Supporting Statistics and Data Science Education with *learnr*

Sara Stoudt
Department of Mathematics, Bucknell University

Anthony D. Scotina
Division of Mathematics, Computing, and Statistics, Simmons University

Karsten Lübke
Institute for Empirical Research and Statistics, FOM University of Applied Sciences

Abstract

A modern statistics or data science course aims to equip students with both conceptual and computing skills. This is a challenging task as instructors do not want to increase students' cognitive load with new tools and technical details and have to balance limited teaching time to help students in achieving the learning outcomes of both content and tool use. Interactive tutorials, built with the *R* package *learnr*, can support student learning with progressive reveal of content, interactive code exercises, and quizzes with automatic feedback, and an interface with the potential to reduce technical burdens via deployment as a web application. We describe different use-cases for *learnr* tutorials including introductory and upper-level statistics and data science courses based on our own teaching experiences. We also discuss the common benefits and lessons learned from implementing and teaching with *learnr* tutorials.

Keywords: cognitive load theory, interactive tutorial, *learnr*, pedagogy, statistical computing

1. Introduction

Since the turn of the twenty-first century, there has been an increasing need for educational and curricula reform in statistics with an added emphasis on computational literacy, programming, and statistical computing at all levels (Nolan & Lang 2010). Indeed, programming and statistical skills are highly sought-after on the job market, and "data literacy" is listed as an essential skill in many graduate programs (Marx 2013; Blickley, Deiner, Garbach, Lacher, Meek, Porensky, Wilkerson, Winford & Schwartz 2013). However the simultaneous instruction of statistical concepts and statistical computing can be challenging for two main reasons: (i) statistics instructors might be left with difficult decisions to make regarding what material to remove from the course in order to devote adequate time to computing, which is especially challenging in service courses which have to cover certain topics (e.g., Introductory Statistics or calculus-based Probability); (ii) students might face an increased cognitive load, particularly those who are learning both statistics and computing for the first time.

On the surface, statistical computing might seem like an addition to a statistics curriculum with very little room for additions. However, the undergraduate statistics curriculum has undergone much change over the past several decades, from the added emphasis on simulation-based inference and the integration of data science tools in introductory courses (Tintle, Chance, Cobb, Roy, Swanson & VanderStoep 2015; Kaplan 2018), to fostering conceptual understanding, active learning, and communication in the upper-level mathematical statistics course (Nolan & Speed 1999; Horton 2013; Green & Blankenship 2015). Much of this reform has focused on the integration of computing into the undergraduate statistics curriculum. For example, Nolan & Lang (2010) advocate for teaching statistical computing in the context of solving scientific problems, which gives students a "deeper, richer appreciation for the practice of statistics." In addition, others advocate for project-based learning to give students exposure to statistical

computing in the context of working through a statistical analysis from beginning to end (Nolan & Lang 2010; Kim & Hardin 2021). However, as with any curriculum design, one must be mindful of the impact that course materials have on students' cognitive load.

Cognitive load theory suggests that students have a limit to their mental workload in working memory and that instructional design should give careful consideration to the roles and limits of working memory (Sweller, Chandler, Tierney & Cooper 1990; Cooper 1990). Three types of cognitive load may simultaneously affect learners. *Intrinsic* load refers to the inherent level of difficulty of the information presented; intrinsic load cannot be reduced through the instructor or the design of the material and is related in part to the prior knowledge of the learner (Chandler & Sweller 1991; Sweller & Chandler 1994). *Extraneous* load refers to the information that is not essential to instruction and is related to the manner in which the instructional material is presented. Due to the limited cognitive resources, materials designed in ways that increase extraneous load will reduce the number of resources available to process intrinsic load. *Germane* load refers to the processing of thought patterns, or *schemas*. Germane load can be modified by instructors through the design of course materials in such a way that reduces extraneous load; for example, by implementing concept mapping and self-reflection exercises.

Examples of pedagogical methods that may be used to reduce learners' extraneous cognitive load include worked examples and the completion of partially-solved problems (Heo & Chow 2005; Guzman, Pennell, Nikelski & Srivastava 2019). In the *worked example effect*, annotated examples are used as part of the instruction, reducing initial extraneous and germane cognitive load during skill acquisition. Generally the worked example effect is optimized when the scaffolding provided for certain types of problems fades during skill acquisition (Renkl 2005). However, worked examples are not necessarily guaranteed to reduce cognitive load, for example if learners are forced to split their attention across different learning materials. Especially relevant to the

discussion of statistical computing, Cooper (1990) provides an example of how ineffective worked examples might appear in a coding setting, where code is used to explain a concept but the discussion of the code is left until the end of the example.

In the *problem completion effect*, students are provided with a partial solution that must be completed. Partial solutions and worked examples reduce extraneous cognitive load by reducing the size of the problem space while still encouraging processing of the material through completion of the remaining elements (Paas & van Merrienboer 1994). Guzman et al. (2019) used cognitive load theory to introduce R programming in undergraduate biostatistics courses by presenting partially completed exercises after worked examples, where template code was given and students had to fill in missing components. Guzman et al. (2019) found that designing course materials, with cognitive load theory in mind, to teach R was associated with increased student motivation and lower levels of frustration and stress when using R.

When integrating statistical computing in any course, ideally the tool being used will not add to students' extraneous load, while also limiting the additional instructional time spent teaching the computing separate from the course's core topics. In a coding context, both worked problems and problem completion approaches risk the *split-attention effect* where students are forced to split their attention between multiple sources of information, like between slides and a statistical computing tool. Instructors can reduce students' extraneous load through the split-attention effect by physically integrating multiple delivery modalities into a single information source (Chandler & Sweller 1992). R Markdown is a tool used to teach statistical computing early in the curriculum that does just this, integrating code, output (e.g., tables and graphs), and written analyses (Baumer, Cetinkaya-Rundel, Bray, Loi & Horton 2014). R Markdown enables students to produce a single reproducible document such that they do not need to divide their attention between multiple sources while learning through statistical computing. However, learning *from* an

already-compiled R Markdown document can be challenging due to the inherently static and text-based nature of the document. The *learnr* package (Schloerke, Allaire, Borges & Aden-Buie 2021) provides a way to build interactive tutorials with R Markdown.

This article follows a similar format to Beckman, Cetinkaya-Rundel, Horton, Rundel, Sullivan & Tackett (2021) by describing the experiences of implementing statistical computing in different statistics courses through the use of *learnr*. We begin by providing a broad introduction to *learnr* as a tool, and then provide descriptions of the courses taught by the three contributing faculty and summaries of the implementation of *learnr* in each course. We conclude by summarizing common benefits and limitations encountered, and offering recommendations to instructors who might consider using *learnr* in their course.

1.1. Introduction to the learnr ecosystem

The *learnr* package, developed by Barret Schloerke, JJ Allaire, Barbara Borges, and Garrick Aden-Buie (Schloerke et al. 2021), enables one to turn R Markdown (Xie, Dervieux & Riederer 2020) documents into interactive tutorials. *learnr* tutorials enjoy all of the benefits of R Markdown, such as the integration of text, equations, figures, code and results, with the added benefits of Shiny-based interactivity (Chang, Cheng, Allaire, Sievert, Schloerke, Xie, Allen, McPherson, Dipert & Borges 2021) including a "progressive reveal" feature, that limits the amount of text that students see at one time, *R* coding exercises (with optional hints, solutions, and automatic feedback), and quiz questions with feedback. RStudio provides example tutorials and guidance about how to create, publish and customize *learnr* tutorials (Schloerke et al. 2021), and Aberson (2021) supplements this by giving more details and a how-to guide for instructors.

In *learnr* tutorials students can edit and run *R* chunks directly in the tutorial with results appearing as if they were working in the console, dampening extraneous cognitive load. *R* chunks

can be pre-populated with worked examples to be run interactively or a code scaffold with blanks denoted so students know where to fill in (problem completion; see Figures 1 and 2). Creators of tutorials can also add a series of hints that complete more and more of this scaffold until the final solution is reached.

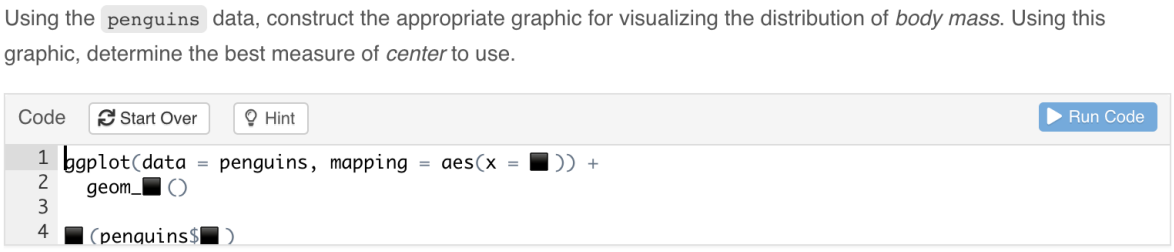

Figure 1: An example of a fill-in-the-blank exercise in *learnr*.

Figure 2: Optional hint feature in *learnr*.

To give students instantaneous feedback, *learnr* can be paired with *gradethis* which automatically compares a student's code chunk to a given solution code chunk (Aden-Buie, Chen, Grolemund & Schloerke 2021) and gives a signal about correctness. Using a problem completion format can help structure a student's solution so that it matches the solution an instructor has in mind, however it is also possible to automatically compare the output of a student's code chunk to a given solution or to check conditions after running a student's code chunk to anticipate different approaches to the problem (Chen 2020). The returned feedback on these code chunks can provide some narration, as anticipated by the creator of the tutorial, or random encouragement using the *praise* package (Csardi & Sorhus 2015).

To collect proof of progress from these tutorials *learnr* can be paired with *learnrhash* which keeps track of exercises engaged with and solutions given to quiz questions and code exercises in the form of a hash, a long string of characters, that can be "decoded" by the instructor (Rundel 2021). *learnrhash* tracks the student's progress behind the scenes as they go through the tutorial. At the end of a tutorial, the student is then prompted to create a hash which can be copied and pasted into an embedded Google Form and submitted to the instructor. The *learnrhash* package then provides a way for instructors to transform that hash back to the context of the tutorial by giving information about important "clicks" that students made. This information can be used as a form of "soft" accountability to give instructors a rough sense of a student's completion of the tutorial or can be translated into a formal score for the assignment by further *learnrhash* capabilities.

learnr and its companion packages help to address many of the previously mentioned challenges in statistics and data science education. As pointed out by Aberson (2021), a *learnr* tutorial enables self-regulated study and low-stakes assessments. The tool allows for a seamless integration of statistical content and statistical computing. *learnr* tutorials are dynamic and can be designed to reduce cognitive load of learners by using worked examples, to be run interactively and progressively revealed, as well as partial solutions, to be completed as exercises. The tutorials can be deployed like Shiny (Chang et al. 2021) applications, so students may use them without the hurdle of an R installation or distributed as an R package (Aberson 2021).

Next we describe *learnr* use in a series of courses that are traditionally part of a statistics or data science curriculum at increasing levels. The first use-case is in an "Introduction to Statistics and Probability" course with a calculus prerequisite. No prior experience with R is expected, but it is not uncommon for students at this institution to come in with some exposure to R through an Introduction to Data Science course or research experience in another field or to Python through an Introduction to Computer Science course. In this class, *learnr* tutorials were distributed via an

R package as pre-lab activities. Students then completed lab activities using R Markdown. The second use-case is in a "Probability Theory" course which has Introductory Statistics (including *R*) and two calculus classes as prerequisites. The *learnr* tutorials were made available as R Markdown documents from the course webpage or GitHub. The third use-case is in a "Statistical Inference Theory" course with a probability theory course as a prerequisite. No prior experience with *R* is expected. Interactive *learnr* tutorials and R Markdown templates were distributed via an *R* package pre-installed by the instructor in an RStudio Cloud project. The last use-case is in a large "Introductory Statistics" course where student backgrounds are heterogeneous and no prior experience with *R* is expected. In this class *learnr* tutorials were hosted on an institutional Shiny server account.

All three faculty use *learnr* for formative self-assessment in these courses. This tool provides an interactive and accessible entry to the course content to support students at different levels. In addition, all three faculty have students build upon the skills learned in an interactive *learnr* format to complete summative assessment activities in R Markdown. Using both of these tools together ensures that students experience many of the attributes required of modern statistical computing tools (e.g., "easy entry" and "support narrative, publishing, and reproducibility") (McNamara 2019).

2. Introduction to Statistics and Probability

The first author taught an Introduction to Statistics and Probability course remotely at Smith College in the fall of 2020 and the spring of 2021. This class of about 45 students met three times a week for an hour and 20 minutes for course content and met an additional hour and 20 minutes for an *R* lab period that split the 45 students across two lab sections. No coding experience was

required for this class, and students started from scratch in lab, downloading *R* and RStudio on their own devices, with an institutional RStudio server as a backup option. The second time the first author taught this course, they converted labs that were modified from those initially provided by OpenIntro (Randomization and Simulation Version) to *learnr* tutorials, informed by pain points in the previous semester (OpenIntro Team 2022).

The introductory text and code examples in each OpenIntro lab were translated to *learnr* tutorials (with some supplementary material added) with the coding-focused exercises interspersed (see Figures 3 and 4 for a supplementary example).

To see how the translation process works, consider one of these labs, available as an R Markdown file. To convert this document into a *learnr* tutorial, first, the YAML header must be adjusted (see Figure 5). The YAML controls elements of how the final output is rendered including final formatting and display options (Xie et al. 2020). To turn a regular code chunk into an interactive code chunk, the header of the code chunk needs to be adapted (see Figure 6). To turn a coding-related question into an auto-graded exercise, an instructor can start with the code from their answer key and "thin" the solution out to create a scaffolded answer. Then the instructor must provide the correct answer and a code chunk to specify that the code should be "graded" (see Figure 7). The questions that are more conceptual or require communicating results can be saved for a lab question (as opposed to a pre-lab question) completed in R Markdown and graded by hand.

These converted tutorials were combined into an *R* package that could be installed from GitHub (De Leon 2020; Stoudt 2021). To avoid the need for students to re-install this package at multiple points throughout the semester, the tutorials had to be completed before the semester started. Students were to go through these *learnr* tutorials individually as a "pre-lab" before coming to each week's lab session. Coding questions within the tutorials could be auto-graded using the

gradethis package without penalties for some amount of trial and error as students first learned, and the *learnrhash* package was used to keep track of student progress and auto-grade the pre-labs based on completion only (Aden-Buie et al. 2021; Rundel 2021).

Figure 3: Interactive exercise about the bootstrap with two hints.

````
## Small Data

The national metrology (measurement science) institute of Argentina made triplicate determinations
of the mass fraction of fructose in honey using
high-performance liquid chromatography, and obtained the following values.

```{r, data, exercise = T}
fructose_measurements <- tibble(measured = c(39.5, 39.3, 40.3))
```

Use R code to get 50 resamplings of three measurements each (with replacement). Name this dataframe
`bootstrap_samples`.

```{r dxqjmkiirhzxgmzj, exercise = TRUE}
bootstrap_samples <- ___ %>%
    ___(size = ___, reps = ___, replace = ___)

head(bootstrap_samples)
```

```{r dxqjmkiirhzxgmzj-hint-1}
bootstrap_samples <- fructose_measurements %>%
    rep_sample_n(size = ___, reps = ___, replace = ___)

head(bootstrap_samples)
```

```{r dxqjmkiirhzxgmzj-solution}
bootstrap_samples <- fructose_measurements %>%
    rep_sample_n(size = 3, reps = 50, replace = TRUE)

head(bootstrap_samples)
```

```{r dxqjmkiirhzxgmzj-check}
# check code
gradethis::grade_code()
```
````

Figure 4: Code chunks for interactive exercise about the bootstrap with two hints.

```
---
title: "The normal distribution"
output:
  html_document:
    css: ../lab.css
    highlight: pygments
    theme: cerulean
    toc: true
    toc_float: true
---
```

```
---
title: "The normal distribution"
output:
  learnr::tutorial:
    progressive: true
    allow_skip: true
runtime: shiny_prerendered
---
```

Figure 5: On the left, the header of an OpenIntro lab. On the right, the header of the *learnr* lab based on the original.

```
```{r male-female}
mcdonalds <- fastfood %>%
  filter(restaurant == "Mcdonalds")
dairy_queen <- fastfood %>%
  filter(restaurant == "Dairy Queen")
```
```

```
```{r male-female, exercise = T}
mcdonalds <- fastfood %>%
  filter(restaurant == "Mcdonalds")
dairy_queen <- fastfood %>%
  filter(restaurant == "Dairy Queen")
```
```

Figure 6: On the left, a code chunk from an OpenIntro lab. On the right, the code chunk adapted to make it interactive.

````
Make plots to visualize the distributions of the amount of calories
from fat of the options from these two restaurants.  How do their centers, shapes,
and spreads compare?

```{r dghqyduwvsgyzshm, exercise = TRUE}
ggplot(mcdonalds, aes(x = ___)) +
  ___
ggplot(dairy_queen, aes(x = ___)) +
  ___
```

```{r dghqyduwvsgyzshm-solution}
ggplot(mcdonalds, aes(x = cal_fat)) +
  geom_histogram()
ggplot(dairy_queen, aes(x = cal_fat)) +
  geom_histogram()
```

```{r dghqyduwvsgyzshm-check}
# check code
gradethis::grade_code()
```
````

Figure 7: The code chunks necessary to auto-grade a coding question.

Where before students might quickly skim through the worked examples presented in the lab instructions without running the code and internalizing the content, students now were incentivized (by the completion check via *learnrhash*) to interactively run each code chunk as they worked their way through the progressively revealed tutorial. The coding exercises that were originally part of the lab questions could now be scaffolded by being translated into a problem completion format, and the *learnr* and *gradethis* partnership allowed students to get immediate feedback (Cetinkaya-Rundel 2021*b*). With the *learnr* tutorials acting as pre-labs, students then came to lab with the same exposure to the new functions and approaches that would be useful for solving the lab questions. This helped alleviate some tension experienced in the previous iteration of the course where students came in with heterogeneous experience of coding and *R*, leading to unbalanced lab pairings.

During the lab session, students came together to work in a team of three to answer questions in an R Markdown document since learning to communicate via this reproducible medium was still a learning goal for this course. The lab questions were more open ended than the coding-focused exercises that students had previously worked on and included interpreting and reasoning about findings in context as well as communication skills more broadly. Students also got a chance to write code from scratch without the problem completion format. These lab reports were then graded based on correctness. Since the pre-lab and lab questions were focused on similar material, students who were stuck on a lab question had a reference with solutions to refer back to for guidance. In the second half of the course, the lab questions were designed to relate to the students' final projects, so the lab reports played a double role as project checkpoints for teams to get feedback from the instructor ahead of a deadline for the final report (completed in R Markdown) and discourage procrastination.

Splitting the labs into a *learnr* pre-lab and R Markdown lab with accompanying lab report that had more targeted questions was a time-saver for both the students and the instructor. Having the more coding-focused questions auto-graded gave the instructor more time to devote to feedback on writing and interpretation. Moving some content outside of the lab session alleviated the time crunch that students had felt in the previous semester. Since students could work through the tutorials at their own pace ahead of lab while getting feedback that did not require waiting for instructor availability, they could come together more efficiently to answer the smaller subset of questions that were assigned as part of the lab report.

Some challenges remained. The first lab where *R*, RStudio, the package that contained all of these tutorials, and all of its dependencies were installed took time and patience. Due to the development-stage nature of some of the *learnr*, *gradethis*, *learnrhash*, and the tutorial delivery package functionality, errors did occur and required troubleshooting. However, the first author

had tested the install process with another faculty member on different operating systems and was able to make a fairly comprehensive list of what could happen and how to fix it ahead of time. Occasionally a student would report a bug in a tutorial, that would be filed as a GitHub issue as a reminder to fix at the end of the semester. Some students who had seen *R* before were confused by the initial setup. They wanted to know how running chunks in the *learnr* tutorial interfaced with the RStudio console. Once it was explained that code did not need to be copied and pasted back and forth between the tutorial and the console and that students could interact with the tutorial as if it were the console itself, the confusion abated.

These *learnr* labs were adapted and used in the same setting by two other instructors, this time teaching in person, but there is still room for improvement in these tutorials. The first time around, the hint structure was not used to its full advantage due to time constraints in teaching preparation. It is possible to scaffold a series of hints before providing the solution, and if the first author were to use these labs again, they would spend some more time fleshing those out. In addition, the OpenIntro community has launched its own set of *learnr* tutorials which span the whole introductory statistics course (Cetinkaya-Rundel, Hardin, Baumer, Bray, Saibene, D'Andrea, Villafane & Theobold 2021).

3. Probability Theory

The second author taught a calculus-based Probability course in-person at Simmons University in the fall of 2021. The course enrolls between 10 and 20 students in a typical semester, with 20 students enrolled during the Fall 2021 semester, and meets three times a week for 50 minutes. This course is designed for upper-level undergraduate students in Mathematics, Economics, Statistics, and Data Science programs, and has prerequisites of Introductory Statistics (with *R*),

and Calculus I and II.

Because *R* programming is introduced in the Introductory Statistics course at Simmons, all students in this class have at least some exposure to *R*. However many of the students enrolled in this course, particularly Statistics and Data Science majors, have several semesters' worth of experience in *R*, including data visualization and wrangling, building linear and non-linear models, and simulation-based inference. However, none of the students have been introduced to *R* topics in *learnr* prior to taking this class.

Major topics in *Probability Theory* include counting techniques, conditional probability and Bayes' Rule, discrete and continuous random variables, expected value and variance, moment-generating functions, multivariate probability distributions, functions of random variables, and sampling distributions. While two semesters of calculus are required, the use of calculus is not an integral part of the course. Rather, additional emphasis is placed on understanding results via Monte Carlo simulations designed in *R*. Students are taught how to derive both analytical solutions and empirical, simulation-based solutions to problems.

Even though students have at least some exposure to *R* prior to the beginning of this class, due to the prerequisite of Introductory Statistics, this class includes one "live coding" lecture in the first week that reviews requisite *R* material within the RStudio IDE to ease the learning curve for the tools that follow. Probability topics are introduced using a combination of "chalk talk" and in-class examples before illustration of examples in *R*.

Approximately every two weeks, one class is devoted entirely to illustrating core course concepts in *R*. During these classes, students work through *learnr* tutorials assigned to them in advance, while the instructor circulates to provide assistance when needed. Students are encouraged to work in pairs, and their work during these labs was not graded. These *learnr* labs covered the following topics:

1. **Designing Simulations in *R***: This tutorial focuses on using the base *R* *sample()* and *replicate()* functions to design simulations to estimate probabilities. Examples are introductory in nature and focus primarily on scenarios involving coin flipping, drawing cards, and rolling dice.

2. **Famous Probability Problems**: This tutorial extends the concepts covered in the previous tutorial, focusing on simulating probabilities from well-known problems, including the Monty Hall Problem, the Birthday Problem, and the Hatcheck Problem (a variation of De Montmort's Matching Problem) (Gill 2011; Borja & Haigh 2007; Scoville 1966).

3. **Discrete Random Variables**: This tutorial walks students through the built-in *d*, *p*, *q*, and *r* functions associated with named *discrete* probability distributions. Additional topics include visualizing probability distributions simulated with the *r* function, and answering probability questions analytically with the built-in probability density and distribution functions in *R*.

4. **Iteration and the Law of Large Numbers**: This tutorial illustrates the Law of Large Numbers with explicit (*for*) loops. We revisit problems computed "by-hand" at the beginning of class, including the Newton-Pepys Problem, by showing approximately how many iterations are needed until the estimated probability converges to the analytical probability (Stigler 2006). Figures 8-10 show what this *learnr* example looks like along with the exercise's scaffolding and solution.

5. **Simulating Continuous Random Variables**: This tutorial extends topics covered in the Discrete Random Variables tutorial to the continuous case. Additional time is spent teaching students how to write functions in *R*, in order to sample from non-named continuous probability distributions.

6. **Joint Probability Distributions**: This tutorial continues on the topic of writing functions in

R, this time with functions of two variables to illustrate simulating bivariate distributions. Additional topics include visualizing bivariate distributions with contour plots and estimating marginal and conditional probabilities using simulated draws from a bivariate distribution.

7. **Conditional Distributions and Naive Bayes**: This tutorial focuses on Naive Bayes classifiers as an application of multivariate and conditional distributions. Examples of Naive Bayes with multiple predictors are used to illustrate the concept of conditional independence between random variables.

8. **The Central Limit Theorem**: This tutorial focuses on simulating sampling distributions, primarily sampling distributions of the sample mean, using a sample of random variables from a given probability distribution.

The `replicate()` Workflow

In MATH/STAT 338, we'll encounter many probability examples that we can derive *by hand* and/or *estimate through simulation*. Though there will be many cases where the simulation-based approach will provide a simpler and more intuitive way to calculating probabilities.

For the especially complicated simulation-based exercises that you'll encounter, it is a good idea to follow a **workflow**, or a set of steps that you can repeat for similar exercises. For example, I recommend you follow this workflow from Speegle and Clair (2021) when using `replicate()` to estimate probabilities:

1. Write code that performs the experiment a single time. For example, if we are estimating the **probability of obtaining at least 12 heads in 20 coin tosses**, a single experiment would be *one set of 20 coin tosses*:

R Code | Start Over | Run Code

```
1 coin_tosses = sample(c("H", "T"), size = 20, replace = TRUE)
2 sum(coin_tosses == "H") >= 12
3
```

2. Replicate the experiment a small number of times and **check** the results:

```
replicate(100, {
  EXPERIMENT GOES HERE
  ...
})
```

3. Replicate the experiment a large number of times and **store** the results:

```
event = replicate(10000, {
  EXPERIMENT GOES HERE
  ...
})
```

4. Estimate the probability with `mean(event)`. Note that `event` will *always* be a **logical vector**, so you'd be taking the mean of a bunch of 1s and 0s.

Figure 8: Scaffolded instructions for a *learnr* exercise.

Exercise 5

In class we worked through the *Newton-Pepys Problem*. Try to simulate each of the three events using `replicate()`. The problem is stated below.

Isaac Newton was consulted about the following problem by Samuel Pepys, who wanted the information for gambling purposes. Which of the following events has the highest probability?

- *A*: At least one 6 appears when 6 fair dice are rolled.
- *B*: At least two 6's appear when 12 fair dice are rolled.
- *C*: At least three 6's appear when 18 fair dice are rolled.

R Code | Start Over | Solution | Run Code

```
1
2
3
```

Continue

Figure 9: The exercise formatting.

Figure 10: The solution for the exercise.

Students accessed the *learnr* tutorials via R Markdown files posted to the course webpage. (Note that tutorials can now be accessed from GitHub (Scotina 2022).) They then opened these files in the RStudio IDE on their own computers and ran them to view the *learnr* tutorial in their web browser. While students were not graded on completion of the tutorial, several exercises in graded, weekly problem sets required students to utilize topics covered in the most-recent tutorial. Alternatively, students had the option to access the *R* tutorials via "static" HTML files knitted from R Markdown. Students could therefore code along within the R Markdown document as they read through the HTML file, or download and work through the corresponding source file.

While this was the first iteration of *Probability* at Simmons that utilized *R* in any format, initial informal student feedback was positive. Students appreciated the guided and incremental nature of the tutorials as an alternative to viewing the complete tutorial all at once in the static HTML format. Because the *learnr* tutorial presented topics in small batches, and like the first author's experience in an introductory statistics context, students were also more likely to read the text

that accompanied coding examples in the *learnr* tutorial. As an added bonus, because students had access to the *learnr* tutorial source file, students had additional resources at their disposal should they choose to make *learnr* tutorials of their own. Several students took advantage of this for their probability "mini-project", which asked them to write a blog post on a named discrete probability distribution of their choice. The instructions for that activity follow.

> In your first Probability Mini-Project, you will write a blogpost on a named (discrete) probability distribution of your choice. Your goal is to research this probability distribution in-depth, and showcase and effectively communicate your understanding of the probability distribution. If it helps, you could frame this as a tutorial blog post!
>
> While I'm leaving some room in this project for creativity, the blogpost should include the following, in no particular order:
>
> - **Motivating example/Real-world application**: In what real-world scenarios might this probability distribution arise?

- **Properties**: What phenomena does your distribution typically model? What is its probability function? Expected value and variance? Any connections to other named distributions that we've seen so far?
- **Visualization**: What does your distribution look like? How does the distribution's shape change based on different values of its parameters?
- **Worked-through simulation example**: Include an example that utilizes R to simulate an example involving your distribution.

One student who did their project on the negative binomial distribution first walked the reader through the technical details, such as the probability mass function, and then used the interactive code chunk capability to show the reader how they can use *dnbinom()* and *rnbinom()* to calculate negative binomial probabilities and simulate distributions with *R*. The tutorial ended with a knowledge check, where the reader was asked to complete a series of multiple-choice exercises on negative binomial probabilities and expected values.

Several students opted to engage with the tutorials via the static HTML format, where they would code along in their own R Markdown documents. These students consisted primarily of Data Science majors who had taken several prior classes that used *R* beyond Introductory Statistics, and seemed to prefer the familiarity of the RStudio IDE over the browser-based coding environment provided by *learnr*. This provides a potential example of the *expertise reversal effect*, which posits that the effectiveness of certain instructional techniques depends on levels of learner expertise (Kalyuga, Ayres, Chandler & Sweller 2003).

The two-semester calculus-based probability and statistical theory sequence has been a cornerstone of Mathematics and Statistics programs for many years (Green & Blankenship 2015). As such, the implementation of *R*, simulation, and real data into the "rethinking" of the probability and statistical theory sequence is a novel approach and an area for future research and

course development. After one semester of teaching the course using this approach, several adjustments are planned, including a more-intentional integration of *R* into the fabric of the course by using it more often during lecture, in addition to live-coding demos.

4. Statistical Inference Theory

The first author taught Statistical Inference Theory in person at Bucknell University in the fall of 2021 and again in the spring of 2022. This class of a little under 10 students met three times a week for 50 minutes for course content and met occasionally during an evening session of one hour and 50 minutes for supplementary coding instruction, project work time, and exams. No coding experience was required for this class, and students used RStudio Cloud where an instructor can pre-install packages and upload templates and data ahead of time for students (Cetinkaya-Rundel 2021*c*). This structure has the added benefit of allowing the instructor to continue to update the package containing *learnr* tutorials throughout the semester instead of having to front-load the preparation of materials.

Despite this being a theoretical course, the first author wanted to make sure the class had an applied portion given their own background and expertise. For example, students could check a gnarly calculation empirically to see if their derivation went astray somewhere (see Figures 11 and 12 for an example), test how confidence intervals and hypothesis tests work, and show what happens when assumptions are broken via a simulation study. There was not enough class time to completely teach *R* on top of the course material, but the goal was to get students comfortable with a subset of functionality most relevant for statistical inference.

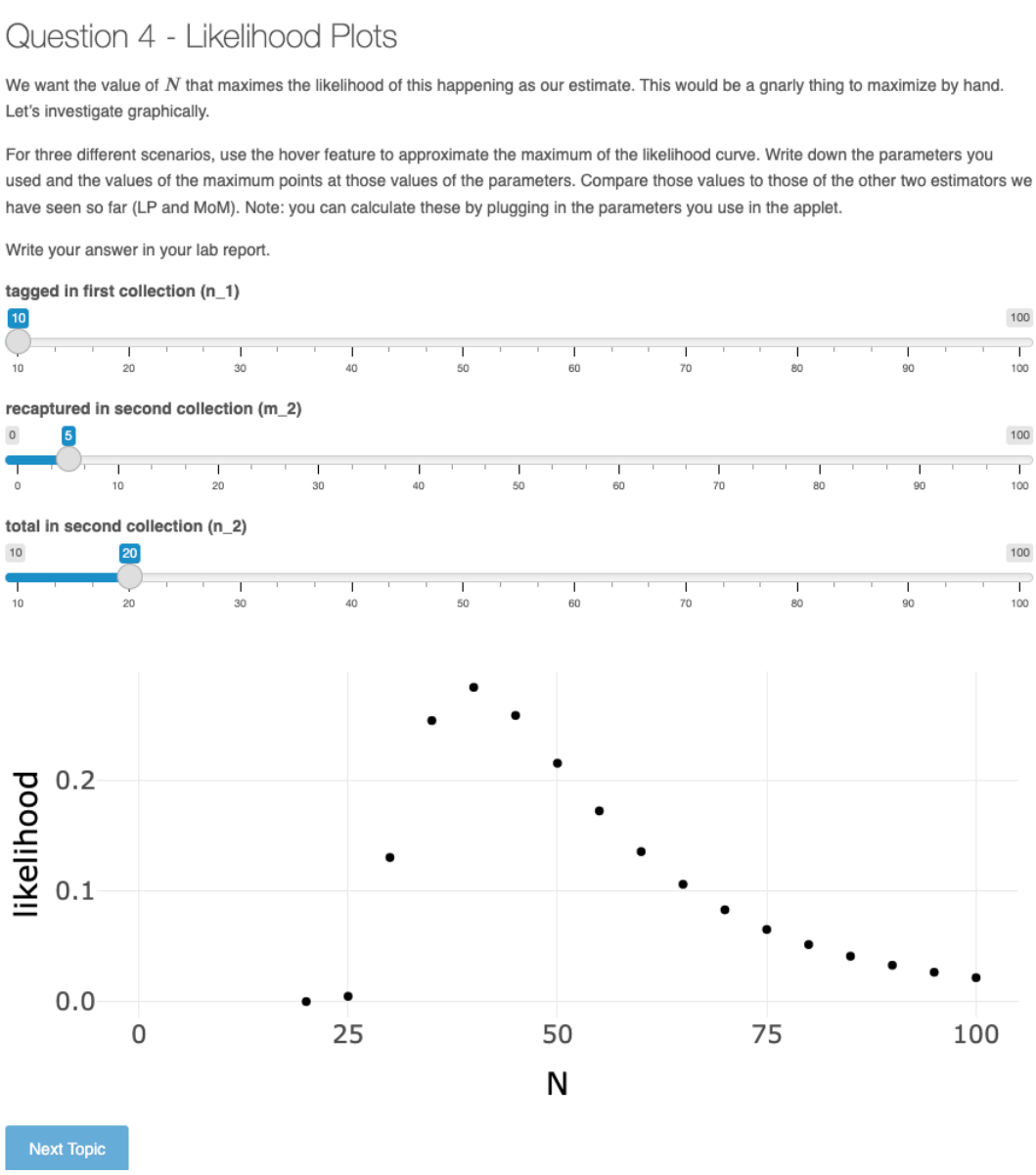

Figure 11: Shiny app within a *learnr* tutorial.

````
```{r, echo=FALSE}
sliderInput("n_1", "tagged in first collection (n_1)", min = 10, max = 100, value = 10,step = 5)
sliderInput("m_2", "recaptured in second collection (m_2)", min = 0, max = 100, value = 5,step = 5)
sliderInput("n_2", "total in second collection (n_2)", min = 10, max = 100, value = 20,step = 5)

plotlyOutput("likPlot")
```

```{r, context="server"}
output$likPlot <- renderPlotly({
  N <- seq(0,100, by = 5)
  a <- choose(input$n_1, input$m_2)
  b <- choose(N-input$n_1, input$n_2-input$m_2)
  c <- choose(N, input$n_2)
  toPlot <- cbind.data.frame(N = N, lik = (a*b)/c)
 g <- ggplot(toPlot, aes(N, lik))+geom_point()+theme_minimal(base_size = 20)+xlab("N")+ylab("likelihood")
 ggplotly(g)
})
```
````

Figure 12: Associated code chunks for Shiny app.

Tutorials created using *learnr* were tried in a variety of formats for the first iteration of the class including supplementary tutorials, lab and project overview tutorials, and short homework companions to check theoretical calculations. These approaches did not work as smoothly as when this author used *learnr* in their introductory statistics class. However, the challenges were not due to the performance of the tool but more about the level of familiarity with R that the students came in with and the need for extra scaffolding and explanation of coding fundamentals. For example, some students in the Introductory Statistics class already had some experience with R from an Introductory Data Science class and could take a leadership role in initial lab settings. The Introductory Statistics class also met for longer class periods and had a dedicated lab period to attend to computational learning goals while the Statistical Inference Theory course faced a tighter schedule with no regular lab time, making it challenging to balance content with computational training. In addition, a typical student's first exposure to R comes later in the curricular sequence while Statistical Inference Theory comes earlier in the curricular sequence as it is a prerequisite for many upper-level courses at this institution. However, this experience did inform a plan for the second iteration of the course that aimed to better utilize *learnr*. What follows are some lessons learned and changes made for the second iteration.

The labs and project for this course were not structured in the same way as they were for an introductory statistics class where students typically learned a set of tools and then applied them to a dataset of interest. Instead, these labs and project checkpoints required students to build upon and revise code they had already written as they worked their way through project checkpoints towards a final product, so there needed to be a static document that they could add to as the semester went on. Woodard & Lee (2021) provide labels for this distinction: the *learnr* tutorials used in an introductory statistics course were more focused on "automation of computational procedures" while the *learnr* tutorials used in a statistical inference theory course were more focused on "computational thinking". In light of this framework, with the statistical computing actions focused on in each course being fundamentally different, it seems less surprising that what worked fairly well in one setting did not translate perfectly to the other.

For example, in the fall, the first author noticed students struggling with going back and forth between a *learnr* tutorial and an R Markdown file, an example of the split-attention effect. The students could also have benefitted from better on-ramps to these coding activities that were not directly tied to the statistical content such as some more explicit introduction to coding constructs such as loops for simulation, basic data structures like vectors and matrices for storing results, how to manipulate data frames, and the basics of R Markdown usage.

In the spring, the first author switched to an independent tutorial system for coding skill building outside of the classroom, graded for completion. R Markdown templates were also used to scaffold inference related exercises and activities for project checkpoints (used primarily for instructor feedback and to mitigate procrastination), and graded in-class labs (Xie, Allaire & Grolemund 2021). These materials were pre-installed on the RStudio Cloud and are also hosted on GitHub (Stoudt 2022).

learnr tutorials cover the following topics:

1. **Guess the Color Activity**: This tutorial steps through simulations that accompany the "Guess the Color" activity described in Green and Blankenship (Green & Blankenship 2015). This is used in class on the first day to get students thinking about creating and evaluating estimators.

2. **Introduction to Interactive Tutorials**: This tutorial explains the structure of a *learnr* tutorial and how to submit progress using a hash and the embedded Google Form.

3. **Estimation in Capture-Recapture**: This tutorial contains Shiny applets to explore likelihood profiles in a capture-recapture context.

4. **Data Structures and Subsetting**: This tutorial introduces vectors, matrices, and logicals including how to subset data of various forms.

5. **Loops**: This tutorial introduces the concept of loops and creating blank objects of appropriate type and size to fill in with results.

6. **Plots and Probability Distributions**: This tutorial introduces students to *ggplot2* and drawing random values from built-in distributions (Wickham 2016).

7. **Plot Expectation Sketching**: This tutorial primes students to interpret plots about power from their own simulation study (related to the the final project for this class).

R Markdown templates cover the following topics:

1. **Reproducibility and Documentation**: This vignette (as students have not yet been introduced to the concept of a template) shows students RMarkdown's capabilities, the importance of setting the seed, and how to use R's documentation to learn about functions

that are new to them. This is paired with the "Introduction to Interactive Tutorials" tutorial.

2. **Estimation in Capture-Recapture**: This template is a companion to the capture-recapture *learnr* tutorial and guides students through the exploration of Shiny applets to learn about the performance of various estimators.

3. **Optimization for Bivariate MLE**: This template provides a starting point for students to learn optimization in R, motivated by a bivariate MLE problem that would be pretty gnarly to do by hand.

4. **Sampling Distributions**: This template walks students through building sampling distributions of various statistics. This template builds on the "Data Structures and Subsetting" and "Loops" tutorials.

5. **Introduction to Nonparametrics**: This template introduces students to nonparametric inference and provides a code structure for the bootstrap and permutation tests to be built upon and customized by students in later assignments. This is designed as a code-along for a live-coding session.

6. **Bootstrap**: This template walks students through crafting various bootstrap intervals. This template builds on the "Loops" tutorials.

7. **Confidence Intervals**: This template introduces students to parametric confidence intervals and the functions that calculate these in R.

8. **Hypothesis Testing**: This template introduces students to parametric hypothesis testing and the functions that compute these in R.

9. **Introduction to Final Project**: This template introduces students to the final project. Each student gets a parametric two-sample test and a related nonparametric two-sample test and

is tasked with designing, performing, and interpreting a simulation study to assess the strengths and weaknesses of the two tests as measured by statistical power.

10. **Functions and Simulating Data**: This template introduces students to creating their own functions to simulate data. This template builds on the "Plots and Probability Distributions" tutorial which primes students to think about which distributions might be particularly interesting to investigate in the context of their assigned hypothesis tests.

11. **Functions and Simulating Power**: This template introduces students to translating a calculation to be done once into a reusable function and operationalizes an empirical estimate of statistical power via simulation.

12. **Simulation Scenarios and Preliminary Plots**: This template helps students structure a set of simulation scenarios to use their previously defined simulating data and power functions on. This template also introduces students to more complex *ggplot2* functionalities.

13. **Tidy Data and Better Plots**: Because students are comparing results of two different tests, it can be helpful to reshape their data to better facilitate comparisons in *ggplot2*. This tutorial guides them through that process.

The goal was to start tutorials with worked examples in the form of interactive code chunks that students could investigate and annotate with comments as they discovered what the code is doing. Then the tutorials transitioned to problem completion, showing students the structure of a coding solution in a *learnr* exercise chunk, or even as part of an R Markdown template, while leaving explicit blank spaces for pieces that students need to fill in, as a way to ease students into coding when buy-in for learning *R* on top of the theoretical material is weak. As students got more comfortable with the basic building blocks of code related to statistical inference, the templates got more and more sparse, in the spirit of faded worked examples (Renkl, Atkinson & Maier

2000). The hope was that this would prevent students from copying and pasting code chunks from a prior tutorial without knowing what each component did and help them recognize which pieces may need to be adapted in the new context. *learnrhash* capabilities as accountability checks were also used to nudge students towards prioritizing this form of skill building as they juggled more traditional problem sets and readings.

Overall, slowly removing the scaffolding of problems assessing coding concepts that were repeated in almost every lab like loops, sampling, and plotting worked well. Students were able to recognize situations where these computational tools would be useful and knew the basic structure of them. Other concepts, like writing and calling custom functions, although used a few times throughout the semester, did not seem to click in the same way. Part of this might be the timing of the *learnr* training and the follow-up assignment where the skills were used. Because of the irregularity of the computational assignments amongst more classical problem sets, there were often gaps between training and application. Contrast this with the "pre-lab before lab every week" structure of Introductory Statistics where students immediately get a sense whether their conceptual understanding is enough to complete lab questions successfully. Another aspect may be that the computational learning objectives need to be further streamlined to the even more essential. We might just be trying to do too much under too many time constraints. Going more in depth into the foundational topics of sampling distributions, confidence intervals, and hypothesis testing rather than aiming for a breadth of labs may be the way to go. Alternatively, a strategic "flip" of this class may free up more time for explicit coding instruction (Horton 2013).

5. Introductory Statistics at Scale

Most of the over 30 different study programs offered at the FOM University of Applied Sciences

in Germany (57,000 students), where the third author teaches, include (applied) data-centric statistic courses (Gehrke, Kistler, Lübke, Markgraf, Krol & Sauer 2021). The different courses are offered by six departments in 35 study centers throughout Germany and in Vienna, Austria. In this heterogeneous setting the class sizes vary between 15 and 150 students. There is also no consistent class schedule across the courses. Some courses meet for 180 minutes almost every week in a semester while some courses with less workload meet less regularly.

Although there are course-specific differences in the syllabi, most statistics courses cover the basics of quantitative data analysis, an introduction to *R* (including elementary data preprocessing with *dplyr*) (Wickham, Francois, Henry & Muller 2018), exploratory data analysis, linear regression, and inference.

Due to the different study programs the scientific background and interest of students is very heterogeneous. Before attending the course most (Bachelor) students are not familiar with *R*. Since 2019, *learnr* is used in a variety of settings in these courses taught by the third author. Fourteen short formative self-assessments are offered as short, "recap" *learnr* tutorials. The focus is not on learning new concepts or coding but rather on 1-3 multiple choice quiz questions, most often integrated in a short case study in *R*, which aim to help students' understanding of the topics covered in class together with (informal) self-assessment by the provided feedback on their answers. Six different short tutorials cover topics such as exploratory data analysis, linear regression, or simulation based inference. They usually include 3-5 multiple-choice quiz questions as well as 3-5 coding exercises, mainly as problem completion, again integrated in small case studies. It is estimated that it takes 10-20 minutes to complete one of these short tutorials. They are designed so that they can be used synchronously or asynchronously. The long tutorials also cover the main topics and concepts of the course, but compared to the short tutorials they are longer (30-45 minutes each), including more quiz questions and coding exercises. While most

tutorials are in German (the native language of students), an English example is available (Cummiskey, Adams, Pleuss, Turner, Clark & Watts 2020; Lübke, Gehrke, Horst & Szepannek 2020, 2022).

Many courses require a reproducible data-analysis by means of R Markdown as part of the assessment. At the beginning of a course some students are missing an *R* installation or have yet to set up their *RStudio Cloud* account. For these students we offer on day one a *learnr* tutorial to expedite their first steps in *R* – without the hurdle of installation or account management. Overall the *learnr* tutorials are used to offer flexibility for students and instructors and ease the integration of computing into the content of the class. They also aim to reduce the extraneous cognitive load by (interesting) worked examples with partial solutions. The integrated quiz questions provide additional feedback for self-assessment.

All of the previously mentioned *learnr* tutorials are open educational resources, available via GitHub (Lübke, Gehrke, Kistler, Krol, Markgraf & Griesenbeck 2022). Development and version control of the R Markdown source files of the tutorial enables easy collaboration across different instructors. Deployment to the students is done via the professional Shiny account of the institution (RStudio 2022). For the specific settings of the apps within the account, such as how many users to allow per instance of the app and how many instances of the app to launch as demand increases, the *learnr* vignette by Angela Li provides valuable tips for a good user experience and with monetary costs in mind (Li 2020). Links to the different tutorials are provided by so called *course books* for asynchronous learning settings. These course books guide and support students self-learning by providing additional resources for a specific topic. The tutorials are also linked in the lecture slides available to the students or shared within the learning management system of the institution. Although students have constant access to the tutorials, they are encouraged to use the "recaps" before or after every lecture starting on day one of the

course. The other styles of tutorials are also integrated throughout the whole semester.

The informal feedback of the students as well as the corresponding comments in the student evaluations are very positive. Students especially appreciate the feedback within the tutorials on quizzes and exercises as well as the progressive reveal. The same is true for feedback from the many teachers who use the recaps and tutorials in their courses. For them, the simple accessibility of the delivery as well as the flexibility of integration in class offer an additional benefit. Once developed and deployed, the tutorials can be used by many students in many courses in very different settings with appropriate server capacity.

In addition to the pre-existing materials, a *learnr* based course, *Introduction to causal inference*, is currently under development. This course is designed as a self-paced online course and will be hosted by the learning platform, AI Campus, a project funded by the German Federal Ministry of Education and Research (BMBF) focused on developing the prototype for a digital learning platform specifically geared towards AI (AI Campus 2022; Lübke & Rohrer 2022). Figures 13 and 14 give an example of a quiz question with feedback within this setting.

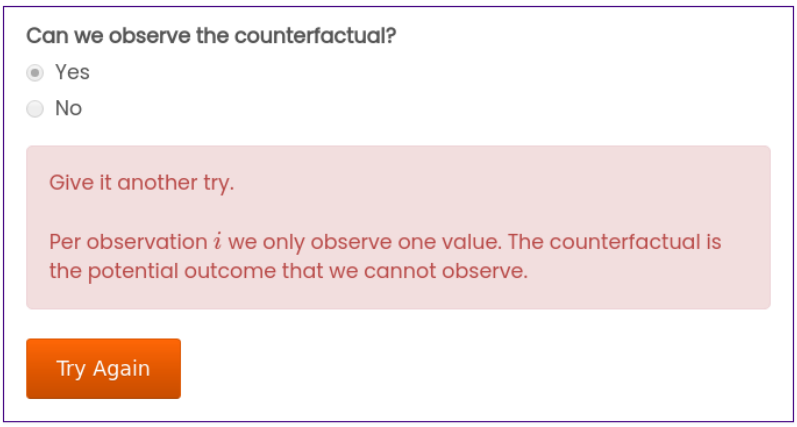

Figure 13: Quiz question with feedback.

```
```{r cf, echo=FALSE}
question("Can we observe the counterfactual?",
         answer("Yes", message = "Per observation $i$ we only observe one value.
The counterfactual is the potential outcome that we cannot observe."),
         answer("No", correct = TRUE, message = "This is correct – the
counterfactual is the potential outcome that we do not observe,"),
         allow_retry = TRUE,
         correct = random_praise(),
         incorrect = random_encouragement()
         )
```
```

Figure 14: Associated code chunks for quiz question with feedback.

6. Summary and Discussion

Whether teaching coding is a core learning objective of a class or supplementary to statistical or data-related content, *learnr* tutorials can help support instructors and students alike in statistics and data science classes. Throughout all of our use-cases, some common benefits of using *learnr* tutorials to deliver content and provide students with opportunities to practice their skills emerge. If deploying via a server or using in conjunction with RStudio Cloud services, much of the *R*, RStudio, and package installation steps are avoided, allowing students to jump directly into the content of the tutorials without extra logistical overhead or extraneous cognitive load. This may also be a good path forward, although not a free one, for shorter term trainings such as coding bootcamps, workshops, or guest lectures.

Tutorials built using *learnr* also help mitigate many problems for instructors that occur due to scale. For example, the ability to anticipate questions by adding hints and solutions and to give instantaneous feedback (via tools like the *gradethis* package) can ensure students get real time guidance even if an instructor is busy helping another student during class or learning is being done after hours when access to an instructor is not available. Combined with tools like the

learnrhash package these tutorials can have a built in sense of "soft" accountability, and completion-grading based on auto-grading of code chunks can free up instructors to evaluate students' conceptual reasoning and communication of results. These benefits will be especially useful since demand for statistics and data science classes seems likely to continue to rise.

Tutorials built using *learnr* can also benefit the student experience. Making mistakes while being provided with "corrective feedback" can benefit learning (Metcalfe 2017). The interactivity, built with less of a learning curve than directly using tools such as Shiny (Chang et al. 2021), hints, solutions, and auto-grading capabilities promotes exploration and supports trial-and-error. Progressive reveal can encourage students to take their time stepping through new content and reduce the overall cognitive load at any given point in the tutorial.

There are many resources to help instructors get familiar with the technical points of *learnr* including overviews of *learnr* in various educational contexts (Grolemund 2017; Cetinkaya-Rundel 2018; Karaesmen 2020; Correa & Milz 2021; Grosjean & Engels 2021), a behind the scenes look at the architecture underlying the package (Shrestha 2020), and example implementations of educational materials using *learnr* (Horst 2020; Silge 2021; Concord Consortium 2022; Pratt 2021; Cetinkaya-Rundel 2021*a*). From a technical standpoint, instructors familiar with R Markdown can straightforwardly develop interactive *learnr* tutorials thanks to the many resources created by the *R* user community. For example, code chunks can be quickly changed to exercises as seen in the Introduction to Statistics and Probability section. However, the potentially time consuming work comes in designing coding and quiz questions that are amenable to the format of the *gradethis* tools. For instructors wanting to try *learnr* in their own classroom, we recommend starting by using a code template or tutorial that is already being used statically in their classroom and converting it into an interactive *learnr* tutorial. This way they can focus on learning about the new tool rather than simultaneously creating content. Once they have some familiarity with the

mechanics, a further conversion to *learnr* tutorials becomes a great opportunity to get students involved in the process. Students with an introductory knowledge of *R* can help transfer pre-existing materials to *learnr*, using the instructor's case study as a model and thereby learn more about the inner workings of *R* in the process. Students who have completed a particular course can also contribute suggestions for how to improve the content and delivery of the material.

A potentially challenging aspect faced by instructors contemplating using *learnr* in their classroom is deployment. There are several options, each with their own pros and cons. Perhaps the most straightforward deployment option is to publish each *learnr* tutorial to *shinyapps.io*, the self-service platform for securely hosting Shiny applications. While straightforward to use, the free *shinyapps.io* plan allows for only five active applications at a single time. Additionally, the maximum of 25 active hours per month would make this option untenable for a typical class size (RStudio 2022). Institutions can either invest in a paid account or in an own Shiny server to accommodate the use of *learnr* across multiple, large classes. A second deployment option is to build the *learnr* tutorials into an R package, which students can then access in their own R session via `learnr::run_tutorial()`. While this solves the problem of potential cost associated with deploying more than several *learnr* tutorials in a medium-to-large class to *shinyapps.io*, instructors need to be well-versed in *R* package design. However, the *usethis* package streamlines a lot of the overhead (De Leon 2020; Wickham, Bryan & Barrett 2022). Another potential pain point is that these tutorials would optimally need to be ready at the start of the semester in order to avoid complications from students having to continuously update the R package. When the first author used the RStudio Cloud, they could update the package for each student when necessary. Alternatively, when the first author did not use the RStudio Cloud, they asked students to update the package in the middle of the course and it was more straightforward than the initial setup with all the dependencies. A third deployment option is to host each *learnr* tutorial as a .Rmd file on GitHub for students to download and run as needed. While this is the

least technically challenging option for instructors, students have direct access to the source files and any solutions that the instructor might wish to keep hidden until students attempt an exercise within the *learnr* tutorial. The second author used this approach, but since *learnr* was used as a summative assessment, having the solutions available was not a problem.

We have felt the challenges associated with teaching both statistical content and the statistical computing skills required for the modern statistician or data scientist in training, especially in introductory courses where both of these things may be new to students. *learnr*'s interactive user interface and the ability to build in automatic feedback make these tutorials amenable to self-study, allowing instructors to move some of the statistical computing content out of the classroom instruction time. Students may then use this training when completing more complicated computational tasks in an R Markdown document. *learnr* tutorials also help lighten the extrinsic cognitive load that comes with juggling content and tools across multiple modalities by building upon the split-attention effect mitigation of R Markdown documents. This is especially helpful for courses where *R* in itself is not a learning outcome. If learning to program is not a main goal but only a "side-effect" of building conceptual understanding, then *learnr* has the potential to reduce the cognitive overload compared to R Markdown. In the extreme, if tutorials are deployed via a server, students do not have to interact with RStudio at all.

The argument of Wild, Pfannkuch, Regan & Parsonage (2017) for teaching the bootstrap method, another computational tool making an appearance in modern statistics courses, applies to *learnr*: "With the rapid, ongoing expansions in the world of data, we need to devise ways of getting more students much further, much faster." We believe that *learnr* is a technology innovation that can be used in statistics education to achieve this goal. We hope that this paper provides a jumping off point for other instructors facing the challenges therein.

Overview of Supplementary Materials

A variety of *learnr* materials were described in this manuscript. These are all publicly available for reference and adaptation by the reader. In order of appearance in the manuscript these are:

- Stoudt (2021): materials for an Introductory Statistics course
- Scotina (2022): materials for a Probability course
- Stoudt (2022): materials for a Mathematical Statistics course
- Lübke, Gehrke, Horst & Szepannek (2022): English example of Introductory Statistics at scale
- Lübke, Gehrke, Kistler, Krol, Markgraf & Griesenbeck (2022): materials for Introductory Statistics at scale
- Lübke & Rohrer (2022): materials for Introduction to Causal Inference (under development)

Acknowledgments

The first author would like to thank Marney Pratt for testing, troubleshooting, and providing feedback and accountability during the process of translating the introductory statistics labs into *learnr* tutorials. They would also like to thank Katherine Kinnaird and Fatou Sanogo for continuing the usage of these tutorials in some form and for providing feedback on the content and workflow. Finally, they would like to thank Abby Flynt for sharing her computational labs for the Statistical Inference Theory course which informed the content of some of the activities described here.

The third author would like to thank Julia Rohrer and Tabea Griesenbeck for the joint work on the course *Introduction to causal inference*. This course was supported by a grant from the German Federal Ministry of Education and Research, grant number 16DHBQP040. The third author also would like to thank Matthias Gehrke, Tabea Griesenbeck, Bianca Krol and Norman Markgraf for feedback on the content.

References

Aberson, C. (2021), 'Building interactive tutorials for teaching psychological statistics online with learnr', *Technology Innovations in Statistics Education* **13**, 1–13. https://doi.org/10.5070/T513153822.

Aden-Buie, G., Chen, D., Grolemund, G. & Schloerke, B. (2021), 'gradethis: Automated feedback for student exercises in 'learnr' tutorials', https://pkgs.rstudio.com/gradethis/, https://rstudio.github.io/learnr/, https://github.com/rstudio/gradethis.

AI Campus (2022), 'The learning platform for artificial intelligence'. https://ki-campus.org/.

Baumer, B., Cetinkaya-Rundel, M., Bray, A., Loi, L. & Horton, N. (2014), 'R Markdown: Integrating a reproducible analysis tool into introductory statistics', *Technology Innovations in Statistics Education* **8**. https://doi.org/10.5070/T581020118.

Beckman, M., Cetinkaya-Rundel, M., Horton, N., Rundel, C., Sullivan, A. & Tackett, M. (2021), 'Implementing version control with git and GitHub as a learning objective in statistics and data science courses', *Journal of Statistics and Data Science Education* **29**(SUP1), S132–S144. https://doi.org/10.1080/10691898.2020.1848485.

Blickley, J., Deiner, K., Garbach, K., Lacher, I., Meek, M., Porensky, L., Wilkerson, M., Winford, E. & Schwartz, M. (2013), 'Graduate student's guide to necessary skills for nonacademic

conservation careers', *Conservation Biology* **27**, 24–34. https://doi.org/10.1111/j.1523-1739.2012.01956.x.

Borja, M. C. & Haigh, J. (2007), 'The birthday problem', *Significance* **4**(3), 124–127. https://doi.org/10.1111/j.1740-9713.2007.00246.x.

Cetinkaya-Rundel, M. (2018), *Interactive R tutorials with learnr*. https://github.com/mine-cetinkaya-rundel/cause-learnr.

Cetinkaya-Rundel, M. (2021*a*), *Data Science in a Box*. https://datasciencebox.org/interactive-tutorials.html.

Cetinkaya-Rundel, M. (2021*b*), Feedback at scale, *in* 'rstudio::global'. https://www.rstudio.com/resources/rstudioglobal-2021/feedback-at-scale/.

Cetinkaya-Rundel, M. (2021*c*), *Teaching and learning R in the cloud with RStudio Cloud*. https://mine-cr.com/talk/2021-rstudio-cloud-whyr/.

Cetinkaya-Rundel, M., Hardin, J., Baumer, B., Bray, A., Saibene, Y. B., D'Andrea, F., Villafane, R. N. & Theobold, A. (2021), *OpenIntro::Introduction to Modern Statistics Tutorials*. https://openintrostat.github.io/ims-tutorials/.

Chandler, P. & Sweller, J. (1991), 'Cognitive load theory and the format of instruction', *Cognition and Instruction* **8**, 293–332. https://doi.org/10.1207/s1532690xci0804_2.

Chandler, P. & Sweller, J. (1992), 'The split-attention effect as a factor in the design of instruction', *British Journal of Educational Pscyhology* **62**, 233–246. https://doi.org/10.1111/j.2044-8279.1992.tb01017.x.

Chang, W., Cheng, J., Allaire, J., Sievert, C., Schloerke, B., Xie, Y., Allen, J., McPherson, J., Dipert, A. & Borges, B. (2021), *shiny: Web Application Framework for R*. R package version 1.7.1 https://CRAN.R-project.org/package=shiny.

Chen, D. (2020), *Grading Code with gradethis*. https://speakerdeck.com/chendaniely/satrday-grading-code-with-gradethis.

Concord Consortium (2022), 'Math modeling with R'. https://learn.concord.org/rmath.

Cooper, G. (1990), 'Cognitive load theory as an aid for instructional design', *Australian Journal of Educational Technology* **6**, 108–113. https://doi.org/10.14742/ajet.2322.

Correa, F. & Milz, B. (2021), *Scaling feedback using learnr and gradethis in a introductory R course*. https://curso-r.github.io/LatinR2021/#1.

Csardi, G. & Sorhus, S. (2015), *praise: praise users*. https://cran.r-project.org/web/packages/praise/index.html.

Cummiskey, K., Adams, B., Pleuss, J., Turner, D., Clark, N. & Watts, K. (2020), 'Causal inference in introductory statistics courses', *Journal of Statistics Education* **28**(1), 2–8. https://doi.org/10.1080/10691898.2020.1713936.

De Leon, D. (2020), *How to deliver learnr tutorials in a package*. https://education.rstudio.com/blog/2020/09/delivering-learnr-tutorials-in-a-package/.

Gehrke, M., Kistler, T., Lübke, K., Markgraf, N., Krol, B. & Sauer, S. (2021), 'Statistics education from a data-centric perspective', *Teaching Statistics* **43**(S1), S201–S215. https://doi.org/10.1111/test.12264.

Gill, R. D. (2011), *International Encyclopedia of Statistical Science*, Springer, Berlin, Heidelberg, chapter Monty Hall Problem: Solution. https://doi.org/10.1007/978-3-642-04898-2_377.

Green, J. & Blankenship, E. (2015), 'Fostering conceptual understanding in mathematical statistics', *The American Statistician* **69**, 315–325. https://doi.org/10.1080/00031305.2015.1069759.

Grolemund, G. (2017), *Introducing learnr*.
https://www.rstudio.com/blog/introducing-learnr/.

Grosjean, P. . & Engels, G. (2021), Teaching biology students to code smoothly with learnr and gradethis, Technical report, University of Mons, Belgium.
https://user2021.r-project.org/participation/technical_notes/t208/technote/.

Guzman, L., Pennell, M., Nikelski, E. & Srivastava, D. (2019), 'Successful integration of data science in undergraduate biostatistics courses using cognitive load theory', *Life Sciences Education* **18**(ar49), 1–10. https://doi.org/10.1187/cbe.19-02-0041.

Heo, M. & Chow, A. (2005), 'The impact of computer augmented online learning and assessment tool', *Educational Technology and Society* **8**, 113–125.
https://www.jstor.org/stable/jeductechsoci.8.1.113.

Horst, A. (2020), 'Exploring missing values in naniar'.
https://allisonhorst.shinyapps.io/missingexplorer/.

Horton, N. (2013), 'I hear, I forget. I do, I understand: A modified Moore-method mathematical statistics course', *The American Statistician* **67**, 219–228.
https://doi.org/10.1080/00031305.2013.849207.

Kalyuga, S., Ayres, P., Chandler, P. & Sweller, J. (2003), 'The expertise reversal effect', *Educational Psychologist* **38**, 23–31. https://doi.org/10.4018/978-1-60566-048-6.ch003.

Kaplan, D. (2018), 'Teaching stats for data science', *The American Statistician* **72**, 89–96.
https://doi.org/10.1080/00031305.2017.1398107.

Karaesmen, E. (2020), 'Data science summer camp for high schoolers'.
https://education.rstudio.com/blog/2020/06/summer-camp-hs/.

Kim, A. & Hardin, J. (2021), "'playing the whole game": A data collection and analysis exercise with Google Calendar', *Journal of Statistics and Data Science Education* **29**, S51–S60. https://doi.org/10.1080/10691898.2020.1799728.

Li, A. (2020), *Publishing learnr Tutorials on shinyapps.io*. https://cran.r-project.org/web/packages/learnr/vignettes/shinyapps-publishing.html.

Lübke, K., Gehrke, M., Horst, J. & Szepannek, G. (2020), 'Why we should teach causal inference: Examples in linear regression with simulated data', *Journal of Statistics Education* **28**(2), 133–139. https://doi.org/10.1080/10691898.2020.1752859.

Lübke, K., Gehrke, M., Horst, J. & Szepannek, G. (2022), 'Causal inference'. https://fomshinyapps.shinyapps.io/CausalInference/.

Lübke, K., Gehrke, M., Kistler, T., Krol, B., Markgraf, N. & Griesenbeck, T. (2022), 'FOM ifes'. https://github.com/FOM-ifes/.

Lübke, K. & Rohrer, J. (2022), 'Introductory Course in Causal Inference'. https://github.com/luebby/WWWEKI-EN.

Marx, V. (2013), 'The big challenges of big data', *Nature* **498**, 255–260. https://doi.org/10.1038/498255a.

McNamara, A. (2019), 'Key attributes of a modern statistical computing tool', *The American Statistician* **73**(4), 375–384. https://doi.org/10.1080/00031305.2018.1482784.

Metcalfe, J. (2017), 'Learning from errors', *Annual Review of Psychology* **68**, 465–89. https://doi.org/10.1146/annurev-psych-010416-044022.

Nolan, D. & Lang, D. (2010), 'Computing in the statistics curricula', *The American Statistician* **64**, 97–107. https://doi.org/10.1198/tast.2010.09132.

Nolan, D. & Speed, T. (1999), 'Teaching statistics theory through applications', *The American Statistician* **53**, 370–375. https://doi.org/10.1080/00031305.1999.10474492.

OpenIntro Team (2022), 'Introductory statistics with randomization and simulation labs'. https://www.openintro.org/book/isrs/.

Paas, F. & van Merrienboer, J. (1994), 'Variability of worked examples and transfer of geometrical problem-solving skills: A cognitive-load approach', *Journal of Educational Psychology* **84**, 122–133. https://doi.org/10.1037/0022-0663.86.1.122.

Pratt, M. (2021), 'Intro to R for biology learnr tutorials'. https://github.com/marneypratt/r4bio.

Renkl, A. (2005), 'The worked-out examples principle in multimedia learning', *The Cambridge Handbook of Multimedia Learning* pp. 229–245. https://doi.org/10.1017/CBO9780511816819.016.

Renkl, A., Atkinson, R. K. & Maier, U. H. (2000), 'From studying examples to solving problems: Fading worked-out solution steps helps learning', *Proceedings of the Annual Meeting of the Cognitive Science Society* **22**(22). https://escholarship.org/uc/item/81b9j9hs.

RStudio (2022), 'Share your Shiny applications online'. https://www.shinyapps.io/.

Rundel, C. (2021), 'learnrhash', https://github.com/rundel/learnrhash.

Schloerke, B., Allaire, J., Borges, B. & Aden-Buie, G. (2021), *learnr: Interactive Tutorials for R*. https://rstudio.github.io/learnr/.

Scotina, A. (2022), 'Probabilitylabs: learnr labs for probability'. https://github.com/ScotinaStats/ProbabilityLabs.

Scoville, R. (1966), 'The hat-check problem', *The American Mathematical Monthly* **73**(3), 262–265. https://doi.org/10.2307/2315337.

Shrestha, N. (2020), *Learning learnr*.
https://education.rstudio.com/blog/2020/07/learning-learnr/.

Silge, J. (2021), *Learn tidytext with my new learnr course*.
https://juliasilge.com/blog/learn-tidytext-learnr/.

Stigler, S. M. (2006), 'Isaac Newton as a probabilist', *Statistical Science* **21**(3), 400–403.
https://www.jstor.org/stable/27645773.

Stoudt, S. (2021), 'nudgestatlabs: learnr labs for intro stat'.
https://github.com/sastoudt/nudgeStatLabs.

Stoudt, S. (2022), 'StatTheoryLabs'. https://github.com/sastoudt/StatTheoryLabs.

Sweller, J. & Chandler, P. (1994), 'Why some material is difficult to learn', *Cognition and Instruction* **12**, 185–233. https://doi.org/10.1207/s1532690xci1203_1.

Sweller, J., Chandler, P., Tierney, P. & Cooper, M. (1990), 'Cognitive load as a factor in the structuring of technical material', *Journal of Experimental Psychology. General* **119**, 176–192.
https://doi.org/10.1037/0096-3445.119.2.176.

Tintle, N., Chance, B., Cobb, G., Roy, S., Swanson, T. & VanderStoep, J. (2015), 'Combating anti-statistical thinking using simulation-based methods throughout the undergraduate curriculum', *The American Statistician* **69**, 362–370.
https://doi.org/10.1080/00031305.2015.1081619.

Wickham, H. (2016), *ggplot2: Elegant Graphics for Data Analysis*, Springer-Verlag New York.
https://ggplot2.tidyverse.org.

Wickham, H., Bryan, J. & Barrett, M. (2022), *usethis: automate package and project setup*.
https://cran.r-project.org/web/packages/usethis/index.html.

Wickham, H., Francois, R., Henry, L. & Muller, K. (2018), *dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.

Wild, C. J., Pfannkuch, M., Regan, M. & Parsonage, R. (2017), 'Accessible conceptions of statistical inference: Pulling ourselves up by the bootstraps', *International Statistical Review* **85**(1), 84–107. https://doi.org/10.1111/insr.12117.

Woodard, V. & Lee, H. (2021), 'How students use statistical computing in problem solving', *Journal of Statistics and Data Science Education* **29**(1), 145–156. https://doi.org/10.1080/10691898.2020.1847007.

Xie, Y., Allaire, J. J. & Grolemund, G. (2021), *R Markdown: The Definitive Guide*, Chapman & Hall/CRC, chapter Document Templates. https://bookdown.org/yihui/rmarkdown/document-templates.html.

Xie, Y., Dervieux, C. & Riederer, E. (2020), *R Markdown Cookbook*, Chapman and Hall/CRC, Boca Raton, Florida. https://bookdown.org/yihui/rmarkdown-cookbook.