students can use it in computer labs or on their own machines. For centralized access, our courses utilized RStudio Cloud, as described here, but a course could use RStudio Server or Workbench [9]. For classrooms or collaborative research groups, these options allow scripts to be accessed from a single location with various permissions. These environments are helpful for instructors, both because they remove the need to troubleshoot students’ individual operating systems and they provide an easy mechanism for sharing course material. For example, in RStudio Cloud, projects can include pre-loaded scripts, datafiles, and installed packages ready for students; this is particularly helpful when students are new to R. RStudio Cloud Free currently has a limit of 50 projects using up to 1 GB RAM per project with 25 h of usage per month or 75 h for $5/ month (April 2022 rates). Under the free plan, projects must be shared by links and stored in individual workspaces for students to have individual copies of scripts. Cloud Instructor or Cloud Organization paid plans enable features for teaching, such as posting projects as assignments that are automatically cloned as they are opened by each student under a central workspace for the course. The course administrators have access to all the cloned versions, making it easy to access student work for troubleshooting and grading. Cloud Instructor plans allow unlimited projects with up to 16 GB RAM per project and cost $15/ month, with student access costing $5/ student/ month or the institution paying for workspace hours over 300 h per month (April 2022 rates). One limitation in RStudio Cloud, at the time of this writing, is that each project can only be open by a single user at a time, meaning that viewing and editing the same project cannot be done concurrently; however, beta testing for collaborative editing for up to 5 users at a time was announced in April 2022.

In this paper, we describe the structure and functions of R Markdown (Section 2) and ways it can be used in teaching, from highly structured assignments to building student confidence for independent coding and reproducible analyses (Section 3). We provide examples from two courses at the University of Richmond taught in Fall of 2020 and Spring 2021. One course was an upper-level undergraduate course, *Data Visualization and Communication for Biologists* (shortened to *Data Visualization* in this manuscript), taught in the Biology Department by Dr. Angie Hilliker and Dr. Kristine Grayson. This course used R Markdown to teach students R programming to manage large datasets and produce creative and compelling data visualizations. The other course was an upper-level undergraduate course, *Computational Modeling in Public Health and Injury Science* (shortened to *Computational Modeling* in this manuscript), taught in the Department of Mathematics and Computer Science by Dr. Joanna R. Wares. R Markdown was used to guide students through mathematical modeling and using R coding to develop, numerically solve, and analyze ordinary differential equations models. In Section 4, we provide example course modules to illustrate different ways of using R Markdown in teaching.

1.1. Data visualization and communication for biologists

The Biology curriculum at the University of Richmond emphasizes quantitative literacy throughout the introductory course sequence, but these courses are limited by the tradeoff between teaching content and data skills [3]. We noticed students in upper-level courses still struggled with organizing, summarizing, and graphing data, and wanted more time to engage students in thinking deeply about the presentation of data in biology and society. In Spring 2020, A. Hilliker and K. Grayson designed a *Data Visualization* course for upper-level biology majors. Students had taken a three-course introductory biology sequence, but ranged in experience from second year students taking their first upper-level biology course to fourth year students with extensive experience in upper-level coursework (enrollment = 18 students, 2nd year = 2, 3rd year = 3, 4th year = 13).

We aimed to introduce software tools and programming for working with complex datasets to biology students with little or no coding experience. There was a wide range of coding experience at the start of the course, with nearly 40% of students reporting they had never heard of R and 17% rating themselves as experienced with R. The main texts we used for teaching the basics of R programming were *Data Visualization: A Practical Introduction* [10] and *R for Data Science* [11], both with versions available as open education resources (OER). For the data visualization and communication concepts in the course, the main texts we used were *Fundamentals of Data Visualization* [12] and *Calling Bullshit: The Art of Skepticism in a Data-Driven World* [13]; the former text is available as OER and the later has teaching resources freely available online. On the topic of data communication, we emphasized issues of scientific ethics, appropriate practices in data inference, refuting misinformation, and equity in data collection and usage. Students learned how to create truthful, beautiful data visualizations and to recognize common pitfalls in data management and presentation. The course explored these principles through examples from both the media and scientific publications.

In the first portion of the course, we introduced students to Tableau to allow students to quickly experiment with graph type selection and stylistic choices. The second half of the course focused on introducing students to programming logic and syntax, teaching the importance of reproducible workflows, and demonstrating how coding can provide easier processing of large datasets and flexibility in visualization choices. While students appreciated exposure to multiple platforms, we shifted the majority of class exercises to R for the next offering of the course.

Given the increasing recognition that the process of science is more iterative and discovery-based than the linear scientific method, we view exploratory data analysis skills as essential for current biology students. Using R in undergraduate biology courses takes time for students to develop confidence and independence with coding [3]. Having an entire course focused on data skills allowed our students to develop their programming skills more than time allows in most biology courses. By the end of the semester our students could investigate complex data sets, discover trends in the data, and communicate evidence-based findings from biological hypotheses, as shown by demonstrating competency in these learning objectives in their capstone project (Section 4.3). In this capstone, students worked with large public datasets requiring data organization and processing, and constructed graphs to communicate the conclusions from their analysis. Students created an R Markdown report that presented their topic and documented their code and visualization choices.

1.2. Computational modeling in public health and injury science

In the fall of 2020, at the height of the COVID-19 pandemic and before a vaccine was authorized for use, J. Wares developed a *Computational Modeling* course and used R Markdown as a way to flip the classroom. The technique of teaching mathematical modeling in R Markdown was implemented during a rapid shift to online and hybrid teaching, but ended up providing a pedagogically useful tool that was later adopted for in-person course offerings. The scaffolded teaching approach presented in the *Computational Modeling* module (Section 4.2) can work in almost any learning environment.

The focus of the class was learning how to model COVID-19 using ordinary differential equation models and agent-based models. There was also some attention to parameter estimation but the focus was dynamic models, as opposed to statistical models. For this course, the main ideas needed to complete the modules were first presented by the instructor during class. Next, students would work through the R Markdown file or other activities with the aid of the instructor and their classmates. The primary foci of the course were (1) determining which assumptions were needed or were being made in the model, (2) learning how to numerically solve the differential equations, in R, and agent-based models, in NetLogo, and (3) mastering the skill of interpreting the numerical and visual results. Work was primarily completed in R Markdown with a focus on reproducibility. In 2020, the *Computational Modeling* course was taught completely online (synchronously over Zoom) but the modules that were created would also work equally well in a live classroom.

Students began the course by learning the fundamentals of the R language, then discussed some simple regressions and why they do not make good predictive models for epidemics. Students then studied models that included more epidemiology. The focus was the standard susceptible–infected–recovered (S-I-R) and susceptible–exposed–infected–recovered (S-E-I-R) models, which were later extended by the students. R Markdown modules were used to introduce these topics. The combination of text explanations with coding examples facilitated teaching students how to model various problems and generate numerical solutions all in one document.

Throughout the semester, students were also designing models of their own to answer questions of their choosing for semester-long projects. They extended the S-I-R models or agent-based models, to include other important components of COVID-19 dynamics, such as isolation and quarantine, mask usage procedures, and testing. Students also focused on developing models that could provide insights as to why distinct demographic groups were having different outcomes. The modeling and simulation skills learned in the R Markdown activities provided students with enough knowledge to create their own models and simulations in R, and to build concise reproducible reports.

2. Structure of R Markdown

In this section, we provide a short explanation of the components that create an R Markdown document, as well as particular optional features that we used in our assignments. RStudio provides a thorough tutorial for creating R Markdown documents [14] and a helpful cheatsheet [15].

2.1. Installing R Markdown

RStudio has R Markdown integrated within the IDE, where users create a new R Markdown file directly from the “new file” menu in RStudio. However, the ‘knitr’ package must be installed on each machine or server. In RStudio, select “Tools”, then “Install Packages” and type ‘knitr’ into the search bar. To use R Markdown without RStudio, install the ‘rMarkdown’ package in R with the command install.packages(“rMarkdown”)

Then create files with the file extension ‘.Rmd’ to signify to R that this is an R Markdown file.

The files we have created contain the following components (Fig. 1):

1. A header written in YAML (optional, defined below)
2. Code chunks that are executable inline or when the whole document is rendered
3. Text that can be formatted using the syntax of Markdown

4. LaTeX (where needed) to format mathematical equations

2.2. YAML header

YAML is a recursive acronym that stands for “YAML ain’t markup language” [16] and is used at the top of a file to assign different attributes to the file. YAML code is placed inside of “−−−” tags. In the YAML code, the author sets features of the document. For instance, in the Herd Immunity example (Section 4.2), we use the following YAML header code:

```
−−−
title: “Herd Immunity”
author: “Joanna R. Wares”
date: “‘r format(Sys.time(), ‘%d %B %Y’)’“
output:
_html_document: default
_pdf_document: default
bibliography: S1_References.bib
csl: S1_References.bib
−−−
```

The syntax of the paired code indicates the formatting of the final document. In the code block above, we have introduced a title, an author, the date of production, and have also set output file types.

The author can optionally also add a bibliography, included in the header code as above by referencing a “.bib” file (S1), which is a type of file usually used inside of LaTeX documents to create bibliographies. Most reference manager software allows the author to export their citations to a .bib file. The author then uses the @ symbol, with the name given to the citation in the bib file, to reference the document inline (below was referenced as @van_den_driessche_reproduction_201). The author can also put brackets around the tag for formatting if preferred. These files can also be created in any regular text software as a list of citations with a particular format [17], however sometimes exported citations have characters that pandoc-citeproc (which processes the file in R Markdown) does not support and the author must delete some fields to knit the document properly. An example of a citation in this format used in the example in 4.2 is:

```
@article{van_den_driessche_reproduction_2017,
title = {Reproduction numbers of infectious disease models},
volume = {2},
issn = {2468-2152},
doi = {10.1016/j.idm.2017.06.002},
number = {3},
journal = {Infectious Disease Modelling},
author = {van den Driessche, Pauline},
month = jun,
year = {2017},
pages = {288--303},
}
```

There are different field options for various citation source types [17]. Optionally, the author can control the style of the bibliography using the YAML tag “csl” for Citation Style Language followed by the appropriate filename defining the style that ends with the .csl extension. Make sure this file is placed in the same directory as the R Markdown file. Here we used S1_Mathbiosciences.csl (S1), which is a file that we downloaded and renamed that is generated by Zotero.

2.3. R code chunks

To execute code within the R Markdown file, the author must create an R code chunk. A code chunk is created by typing “‘{r}”’ (Fig. 1). Alternatively, the author can insert a code chunk by using the “Insert” button in RStudio or using a keyboard shortcut. The {r} syntax directs the ‘knitr’ package to execute anything between the “‘tags as R code. Within the {r} syntax, users should add a unique chunk name (such as ‘setup’ in the sample code below) to indicate the purpose of that code chunk and add optional parameters (such as ‘include=FALSE’) to control the execution and rendering of that code chunk. Further discussion of the use of code chunk parameters are discussed in Section 3.3.

Typically, the first code chunk will load the packages with the R functions needed for all the code chunks in the document. If a student has never used these packages in their local version of RStudio Desktop, then they will need to install the packages on their machine using install.packages(“tidyverse”), for example, where ‘tidyverse’ is the specific package of functions being installed. The user only needs to run ‘install.packages()’ once for each machine, so this function is only needed when a new package is used. In RStudio Cloud, packages need to be installed for each new project, but instructor-installed packages remain if students clone their own copy of a project. The user does need to load the package libraries in every R Markdown file where the functions will be used. Within the first code chunk, typically one would use the ‘library()’ function to complete this task. For instance, in the Herd Immunity example explained in Section 4, after the YAML header, we had the following code snippet to import the ‘tidyverse’, ‘ggrepel’, and ‘deSolve’ packages:

```
“‘{r setup, include=FALSE}
library(tidyverse)
library(ggrepel)
library(deSolve)
”’
```

Compared to an R script file, R Markdown files naturally isolate blocks of code into code chunks, making it easier to run either the entire file or specific pieces of code. Within the RStudio IDE, one can run all of the code chunks at once by hitting the “Run” button at the top of the file window. However, this can be time consuming for long files and will generate errors if there are code chunks that the students need to edit or complete. To run a specific code chunk, use the run button within each code chunk, shown as a green arrow in the upper right corner. This button will execute the code and print anything that evaluates below the code chunk. Keep in mind that the order one runs the code chunks does matter, so code chunks that load libraries or data need to be run before code chunks that use these functions or objects.

Typically, each code chunk is focused on a specific discrete task, which allows for troubleshooting small sections of related code. Throughout the document, the author can add other R code chunks that can be executed either inline or when the entire document is rendered. As we detail below, these code chunks can be used in a variety of ways such as demonstrating code to students, providing partial code for students to alter, or reserving a dedicated spot for students to create code in response to questions in the text.

2.4. Formatted text

Anything outside of the code chunks becomes part of the narrative text within the rendered document (Fig. 1). Code chunks can be interspersed within the longer form text to allow the instructor to provide background information, links, figures, explanations of code syntax, or questions or scenarios for students to answer, either in narrative text or within a code chunk. The text can also provide an organizing framework for the document with simple tag formatting. For example, one can create bold section headers of various sizes by placing one to six hashtag symbols in front of text, with # giving the largest font size and ###### giving the smallest.

Specific formatting can also alert students to questions inside of the document that they need to answer. Using consistent formatting to ask a question and indicate where students should write their answer can help students navigating the document and educators when grading. In this example, the text between the single asterisks will be italicized, while the text between the double asterisks will be bold.

```
*Question: WRITE QUESTION HERE*

**Answer**
```

More comprehensive guides for key text formatting are available from RStudio's cheat sheet for R Markdown [15] or open access guide-books [6]. Formatting text with colorblind-friendly colors is also a great way to highlight results and this can be done with html or LaTeX tags. There are many options for finding colorblind-friendly options, not only for LaTeX tags, but also for visualizations made in R (see Section 4.1).

2.5. LaTeX

It is also possible to include basic LaTeX tags in R Markdown. With LaTeX, the end result is professional looking equations throughout the document (Fig. 1). When using R Studio, install MiKTeX first to be able to compile LaTeX commands [18]. Also, install 'tinytex' by typing the following command in the R Studio console: tinytex::install_tinytex

For instance, in the Herd Immunity example, when we discuss the S-I-R model, we include the following text:

```
\begin{align}
\dfrac{dS}{dt} &= - \beta S I\\
\dfrac{dI}{dt} &= \beta S I - \gamma I\\
\dfrac{dR}{dt} &= \gamma I
\end{align}
```

This appears in our final rendered document as:

$$\frac{dS}{dt} = -\beta SI$$

$$\frac{dI}{dt} = \beta SI - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$

2.6. Rendering the document

There are two ways to render R Markdown code into a final, formatted document. The simplest is to use RStudio and “knit” the document with the knit icon. All of the code will be executed and the final document will include all formatted text, printed code, and executed results. If the author does not want to use RStudio, they can still create finished documents with the ‘render’ function in R Markdown. For our purposes we had students knit a PDF for submitting their work, but html files can also be created and may be more accessible for screen readers [19]. There are a wide variety of output format types, including slides or dashboards, that can be customized in the YAML header [6,20].

Instructors should also be aware of RStudio accessibility features [21] and check that the technology selected does not pose barriers to participation for students. For example, Godfrey presented recommendations for making R Markdown documents accessible for users with vision impairments [22].

3. Using R Markdown with students

When using R in the classroom, students need some basic familiarity with R syntax and functions, but R Markdown documents allow instructors to customize the amount of independent coding needed for assignments, depending on the experience of the students and the goals of the assignment. We discuss three scenarios for using R Markdown in the classroom: (1) as the interface for teaching students coding, (2) for structured exercises that guide students through analyses, or (3) for students submitting projects or independent analyses with a formatted output. We provide examples of each type from the teaching of our courses in Section 4.

Both courses described in this paper successfully used R Markdown files to structure learning in a variety of settings, including online teaching, homework, and active learning within the classroom. In the *Data Visualization* course, R Markdown files were used in class to teach programming and by students completing homework or projects. The upper-level *Computational Modeling* course was a partially flipped class, with students working on modules in and out of the classroom.

We found several pedagogical benefits to using R Markdown in our teaching. In one file, students can explore code in a guided way, while taking meaningful notes on what they learned about the code. Rather than spending time trying to type code from scratch and getting stuck on syntax errors during live coding demonstrations, students could focus on experimenting with the provided code to learn how the R functions work. The instructor is then available to circulate around the room, checking in on students and addressing questions as they arise, rather than being stuck at the keyboard demonstrating the code.

3.1. Teaching students how to code in R

Most of the students entering our courses had little to no R programming experience. The fundamentals of learning R can be taught using the R Markdown framework from the first class period. The combination of text and code chunks allows an instructor to build an active learning experience that can be as guided or as free form as the instructor wishes. Guided activities can introduce students to all aspects of working in R, including exploration of the RStudio environment, introduction to R syntax, and exploration of help documentation for R functions and their parameters (S2). Initial exercises can provide the majority of the code that students run and edit, minimizing the amount of independent coding as students learn the syntax. As the students become more independent, the R Markdown documents can provide less directive information, allowing the students to explore new packages and make independent decisions to perform data tasks.

While the capabilities of R Markdown may seem advanced for beginning students, there are only a few things the student needs to know to get started using these files. First, the students must learn the difference between the text areas and the code chunks, understanding that they can edit both sections. Instructors must clarify that the formatted text can be used for longer prose. The code chunks can only contain valid R syntax. Second, students must know how to comment code within the code chunk by typing a hashtag '#' before any text. These comments will be ignored when the code chunk is executed. Third, they need to know how to run individual code chunks (see Section 2.2) and to think about the order that one should run code chunks. Finally, when students need to render a document into a report, the students need to learn how to modify the YAML Header (Section 2.1) and knit the document (Section 2.5).

A structured R Markdown document can serve both as an arena for students to practice their coding skills and a notebook to record their own ideas and comments, rather than having to capture everything in separate notes while simultaneously learning syntax or exploring new functions in R. For students new to programming, it is important for students to know when to use the formatted text outside of code chunks versus when they should annotate within code chunks via commenting. Students should use the text sections for long form notes about the material, as more options are available for formatting text outside of code chunks. Long comments within code chunks can run outside of the page margin of knitted PDFs, but installing the 'formatR' package or knitting to html can address this issue. Commenting code is an important best practice in programming that students should practice, as those comments are essential for clear, reproducible code and for making the syntax easily usable for collaborators. Short comments within the code block can help students articulate the purpose of a function in their own words and within the context of the syntax, which will aid their learning [23]. Additionally, when experimenting with code, students can "comment out" lines of unnecessary code by turning it into a comment with a # symbol. This is an important tool for troubleshooting code, but also for tracking variations in code within their notes. By adding customized notes and comments to code, the students can augment files from the instructor to create a useful study tool or reference for future work.

When introducing students to new syntax, we would often scaffold their learning experience in three phases: (1) introduce specific R functions, (2) experiment with provided syntax, and (3) use what they learned to build new code to address a specific question. For the first phase, we would provide code within the code chunks and ask students to run the code chunk and answer questions about the code's output. We asked students to provide long form answers as formatted text and/or comments within the code chunks. By giving students time and incentive to explain the operation of the code in writing, we forced them to process and articulate how the code was functioning. Additionally, by asking them to experiment with the code and summarize, in their own words, the results of their trial and error, we helped students create novel connections, which is known to increase learning gains [24]. In the second phase, we would ask them to experiment with the R function(s) they just learned. For example, if they have been introduced to specific parameters of a function, they might alter the arguments of those parameters to see how the output is altered. Or students might be asked to explore additional parameters of that function or related functions and record the results of their exploration. In the third phase, we would provide a question that uses some combination of the functions they have learned, but with little or no guidance in the code chunk. A single assignment could contain multiple iterations of this cycle to teach different concepts. As the students gained experience, the open-ended problems would get more complex and incorporate content from previous classes. This approach is useful for teaching fundamental programming principles, such as defining objects and exploring functions and their parameters, but is flexible enough to be used in more complex tasks, such as guiding students through the use and syntax of packages, such as 'ggplot2' in the *Data Visualization* course (S3) or 'deSolve' in the *Computational Modeling* course (S4). We were mindful to introduce students to functions that they would need later in the class.

R Markdown files also provide an easy way to share educational materials to other instructors. An instructor can annotate their documents further to provide answer keys or pedagogical information for instructors. If one is new to incorporating R into their teaching, there is no need to create materials from scratch. There are several repositories of R Markdown files that other instructors have made publicly available to use or edit, some that accompany textbooks. In the *Data Visualization* course, we mainly modified exercises associated with Kieran Healy's text [10,25]. Claus Wilke also has course materials that align with his text [12,26], which use the 'learnr' package to convert R Markdown files into active tutorials [27].

Depending on your students' prior programming experience and the emphasis the instructor wants to place on coding skills, there may be some skill building that happens outside of class. There are many online tutorials to help new users learn R independently. In the *Computational Modeling* class, the students were already proficient programmers, but not in R specifically. These students developed the R skills needed for the class by completing four Datacamp tutorials outside of class [28]. At the time of this writing, Datacamp is available for free for use in courses at universities and using this service in teaching has both benefits and drawbacks [29]. There are several other options for short tutorials [e.g. 4,30–32] as an alternative for getting students familiar with R syntax.

3.2. Using R Markdown to build class activities and homework

In both classes, we used R Markdown to structure in-class activities and homework on a variety of topics. Some of these examples were built to provide code that the students could experiment with and alter; other examples required students to write their own code. Well-structured documents can create an environment that can support new students while they experiment with coding, which can be adjusted as students advance in competency.

The R Markdown activities had several benefits for students, including interactive learning. During class time, students worked in collaborative groups and were encouraged to help each other and to discuss their results. We found that the collaborative nature of these activities was strong whether the students were in Zoom breakout rooms or physically together in a classroom. The interactive nature of the document combines the information needed for the student to learn the material with the outcomes of learning together in a clear reproducible document that the students can later reference. When used outside of class, the R Markdown documents provided as much guidance as the instructor found necessary for the learning goals and student experience level. Overall, students praised the modules as successful learning tools. They enjoyed the interactive nature of the work, as well as the collaborative experience. Based on the assessment of their coursework, the activities were successful in helping students meet course learning objectives.

We also perceived several benefits for instructors. First, combining instructions, examples, links to coding help, LaTeX, figures, and executable code blocks into one document makes R Markdown a pedagogically useful tool where a variety of material and documentation can be delivered to students in a single interface. Second, as students could add their notes and thoughts to the documents, the instructors had greater insight into what material was clear or confusing to students. Finally, the standardized formatting of a knitted report made these assignments easier to assess, due to their uniformity.

R Markdown documents supported learning content knowledge separate from learning objectives related to coding in R. In the sample module from the *Computational Modeling* class (S4), the activity focused on mathematical modeling to explore the idea of herd immunity (see section 4.2.2). In the provided example, code that numerically solves the differential equations model with the 'deSolve' package is provided at the top. Alternatively, as an additional learning experience, the instructor could remove the sample code and have the students provide proper code to numerically solve the equations on their own. Students then plot the results, interpret the output, and answer directed questions about their interpretation. Subsequently, students are asked to change various assumptions of the model by setting or changing the parameters or initial conditions, generate particular output by numerically solving the model, and plot the output. They then write text answers to explain what the results from the code mean. Here, they can use LaTeX inside of the file to write properly formatted mathematics that, when knit, will look professional. The module provided links to coding help, summaries about needed parameter or model information, and mathematical epidemiology basics so the students had the necessary background for the activity.

3.3. Teaching students to create novel R Markdown files

For open-ended projects, where each student is working with their own empirical data or exploring a topic of their choosing, having an instructor-guided R Markdown document might not be useful or desirable. In this case, it is worthwhile to teach students how to make their own R Markdown file and render it into a clear, well-formatted report. We created a guided document to teach students how to generate their own R Markdown files (S5). In this assignment, students followed the instructor-provided example in R Markdown, rather than in a static text document, allowing students to could render the instruction file and see another example of how the syntax becomes formatted text.

When simply using files made by someone else, students are likely to overlook several aspects of the document, such as the syntax that generates formatted text. We provided both the knitted report and the corresponding R Markdown file that contained the instructions. The students were prompted to compare the appearance of both documents and then taught how to create their own file, where they could edit the YAML header and experiment with formatting text, as detailed in Section 2. In RStudio, R Markdown files are automatically populated with a few code chunks to demonstrate their use. We used the default code chunks to help students explore some useful functions and parameters specific to R Markdown files, separating the creation of documents from any coding practice (S5). However, one could integrate this exploration within an assignment that builds coding skills as well.

We found it was important to teach students specific functions and parameters particular to the R Markdown code chunk format. Within the {r} tag of each code chunk, users should add a unique name (for example, "setup" in Fig. 2). Unique {r} tag names are particularly important in long documents and can help a user find areas that require troubleshooting. While the code will run without a name in the {r} tag, duplicate names within the same R Markdown file will cause errors. The output of code chunks can be modified with parameters within the header. To create organized reports, it is important to teach students about the parameters 'eval' and 'echo'. 'Echo' determines whether the code will be shown in the final report. If the instructor needs to see the code, then 'echo' should be set to 'TRUE'. By default, R Markdown files are generated to include a function from the 'knitr' package that sets all code chunks to print in the knitted report: knitr::opts_chunk$set(echo=TRUE). The students need to understand the importance of this function so they do not delete it, especially if instructors want to assess the code chunks in the final report. If there are particular code chunks that should not be printed in the final report, students can override the default setting by adding the argument ‘echo=FALSE’ within that chunk header. The ‘eval’ parameter determines whether a chunk of code is evaluated or executed. The default setting is ‘TRUE’ (i.e. that the code will run), but can be overridden by adding the argument ‘eval=FALSE’ to the chunk header. Students might find it useful to modify these parameters to keep their knitted file concise. For example, while working through an assignment, students may need to print dataframes to examine the data while working in RStudio, but do not need it printed in the knitted report. In some cases, the students might experiment with code that does not quite work or that they do not need in their final analysis. However, they might be reluctant to delete code if they want to ask questions about it or experiment with it later. In these cases, the student can maintain the code chunks in their file for their own reference, but prevent the code from printing and executing in the knitted report by specifying ‘echo=FALSE’, ‘eval=FALSE’. Alternatively, one can use the parameter ‘include’; ‘include= FALSE’ will override any default ‘echo’ and ‘eval’ parameters to prevent that code chunk from printing and evaluating (Fig. 2).

We used this same assignment to introduce the syntax of formatting text within R Markdown documents [14]. By giving a few examples and linking to online resources, the students could explore the extent of formatting options available [15,20]. This assignment allowed them to test some of the most common formatting needs, such as creating headers, bold text, italic text, and ordered lists. Students new to coding will need to know that paragraph breaks are triggered by two spaces or a backslash at the end of each line, which will differ from their word processing experience. The instructions guided students through rendering their document (Section 2.5) into a PDF, so that they could compare the formatting syntax to the formatted outcome. Another approach to check student understanding is to give students a formatted text document and assign them to produce an R Markdown file that replicates the same formatting [33]. A new feature in RStudio v1.4 (also the version used in RStudio Cloud as of January 2021) is the visual R Markdown editing mode, which allows the user to edit in a mode displayed as the formatted document instead of the source code. This mode also has additional command features for editing and citations. We shared this feature with students after they had learned to format R Markdown documents from the source code to assist with polishing the final document [34].

Taking the time to teach students how to create and format their own R Markdown files has several advantages. First of all, the knitted file creates a clean report that can be easier to grade for the instructor. Second, the students can integrate all aspects of their work, including commenting code, writing and executing code, and written analysis of data, in one cohesive document. Finally, we teach students that R Markdown can be used as a method to document and annotate code for reproducible workflows. These files can become the equivalent of a lab notebook entry, allowing documentation of trial and error of coding. These documents can be streamlined into clean, accessible code, which is increasingly included in publications to facilitate peer review of data processing and reproducibility by others in the field. It is important for undergraduates to see the importance of tracking their computational methods just as they would track lab methods. By introducing R Markdown files in class, we taught the students how to use a common record keeping tool used by professional computational scientists.

4. Examples of R Markdown assignments in math and biology classrooms

We provide specific examples from our two courses to showcase the breadth of uses for R Markdown files in teaching. In Section 4.1, we highlight two sample assignments from the *Data Visualization* course, illustrating how to use R Markdown to help students learn R. The first activity guided students through their first encounter with coding in R and RStudio, including accessing help features. The second activity was used after students developed some comfort with basic R syntax and introduced students to creating data visualizations with 'ggplot2'. In Section 4.2, we describe a class activity from the *Computational Modeling* course that demonstrates the use of R Markdown in structured delivery of course concepts; in this case, students explored and analyzed mathematical models in order to understand the principle of herd immunity. In Section 4.3, we discuss two ways of using R Markdown to support larger student-driven projects in the *Data Visualization* course. The first project, at mid-semester, required more scaffolding and the capstone project was more independent, with the students creating their own report in R Markdown to demonstrate the coding of their figures as well as communicate conclusions from their data analysis.

4.1. Teaching coding in R Markdown

In the *Data Visualization* course, students worked in R Markdown from their very first R exercise. For students new to R and programming, we took the approach of showing students the power of R and R Markdown together to motivate learning the fundamentals [7,35]. In keeping with this philosophy, our introductory activities focused on orienting students to key tasks in the IDE and getting started running code, rather than the details of R syntax or formatting and functions in R Markdown files.

As an instructor, an important first step is determining how to deliver files for students to easily access in the environment they are using. We used RStudio Cloud, where we were able to pre-load datasets, packages, and R Markdown files and students cloned these projects. If students are using RStudio on local machines, the instructor will need to help students load files and share resources in some other way (either through a Learning Management System, pulled from GitHub or posted on the web, or using a server).

During our first class period using R, we oriented students in RStudio Cloud and emphasized the growth mindset we aimed for students to adopt on their programming journey [36,37]. We gave a brief introduction to the general structure of R Markdown documents and demonstrated how to run chunks of code, annotate the code with comments, and write longer form text (as described in Section 3.1). In their first document, students proceeded through assigning objects, accessing data, and running code to create a figure. We created our first R Markdown assignment file (S2) based on materials provided to accompany the K. Healy text [25].

One of the challenges of teaching with R is its continuous and active development, resulting in a wide range of R packages and functions for performing similar tasks. Finding key and consistent resources where students can look for help, and knowing ways to search for help, is essential for beginning students. Broadly web searching or using StackOverflow can be overwhelming for students new to R syntax. One of our class exercises focused on basic troubleshooting practices, where the instructions guided the student through making use of the help options within RStudio. We taught them practices such as checking for common syntax mistakes, running individual lines or small blocks of code in isolation, and commenting out code temporarily while testing other lines of code (see other tips shared with students in S2). Another of our pedagogical decisions was to teach R using the 'tidyverse' suite of packages and provide a consistent set of syntax and vocabulary for our students [38], but courses can use functions with base or formula syntax as well [39].

When making data visualizations in genuine scientific research, much of one's time is spent on cleaning and organizing data before any graphs are made. However, to build excitement for coding we intentionally started students with clean data sets and taught them to use 'ggplot2' (a package within the 'tidyverse' collection) to make a variety of graph types. One benefit of the 'tidyverse' is that the default appearance of plots generated in 'ggplot2' provides a fairly polished look even before customization [40]. This approach allowed students to produce fast, compelling visuals from their early coding efforts, which we believe helped maintain student engagement. Students then built upon their knowledge of basic 'ggplot2' functions to customize the appearance of plots, all while becoming more familiar with 'tidyverse' structure and vocabulary and R syntax.

We scaffolded our students' introduction to 'ggplot2' by demonstrating functions with complete code, asking students to practice and customize provided prompts, and applying their knowledge to trying new plot types (S3). The theoretical alignment of 'ggplot2' with the grammar of graphics [41,42] provided the opportunity to teach data visualization principles and then have students learn the coding layers to change various graphic features. The basic principles of 'ggplot2' include learning the 'geom' functions to designate plot types and specifying the x and y aesthetics to map the variables on the plot [11,43]. The basic structure for any graph starts with the 'ggplot' function, the selection of the geometric object, and the variables mapped to that object, such as this code for a scatterplot: ggplot(data = dataset1) + geom_point(mapping = aes(x = variable1, y = variable2)). The most important basic principle for beginning students is differentiating the variables included in the mapping argument, aes(), and functions outside this argument to customize appearance. Students later learn to add scaling features and faceting to produce multiple subset plots, as well as adjusting appearance and placement of chart elements.

As an example of adjusting design features, we taught students about color selection and considering the accessibility of their final products. Color can map a new variable onto the data or emphasize trends, but using a color-blind friendly palette is critical for universal accessibility. We recommended that students use either ColorBrewer 2.0 [44] or Chroma.js [45] to find palettes for sequential or diverging data used in graphs. ColorBrewer has an option to select only color-blind friendly palettes, while Chroma.js will indicate whether palettes are color-blind friendly. To find more diverse color palettes, such as a triad or complementary colors, we used Adobe Color Wheel [46], which is a free web source to pick various color schemes with the option to indicate color-blind safe selections. Both Chroma.js and Adobe Color Wheel have built in color-blind simulators, which are good teaching tools for students.

After teaching 'ggplot2' syntax, we progressed to other tidyverse packages ('dplyr' and 'tidyr') to build skills to explore authentic datasets, such as cleaning and organizing data. There are great chapters in both Wilke's and Healy's texts guiding students through these functions [10,25]. We covered fundamental skills in modifying data, such as filtering, adding or deleting columns/rows within dataframes, pivoting or joining dataframes, replacing text, and changing data type (e.g., from string to numbers). We also covered functions for processing the data, including how to group or filter data and perform basic calculations, such as counting and averaging. Students practiced these skills via guided questions using small datasets [25].

4.2. Using R Markdown to build complete activities

This section describes a class module that examines the principle of herd immunity during an active pandemic (S4). It was used in J. Wares' *Computational Modeling* course, where structured modules in R Markdown were used to teach students how to compute numerical solutions for differential equations models, create data-driven plots, and analyze simulation-created data or data from outside sources. R Markdown's flexible formatting allowed the module to weave text explanations and links to resources with R coding exercises and other questions for the students to answer. We used LaTeX throughout to create professionally formatted equations in the final knitted PDF. This module can be broken into smaller pieces, worked on in the classroom, or used as a take-home assessment.

As an online synchronous course (due to pandemic teaching conditions in Fall 2020), students worked together in Zoom breakout rooms on the R Markdown modules, but in a live classroom students could cluster together and work the same way. The instructor floated between groups (either entering breakout rooms or in the classroom) and answered questions and helped with coding. Although students were allowed to collaborate, each student was responsible for turning in their own knitted PDF report. Students were given time in class to work on the modules, but all of the assignments were continued outside of class and completed by each student on their own.

In the R Markdown module, students investigated early claims that "herd immunity" was a goal to be achieved that could reduce the deleterious effects of COVID-19 [47]. The main idea enumerated in supporting arguments was that if enough people became immune to SARS-CoV-2 (the herd), then transmission would cease, and others who had not been infected would be protected. At the time of these claims, there was no vaccine available and the only road to herd immunity was through people getting infected with the virus.

For this module, we first explain the topic and have students read an article describing the problem [48]. Beyond this preparation, we assume students have knowledge of ordinary differential equations (ODEs) models, since a course in ODEs was a prerequisite for this course, and also assume they know what numerical simulations of the solutions represent.

Below we walk through the entire module. This can be broken into three separate exercises with each major point of each exercise being made the focus of a class session or homework assignment.

Exercise 1 - Herd Immunity Percentages—In the first exercise of the module, students vary the amount of people that begin in the recovered/immune state to get a sense of how herd immunity prevents an epidemic from occurring in a population. Since we want to discuss herd immunity in the context of no vaccine, we assume all people in the recovered state have previously been infected and are now permanently immune. We start with a base case, where no people start off recovered/immune, and then change the amount of people recovered/immune at the beginning of the simulation to be first lower than the herd immunity threshold, and then greater than the herd immunity threshold.

CASE 1: The first aim of this module is to teach students how to compute numerical solutions for the S-I-R model using R. A preliminary exercise that covers this topic more in depth can be found in the book chapter *Supporting Teaching and Learning Mathematics During COVID-19* [49]. The students are first instructed to watch a video about the S-I-R model [50]. The basic S-I-R model is

$$\frac{dS}{dt} = -\beta SI$$

$$\frac{dI}{dt} = \beta SI - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$

Here, S(t) is the number of susceptible people at time t, I(t) is the number of infectious people at time t, R(t) is the number of recovered/immune people at time t, β is the transmission rate, and γ is the reciprocal of the amount of time that people are infected with the disease on average.

To introduce the students to the topic, we use text and LaTeX in the R Markdown file. We also present them with some initial code, in the incorporated code snippets sections, so they can learn how to properly enter and compute numerical solutions for the differential equations model. We then use the output from numerically solving this model to determine the predicted proportion of the population that would become infected with SARS-CoV-2 over time under simple assumptions.

First, a base case example of SARS-CoV-2 transmission is developed and numerical solutions are found. The main assumptions of the base case are that only *one person* starts off infected with SARS-CoV-2 in the population and all other people are considered susceptible (with none recovered/immune at t=0). For this case, we are assuming there is no vaccine, as we investigate claims made early in the pandemic when no vaccine existed. We also assume that on average infections last 14 days and that the basic reproduction ratio for SARS-CoV-2 is 2.5, which was in the range of predicted values for the alpha strain. These values are entered into the R Markdown file.

The base case represents transmission dynamics over time if no transmission-suppressing mitigation efforts were made and if no vaccine was implemented. Students are then given code in the file that stores the parameters properly, sets the initial conditions, and then numerically solves the associated S-I-R model. This comes from an earlier exercise in the course and other instructors could easily remove this code and have the students write the code themselves as an exercise [49].

Students are asked to create particular plots of the output using ‘ggplot2’ and are then asked to determine the total number of people that become infected with the given infection over time for the base case. Coding is interspersed between text questions and answers, providing a clear final reporting format. Students must realize that to determine the total number of people that become infected over time, they must determine the endpoint of the recovered function, not the infected function as would be their first guess. Here we see that under basic assumptions, if SARS-CoV-2 ran through the population unabated, about 88% of the population would become infected before the disease died out naturally.

CASE 2: In the second exercise, we investigate what would happen in a partially immune population. In this case, we assume that 50% of the population is immune to SARS-CoV-2 at the beginning of the simulation but still assume only one person is infected at this time. Students must change the initial conditions and rerun the numerical solutions to find new estimates for the final size of the population that became infected. The point of this exercise is to show the students that if some of the people are immune at the beginning of an epidemic, then the total number of people that become infected over the life of the epidemic is reduced, even if we consider those that were originally immune as part of the population that was ever infected. We could also use this type of simulation if there was a vaccine available to view how the results change for various proportions of the population being vaccinated.

CASE 3: In the last exercise, we start with more than the herd immunity threshold of people already recovered/immune and still only one person infected. Students do this without knowing what the herd immunity threshold is at this time. In this case, no epidemic occurs and only a tiny number of people become infected. This example is supposed to show the power of herd immunity. After the students run the files and collect the output in R, they answer questions in the text sections of the R Markdown file.

In review, Case 1 provides us with an estimate of the total percentage of people that would become infected with SARS-CoV-2 over time under basic assumptions with no vaccine or interventions, 88%. Something that might surprise the students at this point is that not all people will become infected. Students then see from Case 2 that if some people start off already immune, when one person enters the population infected, that the total final infected percentage would decrease. If the model starts off with the herd immunity percentage immune, as in Case 3, that will be the final percentage of people infected.

There are two goals to this exercise: (1) If the model starts with some people recovered/ immune, the overall percentage of people that become infected over time would be reduced. (2) If the herd immunity threshold is reached before the infected person enters the population, an epidemic will not occur and the final percentage of people that become recovered/immune will be the same as the herd immunity threshold. A note can be added here about the power of vaccination to protect people that cannot be immunized.

Exercise 2 - Herd Immunity with Definition and Equations—At the beginning of the second exercise, we explain that the equation for determining the herd immunity threshold, θ, comes from determining the minimal proportion of the population $\theta = \frac{R}{N}$ that would cause infections to decrease $\frac{dI}{dt} < 0$, and therefore for no epidemic to occur because infections are decreasing. The easiest way to think about this is to find the proportion of the population that needs to be immune so that the rate of change of the number of infected people is decreasing over time, so that new infections stop occurring. Here again, herd immunity is defined when only one person is infected. So $S \approx N\text{-}R$ and we can equivalently think about the proportion of the population $\frac{S}{N} \approx 1 - \theta$ that must be susceptible for $\frac{dI}{dt} < 0$.

This latter definition, using the proportion of people that must be susceptible in a population for $\frac{dI}{dt} < 0$, is needed in this exercise, because later we look at the case when there are many more than one people infected (during an active pandemic), and at that time the number of recovered/immune people does not equal the total number of people minus the number of susceptible people ($R = N\text{-}S\text{-}I$ and I is not trivial in this case).

So for instance, if for a particular pathogen we find that 60% or more of the population must be immune for an epidemic to not occur if someone enters the population infected, we would say that 40% or less of the population must be susceptible for an epidemic to not occur. When 60% of the population is immune, the likelihood of a contact between the one infected person occurring with a susceptible person and then also resulting in a transmission is very low, so we do not expect the pathogen to spread. If only 50% of the population was immune, the resulting likelihood would be higher and the pathogen would spread for some time. And in this case, since more than one person is infected at a time, it can be the case that the susceptible population meets the herd immunity threshold criterion, but whose recovered/immune population does not (because there is a large infectious population).

CASE 1: Students are asked to find the herd immunity threshold (in terms of S/N) for the base case. Students write their answers in the text of the R Markdown file and can use LaTeX to format them. Here, with the parameters given, students find that 40% or less of the population must be susceptible to the SARS-CoV-2 (the herd immunity threshold) when an infected person enters the population to prevent an epidemic. They then use the simulating and graphing features of R to investigate this relationship further. They find that if 40% of the population is susceptible when a single infected person enters the population, that by the end of the pandemic, only a few more people will have become infected. We would say the final size of the pandemic was 60% here because before there was a vaccine, the only way that people could have entered the recovered/immune state would have been by becoming infected. Note here that we assume the people became immune through infection and that no one is currently infected when the one infected person enters the population.

One thing to note is that if we start with all people susceptible, under the same transmissibility conditions, the final size of the epidemic will be 88% as in the base case, even though the herd immunity threshold for recovered/immune is only 60%. This surprising result is due to the fact that during an active pandemic the herd immunity threshold is reached when more than one person in the population is infected. This is explored more later in the exercise.

CASE 2: Students are then asked what the final size of the pandemic would be if more people, 50% of the population, started off susceptible. Note that more people being susceptible to the disease means more transmission (since transmission is proportional to the product of S and I). In this case, students see that even with only one person initially infected, that the final size of the pandemic would increase by nearly 10% past the herd immunity threshold. Notice that, the herd immunity threshold and the final size of the epidemic again differ. The point to be made here is that even though the herd immunity threshold has not changed, 60% recovered/immune, that if this threshold is not reached when someone enters the population infected that the total infections over time will be greater than the herd immunity threshold. This is called overshoot [51]. This is due to there being many more than one person infected with SARS-CoV-2 at the time that the herd immunity threshold is reached. Since there are many more than one infected persons at this time, the disease continues to spread because $\frac{dI}{dt} > 0$.

Exercise 3 - Final Number of Cumulative Cases—In the final exercise, we further explore what happens in a simple model during an active pandemic. We use the text section to give some brief explanations before having students code and answer questions in the coding snippets and text sections. Our explanations, along with our students' answers, flow together, and in the end, form a complete report, knitted from the completed R Markdown file.

In this section, we want to explore why more people than the herd immunity threshold become infected over time in our reference case. In that case, the herd immunity threshold was 60% of people needed to be recovered/immune for herd immunity, but the simulations showed that 88% of people would become recovered/immune/infected over time. This overshoot is due to there being many more than one person infected at the time that herd immunity threshold is reached in the population. Therefore, $\frac{dI}{dt} > 0$ at this time and many more people become infected before $\frac{dI}{dt} < 0$, at which point, infection numbers begin to decrease.

The goal is for the students to recognize that the herd immunity threshold is not the same thing as the final size of the epidemic. When herd immunity is reached through infection instead of through vaccination, the final size of the pandemic will be much higher than the herd immunity level. We then look at a similar example with a different basic reproduction ratio, R_0, value to drive the point home. For pathogens that are less infectious, or if counter measures are in place, we would expect the R_0 to be lower. If the pathogen is more infectious, the R_0 is higher. In the case of a lower R_0, the total number of people that become infected over the pandemic should decrease and the herd immunity threshold is lower. In this exercise, the R_0 is lowered from 2.5 to 1.5 when we add the assumption that some interventions are imposed to lower transmissibility.

After each exercise, students are asked to reflect on the assignment and write summary statements to generalize what they have learned. In the first exercise, students should conclude that the final size of the epidemic is smaller if the model starts with some people immune, even if the model always starts with one person infected. In the second exercise, they should relate this to the concept of herd immunity. In the third exercise, students should learn that if the herd immunity threshold is reached when there is an active pandemic, then many more people in the population will become infected than the herd immunity threshold. From this exercise, students should also see that almost the entire population would become infected without a vaccine and that arguing for a herd immunity strategy to prevent infections does not make sense without a vaccine.

This exercise can also show the power of a vaccine. If the herd immunity threshold can be met by vaccination before an infected person enters a population, others will not become infected and the overall number of people affected remains small.

In this example, students progress through an instructor-designed R Markdown file, build technical skills, and investigate a course topic. They learn how to code and solve ODEs in R, how to create relevant plots and work with dataframes, and to create clear arguments from the results of their code. Similar types of activities could be developed for a range of other course content or for course-based research experiences.

4.3. Scaffolding student projects in R Markdown

After students build skills over the progression of a semester, many courses include projects where students have more independence in designing the topic or question explored. Setting clear guidelines and expectations, based on the experience level and coding comfort of students, is critical for successfully meeting the learning goals of a more independent project. Challenging students to do independent analyses beyond their experience level can lead to fixed mindsets in students about their ability to learn and use coding. With a fixed mindset, students believe their ability level is predetermined and can become discouraged with complex tasks instead of viewing challenges as opportunities to learn [52]. To build a growth mindset in the *Data Visualization* course, we emphasized the continual learning process with coding. We designed the mid-semester project as highly structured with curated data, followed by a more open-ended capstone project where students sourced public data and designed their own exploration. Key considerations for student independent projects include:

1. What is the degree of independence in topic or scope for student exploration?
2. Will datasets or models be provided as a starting point or do students source or create these on their own?

3. What output do students create and does the instructor provide an R Markdown template or just guidelines? What guidance or criteria for success does the instructor provide to students?

For a mid-semester summative assessment, we assigned a project where students had limited latitude in both choice of data and types of visualization they were asked to produce. At this stage of the course, students had experience generating graphs from cleaned, organized data and had been practicing data visualization principles, but their experience in R was still limited as we used Tableau during the early portion of the semester in our first iteration of the course [53]. Asking students to select their own topic and find associated datasets or models may seem fairly straightforward, but instructors can find themselves troubleshooting a wide range of challenges, including importing various data files, degree of data cleaning, and differing levels of documentation. Guiding the appropriate identification of topics and feasibility is highly important for more independent assignments.

To reduce the challenges for students (and instructors) from using and troubleshooting a wide range of data sources, the mid-semester project tasked students with examining a topic from Our World in Data, an open resource for well-cleaned and accessible data [54]. We pre-selected topics that aligned with the biology focus of our course with adequate scope for additional exploration by students. Each topic page from Our World in Data has a narrative on a global issue (e.g., air pollution, cancer, deforestation, life expectancy) and interactive visualizations using their standard chart templates. We provided an R Markdown file to serve as the template for their report, as they had used the interface, but not yet created their own R Markdown files yet (S6). Students were tasked with recreating and reconceptualizing charts on their topic from Our World in Data (S7). The default chart types ranged from line and scatter plots to stacked area and bar charts to choropleth maps, providing students the opportunity to select the chart type they wanted to customize further in ‘ggplot2’. Two to three students worked together on a topic so they could share background research and ideas, and each was asked to tackle a different aspect of the data. The students benefited from the advice of their group, but the final products and grades were individual. Our World in Data provided the ideal inspiration and well-curated data for students to start exploring immediately and focus on building their coding skills for graphing and comfort in R Markdown towards their final product.

The final project in our course tasked students with selecting a biological topic and making a claim or demonstrating the scope of an issue by creating visualizations from publicly accessible data. We wanted students to source raw data and create their own R Markdown file from scratch to document their data curation, figures, and analysis of their topic. As we were intentional about giving the students wider freedom to explore their interests, we wanted them to have responsibility for the aesthetics of their final report. Leading up to the capstone project, we taught students how to create their own R Markdown files and customize the formatting and appearance of the output document (see Section 3.3).

We designed the final projects as a balance between turning students completely loose to source their own topic and data, and giving them ownership of their analysis. To narrow the focus, we pre-selected a range of topics and key data sources that appeared reliable and easier for students to use, while encouraging the exploration of additional sources (S8). We asked students to compare the data availability, quality, and accessibility between the sources they consulted. By using R Markdown for the final product, students were able to combine aspects of a final paper with analysis and code in a complete project package with a cleanly formatted export format. The class used RStudio Cloud, however, some of these larger data sets required the greater processing power of desktop RStudio.

While the access and availability of public datasets has substantially increased in the last decade, challenges related to access and formatting can result in high levels of individual troubleshooting with students without some instructor curation [55,56]. Students and educators can source data sets using search tools specific for open datasets (e.g., Google Dataset Search or Kaggle), governmental organizations that provide open data (World Health Organization, World Bank, or country-specific agencies such as the National Oceanic and Atmospheric Administration or Centers for Disease Control and Prevention in the United States), and research data repositories (Dryad, Zenodo, figshare, and other discipline-specific or institutional repositories). The challenges we commonly encountered included unfamiliar file formats, dataset and file size, restricted access, availability of metadata, and the amount of data cleaning and processing required for student useability. For example, the Drug Enforcement Administration's database of pain pill prescriptions was published by the Washington Post [57], but the full national database is over a million rows and downloads as a .tsv file; due to its size the data is set up to be accessed through an API (Application Programming Interface), which we had not covered in our course. For beginning students, reducing the barriers in accessing and formatting public data was one of our major considerations in scaffolding the mid-semester and capstone assignments.

5. Discussion and conclusion

The two classes discussed in this paper have very different content and learning objectives, yet both demonstrate how creative uses of R Markdown can provide an interactive arena for learning. In *Computational Modeling*, the students had previous programming experience in other languages and learned R syntax via a few beginning R Markdown modules and further self study. Extensive use of new coding was not a major learning objective and R Markdown files were used to build activities that taught students foundational material about mathematical modeling: including how to build models based on well-defined assumptions, how to numerically solve those models, and how to interpret the results to answer questions of interest. In *Data Visualization*, many students had no coding experience and learning the basics of using R was a central learning objective, along with analyzing and visualizing biological data. R Markdown files were used throughout the course to support student exploration of basic coding principles, using R to wrangle data and make compelling visualizations, and providing structure for large, open-ended projects.

As the examples from these two courses show, R Markdown files lend an enormous amount of flexibility as a dynamic teaching tool. For instructors, one of the key benefits of R Markdown is the scaffolding and explanations that instructors can include with class exercises and assignments, and the associated enhancements to student learning [8]. The ability to include sections of text, images, and equations alongside the code allows students to personalize the documents to become individualized records of their learning. Finally, the output options with R Markdown allows students to display their work in a variety of formats that can be easily updated as needed.

We built many of these materials when we were still new to teaching with R and we benefited immensely from the large amount of open educational resources available. The open-source nature of the R community is an enormous benefit, financially and intellectually, to our students as well. We encourage the sharing of classroom materials to further build teaching resources that use coding and dynamic computation tools for teaching. In addition to the modules included as supplements in this manuscript, both courses have shared additional teaching materials on open access platforms [58–60]. Faculty can obtain suggested solutions and keys to the modules in the supplements by emailing Dr. Grayson (kgrayson@richmond.edu) or Dr. Hilliker (ahillike@richmond.edu) for *Data Visualization* modules and Dr. Wares (jwares@richmond.edu) for the *Computational Modeling* herd immunity module.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

We thank the University of Richmond Mathematics Department, Biology Department, and the UR Faculty Hub for supporting our visions for these courses, the Office of the Provost for subscribing to RStudio Cloud during the first year of pandemic teaching, and the Stokes Fund for Mathematics and Computer Science as well as the School of Arts and Sciences for supporting the open access publishing of this manuscript. As we developed our courses, we were grateful for advice and helpful discussions with Andrew Bell, Taylor Arnold, Fred Hagemeister, Krista Stenger, and Carrie Eaton.

Abbreviations:

IDE	Integrated development environment
OER	Open education resources

References

[1]. Lai J, Lortie CJ, Muenchen RA, Yang J, Ma K, Evaluating the popularity of R in ecology, Ecosphere 10 (2019) e02567, 10.1002/ecs2.2567.

[2]. Tippmann S, Programming tools: Adventures with R, Nature 517 (2015) 109–110, 10.1038/517109a. [PubMed: 25557714]

[3]. Auker LA, Barthelmess EL, Teaching R in the undergraduate ecology classroom: approaches, lessons learned, and recommendations, Ecosphere 11 (2020) e03060, 10.1002/ecs2.3060.

[4]. Custer GF, van Diepen LTA, Seeley J, Student perceptions towards introductory lessons in R, Nat. Sci. Educ 50 (2021) e20073, 10.1002/nse2.20073.

[5]. Baumer B, Cetinkaya-Rundel M, Bray A, Loi L, Horton NJ, Markdown R, Integrating a reproducible analysis tool into introductory statistics, Technol. Innov. Stat. Educ 8 (2014) 10.5070/T581020118.

[6]. Xie Y, Allaire JJ, Grolemund G, Markdown R, The Definitive Guide, CRC Press, 2018, https://bookdown.org/yihui/rmarkdown/.

[7]. Çetinkaya-Rundel M, Ellison V, A fresh look at introductory data science, J. Stat. Data Sci. Educ 29 (2021) S16–S26, 10.1080/10691898.2020.1804497.

[8]. Finch S, Gordon I, Patrick C, Taking the aRghhhh out of teaching statistics with R: Using R Markdown, Teach. Stat 43 (2021) S143–S147, 10.1111/test.12251.

[9]. RStudio products, (n.d.), 2021, https://www.rstudio.com/products/rstudio/. (Accessed 27 December 2021).

[10]. Healy K, Data Visualization: A Practical Introduction, Princeton University Press, Princeton, NJ, 2018.

[11]. Wickham H, Grolemund G, R for Data Science, O'Reilly Media, 2017, https://r4ds.had.co.nz/.

[12]. Wilke C, Fundamentals of Data Visualization: A Primer on Making Informative and Compelling Figures, first ed., O'Reilly Media, Sebastopol, CA, 2019.

[13]. Bergstrom CT, West JD, Calling Bullshit: The Art of Skepticism in a Data-Driven World, first ed., Random House, New York, 2020.

[14]. Introduction to R Markdown, (n.d.), 2021, https://rmarkdown.rstudio.com/lesson-1.html. (Accessed 12 December 2021).

[15]. Rmarkdown : : cheatsheet, 2021, https://github.com/rstudio/cheatsheets/blob/main/rmarkdown-2.0.pdf. (Accessed 27 December 2021).

[16]. YAML tutorial: Everything you need to get started in minutes, Cloud-Bees. (n.d.), 2021, https://cloudbees.com/blog/yaml-tutorial-everything-you-need-get-started. (Accessed 12 December 2021).

[17]. The definitive guide to BibTeX - BibTeX.com, Paperpile. (n.d.), 2021, https://www.bibtex.com/. (Accessed 20 December 2021).

[18]. Schenk C, Getting MiKTeX, get. MiKTeX (n.d.), 2022, https://miktex.org/download. (Accessed 13 May 2022).

[19]. Xie Y, Dervieux C, Riederer E, Markdown R, Cookbook, CRC Press, 2020, https://bookdown.org/yihui/rmarkdown-cookbook/.

[20]. Markdown R, Gallery, (n.d.), 2021, https://rmarkdown.rstudio.com/gallery.html. (Accessed 12 December 2021).

[21]. Ritchie G, Rstudio accessibility features, rstudio support, 2021, https://support.rstudio.com/hc/en-us/articles/360044226673-RStudio-Accessibility-Features. (Accessed 10 May 2022).

[22]. Godfrey AJ, Accessible R Markdown documents, 2018, https://r-resources.massey.ac.nz/rmarkdown/. (Accessed 10 May 2022).

[23]. Hundhausen CD, Agrawal A, Agarwal P, Talking about code: Integrating pedagogical code reviews into early computing courses, ACM Trans. Comput. Educ 13 (2013) 10.1145/2499947.2499951, 14:1–14:28.

[24]. Lopez M, Whalley J, Robbins P, Lister R, Relationships between reading, tracing and writing skills in introductory programming, in: Proc. Fourth Int. Workshop Comput. Educ. Res, Association for Computing Machinery, New York, NY, USA, 2008, pp. 101–112, 10.1145/1404520.1404531.

[25]. Healy K, Utility functions and data sets for data visualization, socviz 1.2, (n.d.), 2021, https://kieranhealy.org/blog/archives/2018/12/12/teaching-and-learning-materials-for-data-visualization/, https://kjhealy.github.io/socviz/. (Accessed 11 December 2021).

[26]. Wilke C, SDS 375 data visualization in R, (n.d.), 2021, https://wilkelab.org/SDS375/, https://github.com/wilkelab/SDS375. (Accessed 11 December 2021).

[27]. Aden-Buie G, Schloerke B, Allaire J, learnr: Interactive Tutorials for R, n.d. https://rstudio.github.io/learnr/, https://github.com/rstudio/learnr.

[28]. R. Learn, Python & data science online, (n.d.), 2021, https://www.datacamp.com/. https://plus.google.com/u/0/+Datacamp. (Accessed 29 December 2021).

[29]. Baumer BS, Bray AP, Çetinkaya-Rundel M, Hardin JS, Teaching introductory statistics with DataCamp, J. Stat. Educ 28 (2020) 89–97, 10.1080/10691898.2020.1730734.

[30]. Fung J, Li A, Theobold A, Data carpentry: R for social scientists, (n.d.), 2021, https://datacarpentry.org/r-socialsci/. (Accessed 27 December 2021).

[31]. Walum H, De Leon D, Teacups, giraffes, & statistics, (n.d.), 2021, https://tinystats.github.io/teacups-giraffes-and-statistics/. (Accessed 27 December 2021).

[32]. Daskalova G, et al., Coding club: a positive peer-learning community, (n.d.), 2021, https://ourcodingclub.github.io/. (Accessed 27 December 2021).

[33]. Stander J, Dalla Valle L, On enthusing students about big data and social media visualization and analysis using R, rStudio, and rMarkdown, J. Stat. Educ 25 (2017) 60–67, 10.1080/10691898.2017.1322474.

[34]. Rstudio team, visual R markdown, 2021, https://rstudio.github.io/visual-markdown-editing/. (Accessed 13 May 2022).

[35]. Wang X, Rush C, Horton NJ, Data visualization on day one: Bringing big ideas into intro stats early and often, Technol. Innov. Stat. Educ 10 (2017) 10.5070/T5101031737.

[36]. Tanner KD, Promoting student metacognition, CBE—Life Sci. Educ 11 (2012) 113–120, 10.1187/cbe.12-03-0033. [PubMed: 22665584]

[37]. Flanagan KM, Einarson J, Gender, math confidence, and grit: relationships with quantitative skills and performance in an undergraduate biology course, CBE—Life Sci. Educ 16 (2017) ar47, 10.1187/cbe.16-08-0253. [PubMed: 28798209]

[38]. Çetinkaya-Rundel M, Hardin J, Baumer BS, McNamara A, Horton NJ, Rundel C, An educator's perspective of the tidyverse, 2021, ArXiv210803510 Stat. (2021). http://arxiv.org/abs/2108.03510. (Accessed 12 December 2021).

[39]. McNamara A, Teaching modeling in introductory statistics: A comparison of formula and tidyverse syntaxes, 2022, 10.48550/ARXIV.2201.12960.

[40]. Myint L, Hadavand A, Jager L, Leek J, Comparison of beginning R students' perceptions of peer-made plots created in two plotting systems: A randomized experiment, J. Stat. Educ 28 (2020) 98–108, 10.1080/10691898.2019.1695554. [PubMed: 33762806]

[41]. Wickham H, A layered grammar of graphics, J. Comput. Graph. Stat 19 (2010) 3–28, 10.1198/jcgs.2009.07098.

[42]. Wilkinson L, The grammar of graphics, in: Gentle JE, Härdle WK, Mori Y (Eds.), Handb. Comput. Stat. Concepts Methods, Springer, Berlin, Heidelberg, 2012, pp. 375–414, 10.1007/978-3-642-21551-3_13.

[43]. Wickham H, Ggplot2: Elegant Graphics for Data Analysis, Springer, 2021, n.d. https://ggplot2-book.org/. (Accessed 27 December 2021).

[44]. Brewer C, Harrower M, Color Brewer 2.0: Color advice for cartography, color advice for (n.d.), 2022, https://colorbrewer2.org/ (Accessed 8 May 2022).

[45]. Aisch G, Chroma.js color Palette Helper, Chromajs Color Palette Help (n.d.), 2022, https://gka.github.io/palettes. (Accessed 8 May 2022).

[46]. Adobe development team, color wheel, a color palette generator, adobe color. (n.d.), 2022, https://color.adobe.com/. (Accessed 8 May 2022).

[47]. The data is in — stop the panic and end the total isolation | The-Hill, (n.d.), 2021, https://thehill.com/opinion/healthcare/494034-the-data-are-in-stop-the-panic-and-end-the-total-isolation. (Accessed 13 December 2021).

[48]. Wares J, Krehbiel S, Herd immunity won't solve America's COVID-19 problem, the conversation. (n.d.), 2021, http://theconversation.com/herd-immunity-wont-solve-americas-covid-19-problem-139724. (Accessed 13 December 2021).

[49]. Stone A, Teymuroglu Z, Torres M, Wares J, Supporting teaching and learning mathematics during COVID-19, in: Contemp. Res. Math. Biol, 2022.

[50]. Bazett Trefor Dr., The MATH of pandemics | intro to the SIR model, 2020, https://www.youtube.com/watch?v=Qrp40ck3WpI. (Accessed 13 December 2021).

[51]. Miller JC, A note on the derivation of epidemic final sizes, Bull. Math. Biol 74 (2012) 2125–2141, 10.1007/s11538-012-9749-6. [PubMed: 22829179]

[52]. Cutts Q, Cutts E, Draper S, O'Donnell P, Saffrey P, Manipulating mindset to positively influence introductory programming performance, in: Proc. 41st ACM Tech. Symp. Comput. Sci. Educ, Association for Computing Machinery, New York, NY, USA, 2010, pp. 431–435, 10.1145/1734263.1734409.

[53]. Nolan D, Perrett J, Teaching and learning data visualization: Ideas and assignments, Am. Stat 70 (2016) 260–269, 10.1080/00031305.2015.1123651.

[54]. Roser M, Our world in data, (n.d.), 2021, https://ourworldindata.org/about. (Accessed 11 December 2021).

[55]. Kjelvik MK, Schultheis EH, Getting messy with authentic data: exploring the potential of using data from scientific research to support student data literacy, CBE—Life Sci. Educ 18 (2019) 10.1187/cbe.18-02-0023, es2. [PubMed: 31074698]

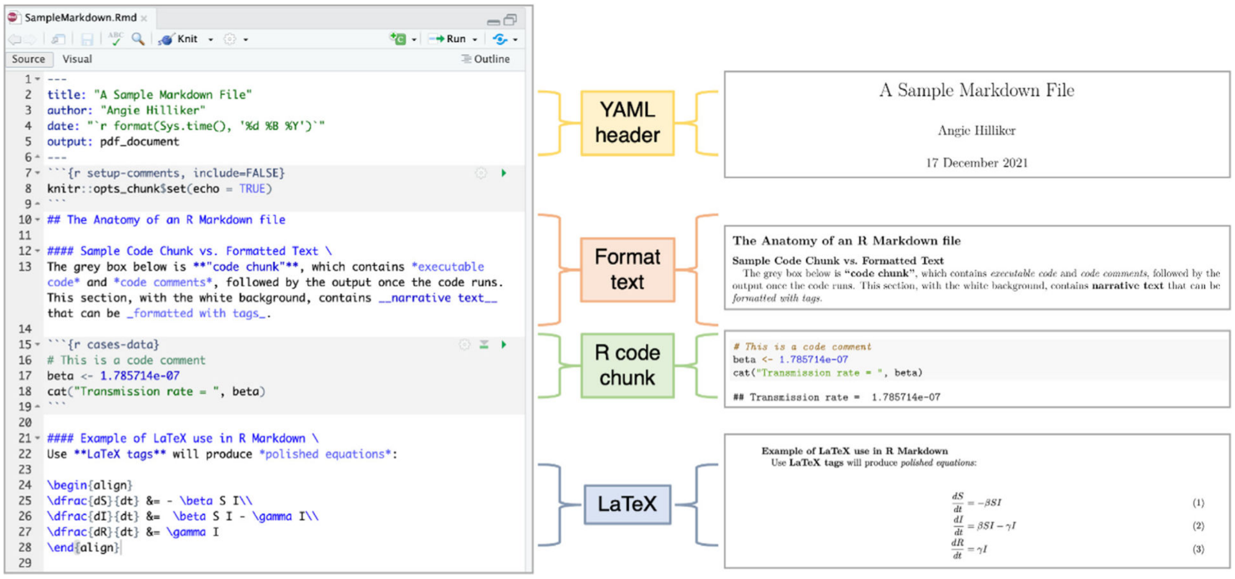
[56]. Kastens K, Krumhansel R, Baker I, Thinking big—Transitioning your students from working with small, student-collected data sets towards big data, Sci. Teach 82 (2015) 23–31.

[57]. Rich S, Sánchez Díez M, Vongkiatkajorn K, How to download and use the DEA pain pills database, wash. Post (n.d.), 2021, https://www.washingtonpost.com/national/2019/07/18/how-download-use-dea-pain-pills-database/. (Accessed 27 December 2021).

[58]. Hilliker A, Grayson K, Teaching data viz and communication as an undergraduate biology course: syllabus and resources, 2021, 10.25334/G5CQ-JK91.

[59]. Grayson K, Hilliker A, Teaching data viz and communication as an undergraduate biology course: assignments and projects, 2021, 10.25334/5C87-YE71.

[60]. Wares J, Torres M, Teymuroglu Z, Stadnyk G, Hawthorne C, Balreira EC, COVID-19 teaching modules, (n.d.), 2021, https://sites.google.com/view/covid-19teachingmodules/covid-19-teaching-modules. (Accessed 27 December 2021).



~~~
---
title: "A Sample Markdown File"
author: "Angie Hilliker"
date: "`r format(Sys.time(), '%d %B %Y')`"
output: pdf_document
---
```{r setup-comments, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
~~~
The Anatomy of an R Markdown file

Sample Code Chunk vs. Formatted Text \
The grey box below is **"code chunk"**, which contains *executable
code* and *code comments*, followed by the output once the code runs.
This section, with the white background, contains __narrative text__
that can be _formatted with tags_.

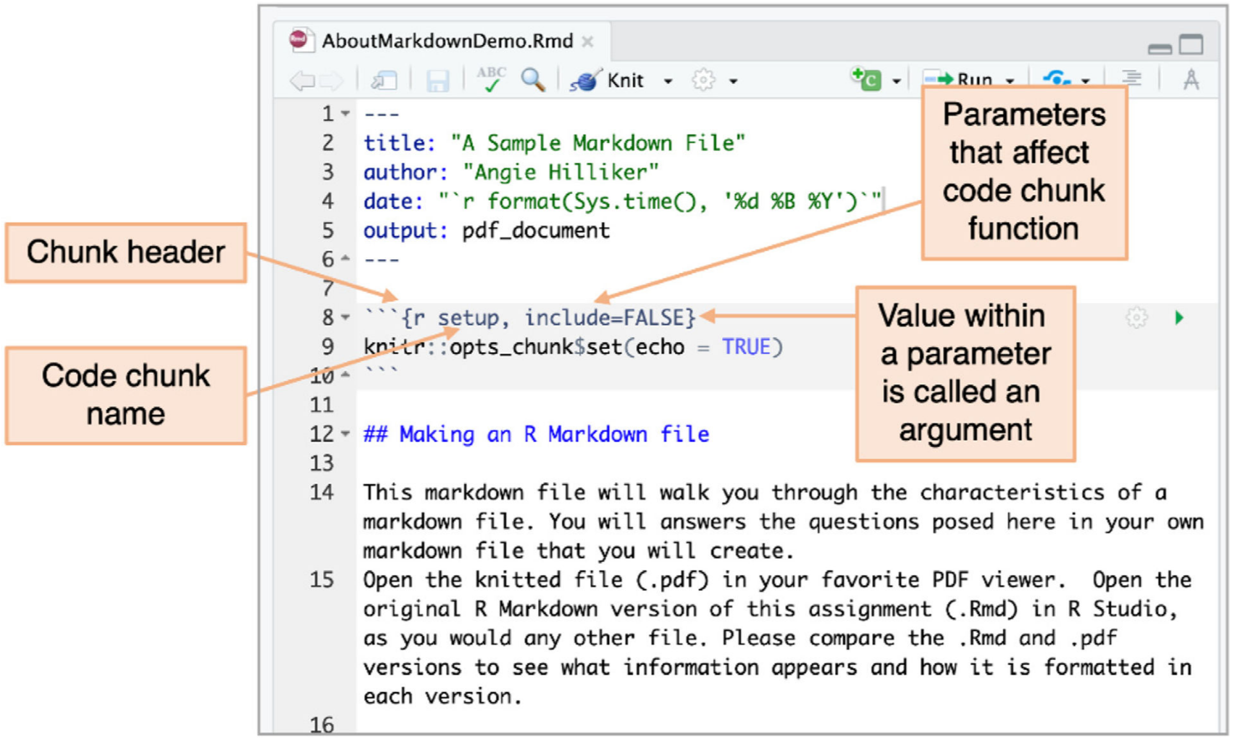
```{r cases-data}
# This is a code comment
beta <- 1.785714e-07
cat("Transmission rate = ", beta)
```

Example of LaTeX use in R Markdown \
Use **LaTeX tags** will produce *polished equations*:

\begin{align}
\dfrac{dS}{dt} &= - \beta S I\\
\dfrac{dI}{dt} &= \beta S I - \gamma I\\
\dfrac{dR}{dt} &= \gamma I
\end{align}

```

A Sample Markdown File

Angie Hilliker

17 December 2021

**The Anatomy of an R Markdown file**

**Sample Code Chunk vs. Formatted Text**

The grey box below is **"code chunk"**, which contains *executable code* and *code comments*, followed by the output once the code runs. This section, with the white background, contains **narrative text** that can be *formatted with tags*.

```
# This is a code comment
beta <- 1.785714e-07
cat("Transmission rate = ", beta)

Transmission rate = 1.785714e-07
```

Example of LaTeX use in R Markdown

Use **LaTeX tags** will produce *polished equations*:

$$\frac{dS}{dt} = -\beta SI \tag{1}$$

$$\frac{dI}{dt} = \beta SI - \gamma I \tag{2}$$

$$\frac{dR}{dt} = \gamma I \tag{3}$$

Fig. 1.

The anatomy of an R Markdown file. R Markdown files contain a YAML header, code chunks, and sections of formatted text. LaTeX tags can be used to create mathematical equations. The R Markdown file (left) and the rendered PDF (right) are shown.
```


AboutMarkdownDemo.Rmd
Knit
Run
Parameters that affect code chunk function
Chunk header
Value within a parameter is called an argument
Code chunk name

~~~
---
title: "A Sample Markdown File"
author: "Angie Hilliker"
date: "`r format(Sys.time(), '%d %B %Y')`"
output: pdf_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

Making an R Markdown file

This markdown file will walk you through the characteristics of a markdown file. You will answers the questions posed here in your own markdown file that you will create.
Open the knitted file (.pdf) in your favorite PDF viewer. Open the original R Markdown version of this assignment (.Rmd) in R Studio, as you would any other file. Please compare the .Rmd and .pdf versions to see what information appears and how it is formatted in each version.

~~~

Fig. 2.

Understanding code chunk headers is important for understanding the rendering of an R Markdown file. Unique code chunk names help organize and identify specific code chunks. The parameters within a header will determine how the code chunk is shown and/or executed in the final report.