DESIGNING POSITIVE FIRST EXPERIENCES WITH CODING FOR INTRODUCTORY-LEVEL DATA SCIENCE STUDENTS

Anna Fergusson
University of Auckland, New Zealand
a.fergusson@auckland.ac.nz

There is a need to explicate the design of learning tasks that introduce coding to introductory-level data science students. Computational lab components of an introductory statistics and data science course at the University of Auckland were written so students could complete them online. The R package learnr was used to create interactive lab tasks that featured videos, code exercises, progressive revealing of task components and quiz questions. This paper provides teachers with practical guidance on how to design and implement "first experience" tasks with coding using learnr. Consideration is given to balancing the learning of new statistical, computational, data-related, and tool-related knowledge.

INTRODUCTION

Data science students need learning experiences that support their development of integrating both statistical and computational thinking to learn from data. Developing at least some coding skills will support students to access and manipulate a wide variety of digital data sources (e.g., Gould, 2010) using automated and reproducible approaches. A teaching challenge is how to introduce coding within an introductory-level data science course, so that the learning of new statistical, computational, data-related, and tool-related knowledge is not overwhelming for students. Additionally, students' first experiences with coding should be positive and support future engagement with learning data science, and the tasks used will influence the nature of the learning that takes place (e.g., Doerr & Pratt, 2008). Data science educators have been advised to look to the computer science education community to inform their teaching of coding (McNamara, 2015). Pedagogical approaches recommended by computer science educators are to scaffold and structure programming tasks (e.g., Lee et al., 2011), and to align the task to a clear purpose for learning from data (e.g., Cunningham, 2021). Although statistics education task design research has not specifically explored coding as the main computational tool, researchers have attempted to define the complex relationship between software features, task design and learners' statistical conceptions (e.g., Ben-Zvi, 2000).

When designing tasks that introduce data science students to coding, further consideration is needed about how both the tool and task support learners to access statistical concepts graphically (e.g., Ben-Zvi, 2000), as for novices the mental models needed to produce graphics using code-driven tools are different from those used for generating data from models. Learners may require less *computational transparency* (Fergusson & Pfannkuch, 2022) and greater scaffolding and support when initially using code-driven approaches to produce graphics. The task designer needs to make several decisions, for example: which computer programming language to use; the environment within which to read, write and execute code; the syntax and layout of the code used, including what packages to use; and how much of the code to reveal or how much to "hide" within functions. These design decisions clearly link to pedagogical goals. Interactive documents such as *RMarkdown* (Allaire et al., 2018) can be used as "computational templates" (Wickham, 2018) to assist learners to engage with the computer programming language *R* (R Core Team, 2020), while simultaneously providing structure and support for new computational ideas. Alternatively, the *R* package *learnr* (Schloerke et al., 2018), provides a way to produce an interactive web-based task where students can execute small "chunks" of R code within a web browser. For learners new to computer programming, two advantages of using the interactive web-based environment provided by *learnr* are: (1) the user interface is simple and so potentially less overwhelming, and, (2) the desired interactions between the tool and the learner's statistical and computational thinking can be embedded within both the tool and task design.

In general, it is difficult to find substantial literature that explicitly communicates specific *pedagogica*l strategies for designing tasks that introduce coding to data science students. In this paper, I focus on the design of the lab tasks used in an introductory statistics and data science course at the University of Auckland.

In: EM Jones (Ed.), *Fostering Learning of Statistics and Data Science*
Proceedings of the Satellite conference of the International Association for Statistical Education (IASE),
July 2023, Toronto, Canada. https://iase-web.org

TEACHING CONTEXT

The teaching context for this paper is STATS 100 (Concepts in Statistics), an introductory statistics and data science course offered at the University of Auckland. STATS 100 serves as both a foundation course for more advanced introductory level courses in Statistics or Data Science, as well as a service course for several client departments including Business and Psychology. The course is designed to support students who have limited experience or confidence with Grade 12 Mathematics and/or Statistics, and students are not assumed to have any prior experience with coding. Typically, the course has around 200 students enrolled per semester. STATS 100 is structured as 12 different week-long topics which are arranged within four modules: (1) making predictions, (2) conducting tests, (3) building models, and (4) informing decisions. Students have access to an online coursebook that provides notes, examples, and interactive "self-marking" exercises. Each week, students attend three activity-based lectures, complete an online lab task, and submit a weekly assignment. The weekly assignment involves a mini data investigation that combines use of GUI-driven tools (tools where computational actions are initiated by pointing, clicking, or gesturing within visual environments) and code-driven tools (tools where computational actions are initiated using text commands).

The computer programming language used is *R*. The goal of STATS 100 is not to provide a comprehensive introduction to computer programming with *R* but instead to build greater awareness of the ways code can be used to learn from and create with data. When the course was introduced in 2018, the labs were completed in person and involved the students working through *RMarkdown* documents deployed using RStudio Cloud (now Posit Cloud). A Google form was used to provide structure to the task and to record students answers to questions throughout the task. During 2019 to 2020, the lab tasks were created using the R package *learnr* and deployed using university-hosted *Docker* containers. In 2019, the students worked through the lab tasks in pairs within timetabled lab sessions. In 2020, students worked through the lab tasks independently online.

RESEARCH CONTEXT

At the same time as developing and teaching STATS 100, I undertook research for a PhD in statistics and data science education. The purpose of my PhD research was to explicate design principles for the construction of statistical modelling tasks that introduce code-driven tools (Fergusson, 2023). My research involved designing and implementing tasks with high school statistics teachers and several of these research tasks were derived from tasks I had developed for STATS 100. Accordingly, most of the code-driven aspects of the tasks used for my research were created using the R package *learnr*. My research led to a new task design framework for introducing code-driven tools through statistical modelling. In Fergusson and Pfannkuch (2022), I present the key components of the task design framework and describe how the design of a *learnr* task for introducing APIs (Application Programming Interfaces) and predictive modelling can be explicated by this framework. The task design framework has six design principles which are intended to inform decisions about the learning task in terms of specific actions or experiences for learners and the chronological order of these actions or experiences. These six design principles are: immerse, familiarise, describe, match, adapt and explore.

The six design principles provide guidance as to how to structure the task. At the beginning of the task, students should be *immersed* in the data context and then *familiarised* with key statistical modelling ideas and actions. The activities used should encourage students to engage with the data context and should not involve code-driven tools. The task should then focus on *describing* key computational steps in the statistical modelling process using words, *matching* these modelling steps with specific lines or chunks of code, and *adapting* code to complete a specific modelling action. The activities used should support students to build some awareness of code syntax and structure. The task should end by asking students to modify code that has been provided in order to *explore* a "what if?" scenario or new problem related to the data context. In my research with teachers (Fergusson & Pfannkuch, 2022), I found that carefully ordering the introduction of new statistical and computational ideas and *progressively revealing* each step of the task visually appeared to minimise cognitive load (cf. Wouters et al., 2008). It also appeared that using *tinker questions* within the task supported the introduction of new computational ideas. A tinker question presents a set of related TRUE/FALSE statements that are deliberately written to require action by the learners within a computational learning environment to evaluate each statement. The findings from my research align with my experiences of designing and implementing similar tasks to introduce coding to STATS 100 students.

DESIGNING FIRST EXPERIENCES WITH CODING

In this section, I provide a narrative of the decisions made when designing the first experiences with coding for students in STATS 100. As this first experience is the lab task that students complete in the first topic/week of the course, I focus primarily on the design of the first lab tasks used in 2019 and 2020. I begin by describing key design decisions made when structuring the 2020 task, before identifying examples of how specific learning interactions can be embedded using both tasks. The first topic of the course is called *Exploring time series data*. This topic is taught first as it provides a range of accessible opportunities for introducing and integrating statistical and computational approaches for learning from data, such as:

- collecting data through sensors (e.g., Apple health steps);
- accessing large open data sets (e.g., time series data from StatsNZ);
- restructuring data by different time periods (e.g., turning daily data into weekly data);
- describing key features of data (e.g., trends, seasonality);
- developing informal prediction models (e.g., sketching forecasts by hand);
- creating informative visualisations (e.g., use of titles, labels, colours, and other graphical techniques).

In the lecture-based activities for this topic, food price data from StatsNZ and Google search interest data for different food items had been explored using GUI-driven tools. Activities included: sketching by hand trends for food prices, describing key features of food price data such as seasonality, exploring how search interest for coffee changes by the day and hour using Google trends, and critiquing examples of time series visualisations. Hence, before completing the lab task, students were fully *immersed* and *familiarised* with the food prices and Google trends data contexts and key statistical modelling ideas related to time series data.

The lab task begins by asking students to watch a short video about how they can use Google to find recipes for different foods (see https://youtu.be/IsUN1dUbbM8). The intent of this video is to encourage students to think about reasons why people might search for food items using Google. After clicking the "Next" button that is a feature of the *progressive reveal* setting used for the *learnr* task, students are presented with a familiar representation, a simple line graph of search interest for tomatoes in New Zealand over the last five years (Figure 1).

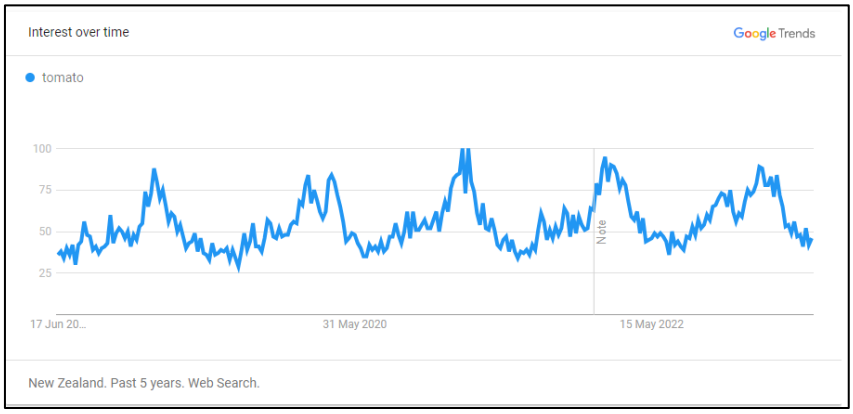

Figure 1: Embedded Google trends plot of search interest for tomatoes within New Zealand over last five years.

The plot is embedded from Google Trends and is interactive, so students can move their mouse/cursor over the graph and read information about each data point. The start of the lab task is designed to link back to lectures and again *immerse* and *familiarise* students with data and statistical knowledge, before introducing new computational knowledge. In the next step of the lab task, students are briefly introduced to the R package *gtrendsR*, before being asked simply to run the code to recreate the plot shown in Figure 1. Figure 2 provides a screenshot of this task step.

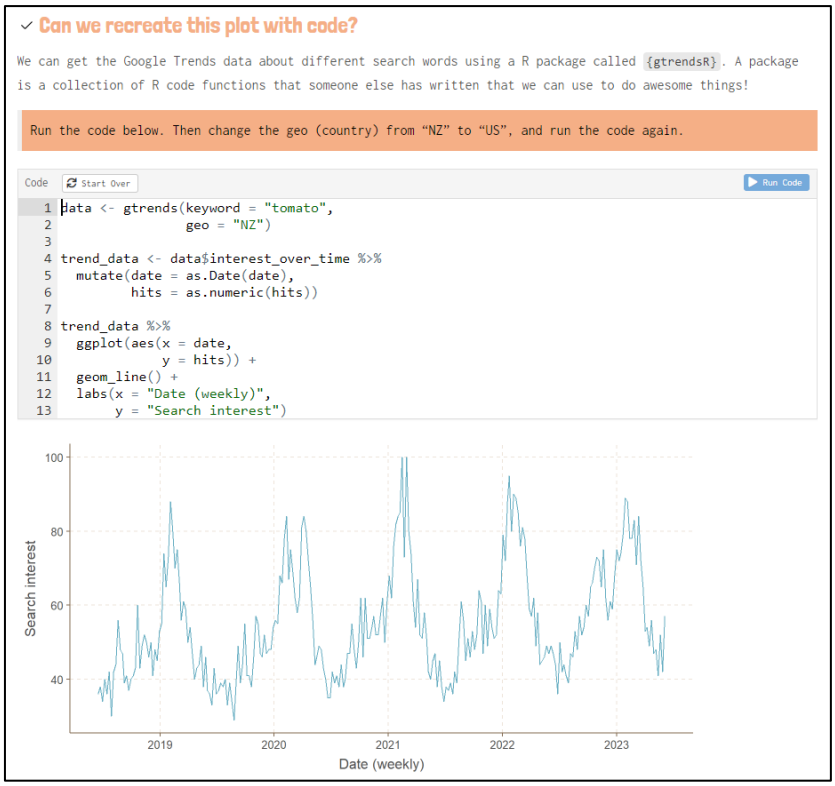

Figure 2: Screenshot of first step in lab task where students first see R code.

Note that students are asked to run the code first, and then change "NZ" to "US" before running the code again. The decision here is purposeful, so that the first time running the code is successful, and the second time is likely to be successful due to only a small change being asked for. Furthermore, the changes support a statistical modelling focus, in that they allow the students to compare search interest between NZ and the US. Additionally, students are not asked to read all the code, just the part that needs changing. Instead, they interact with code like how they would interact with a GUI-driven tool – by simply clicking a button. In the second step, the statistical modelling goal of being able to compare the search interest of tomatoes between NZ and the US provides a reason to *describe* and *match* key features of the code (Figure 3).

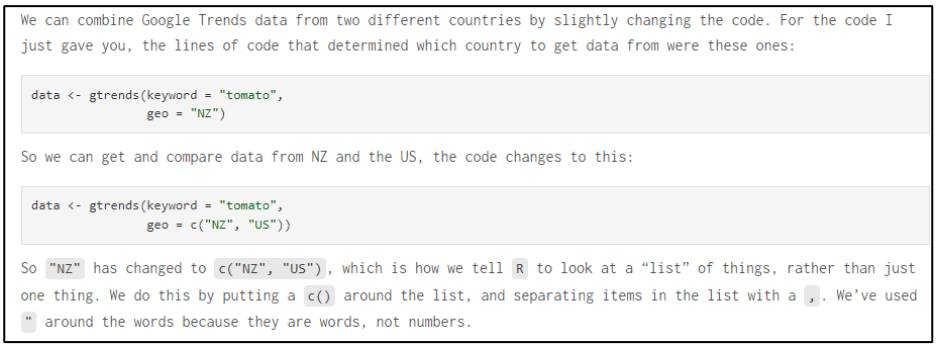

Figure 3: An example of the task describing and matching code.

Rather than introduce vectors (colloquially referred to as lists on purpose) in abstract terms, new computational knowledge is introduced more contextually and naturally. After being introduced to the ISO codes for all countries, students are then asked to *adapt* the code to compare search interest for tomatoes between NZ and another country that is not the US. Students are then given a mini challenge, where they are asked to *adapt* the code so they can compare the search interest for avocado between Mexico, Japan and Spain. Increasing the number of countries compared also supports the introduction of "subsetting" or "facetting" to create visualisations, and the required code needed to accomplish this.

In a similar approach, the lab task then continues to use *familiar* statistical modelling actions or goals to demonstrate how small changes to the code result in changes to the visualisation produced. For instance, in lectures students have seen how the labelling of the time periods on the x-axis can influence how the plot is interpreted, and so learn how they can "tweak" the code to change the date breaks and labels. Students also explore how to change the labels of a plot so that there is a clear title, subtitle, and caption, and how to change the background colour of their plot based on a hex code (e.g., #d3d3d3). The lab task ends by asking students to create their own informative time series plot that uses a keyword (search term) that is different from any of the examples used in the lab and compares search interest for two or more countries different from the examples used in the lab (i.e., not NZ or US). The code supplied for the "lab challenge" does not have informative labels, uses unhelpful breaks for the dates, and uses white as the background colour, so there are quite a few changes each students needs to *explore*. Importantly, the challenge encourages students to be creative, by allowing them to make their own selections for keywords, countries, and background colour, which I believe is an important priority for their first experience with coding.

One of the many affordances of creating a web-based lab task using the R package *learnr* is that the desired interactions between the tool and the learner's statistical and computational thinking can be embedded within both the tool and task design. Interactions can involve more than executing small chunks of code in the browser. In addition to embedding videos and interactive plots in the task, animated GIFs can also be used to demonstrate actions and highlight visually where to look on the screen. When introducing ISO codes for different countries in the lab task, an animated GIF shows students how the URL for a Google trends search result changes when the student changes the country, for example from *trends.google.com/trends/?geo=US* to *trends.google.com/trends/?geo=AL*.

Underlying principles for the design of the lab tasks for STATS 100 are that students will learn by "changing stuff and seeing what happens" and that the risks for students to encounter issues with running code are minimised. One way that students can be encouraged to try out the "mini challenges" embedded with the task is to use tabbed content. The main tab is labelled "Try it out!" and provides initial code with a "challenge" for changing the code to produce a different output. The other tab is labelled "See an answer" and provides students who need some initial help a nonjudgmental way to view examples of code approaches. The use of *tinker questions* is a crucial part of the task design. For example, when students first learn how to read data from a Google sheet published as a CSV into R, the tinker question encourages students to inspect the data they have imported, as well as familiarising students with the interactive data frame produced within the web task (Figure 4). Later in the task, *tinker questions* are used to focus students' attention on identifying and describing key features of the time series data and how these features change when different food items or countries are used. In this way, the *tinker questions* both provoke students to make changes to code but also promote linking computational actions to statistical modelling actions.

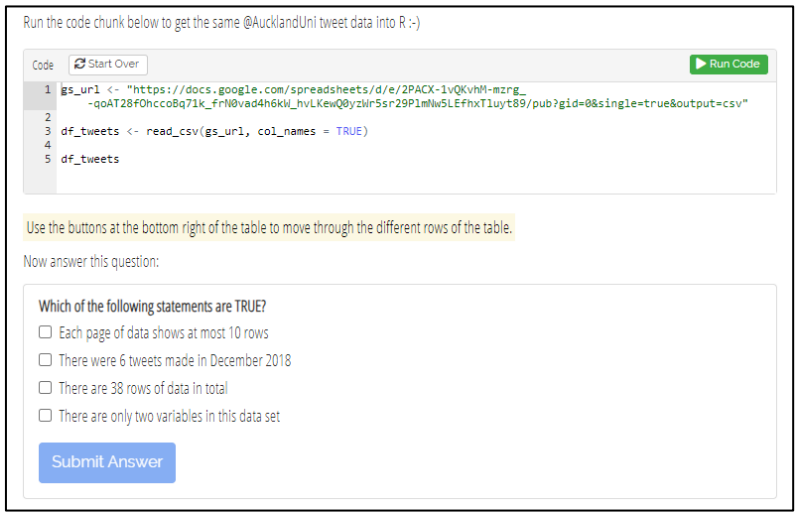

Figure 4: An example of a tinker question

SUMMARY

The first experiences introductory-level data science students have with coding should be situated within modern data contexts and purposeful statistical modelling tasks. Initial learning goals should be building a greater awareness about how code can be used to learn from data and empowering students to create with data and code. To support positive learning experiences, lab tasks should be designed as part of a larger teaching sequence, so that students can make connections to familiar data contexts and statistical modelling actions when encountering new computational ideas and approaches. Web-based tasks created using R packages such as *learnr* should be carefully structured and utilise features such as progressive reveal and tinker questions to support development of the learner's statistical and computational thinking. More task design research is needed to support educators to make effective decisions when creating tasks that introduce coding to introductory-level data science students.

REFERENCES

Allaire, J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W., & Iannone, R. (2021). *rmarkdown: Dynamic documents for R*. RStudio. https://rmarkdown.rstudio.com

Ben-Zvi, D. (2000). Toward understanding the role of technological tools in statistical learning. *Mathematical Thinking and Learning, 2*(1-2), 127–155. https://doi.org/10.1207/S15327833MTL0202_6

Cunningham, K. I. (2021). *Purpose-first programming: A programming learning approach for learners who care most about what code achieves* (PhD dissertation). University of Michigan.

Doerr, H. M., & Pratt, D. (2008). The learning of mathematics and mathematical modeling. In M. K. Heid & G. W. Blume (Eds.), Research on technology and the teaching and learning of mathematics: Volume 1 Research syntheses (pp. 259–285). Information Age Publishing.

Fergusson, A. (2023). *Towards an integration of statistical and computational thinking: Development of a task design framework for introducing code-driven tools through statistical modelling* (PhD thesis). University of Auckland.

Fergusson, A., & Pfannkuch, M. (2022). Introducing high school statistics teachers to predictive modelling and APIs using code-driven tools. *Statistics Education Research Journal, 21*(2), Article 8. https://doi.org/10.52041/serj.v21i2.49

Gould, R. (2010). Statistics and the modern student. *International Statistical Review, 78*(2), 297–315. https://doi.org/10.1111/j.1751-5823.2010.00117.x

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smtih, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32–37. https://doi.org/10.1145/1929887.1929902

McNamara, A. A. (2015). *Bridging the gap between tools for learning and for doing statistics* (PhD dissertation). University of California, Los Angeles.

R Core Team. (2020). R: A language and environment for statistical computing. https://www.Rproject.org

RStudio Team. (2018). RStudio: Integrated development environment for R. http://www.rstudio.com/

Schloerke, B., Allaire, J., & Borges, B. (2018). Learnr: Interactive tutorials for R. CRAN. https://CRAN.R-project.org/package=learnr

Wickham, H. (2018). *Should all statistics students be programmers*? Paper presented at the Tenth International Conference on Teaching Statistics (ICOTS10, July 2018), Kyoto, Japan. Speaker Deck: https://speakerdeck.com/hadley/should-all-statistics-students-be-programmers

Wouters, P., Paas, F., & van Merriënboer, J. J. (2008). How to optimize learning from animated models: A review of guidelines based on cognitive load. *Review of Educational Research, 78*(3), 645–675. https://doi.org/10.3102/0034654308320320