Emerging Models for a Second Undergraduate Course in Data Science

Benjamin S. Baumer
Brianna C. Heggeseth Nicholas J. Horton
Leslie Myint Michael A. Posner
Paul Roback Gillian D. Beltz-Mohrmann

2026-01-29

As data science continues to mature as an academic discipline, curricular standardization remains elusive. While a number of groups have defined program-level learning outcomes for data science, there is no consensus surrounding course-level learning outcomes. We present five different models for a *second* course in data science, each of which comes with its own learning goals, coverage of topics, points of emphasis, prerequisites, and role in our respective (statistics and) data science curricula. We use the data science topic, subtopic, and competency concept framework created by the MASDER project to analyze the content of our courses, and present areas of similarity and difference across them. While we cannot claim that our list of courses is comprehensive, we hope that the material we present will provide other institutions and instructors with vetted ideas for second courses in data science. We also hope that our analysis can move the field towards greater coordination (and perhaps even standardization) within the data science curriculum.

1 Introduction

As programs in data science have sprung up worldwide over the past decade, a consistent vision for an undergraduate curriculum in data science remains insufficiently articulated, let alone adopted. A decade ago, Hardin et al. (2015) described several early courses in data science, revealing some commonalities but also vast differences in approaches and topics covered. Soon after that, De Veaux et al. (2017) imagined an innovative, holistic experience in data science, but such truly interdisciplinary courses remain more exceptional than common. While influential groups like National Academies of Sciences, Engineering, and Medicine (2018) have broadly defined the goals and learning outcomes for data science majors, they have not offered a blueprint for a series of courses. Donoho (2017) grappled with the big, hard questions about what makes data science a *science*, but similarly gave us six divisions of "Greater Data Science" and left us to figure out the details. ABET, formerly known as the Accreditation Board for Engineering and Technology, Inc. (Blair et al. 2021) has brought the imprimatur of accreditation to such programs, but also articulates only program- and student-level outcomes. Posner and Kerby-Helm (2025) have synthesized professional recommendations into a set of Data Science Program Topics, but this has not yet been aligned with curricula.

However, despite a decade of robust curricular growth in data science since Hardin et al. (2015), the data science courses that provide the glue to these data science programs lack the standardization that adjacent disciplines like mathematics, computer science, and even statistics have developed over the years. If a student takes a course in linear algebra, data structures, or introductory statistics at virtually any college in the United States, it is highly likely that they will be able to transfer those credits to another institution and fulfill a corresponding requirement because those courses are relatively standardized across institutions. The Mathematical Association of America (MAA), the Association for Computing Machinery (ACM), and the American Statistical Association (ASA) have expended a great deal of energy to make it so. But it is not yet so with data science, where, for example, an introductory data science

course at one institution is far less likely to fulfill a requirement at another institution. Recently, the MAA, ACM, and ASA along with the Academic Data Science Alliance (ADSA), the Institute for Operations Research and the Management Sciences (INFORMS), and the Society for Industrial and Applied Mathematics (SIAM), created a joint Data Science Curriculum Task Force. This group is focused on program-level curriculum (i.e., core knowledge areas) and not course-specific guidelines, and their recommendations have not yet been shared publicly.

To date, the lack of a standardized curriculum in data science has caused relatively little chaos because many curricula in data science rely largely or entirely on a package of longstanding courses in adjacent disciplines, with perhaps a flexible, project-based capstone experience at the end. Thus, it is currently the standardization of the courses *not labeled data science* that provides some curricular consistency of data science programs. Recent curriculum guides from the MAA (Schumacher and Siegel 2015), ACM (Danyluk and Leidig 2021), and ASA (Carver et al. 2016) all touch on data science, but still fail to deliver standardization. Without consensus and coordination, this situation is likely to get worse in the near future. Baumer and Horton (2023) illustrate how the lack of curricular standards in data science becomes especially problematic at the interface between two-year and four-year colleges.

INSERT: Intro to MASDER project (Posner and Kerby-Helm 2025)

Paul: a necessary paragraph, and a good place for it, Brianna: MASDER's work on data science topics is cited above, but having a few sentences here as it relates to commonality / standardization of topics in curriculum make sense here

As data science matures as a discipline, programs in data science cannot simply be a grab bag of courses from other programs. If students are simply choosing two courses from this pile, one from that pile, etc., there is no explicit scaffolding and connection between courses in their educational journey. Standardized courses in data science, especially early courses, can provide essential foundational structure that future courses can build on in a systematic manner.

1.1 A first course in data science

Apart from the standard courses in adjacent disciplines that provide data science students with material relevant to a data

science major, several visions for a *first course in data science* (a.k.a., DS1) have been articulated. Early examples were Yan and Davis (2019) (targeting first-year students), Baumer (2015) (targeting advanced undergraduates), and Hicks and Irizarry (2018) (targeting graduate students). As expected with an emerging discipline, consensus around what topics should be included in such a course has been slow to build. For example, the question of whether a first course in data science should include statistical inference remains open. The course described in Baumer (2015) later morphed into an introductory course that deliberately does *not* include statistical modeling and inference, while the course described in Çetinkaya-Rundel and Ellison (2021) *does* include statistical modeling and inference. Numerous textbooks (Irizarry 2019; Baumer et al. 2021; Ismay and Kim 2019) support either format.

Nevertheless, despite the lack of standardization, two topics consistently appear as focus areas of a first course in data science: **data wrangling** and **data visualization**. Additionally, we perceive broad consensus that a first course in data science should include significant exposure to a high-level programming language, which is nearly always R or Python. Thus, regardless of whether a first course in data science includes statistical inference, statistical modeling, or a more computational skill (e.g., how to query a database), such a course at any imaginable institution is highly likely to engage students with coding in either R or Python and develop the capacity for students to wrangle and visualize data. Other common data science concepts (not always considered *topics*) that appear in first courses include ethics, workflows, and the data science lifecycle (Wing 2019; Stodden 2020).

1.2 Recent models for a second undergraduate course in data science

While the specifics of a first course in data science continue to coalesce, we turn our attention to the *second course in data science* (a.k.a., DS2). Even though the exact learning outcomes of the first course remain nebulous around the edges, there is enough consensus to proceed with the expectation that students

in a second course in data science will have the aforementioned foundational skills for data wrangling and data visualization in R and/or Python. In this paper, we explicate several recently developed versions of a second course in data science that further develop data science skills. We hope that these courses will provide instructors and curriculum designers with a sufficiently diverse set of courses and topics to build out their own curricula. Here again, we hope these courses will be particularly influential for two-year colleges, who will likely need to develop a second course in data science soon (Baumer and Horton 2023). In Section 7.2, we also situate these courses in relation to the vision for DS2 articulated by De Veaux et al. (2017).

Mercifully, delineating the specific boundaries between *data science*, *statistics*, and *computer science* feels less important now than it did ten years ago. But we should be clear about what a second course in data science (as we see it) is not.

It is **not** a second or third course in statistics. A second course in statistics typically focuses on statistical modeling and/or experimental design, and is well supported by a variety of textbooks, including Cannon et al. (2019), Ramsey and Schafer (2013), Kuiper and Sklar (2012), Sheather (2009), and Cobb (1998). Any number of third courses in statistics might focus on more advanced techniques in statistical modeling, for example hierarchical or multilevel models, random effects models, generalized linear models, or Bayesian models (see, among many, Roback and Legler (2021)).

A second course in data science is also **not** a second course in computer science, which often focuses on data structures (CC2020 Task Force 2020). While not what we have in mind for a second course in data science, these statistics and computer science courses provide good examples of how a second course in a discipline builds on the grounding from a first course to expand fundamental knowledge in that discipline.

Finally, these second courses in data science are also **not** a course in machine learning (Bishop and Nasrabadi 2006) or statistical learning (James et al. 2013), which are important parts of a data science curriculum that we believe merit their own course or courses. We view these courses as excellent applied courses at the intersection of statistics, computer science, and data science,

especially once a foundation has been formed in one or more of those disciplines, and we offer them as standalone courses in our curricula.

Among the courses we describe, there is agreement in further developing data science programming skills and emphasizing data science workflow through the use of version control tools. The particular courses, however, range from the very technical (building a package) to the very broad-minded (data storytelling; see Section 7.1).

1.3 Paper outline

To the best of our knowledge, this paper represents the first wide-ranging attempt to provide models for a second course in data science[1]. Specifically,

- In Section 2, we describe a course taught at St. Olaf College by Paul Roback and Joe Roith called "Data Science 2" that helps students become more proficient in computing aspects of data science while gaining experience with the data science workflow through a series of creative projects.
- In Section 3, we describe a course taught at Macalester College by Leslie Myint and Brianna Heggeseth called "Intermediate Data Science" that develops students' abilities to tell stories with real data, through a variety of mechanisms (e.g., maps, Shiny apps, etc.).
- In Section 4, we describe a course taught at Villanova University by Michael Posner called "Intermediate Data Science" that covers text mining, interactive data visualization (data tables, **plotly**, and **shiny**), and introduces students to data science tools (SQL, GitHub).
- In Section 5, we describe a course taught at Smith College by Ben Baumer, Albert Kim, and William Hopper called "Programming for Data Science in R" that culminates in students building an R package.

[1]The courses described in this paper came together during a session at the 2024 Joint Statistical Meetings titled: "Is There a 'Common Core' for a Data Science 2 Course?".

- In Section 6, we describe a course taught at Smith College by Gillian Beltz-Mohrmann called "Programming for Data Science in Python" that culminates in students building a Python package.
- In Section 7, we map learning outcomes from our five exemplar courses to the topics, sub-topics, and competencies which grew out of the MASDER project, looking for common threads and unique aspects.
- Finally, Section 8 offers a discussion and closing thoughts.

2 A course that focuses on literate programming and data science workflow

SDS 264: Data Science 2 debuted in spring semester of 2024 at St. Olaf College[2], the result of two forces within the Statistics and Data Science program. First, the Introduction to Data Science course that we had been teaching since spring semester of 2018 had become so packed with content and moved so fast for many students that it was no longer an "introductory" experience. As a result, we shifted about 2/3 of the material to a 100-level DS1 course that was truly introductory, with a primary focus on data wrangling and data visualization in R. The other 1/3 of the material, along with many topics we had always wanted to include, became a 200-level DS2 course which built on the fundamentals gained in DS1. Second, in 2024, we were converting a concentration in Statistics and Data Science to a full-fledged major, and we envisioned an early, well-defined core set of two courses in data science to parallel an early core set of two courses in statistics.

Brianna: It is important to use SDS 264 throughout. You refer to the class as DS2 in the next subsection.

SDS 264 recognizes that there are many fundamentals of literate programming (Knuth 1984) that a modern data scientist should possess that are not covered in DS1. We therefore focus on many of those fundamentals in the presence of real data and real questions. Examples include writing functions (to perform common tasks efficiently), iteration (to effectively loop through

[2]St. Olaf College is a private liberal arts college in Northfield, Minnesota, known for its strong programs in mathematics and sciences and its commitment to integrative learning.

similar instances), regular expressions (to clean and analyze text data), data acquisition (to acquire analyzable data from websites, APIs, and other sources), data types (to gain a deeper understanding of the R programming language), version control (to work well in group settings), and structured query language (SQL, to wrangle even larger data sets). Students then apply these newly mastered fundamentals to several areas of application, including mapping, text analysis, webpage creation, and dynamic graphics and dashboards. We hope to expand students' understanding of data beyond traditional columns of numbers and plots like scatterplots and barplots. SDS 264 students become more proficient in the computing aspects of data science while gaining experience with the data science workflow and remaining grounded in real data. They also gain an expanded set of tools for discovery and decision making with data.

We remind students that DS2, like DS1, is not a substitute for coursework in statistics, where they study core ideas like study design, variability, and inference at a deeper level, nor is it a substitute for coursework in computer science, where they study algorithms and systems at a deeper level, or mathematics, where they study patterns and order, learning the language of modeling and probability (among other ideas).

2.1 Learning goals

Students completing DS2 at St. Olaf will be able to:

1. Use efficient and literate coding practices such as iteration, functions, and regular expressions
2. Extract data from databases using structured query language
3. Scrape data from websites and other sources to create analyzable data sets
4. Create meaningful choropleth maps
5. Produce dynamic graphs that a user can interact with
6. Work in data science teams using version control and other sound workflow practices
7. Tell stories with data while carefully considering ethical implications

Additionally, we hope that by the end of SDS 264, each student has developed a skill set and tangible products that would impress a future interviewer, as well as teaching their professor and their classmates at least one new thing about this quickly-moving discipline of data science.

2.2 Prerequisites

Our only prerequisite for DS2 at St. Olaf is DS1 (also taught with R), making this a true two-course sequence on the fundamental building blocks of the field.

2.3 Content

Before the first day of class, students are asked to install R and RStudio, as well as `git` and GitHub, on their own computers. This immediately signals a break from DS1 and any previous statistics courses they may have taken, which typically rely on the RStudio Server maintained by St. Olaf's IT Department. While reviewing data wrangling and data visualization from DS1, students begin to learn version control using GitHub in the RStudio environment, as well as focus more attention on the quality of their code.

Next, strings and regular expressions are presented, allowing students to perform some simple but motivating text analyses. We then shift to key computational building blocks in data science, including functions, iteration, and data types. While computer science majors may have experience with these ideas, many students are exposed to them for the first time. These building blocks are immediately put to use for data acquisition, using APIs and R packages, or by directly (and ethically) scraping tables and web pages. Students are then given the opportunity to create personal web pages with Quarto (Scheidegger et al. 2022), which provide a vehicle for illustrating project work and source code to those students who choose to make theirs public.

A mini-project with choropleth maps provides an immediate, visually-arresting example of data storytelling to include on their

web page. Each student also produces an "R Tip of the Day" as a way to explore new packages as well as features of presentations using Quarto. Before starting their final project, students spend time learning SQL (a popular request of employers). This helps them see how their knowledge of **dplyr** verbs in the **tidyverse** has set them up well to understand SQL logic. We are extremely grateful that our IT Department created a server to house a data warehouse that students can query, since publicly-available SQL databases are hard to find. Our students make their SQL queries within the RStudio environment in order to build custom data sets that they can examine with their data wrangling and visualization skills. Finally, students complete and present final projects in pairs—building **shiny** apps for interactive storytelling based on data they've already acquired and wrangled themselves.

All course materials are posted to a course web page; these materials are based on the open source textbooks by Wickham et al. (2023), Baumer et al. (2021), and Irizarry (2019), materials graciously shared by several co-authors on this paper, and homegrown materials. In the first edition of this course, we also attempted to cover networks, simulation, and spatial data, but these topics proved to be overly ambitious and were removed before the second edition. Simulation and spatial data, in particular, are both covered more fully in other courses available in the department.

2.4 Assessment of students

Most courses in the SDS program at St. Olaf have similar breakdowns of assessment methods, featuring meaningful weekly or bi-weekly homework sets, several exams that combine in-class and take-home components, and a final group project. Assessment in DS2, on the other hand, is heavily weighted toward a series of mini-projects preceding a final project. These mini-projects give students a chance to apply recently-developed skills to data and questions of their own choosing. They can decide which data they'd like to acquire and how they'd like to construct a **shiny** app to allow others to explore and gain insights. They create their own personal web page and then

build pages showing choropleth maps and text analyses of works and variables they selected. Two or three knowledge quizzes are built in to check understanding of technical details, and periodic homework is collected to encourage practice with new ideas. But most of the assessment is weighted toward projects that allow students to follow their passions and display their creativity and storytelling abilities.

2.5 Instructor reflections

By the fourth edition of SDS 264 at St. Olaf (Fall 2025), a more settled curriculum and other changes led to (mostly) positive student evaluations. Comments included:

> "[The] professor lets us do many projects on our own using lessons from class and to me, it's the best way to practice my skills but also combine my interests with those skills."

> "I had a lot of fun working on the projects and R Tip of the Day and felt very proud when presenting my projects."

> "Thank you so much for creating such an exciting class… I honestly learned more than I ever expected, [and] I feel like I am one step closer to becoming a real data scientist."

Besides removing several topics as discussed above, changes from the first to the fourth edition included waiting before creating Quarto (Scheidegger et al. 2022) websites to reduce the initial burden, intentionally teaching students how to resolve merge conflicts in GitHub, adding knowledge quizzes for short check-ins on technical details, improving the project rubrics and instruction to provide more direction, using our own server for SQL queries, and improving the class activities around data acquisition.

Future editions will enact much smaller changes, as recent students showed solid understanding of fundamentals of literate programming for data science through knowledge quizzes and project work, and they produced an impressive and creative

set of Shiny apps for their final projects. Final project topics included text analyses of how newspapers portrayed ethnic groups before and after Pearl Harbor, the presence and migratory movement of loons and geese, and strategies for profiting in a popular online farming simulation game.

Nevertheless, some content of SDS 264 can be a moving target as technology constantly changes. Since the first edition in Spring 2024, new and useful R packages have appeared, Quarto has become more mainstream for the creation of web pages and dashboards, government data sources and APIs became unreliable (sometimes overnight), and large language models have simplified many tasks, such as the creation of **shiny** apps. For professors whose formal training did not include many DS2 topics, keeping up-to-date can be a non-negligible challenge that is simultaneously exhilarating and exhausting.

3 A course that focuses on data storytelling

COMP/STAT 212: Intermediate Data Science at Macalester College[3] is envisioned as a critical bridge course between the introductory data science course (DS1) and the project-based capstone experience. Macalester's data science curriculum dates back to Fall 2017 with the debut of the DS1 course, which expanded upon a successful 1-credit "Data and Computing Fundamental" course developed by Daniel Kaplan. Simultaneously, the department launched its project-based capstone in data science during the same term. Despite early progress at the introductory and advanced levels, a significant gap remained for students to navigate without any intervening opportunity to further develop their technical skills. To address this, the Data Science major was intentionally designed around three core data science courses that gradually build proficiency and share a focus on iterative data storytelling.
These courses are designed to complement the existing Statistics, Mathematics, and Computer Science courses. Although the Data Science major officially launched in Fall 2021, the

[3]Macalester College is a private liberal arts college in Saint Paul, Minnesota, known for its academic excellence, internationalism, multiculturalism, and service to society.

implementation of this intermediate bridge was deferred until Fall 2023.

The intermediate data science (DS2) course builds upon and deepens the data visualization, data wrangling, and data communication skills developed in DS1. The curriculum challenges students to deconstruct the fundamental building blocks of visualizations, expanding their capacity for sophisticated and effective data storytelling. To support these goals, the course incorporates programming in R, which facilitates advanced data acquisition and more complex wrangling and visualization workflows. The course culminates in a final project where students create a digital artifact using at least two data sources. This project requires students to craft an honest and ethical narrative, resulting in a professional output suitable for a public portfolio or sharing with prospective employers.

3.1 Learning goals

Students completing DS2 at Macalester will be able to:

- Work collaboratively with Git/GitHub, and incorporate input to formulate and refine research questions that can be answered with data.
- Use current professional data science tools to acquire, wrangle, and visualize data accurately and ethically.
- Incorporate documentation and reproducibility in data acquisition, management, and analysis to promote transparency and accountability.
- Construct and present effective data narratives, ensuring that they effectively communicate the significance and limitations of the findings through various mediums, including static and interactive data visualizations, presentations, and written reports.

3.2 Prerequisites

The prerequisites for the course includes introductory data science (taught with R), a first course in computer science (taught with Python), and our first course in statistics (taught

with R). The course is required for both the Data Science major and minor and counts as an elective course for the Statistics and Computer Science majors.

3.3 Content

The course is structured to interweave the data science workflow topics with the technical topics. In particular, every class period starts with a data storytelling moment in which students engage with an existing data narrative and reflect on the story communicated, the concrete features that make it effective, and the ways it could be improved.

We start the semester with a discussion of file organization and an introduction to the basics of Git/GitHub, which are important tools for starting a new data project. We then review **ggplot2** functions and consider more customized visualizations by discussing a wider variety of `geom_*()` functions and spatial tools such as coordinate reference systems and functions in the **sf** package. Students attempt skill challenges individually on these topics and use GitHub Classroom to turn in their work. We then move on to advanced data wrangling and cover subtleties of the data types in the Transform Chapters of Wickham et al. (2023) with an emphasis on checking the assumptions you make about the data when you transform it to ensure accuracy.

We discuss missing data assumptions (e.g., missing at random, missing completely at random, and missing not at random), and how these assumptions impact the appropriate wrangling approaches and future data narratives. Demonstrating imputation techniques, such as those in the **mice** package, provides motivation for students to learn base R tools for iteration and custom functions.

By this point, students will have set up a collaborative environment with GitHub for their project work and practice using that tool with each other. We then move to programming in R with a focus on functions and iteration to automate and generalize data wrangling processes when applicable. These programming skills enable them to learn about and implement APIs, web

scraping, and databases as approaches to acquiring data for their projects.

We pair the presentation of interactive visualization tools from the **shiny** and **plotly** packages with a podcast with Cole Nussbaumer Knaflic, author of "Storytelling with Data: A Data Visualization Guide for Business Professionals" (Knaflic 2015), about the advantages and disadvantages of interactive tools over static graphics to facilitate effective data communication and storytelling.

The scaffolding around the project involves a variety of milestones to encourage regular attention and many opportunities for students to give and receive feedback for refining both the questions and the data narrative. The following are the project milestones:

- Propose Data Source + Data Context
- GitHub Repo Setup with File Organization + Codebook
- Draft Written Introduction and Initial Results
- Present a Visual + Peer Feedback
- Project Planning using GitHub Issues
- Code Review + Peer Feedback
- Final Presentation + Digital Product + Code/Data Documentation

3.4 Assessment of students

Student learning is assessed through a combination of individual and group work. Individually, students complete weekly homework challenges on the course topics to deepen their understanding, which is then formally assessed in three in-person concept quizzes. These quizzes were written to motivate students to internalize computational concepts and syntax patterns rather than relying on external support (e.g., Generative AI tools). In groups, individuals are expected to make individual `git` commits to project milestones and the digital artifact.

To cap off the course, students present the final digital artifact to their classmates in a poster session-style environment. Groups are expected to present a brief summary of their data narrative, using the digital artifact as the main visual prop, and to refine

their verbal communication by repeating their presentation multiple times as small audience groups rotate to view each project.

3.5 Instructor reflections

This course has been taught four times and the curriculum is continuously being refined. Students view this as a challenging course that touches on many useful topics. One student said:

> "I learned a lot of specific data science tools, like wrangling and iteration techniques, but also a lot of things that I think I'll use in life even if I don't go into data science as a career. Understanding URLs, how to access information on the internet, and make meaningful visualizations is a useful tool as a human and I really enjoyed learning about it. This class also made me grow as a student."

We have ordered topics so that the first part is a review and expansion of topics from the DS1 course, as students recognize that their understanding of visualization and wrangling deepens alongside their understanding of programming. One student commented:

> "I felt a huge spike in my learning after we had covered more of the coding basics and I really think that I could've understood more of how to do visualizations or wrangling if I had that knowledge."

Not only serving as a bridge to upper-level courses, this course offers a vital first opportunity for students to explicitly integrate their foundational knowledge in statistics, computer science, and data science. In one homework challenge, they use iteration and functions to process and clean hundreds of thousands of JSON files to facilitate a statistical exploration of the art objects in the Minneapolis Institute of Art.

Students particularly appreciated the structure of course activities and assignments, citing that these resources facilitated active group learning in the classroom and review outside of the classroom. One student said:

> "I liked the checkpoints and the class activities, as they helped me understand the concepts through interactive work and reading. The projects helped us put our learning to use with our own topics and datasets."

Generally, students felt that the course-long project experience facilitated good practices in terms of early idea generation, forming connections with teammates, and making incremental progress. They enjoyed the chance to pursue a topic that interested them and felt that the project gave them a chance to practice most of the core ideas in the course in a more complex setting.

4 A course that focuses on data science tools, interactive visualization, and text mining

Intermediate Data Science at Villanova University[4] was created due to a demand from students for a follow-up course to the popular Intro to Data Science (DS1) course (STAT 4380) and to better prepare them for industry jobs after graduation. The content for the course was identified through a mix of discussions with students, alumni, and data science practitioners, as well as discussion among peers, including the co-authors of this paper. There is an existing course on data mining and predictive analytics (a.k.a., statistical/machine learning) (STAT 4480), so modeling content was not considered for inclusion in course content.

The course was first taught in Fall 2023 as a special topics course and is being taught for a second time in Spring 2026. It was taught out of the Department of Mathematics and Statistics, where a Bachelor's in Statistics was first offered in 2020. At Villanova University, Data Science (or Analytics) is also taught in the Department of Computing Sciences, the College of Engineering, and the Business School, without substantial coordination.

[4]Villanova University is a medium-sized, comprehensive, Catholic university located outside of Philadelphia, PA.

4.1 Learning goals

The course, which uses R, was designed to teach students the following:

- Data Science Concepts
  - Understand the role of data science and its demand in society
  - Communicate results to technical and non-technical audiences
  - Discuss and implement good coding practices
  - Explore version control through Git/GitHub
  - Learn about SQL databases and how to interact with them through R
  - Explore methods for extracting data from additional formats / sources
  - Review Large Language Models and improve prompt engineering
- Data Visualization
  - Improve visualization choices using the gestalt of data visualization
  - Discuss dimension reduction and feature engineering
  - Create dynamic / interactive visualizations using data tables (**DT**), **plotly**, **leaflet**, and more
  - Create dashboards using **shiny**
- Working with Text
  - Develop (or refresh) skills to process strings
  - Perform text mining, including term frequencies, TF-IDF, and sentiment analysis

Given that Quarto had been released since these students had taken Intro to Data Science, an introduction to using Quarto was also covered in this course.

Brianna: there is a duplicate in Content section

The class periods involved a mixture of material presented by the instructor, in-class coding time for students, and student presentations. For most sections, a short review of the material from DS1 was covered before engaging with new material. R

and RStudio were run on individual machines. All course materials were posted on a course website that was protected by a firewall.

4.2 Prerequisites

The Intermediate Data Science course requires students to have taken Intro to Data Science (STAT 4380) or its equivalent, which covers data wrangling and data visualization using Baumer et al. (2021), and supplemental material on working with strings (Sanchez 2024), simulation, and data ethics. In turn, the DS1 Course requires students to have taken an introductory statistics course and have some programming experience, so those are hidden prerequisites for the DS2 course as well.

4.3 Content

Paul: there should be a Content section to parallel the other Sections 2-6, Brianna: I suggest taking the last few paragraphs from learning goals and expanding upon them in here; I also wonder if putting some of the assessment part in Content would [illegible] below. Is that sufficient? We could move the projects here and just list them below…

Given that Quarto had been released since these students had taken Intro to Data Science, an introduction to using Quarto was also covered in the first iteration of this course.

The class periods involved a mixture of material presented by the instructor, in-class coding time for students, and student presentations ("R Tip of the Day" and LLM mini-projects and the final project). For most sections, a short review of the material from DS1 was covered before engaging with new content. R and RStudio were run on individual machines. All course materials were posted on a course website that is protected by a firewall.

4.4 Assessment of students

Assessment in the class is done through class attendance and participation (10%), a final project (40%) and four mini-projects (50%). There are homework assignments throughout the semester that students are recommended to complete, but they are not graded.

The final project asks students to choose and define a question, as well as identify data sources that could be used to address

it. Multiple data sources are recommended, but not required. Students are required to get project topic approval from the professor, attend individual meetings with the professor at least once, and present to the class at the end of the semester.

The four mini-projects include interactive data visualization, text mining, an LLM project, and the "R Tip of the Day".

The interactive data visualization mini-project requires them to analyze the World Happiness Data and produce interactive tables (using **DT**) and graphs (using **plotly**).

The text mining mini-project has students read in the Harry Potter books and:

1. identify the most popular house based on word frequency,
2. identify the frequency in each book of "Voldemort", both with and without inclusion of "he-who-must-not-be-named" as a substitute for this name,
3. use Term Frequency - Inverse Document Frequency across the books to identify the most unique words in each book, and
4. perform sentiment analyses across books to see which one was the most negative.

The LLM mini-project engages students (and the professor) in an early use of LLMs. Groups of students are assigned different LLMs and asked to duplicate code that we had previously explored in class together. They present to the rest of the class on how well the LLM did at duplicating the code, how often it hallucinated, and what they had to do to make the code work.

The last mini-project is the "R Tip of the Day" (RTD). This is designed to expose students to other packages or skills that are useful for working with data, but are not covered elsewhere. For the first part of the course, on most days, the professor presents an RTD. For example, keyboard shortcuts, the R Graph Gallery, and more about YAML, as well as packages like **corrplot**, **kableExtra**, **ggrepel**, and **gridExtra**. Students are asked to identify an RTD that they would present later in the semester (a list of suggestions were given to help them identify one). The topics chosen by previous students have included simulation and **ggplot2** themes as well as packages including **janitor**,

patchwork, **naniar**, **hoopR**, **extrafont**, **scales**, **ggthemes**, **ggridges**, **ggThemeAssist**, **leaflet**, **likert**, **esquisse**, and **geniusr**.

Ben: maybe cut and/or condense this list of packages?

4.5 Instructor reflections

The Intermediate Data Science course was taught at Villanova University in Fall 2023. The course consisted of 15 students who were primarily statistics majors. The aspects of the course that most helped with student learning (according to course and teacher evaluations) were live coding in class together and time given to students to code for themselves in order to absorb the material. Two challenges from a teaching perspective, which are common for new content classes in fields like data science, were the work required to stay one step ahead of students as well as the lack of knowledge of the full usage and coding landscape required to answer questions and frame the material appropriately.

The students seemed to enjoy the course and their ability to tackle more complex problems as well as learn skills that might be useful as a practicing data scientist. One student commented:

> "The combination of text mining and interactive visualization really opened my eyes to what's possible with data science. I never thought I could build something like a Shiny dashboard before this class."

Some students have gone on to submit their work from the class to local and national data visualization competitions, including the winner of our local (Villanova) one. Other students have expressed appreciation for continued exposure to coding in R that helped deepen their understanding of how to tackle real-world data science tasks. Another student reflected:

> "Learning about LLMs and how to use them effectively for coding was incredibly valuable. It's definitely a skill I'll use in my career, but now I understand both their potential and limitations."

5 A course that focuses on developing an R package

Brianna: how does this fit within the curriculum more broadly? to mirror the content in the other sections, it would be useful to have a paragraph about that first and then a quick summary

SDS 270 at Smith College[5] focuses on a clear outcome for all students: to build a usable R package that is hosted on GitHub. Taking inspiration from Reinhart and Genovese (2020), course content during the first half of the semester covers roughly the first half of the material presented by Wickham (2019), which deepens students' understanding of how R works. Unlike in the first course in data science, where the emphasis is on rapid development of coding skills, in this second course students learn about computer science concepts like object types, S3 classes and methods, and environments. During the second half of the semester, students work in groups to develop an original R package, following advice provided by Wickham and Bryan (2023). All completed packages must pass an R CMD check.

Students use modern software development workflows, including the extensive use of Git/GitHub throughout the semester. While students are introduced to GitHub in our DS1, this course prepares students for the mission-critical collaborative work they will see in the senior capstone course.

5.1 Learning goals

The learning goals for the course include:

- Contribute to an open-source software project using version control
- Write a robust, encapsulated software package in the R programming language
- Write and debug sophisticated functions in R

Unlike all other courses in the statistics and data science curriculum, this course does not explicitly require students to perform data analysis, data wrangling, or data visualization. Instead, students are forced to grapple with the limitations of their

[5] Smith College is a private liberal arts women's college in Northampton, Massachusetts, renowned for its rigorous academic programs and pioneering work in interdisciplinary studies.

understanding of how R actually works, and to build mental models that will help them with more advanced projects in R, and undoubtedly other programming languages.

5.2 Prerequisites

The prerequisites for the course include a first course in data science (taught using R at Smith) and a first course in computer science (taught in Python at Smith). The course is one of several that satisfies the programming depth requirement for the SDS major, and has become the most popular option.

Brianna: this sentence about how it fits within the major could go in the intro of the section

5.3 Content

SDS 270 is divided into two parts. In the first half of the course, students work through roughly the first half of the material presented by Wickham (2019). For most students, it's the first time that they are asked to approach R as a programming language in a manner similar to how a computer scientist (and *not* a data analyst) might. Early emphasis is on a careful examination of object types and coercion rules. This is followed by a treatment of object-oriented programming using the S3 system. This material builds crucial understanding of generic functions and method dispatch that some students will develop further in the process of writing their R package. Improving students' ability to write robust, well-documented functions, and how to iterate them over a vector (or list) of values is another goal (and one perhaps most germane to data science). Conditional execution, error handling, and environments are also discussed.

In the second half of the course, students work in groups to develop a viable R package on GitHub, following advice provided by Wickham and Bryan (2023). While all students must submit a package proposal and have it approved by the instructor, students are largely free to use their creativity to explore the space of possibilities. As such, these packages run the gamut from those that are more focused on a specific data source (or API), to those like **ggRtsy** that, for example, provide **ggplot2**

color palettes based on the works of the Dutch master Vincent Van Gogh (Diaz et al. 2025). All completed packages must pass an R CMD check via GitHub Actions. Each semester, one or two packages are submitted and accepted to CRAN, despite that goal being beyond the scope of the course.

Brianna: do we need this? we have a whole section for it now, right?

SDS 271 is a recent addition to the curriculum designed to offer a second course in data science in Python. The prerequisites and the position of the course in the curriculum are the same as SDS 270, but the learning goals and content of the course obviously differ. Since our first course in data science is taught in R, while the first course in computer science is taught in Python, students should be prepared to take either (or both) course(s). However, since students work with data in R intensively in our first course in data science, SDS 270 pivots toward developing computer science skills in R in the second course. Conversely, SDS 271 provides a place in the curriculum for students to deepen their data wrangling and data visualization skills *in Python*.

5.4 Assessment of students

Students in SDS 270 are assessed using a variety of common methods: reading quizzes, homework assignments, in-class labs, etc. Since the course is organized so specifically around the creation of an R package, that assignment provides the clearest insight into what the students are capable of, and accordingly bears heavily on the student's grade. We use rubrics to assess the projects. This also includes feedback from automated code checkers (e.g., `R CMD check`, **styler**) and continuous integration tools (e.g., GitHub Actions).

5.5 Instructor reflections

While many of the packages that students create are relatively simple, a few have continued to lead lives of their own beyond the classroom. One notable package is `nwslr`, which provides tools for working with data from the National Women's Soccer League. The package was created by two students in the first version of the course, taught in the fall of 2019. While both

students have gone on to successful careers in data science, one of them (Arielle Dror '20) is now the Director of Data & Analytics for the Bay FC, in no small part because of the interest and ability that she demonstrated in creating the package.

Two refrains appear most commonly in student reflections about their experience in SDS 270. The first is a profound sense of accomplishment, as they have learned a great deal about R. Students often report that even though they did well in their DS1 course, they now realize just how shallow their understanding of R actually was, and how grateful they are to have had the opportunity to deepen it. The second is a revelation that R is "real programming language". Students are able to connect ideas from object-oriented programming (e.g., classes and methods) that they learned in their computer science classes to R, and apply them in context. This leads to a deeper appreciation for R as a programming language, and a reinforcement of their ability to program in *any* high-level programming language.

6 A course that focuses on developing a Python package

Brianna: how does this fit within the curriculum more broadly and interact/compliment with 270? to mirror the other course, it would be useful to start with this.

SDS 271 at Smith College covers the skills and tools needed to process, analyze, and visualize data in Python. Topics include functional and object-oriented programming in Python, data wrangling in NumPy (Harris et al. 2020) and Pandas (The pandas development team 2020; McKinney 2010), visualization in Matplotlib (Hunter 2007) and Seaborn (Waskom 2021), and statistics with SciPy (Virtanen et al. 2020). The ultimate goal of the course is to build a usable Python package hosted on GitHub. Students learn to debug, test, and document their code, as well as how to code collaboratively and effectively use version control.

The course covers much of the content in McKinney (2022), as well as VanderPlas (2022).

6.1 Learning goals

The learning goals for the course include:

- Run Python in Jupyter notebooks and from the command line
- Understand data types, operators, control statements, and loops in Python
- Interpret error messages and debugging code efficiently
- Use and write functions in Python
- Object-oriented programming in Python, including understanding and writing classes
- Use NumPy and Pandas to organize, manage, and manipulate data
- Visualize data clearly and effectively with Matplotlib and Seaborn
- Perform basic fitting and statistics with Python
- Code collaboratively with Github
- Write clear code documentation and useful error messages
- Create a complete Python package that is thoroughly unit-tested and well-documented

6.2 Prerequisites

The prerequisites for the course include a first course in data science (taught using R at Smith) and a first course in computer science (taught in Python at Smith).

The course is one of several that satisfies the programming depth requirement for the SDS major.

6.3 Content

SDS 271 begins with a review of basic Python: data types, arithmetic, operators, conditionals, and loops. It then covers writing and calling functions, including the use of keyword and positional arguments, as well as how to write appropriate docstrings and error messages within functions.

The course then introduces NumPy, and extensively covers data manipulation, including reading and writing files and working

with arrays (e.g. indexing, slicing, masking, array arithmetic, etc.). The course subsequently introduces the Pandas library, and students learn to work with data in Pandas data frames.

Students then learn to visualize data with Matplotlib and Seaborn. They are introduced to various plotting methods and styles, and focus on visualizing data clearly and effectively.

Students then learn to fit lines, polynomials, and user-defined functions to their data using NumPy and SciPy, as well as how to compute basic goodness-of-fit metrics with SciPy. They are also introduced to random number generators in NumPy, and learn how to generate probability distributions, draw from these distributions, and use these distributions to compute p-values. Students also practice using random number generators to perform Monte Carlo simulations.

Students are then given a more in-depth explanation of object-oriented programming, and are introduced to classes in Python. Students learn to create classes with built-in attributes and methods. Students are then introduced to the concept of a Python package, and ultimately work in groups to design a user-friendly package that contains several classes and methods. Students are free to design a package that does whatever they would like, although they must propose their idea to the instructor to receive feedback on its feasibility. Students learn to code collaboratively with Git/Github, and their final packages must be well-documented and thoroughly unit-tested.

For the first half of the course, students mainly work in Jupyter notebooks, but when creating their packages students learn to run Python from the command line. They learn about package organization and file structure, and how to run unit tests from the command line with pytest (Krekel et al. 2004). Students also learn to use `git` from the command line, and are required to create branches and submit pull requests to their repositories.

6.4 Assessment of students

Students in SDS 271 are assessed using a variety of methods: homework assignments, projects, in-class presentations, and oral exams (Theobold 2021).

The mid-semester project is a data wrangling and data visualization project, while the final project is the creation of a Python package. Both of these are partner or group projects, and involve short in-class presentations. The first presentation focuses on the creation of effective plots for a chosen dataset, while the second presentation focuses on a live demonstration of the finished Python package.

Students are also assessed based on two fifteen-minute oral exams (one midterm and one final). During this exam, students meet with the instructor one-on-one and work in a Jupyter notebook on the instructor's computer. Students are asked to perform a few basic operations in Python, and are then asked to explain a piece of code that they wrote for a homework assignment (that the instructor has chosen). Students are asked to explain why they approached the problem the way that they did, and what would happen if they changed certain lines of code.

6.5 Instructor reflections

Many students were surprised at how much they had learned by the end of the semester, and were excited that they were able to build their own Python packages from scratch. Students reported having much more confidence working with Python by the end of the semester compared to how they felt after their introductory computer science course. Several students commented that they had so much fun working on their final projects that they want to continue building their packages on their own time.

7 Summary

7.1 Analysis of topic coverage

We analyze the content of our DS2 courses by cataloging a superset of learning outcomes, and the extent to which our courses cover them. First, we each contributed to a master list of specific learning outcomes that we include in our courses. Next,

Paul: We should remind readers of the goals and products associated with the MADSER project.

we mapped these learning outcomes to the topics, subtopics, and competency concepts created as part of the MASDER project's efforts (Posner et al. 2024). This tree-based structure helps to situate the DS2 topics we teach within the larger framework of the data science curriculum. Finally, for each learning outcome, we determined the extent to which the topic is covered by each of our courses. We used the coding system from Beckman et al. (2021) and reproduced in Table 1 to indicate how essential various learning outcomes are to each course.

Table 1: Levels for the treatment of various topics across the different courses.

Level	Label	Explanation
	None	This is not included in course.
	Incidental	This may or may not occur in the course.
	Teacher	This is demonstrated to students, but they are not necessarily expected to do it independently.
	Student	Students are expected to do this independently, but it may not be formally assessed.
	Assessed	Students are expected to do this independently, AND will be assessed for proficiency.

Source: Article Notebook

7.1.1 Competencies with shared emphasis

Nick: this section needs to be updated once the analysis has been repeated without the Amherst College course

Across the five courses we have described, there was the greatest emphasis in data science skills, data processing, and data visualization. Table 2 summarizes the competency concepts and learning outcomes that received the greatest shared emphasis across our courses. Specifically, Table 2 shows the data science learning outcomes covered at the Student or Assessed level (i.e., a score of 3 or 4, see Table 1) by at least four of our courses (aggregated at the subtopic level), along with their respective

Table 2: Summary of data science learning outcomes covered by at least three of our five DS2 courses to at least level 3.

Dspt Sub Topic	Dspt Competency Concepts
Knowledge (CF)	Computational Thinking, Data Structure
Skill (CF)	Coding Practices, Data Processing (CF)
Acquisition	Find Data, Import Data, Web Scraping
Cleaning	Fix, Identify
Wrangling	Combine, Feature Engineering, Reshape, Subset
Method (DSS)	DS-Specific Programming, Workflow
Professional Tools	Collaboration, Communication, Continuing Education
Knowledge (DSS)	Exploratory Data Analysis (DSS), Problem-solving Process, Subject Matter Expertis
Create	Dynamic, Exploratory Data Analysis (Viz), Multi-dimensional, Select, Static

topics, subtopics, and competency concepts. While this measurement is not precise, it does reveal the general areas which receive more emphasis.

Foundational coding processes and data science-specific programming and workflows (particularly those centered around Git/GitHub) stand out as the competency concepts with many learning outcomes that receive a relatively high level of emphasis, with 18 learning outcomes receiving an average treatment of XX (on a scale of 0–4). Data processing skills (i.e., data wrangling) involving merging, nesting, and subsetting data, writing functions, importing data, or using regular expressions are also noteworthy. We also note that several data visualization learning outcomes—mostly centered around interactive displays—receive heavy emphasis.

Nick: need to check numbers below once analysis is redone without Amherst: is four courses still the best cutoff?

Nick: need to check numbers below once analysis is redone without Amherst: is four courses still the best cutoff? Brianna: I tried updating the table with 3 courses (over half) and adding sd of rating rather than the number of outcomes; the outcomes are listed so that information is present

Source: Article Notebook

Considering the full set of data (see Table 5), there is considerable agreement in content and learning outcomes around using standard programming concepts such as control flow, iteration, data structure, and writing and using functions for data science applications. Whether students are learning these programming concepts for the first time or learning them in a new programming language, these second data science courses focus on programming for data processing and analysis. Additionally, these courses raise student expectations when working with complex data structures (e.g., data frames, lists, arrays, and nested structures). There is also agreement in emphasizing version control in the data science workflow through the use of basic Git/GitHub skills (e.g., commit, push, pull). Other common topics include advanced data visualizations such as static choropleth maps or interactive Shiny apps, advanced data processing with strings and regular expressions, and data communication by iterating on a data story through feedback.

Paul: The "differences" section below is just as long and detailed as the "similarities" section above. I think we want to emphasize the similarities, especially if the differences are mainly details from a single course. Can we make the differences section a more general overview of one-off topics, with just enough detail so readers will understand the general ideas?

7.1.2 Competencies with variable emphasis

At the same time, we observe the greatest heterogeneity in topic coverage in different aspects of data science skills and data processing, as well as statistical foundations and modeling. Table 3 summarizes the same information as Table 2, but for learning outcomes that were covered in exactly one of our courses.

Brianna: the mean doesn't make sense to me here when the data is so skewed, perhaps all we need is a list of learning outcomes that are only covered by 1 or 2 courses

Source: Article Notebook

The variance in topic coverage stems from the diversity of end products, overall goals, prerequisites, and other offerings at the institution. The Smith courses frame some of the learning outcomes around creating a package, while St. Olaf, Macalester, and Villanova focus more on communicating understanding from analyzing a potentially non-standard dataset. Understanding text as data holds an important role in St. Olaf and Villanova's class. Conversely, advanced Git/GitHub tasks (e.g. forks, branches, pull requests) and skills and concepts needed for package development (Model Deployment) are only found in the Smith courses.

Table 3: Summary of data science learning outcomes covered by at least of our DS2 courses to at least level 3 while not covered at all by at least 2.

Dspt Sub Topic	Dspt Competency Concepts	Learning Outc
Knowledge (CF)	Data Management, Data Structure	store larger da
Skill (CF)	Data Processing (CF), Scalability	pass argument
Stewardship	Governance, Lineage	determine per
Acquisition	Create Structure, Extract from Database, Simulation, Web Scraping	tokenize corpo
Method (DSS)	DS-Specific Programming, Model Deployment, Workflow	use iteration f
Knowledge (DSS)	Ethics	scrape data et
Professional Tools	Communication	analyze data v
Create	Dynamic, Static	create interact
Unsupervised	Text Analysis	conduct text a
Knowledge (SF)	Statistical Thinking	identify bias fr

In Table 3, we see data science-specific programming and data processing concepts that are more advanced than those in Table 2. For example, many of the items in Table 3 refer to concepts like variable scoping, function environments and methods, and code deployment that are more relevant in a software development context than a typical treatment of data wrangling. Similarly, Table 3 also reveals Git/GitHub concepts (e.g., GitHub Actions) that are more advanced than the beginner push/pull workflow. This is possible if students have previous exposure to programming concepts through a prerequisite course.

Lastly, most of our institutions touch on the concepts in Table 3 in a separate course such as data visualization, statistical modeling, data structures, or object-oriented programming.

Overall, the depth of programming skills developed (e.g., iteration, object-oriented programming) among courses depends importantly on the end-of-semester deliverable and what is needed to complete that project.

Paul: - some differences emerge from (a) other courses offered at each institution, (b) prerequisites, and (c) faculty expertise and training. For example, St. Olaf has a separate data viz course, which could explain why it covers data viz less deeply than a fewer other courses. Also, St. Olaf has no CS or stats prerequisite, which means many students are seeing loops, functions, and GitHub for the first time, impacting the speed at which we can move. Brianna: I think this is now addressed in the text. Brianna: this line might need adjustment

7.1.3 Omitted competencies

In terms of the topical tree structure of Posner et al. (2024), our courses contain little to no coverage of the topics shown in Table 4. As noted earlier, mathematical foundations (e.g., calculus, linear algebra, and probability) and statistical modeling principles are covered elsewhere in the curriculum (in a sequence of mathematics courses, and in a statistical modeling and/or machine learning course, respectively). Other omitted topics include even more advanced use of Git/GitHub (e.g., reviewing and merging a pull request), use of UNIX command line tools for navigating the file system, and code benchmarking.

Source: Article Notebook

We are somewhat dismayed to note that data stewardship was also a largely omitted topic. We affirm that ideas surrounding data privacy, security, governance, licensing, and lineage are important, and while some of these notions arise during our

Table 4: Data science learning outcomes with little to no coverage across our DS2 courses.

Dspt Sub Topic	Dspt Competency Concepts	Learning Outcomes
Stewardship	Security	who has access to the
Knowledge (MF)	Calculus	Calculus: the use of d
Classification	Algorithmic, Logistic Regression	Algorithmic: relating
Diagnostics	Consistency, Crossvalidation, Precision / Accuracy	Consistency (across v
Structure	Hierarchical, Uncorrelated	Hierarchical: the qual
Tuning	Bagging, Boosting, Bias-Variance Trade-Off, Model Packaging	Bagging, Boosting: te

various student project experiences, it is revealing that those topics appear in Table 4.

7.2 Relationships to other visions for DS2

While several authors and panels of experts have articulated program-level goals for undergraduate data science curricula, De Veaux et al. (2017) were among the first and only commentators to offer details on the composition and learning goals of specific courses in an undergraduate data science program. De Veaux et al. (2017) imagined a curriculum for a data science major that was freed from the constraints of departmental turf wars and existing curricula that reflect the reality at most institutions.[6] Their vision included four foundational courses—a two-course sequence in mathematics specifically for data science and a two-course sequence in data science itself—opening up into upper-level courses ranging from statistical modeling and machine learning to algorithms and data curation, and culminating

[6]We note that at Macalester and St. Olaf, mathematics, computer science, statistics, and data science are all housed in one department.

in a capstone experience. The envisioned two-course sequence in mathematics would combine elements of several traditional courses from the undergraduate mathematics curriculum (e.g., calculus, linear algebra, geometry, and probability), while the two-course sequence in data science would help establish data science as its own field, with its own unique foundational structure.

In the discipline-free vision of De Veaux et al. (2017), the DS1 course has many elements of the courses we described in Section 1.1, although they de-emphasized data wrangling in favor of the inclusion of ideas in modeling, simulation, and algorithmic thinking. But key learning goals still include "exploring and wrangling data" and "summarizing, visualizing, and analyzing data," with "R recommended" as the high-level programming language. They then envisioned a DS2 course that would "help students understand the thinking and structure behind the higher-level functions they experienced in [DS1]." Their learning goals include interacting with many different data types, sources, and interfaces, and building structure from various data forms for analysis. Many of the DS2 courses described in this paper share these objectives with topics such as text as data, regular expressions, SQL, and data acquisition.

However, the De Veaux et al. (2017) vision for DS2 also includes "an introduction to the elementary notions in estimation, prediction, and inference," in addition to study design and data collection. While some data science 2 courses might appropriate touch on these topics, the courses described here leave these ideas to other statistics courses in their curricula, choosing instead to focus on literate programming (St. Olaf), data storytelling (Macalester), LLMs and more visualization (Villanova), and developing an R package (Smith). In their own way, each course helps students develop skills on the computational end of the data science spectrum, rather than focus on statistical inference.

De Veaux et al. (2017) acknowledged that the courses they outlined "are clearly bold steps toward a new integrated program in data science," and expected many iterations. We share the desire for a two-course sequence in data science that will provide solid grounding for upper-level courses, and we offer our courses

as potential iterations of the De Veaux et al. (2017) vision that have fit well within our local contexts.

An alternative approach undertaken at Amherst College has been to offer a capstone course to cover many of the topics we have described in this paper. STAT 495 (Advanced Data Analysis), which has been taught since 2014, was developed to serve as the culminating experience for students in the Statistics major, helping them integrate the theoretical and applied knowledge they have acquired throughout their coursework. Within the course, students are exposed to advanced statistical methods and undertake the analysis and interpretation of complex and real-world datasets that go beyond textbook problems. Course topics and structures vary from year to year depending on the instructor and selected case studies. While STAT 495 is not a DS2 course per se, it shares some of the same learning goals and topics described in this paper, with two overarching goals: 1) students engage with advanced statistical methods, and 2) students complete multiple project-based data challenges (some group, some individual). While there are many attractive aspects of such a capstone, it cannot fully substitute for a 200-level DS2 course that can provide all students with the deeper data acumen skills they need to thrive in later courses and experiences.

8 Conclusion

8.1 Discussion

Over a decade and a half ago, Nolan and Temple Lang (2010) articulated a vision for more computing in statistics (and data science) curricula. Their three key components that should be integrated into statistics and data science curricula include:

1. "broaden statistical computing" (e.g., data wrangling, acquisition, visualization, and alternative formats such as text);
2. "deepen computational reasoning and literacy" (e.g., programming concepts, algorithms, functions, debugging, and data structures); and,

3. "compute with data in practice of statistics" (e.g., problem solving with real data and alternative approaches to statistical thinking).

We see the DS2 courses described in this paper as manifestations of Nolan and Temple Lang's vision, where the common themes described in Section 7 are well-captured by their three components of computing. Component 1 includes the DS1 prerequisites for most of the five courses, as well as expansions into more advanced data visualizations and alternative data formats. Component 2 includes writing functions for data acquisition and processing, more advanced data structures, and good coding practices. Component 3 includes communicating a story about data the students have gathered and cleaned, and also managing workflow using version control software.

Starting with this shared canvas, the five courses in this paper then paint their own pictures. From developing packages to diving deeper into GitHub to acquiring data using SQL to telling stories through text analysis, these examples show that DS2 courses can be customized to students' backgrounds, instructor passions, and program-wide curricula.

8.2 Limitations

While we have provided five examples of DS2 courses that work at four specific institutions, it is challenging to outline a single course that will work in all situations. In fact, we hope that these courses can motivate other instructors to craft exciting DS2 courses at their institutions, and that those courses will help students achieve learning outcomes that are appropriate for their program. We have collected a set of topics that emerged independently; this list could be considered a first draft of key concepts for a second core course in the emerging discipline of data science.

We hope and expect that this list will change as more and more voices enter the conversation to discuss ideas they have developed and courses they have created. It's possible that we will learn that producing a "timeless" list of core topics for a quickly-changing field like data science may be a Sisyphean task,

but we believe there is value in the thought process and the attempt.

We also hope that future data science instructors and researchers will produce more formal assessment methods to measure the gains of students taking DS2 courses. Currently, we rely primarily on anecdotal evidence to assess the success of early versions of our courses.

Finally, we acknowledge the disruption that generative AI might cause to our core learning outcomes. Several recent papers (Ellis and Slade 2023; DeLuca et al. 2025) address the pressures of teaching students computing and programming concepts in the presence of tools like ChatGPT. While we believe that generative AI has already changed some of the ways that students work, it won't change the value of their fundamental knowledge of computing with data. After graduating, our students will still have to understand how code works, why it isn't working as desired, and how it can be fixed. In addition, they will have to be able to tell the stories that emerge from data and use insights gained to make important decisions.

8.3 Future work

If we are able to form a consensus as a discipline about key foundational ideas that every data science student should internalize in their early coursework, then we can develop course materials that will enable undergraduate instructors in many settings to teach effectively, regardless of their background. De Veaux et al. (2017) suggested that "resources for faculty including notes, examples, case studies, and—perhaps most importantly—new textbooks will be essential" for the effectiveness of a two-course sequence as the foundation of a data science curriculum. We also agree with De Veaux et al. (2017) that a common core sequence combined with "interactions [with two-year colleges and high schools] will be crucial in order to coordinate courses and instruction to facilitate transfer to four-year institutions."

We note that most of the courses described here were developed after the introductory and final courses in our data science sequences had already been developed. As such, the topics and

learning goals that have emerged as common themes across these courses provide clarity on core skills and topics that fill gaps in understanding between first and final courses in data science sequences. While particular topics or applications may vary, they are chosen with a specific set of learning outcomes in mind. Some of the courses in this paper represent a "grab bag of topics" chosen to customize the learning experience for students while establishing a fixed knowledge base. This stands in contrast to data science curricula which are organized as a "grab bag of courses," providing little core data science-specific material that can be counted on across (and often even within) institutions.

We can start by encouraging instructors developing their own DS2 courses to make their materials publicly available as much as possible. Then, as many others jump in to experiment, modify, and add their voices to the conversation, we can grow as a community toward the development of a common core in data science, while still allowing great flexibility and creative freedom in designing student experiences. At the end of the day, we hope that the courses described in this paper can serve as early, but hopefully important, steps toward a core sequence for data science undergraduate students that emerges as the discipline itself matures.

References

Baumer, Ben. 2015. "A Data Science Course for Undergraduates: Thinking with Data." *The American Statistician* 69 (4): 334–42. https://doi.org/10.1080/00031305.2015.1081105.

Baumer, Benjamin S., and Nicholas J. Horton. 2023. "Data Science Transfer Pathways from Associate's to Bachelor's Programs." *Harvard Data Science Review* 5 (1). https://doi.org/10.1162/99608f92.e2720e81.

Baumer, Benjamin S., Daniel T. Kaplan, and Nicholas J. Horton. 2021. *Modern Data Science with R.* 2nd ed. Chapman and Hall/CRC Press. https:

//www.routledge.com/Modern-Data-Science-with-R/Baumer-Kaplan-Horton/p/book/9780367191498.

Beckman, Matthew D, Mine Çetinkaya-Rundel, Nicholas J Horton, Colin W Rundel, Adam J Sullivan, and Maria Tackett. 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): S132–44. https://doi.org/10.1080/10691898.2020.1848485.

Bishop, Christopher M, and Nasser M Nasrabadi. 2006. *Pattern Recognition and Machine Learning*. Vol. 4. Springer.

Blair, Jean RS, Lawrence Jones, Paul Leidig, Scott Murray, Rajendra K Raj, and Carol J Romanowski. 2021. "Establishing ABET Accreditation Criteria for Data Science." *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 535–40. https://doi.org/10.1145/3408877.3432445.

Cannon, Ann R, George W Cobb, Bradley A Hartlaub, et al. 2019. *STAT2: Building Models for a World of Data*. 2nd ed. New York: W.H. Freeman.

Carver, Robert, Michelle Everson, John Gabrosek, et al. 2016. *Guidelines for Assessment and Instruction in Statistics Education (GAISE) College Report 2016*. American Statistical Association; AMSTAT. https://www.amstat.org/asa/files/pdfs/GAISE/GaiseCollege_Full.pdf.

CC2020 Task Force. 2020. *Computing Curricula 2020: Paradigms for Global Computing Education*. Association for Computing Machinery. https://doi.org/10.1145/3467967.

Çetinkaya-Rundel, Mine, and Victoria Ellison. 2021. "A Fresh Look at Introductory Data Science." *Journal of Statistics and Data Science Education* 29 (sup1): S16–26. https://doi.org/10.1080/10691898.2020.1804497.

Cobb, George W. 1998. *Introduction to Design and Analysis of*

Experiments. John Wiley & Sons.

Danyluk, Andrea, and Paul Leidig. 2021. *Computing Competencies for Undergraduate Data Science Curricula: ACM Data Science Task Force*. Association for Computing Machinery. https://doi.org/10.1145/3453538.

De Veaux, Richard D., Mahesh Agarwal, Maia Averett, et al. 2017. "Curriculum Guidelines for Undergraduate Programs in Data Science." *Annual Review of Statistics and Its Application* 4 (1): 1–16. https://doi.org/10.1146/annurev-statistics-060116-053930.

DeLuca, Laura S, Alex Reinhart, Gordon Weinberg, Michael Laudenbach, Sydney Miller, and David West Brown. 2025. "Developing Students' Statistical Expertise Through Writing in the Age of AI." *Journal of Statistics and Data Science Education*, 1–13. https://doi.org/10.1080/26939169.2025.2497547.

Diaz, Katelyn, Silas Weden, and Tess Goldmann. 2025. *ggRtsy: Add Some van Gogh Colors and Overlay Colors on Your Ggplot()*.

Donoho, David. 2017. "50 Years of Data Science." *Journal of Computational and Graphical Statistics* 26 (4): 745–66. https://doi.org/10.1080/10618600.2017.1384734.

Ellis, Amanda R, and Emily Slade. 2023. "A New Era of Learning: Considerations for ChatGPT as a Tool to Enhance Statistics and Data Science Education." *Journal of Statistics and Data Science Education* 31 (2): 128–33. https://doi.org/10.1080/26939169.2023.2223609.

Hardin, J., R. Hoerl, N. J. Horton, et al. 2015. "Data Science in Statistics Curricula: Preparing Students to 'Think with Data'." *The American Statistician* 69 (4): 343–53. https://doi.org/10.1080/00031305.2015.1077729.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, et al. 2020. "Array Programming with NumPy." *Nature* 585

(7825): 357–62. https://doi.org/10.1038/s41586-020-2649-2.

Hicks, Stephanie C, and Rafael A Irizarry. 2018. "A Guide to Teaching Data Science." *The American Statistician* 72 (4): 382–91. https://doi.org/10.1080/00031305.2017.1356747.

Hunter, J. D. 2007. "Matplotlib: A 2D Graphics Environment." *Computing in Science & Engineering* 9 (3): 90–95. https://doi.org/10.1109/MCSE.2007.55.

Irizarry, Rafael A. 2019. *Introduction to Data Science: Data Analysis and Prediction Algorithms with R*. Chapman and Hall/CRC Press. https://doi.org/10.1201/9780429341830.

Ismay, Chester, and Albert Y Kim. 2019. *Statistical Inference via Data Science: A ModernDive into r and the Tidyverse*. New York: Chapman; Hall/CRC. https://doi.org/10.1201/9780367409913.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning*. Vol. 112. Springer.

Knaflic, Cole Nussbaumer. 2015. *Storytelling with Data: A Data Visualization Guide for Business Professionals*. Wiley.

Knuth, Donald Ervin. 1984. "Literate Programming." *The Computer Journal* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.

Krekel, Holger, Bruno Oliveira, Ronny Pfannschmidt, Floris Bruynooghe, Brianna Laugher, and Florian Bruhin. 2004. *Pytest x.y*. Https://github.com/pytest-dev/pytest.

Kuiper, Shonda, and Jeff Sklar. 2012. *Practicing Statistics: Guided Investigations for the Second Course*. Pearson Higher Ed.

McKinney, Wes. 2010. "Data Structures for Statistical Computing in Python." In *Proceedings of the 9th Python in Science Conference*, edited by Stéfan van der Walt and Jarrod Mill-

man. https://doi.org/ 10.25080/Majora-92bf1922-00a .

McKinney, Wes. 2022. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter*. 3rd ed. O'Reilly Media, Inc. https://wesmckinney.com/book.

National Academies of Sciences, Engineering, and Medicine. 2018. *Data Science for Undergraduates: Opportunities and Options*. National Academies Press. https://nas.org/envisioningds.

Nolan, Deborah, and Duncan Temple Lang. 2010. "Computing in the Statistics Curricula." *The American Statistician* 64 (2): 97–107. https://doi.org/10.1198/tast.2010.09132.

Posner, Michael A., and April Kerby-Helm. 2025. *The Landscape of Introductory Data Science Courses*. https://ww3.aievolution.com/JSMAnnual2025/Events/viewEv?ev=4272.

Posner, Michael A., April Kerby-Helm, Alana Unfried, Douglas Whitaker, Marjorie E. Bond, and Leyla Batakci. 2024. "A Cross-Disciplinary Review of Introductory Undergraduate Data Science Course Content." *Proceedings of the 55th ACM Technical Symposium on Computer Science Education v. 2* (New York, NY, USA), SIGCSE 2024, 1937. https://doi.org/10.1145/3626253.3635352.

Ramsey, Fred, and Daniel Schafer. 2013. *The Statistical Sleuth: A Course in Methods of Data Analysis*. 3rd ed. Brooks/Cole Publishing Co.

Reinhart, Alex, and Christopher R Genovese. 2020. "Expanding the Scope of Statistical Computing: Training Statisticians to Be Software Engineers." *Journal of Statistics Education*, 1–23. https://doi.org/10.1080/10691898.2020.1845109.

Roback, Paul, and Julie Legler. 2021. *Beyond Multiple Linear Regression: Applied Generalized Linear Models and Multilevel Models in R*. Chapman and Hall/CRC Press. https://doi.org/10.1201/9780429066665.

Sanchez, Gaston. 2024. *R for Strings: Handling Strings with R*. 3rd ed. Gaston Sanchez. https://www.gastonsanchez.com/R-for-strings/.

Scheidegger, Carlos, Gordon Woodhull, Christophe Dervieux, Charles Teague, J. J. Allaire, and Yihui Xie. 2022. *Quarto*. Posit, PBC. https://quarto.org/.

Schumacher, Carol S., and Martha J. Siegel. 2015. *2015 CUPM Curriculum Guide to Majors in the Mathematical Sciences*. The Mathematical Association of America. https://maa.org/wp-content/uploads/2024/06/2015-CUPM-Curriculum-Guide.pdf.

Sheather, Simon. 2009. *A Modern Approach to Regression with R*. Springer Science & Business Media.

Stodden, Victoria. 2020. "The Data Science Life Cycle: A Disciplined Approach to Advancing Data Science as a Science." *Communications of the ACM* 63 (7): 58–66. https://doi.org/10.1145/3360646.

The pandas development team. 2020. *Pandas-Dev/Pandas: Pandas*. V. latest. Zenodo, released February. https://doi.org/10.5281/zenodo.3509134.

Theobold, Allison S. 2021. "Oral Exams: A More Meaningful Assessment of Students' Understanding." *Journal of Statistics and Data Science Education* 29 (2): 156–59. https://doi.org/10.1080/26939169.2021.1914527.

VanderPlas, Jake. 2022. *Python Data Science Handbook: Essential Tools for Working with Data*. 2nd ed. O'Reilly Media, Inc.

Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, et al. 2020. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." *Nature Methods* 17: 261–72. https://doi.org/10.1038/s41592-019-0686-2.

Waskom, Michael L. 2021. "Seaborn: Statistical Data Visu-

alization." *Journal of Open Source Software* 6 (60): 3021. https://doi.org/10.21105/joss.03021.

Wickham, Hadley. 2019. *Advanced R*. 2nd ed. Chapman and Hall/CRC Press. https://adv-r.hadley.nz/.

Wickham, Hadley, and Jennifer Bryan. 2023. *R Packages*. 2nd ed. O'Reilly Media, Inc. https://r-pkgs.org/.

Wickham, Hadley, Mine Çetinkaya-Rundel, and Garrett Grolemund. 2023. *R for Data Science*. 2nd ed. O'Reilly Media, Inc. https://r4ds.had.co.nz/.

Wing, Jeannette M. 2019. "The Data Life Cycle." *Harvard Data Science Review* 1 (1): 6. https://doi.org/10.1162/99608f92.e26845b4.

Yan, Donghui, and Gary E Davis. 2019. "A First Course in Data Science." *Journal of Statistics Education* 27 (2): 99–109. https://doi.org/10.1080/10691898.2019.1623136.

9 Acknowledgments

10 Appendix

10.1 Public course materials

- Intermediate Data Science at Macalester College
- Programming for Data Science in R at Smith College

10.2 Full topic list

The full table of topics covered is shown in Table 5.

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_sub_topic	dspt_competency_concept	learning_outcome	st_olaf	macalester	villanova	smith270	smith271
Computational Foundations	Knowledge (CF)	Computational Thinking	break down complex problems into simpler subproblems					
Computational Foundations	Knowledge (CF)	Data Management	differentiate between relative and absolute paths					
Computational Foundations	Knowledge (CF)	Data Management	store larger data files or the results from larger computations in native format					
Computational Foundations	Knowledge (CF)	Data Structure	recognize augmented vectors/arrays (with attributes)					

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_course	dspt_competency_subtopic	dspt_competency_topic	dspt_competency_learning_outcome	vilanova_course	vilanova_competency_subtopic	vilanova_competency_topic	vilanova_competency_learning_outcome
Computational Foundations	Knowledge (CF)	Data Structure	determine the amount of memory that an object occupies				
Computational Foundations	Knowledge (CF)	Data Structure	identify when a series of operations resulting in a copy (of an object) being made				
Computational Foundations	Knowledge (CF)	Data Structure	Identify the reference location (in memory) for an object				
Computational Foundations	Knowledge (CF)	Data Structure	recognize that a data.frame is just a list of vectors of the same length				

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_cspt	dspt_competency_content	learning_outcome	st_	vil-	macalester	smith270	smith271
Computational Foundations	Skill (CF)	Coding Practices	invoke best practices for code commenting					
Computational Foundations	Skill (CF)	Coding Practices	Differentiate code written using a base type from one in a different idiom (e.g., tidyverse-style in R)					
Computational Foundations	Skill (CF)	Data Processing (CF)	list the arguments that a function takes					
Computational Foundations	Skill (CF)	Data Processing (CF)	distinguish scope between global and local variables					

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_course	dspt_topic	dspt_subtopic	dspt_competency_learning_outcome	vil-lanova_course_number
Computational Foundations	Skill (CF)	Data Processing (CF)	set the default value of an argument to a function	1270 1271
Computational Foundations	Skill (CF)	Data Processing (CF)	handle missing values in functions	
Computational Foundations	Skill (CF)	Data Processing (CF)	determine the environment that provides a function/data.frame	
Computational Foundations	Skill (CF)	Data Processing (CF)	identify object returned by function	

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_course	dspt_topic	dspt_subtopic	dspt_competency_learning_outcome	vil-lanova_st_oakland_smith270	vil-lanova_st_oakland_smith271
Computational Foundations	Skill (CF)	Data Processing (CF)	throw errors, warnings, and messages (from within a function)		
Computational Foundations	Skill (CF)	Data Processing (CF)	pass labels to a plot through a function		
Computational Foundations	Skill (CF)	Data Processing (CF)	determine if a function/object is being masked		
Computational Foundations	Skill (CF)	Data Processing (CF)	differentiate between errors, warnings, and messages		

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_course	dspt_topic	dspt_subtopic	dspt_competency_learning_outcome	vil-lanova_course_number
Computational Foundations	Skill (CF)	Data Processing (CF)	display the code in the body of a function	1270 1271
Computational Foundations	Skill (CF)	Data Processing (CF)	perform basic file system navigation at the command line (e.g., ls, mv, cp, etc.)	
Computational Foundations	Skill (CF)	Data Processing (CF)	pass arguments to another function (e.g., dots (…) or kwargs)	
Computational Foundations	Skill (CF)	Data Processing (CF)	pass arguments through the dots (…)	

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_course	dspt_competency_concept	dspt_competency_concept_topic	learning_outcome	vil-lanova_course_number
Computational Foundations	Skill (CF)	Data Processing (CF)	distinguish between the global environment and a package environment	1270 1271
Computational Foundations	Skill (CF)	Data Processing (CF)	display the code in the body of a method	
Computational Foundations	Skill (CF)	Data Processing (CF)	distinguish between data masking and scoping/variable selection	
Computational Foundations	Skill (CF)	Data Processing (CF)	list the search path/scoping	

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_sub_topic	dspt_competency_concept	learning_outcome	st_olaf	vil-	smith270	smith271
Computational Foundations	Skill (CF)	Scalability	benchmark code efficiency (e.g., bench::mark())				
Data	Acquisition	Create Structure	tokenize corpora / documents				
Data	Acquisition	Extract from Database	recognize differences between SQL keywords and dplyr verbs				
Data	Acquisition	Extract from Database	establish a connection and query a database using SQL				
Data	Acquisition	Extract from Database	integrate SQL queries with R/Python code				
Data	Acquisition	Find Data	finding data relevant to the problem at hand				
Data	Acquisition	Import Data	import text data				

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_subtopic	dspt_competency_learning_outcome	vil-lanova_stat2700	vil-lanova_stat2701
Data Acquisition	Import Data	import CSV and XLSX files		
Data Acquisition	Simulation	data generation/synthetic data/creating distributions (not modeling)		
Data Acquisition	Web Scraping	scrape tables using the rvest package		
Data Acquisition	Web Scraping	accessing web APIs directly (e.g., with httr/httr2)		
Data Acquisition	Web Scraping	use functions and iteration to systematically acquire data		
Data Acquisition	Web Scraping	use web APIs through a wrapper package		
Data Acquisition	Web Scraping	scrape webpages using selector gadget and CSS selectors		
Data Stewardship	Governance	considers whether licensing allows data to be included in a public R/Python package		

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_subtopic	dspt_competency_learning_outcome	vil-lanova_stat_2700	vil-lanova_stat_2710	vil-lanova_stat_2715
Data Stewardship	Governance	determine permissibility to use data (e.g., robots.txt, terms and conditions, or licensing information)			
Data Stewardship	Lineage	reviewing lineage of original data source (before analyst got it)			
Data Stewardship	Lineage	maintaining record of original data source throughout the analysis (what did analyst do)			
Data Stewardship	Privacy	considers who is supposed to see the data			
Data Stewardship	Security	who has access to the data			
Data Processing	Cleaning	Fix	correct errors in source data		
Data Processing	Cleaning	Identify	identify errors in source data		
Data Processing	Wrangling	Combine	join, merge, or combine data sets		

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_topic_sub	dspt_competency_concept_topic	learning_outcome	st_ vil- smith270 smith271
Data Processing	Wrangling	Feature Engineering	capture information from strings using the stringr package	
Data Processing	Wrangling	Feature Engineering	match patterns in strings with regular expressions	
Data Processing	Wrangling	Reshape	nest and unnest list-columns	
Data Processing	Wrangling	Reshape	pivoting wider and longer	
Data Processing	Wrangling	Subset	subset cases and subset features/variables (e.g., filter() and select())	
Data Science Skills	Knowledge (DSS)	Ethics	scrape data ethically (e.g., with the polite and robotstxt packages)	
Data Science Skills	Knowledge (DSS)	Exploratory Data Analysis (DSS)	students undertake exploratory data analysis in iterative fashion (also in Data Visualization / Create)	

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_sub_topic	dspt_competency_concept	learning_outcome	st_[illegible]	vil-[illegible]	[illegible]	smith270	smith271
Data Science Skills	Knowledge (DSS)	Problem-solving Process	students engage in an iterative data analysis cycle					
Data Science Skills	Knowledge (DSS)	Subject Matter Expertise	understand context of data					
Data Science Skills	Methods (DSS)	DS-Specific Programming	vectorize control flow with ifelse()/if_else()					
Data Science Skills	Methods (DSS)	DS-Specific Programming	index a vector/list with a logical vector					
Data Science Skills	Methods (DSS)	DS-Specific Programming	index a vector/list with a character vector					
Data Science Skills	Methods (DSS)	DS-Specific Programming	identify the class of an object					

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_competency	dspt_topic	dspt_subtopic	learning_outcome	vil-lanova_course_number	vil-lanova_course_name	smith270	smith271
Data Science Skills	Methods (DSS)	DS-Specific Programming	differentiate between subsetting operators (e.g., [, [[, and $ in R)				
Data Science Skills	Methods (DSS)	DS-Specific Programming	index a vector/list with a numeric vector				
Data Science Skills	Methods (DSS)	DS-Specific Programming	name the four types of atomic vectors				
Data Science Skills	Methods (DSS)	DS-Specific Programming	iterate a function over a list or vector (e.g., with map()/lapply() in R)				
Data Science Skills	Methods (DSS)	DS-Specific Programming	improve code quality using a style guide				
Data Science Skills	Methods (DSS)	DS-Specific Programming	differentiate between vectorization and iteration				

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_sub_topic	dspt_competency_content	learning_outcome	st_olaf	villanova	macalester	smith270	smith271
Data Science Skills	Methods (DSS)	DS-Specific Programming	write a for() loop					
Data Science Skills	Methods (DSS)	DS-Specific Programming	iterate a function over a list-column					
Data Science Skills	Methods (DSS)	DS-Specific Programming	predict what will happen when vectors are combined (coerced)					
Data Science Skills	Methods (DSS)	DS-Specific Programming	use functions to eliminate loops					
Data Science Skills	Methods (DSS)	DS-Specific Programming	list the attributes of an object					
Data Science Skills	Methods (DSS)	DS-Specific Programming	use iteration for simulation					

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_subtopic	dspt_competency_content_learning_outcome	vil-lanova_stat_2700	vil-lanova_stat_2710	vil-lanova_stat_2711
Data Science Skills	Methods (DSS)	DS-Specific Programming	compare recycling in base R and the tidyverse		
Data Science Skills	Methods (DSS)	DS-Specific Programming	change an attribute of an object		
Data Science Skills	Methods (DSS)	DS-Specific Programming	iterate over multiple arguments (e.g., using pwalk()/mapply())		
Data Science Skills	Methods (DSS)	DS-Specific Programming	determine whether a function is generic		
Data Science Skills	Methods (DSS)	DS-Specific Programming	define new methods for a generic function		
Data Science Skills	Methods (DSS)	DS-Specific Programming	determine which method is dispatched when a generic function is called		

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_sub_topic	dspt_competency_concept	learning_outcome	st_olaf	vil-	macalester	smith270	smith271
Data Science Skills	Methods (DSS)	DS-Specific Programming	define a new generic function					
Data Science Skills	Methods (DSS)	Model Deployment	publish an interactive app					
Data Science Skills	Methods (DSS)	Model Deployment	read and write a DESCRIPTION file					
Data Science Skills	Methods (DSS)	Model Deployment	read a NAMESPACE file					
Data Science Skills	Methods (DSS)	Model Deployment	Use the Imports/Suggests/Depends fields					
Data Science Skills	Methods (DSS)	Model Deployment	add and document a data object in a package					
Data Science Skills	Methods (DSS)	Model Deployment	run R CMD check locally					
Data Science Skills	Methods (DSS)	Model Deployment	fix errors thrown by R CMD check					

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_sub_topic	dspt_competency_concept	learning_outcome	vil-st_oaf	[illegible]	smith270	smith271
Data Science Skills	Method (DSS)	Model Deployment	fix warnings thrown by R CMD check				
Data Science Skills	Method (DSS)	Model Deployment	fix resolve most messages thrown by R CMD check				
Data Science Skills	Method (DSS)	Model Deployment	differentiate between data/ and /data-raw/				
Data Science Skills	Method (DSS)	Model Deployment	Use the export directive in roxygen2				
Data Science Skills	Method (DSS)	Model Deployment	Use the import and importFrom directives in roxygen2				
Data Science Skills	Method (DSS)	Model Deployment	run unit tests via testthat or pytest				
Data Science Skills	Method (DSS)	Model Deployment	write simple unit tests				
Data Science Skills	Method (DSS)	Model Deployment	add and document a function in a package				

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_sub_team	dspt_competency_concept	learning_outcome	st_	vil- mat_chastain	smith270	smith271
Data Science Skills	Method (DSS)	Workflow	make a commit				
Data Science Skills	Method (DSS)	Workflow	push to a repo				
Data Science Skills	Method (DSS)	Workflow	pull from a repo				
Data Science Skills	Method (DSS)	Workflow	establish a file organization system for a course / project				
Data Science Skills	Method (DSS)	Workflow	create an informative README for a repo				
Data Science Skills	Method (DSS)	Workflow	clone a repo				
Data Science Skills	Method (DSS)	Workflow	set up git credentials				
Data Science Skills	Method (DSS)	Workflow	resolve a merge conflict				

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_sub_topic	dspt_competency_content	learning_outcome	st_olaf	villafranca	mclean	smith270	smith271
Data Science Skills	Method (DSS)	Workflow	conduct a peer code review					
Data Science Skills	Method (DSS)	Workflow	create and publish quarto webpages					
Data Science Skills	Method (DSS)	Workflow	create, comment on, and/or assign an issue					
Data Science Skills	Method (DSS)	Workflow	create quarto presentations					
Data Science Skills	Method (DSS)	Workflow	close an issue from a commit message					
Data Science Skills	Method (DSS)	Workflow	reference commits/code line numbers in issue					
Data Science Skills	Method (DSS)	Workflow	fork a repo					
Data Science Skills	Method (DSS)	Workflow	set up a pre-defined GitHub Action to perform R CMD check					

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_sub_topic	dspt_competency_content	learning_outcome	st_olaf	vil-	macalester	smith270	smith271
Data Science Skills	Methods (DSS)	Workflow	differentiate between README.Rmd and README.md					
Data Science Skills	Methods (DSS)	Workflow	make a pull request					
Data Science Skills	Methods (DSS)	Workflow	review a pull request					
Data Science Skills	Methods (DSS)	Workflow	make a branch					
Data Science Skills	Methods (DSS)	Workflow	push a branch					
Data Science Skills	Methods (DSS)	Workflow	sync a fork					
Data Science Skills	Methods (DSS)	Workflow	merge two branches					
Data Science Skills	Methods (DSS)	Workflow	merge a pull request					

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_course	dspt_topic	dspt_competency_subtopic	dspt_competency_learning_outcome	vil-lanova_course_number
Data Science Skills	Professional Tools	Collaboration	engage with diverse perspectives and assumptions	STT 270, STT 271
Data Science Skills	Professional Tools	Collaboration	work with others to solve problems	
Data Science Skills	Professional Tools	Collaboration	various GitHub actions (PR note - is this captured above?)	
Data Science Skills	Professional Tools	Communication	analyze data visualizations and storytelling in popular media	
Data Science Skills	Professional Tools	Communication	iterate on a data storytelling product by reflecting on feedback	
Data Science Skills	Professional Tools	Continuing Education	students reflect on learning process + lifelong learning (learning how to learn)	
Data Visualization	Create	Dynamic	create interactive apps with shiny	

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_course	dspt_competency	dspt_sub_topic	dspt_learning_outcome	vil-lanova_course_number
Data Visualization	Create	Dynamic	create interactive maps with leaflet	5270, 5271
Data Visualization	Create	Dynamic	create interactive apps with flexdashboard	5270, 5271
Data Visualization	Create	Exploratory Data Analysis (Viz)	explore data after read in (see also earlier EDA)	5270, 5271
Data Visualization	Create	Interpret	interpret initial EDA	5270, 5271
Data Visualization	Create	Multi-dimensional	create interactive multidimensional visualizations	5270, 5271

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_competency_topic	dspt_competency_learning_outcome	vil-lanova_stat_2700	vil-lanova_stat_2710	vil-lanova_stat_2715
Data Visualization	Create	Select	identify best visualization		
Data Visualization	Create	Static	create a choropleth map with geom_polygon or geom_sf		
Data Visualization	Create	Static	create maps with shapefiles and the sf package		
Data Visualization	Enhance	Effective	determine effectiveness of visualization		
Data Visualization	Enhance	Refine	improve initial visualization		

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_comp…	dspt_competency_con…	learning_outcome	st_ vil- … olaf charleston smith270 271
Mathematical Foundations	Knowledge (MF)	Calculus	Calculus: the use of differentiation and integration techniques and concepts to understand the underlying ideas behind and implementation of data models	
Mathematical Foundations	Knowledge (MF)	Linear Algebra	Linear Algebra: the use of matrix techniques and concepts to understand the underlying ideas behind and implementation of data models	
Mathematical Foundations	Knowledge (MF)	Optimization	Optimization: the use of numerical methods for minimizing or maximizing a function to understand the implementation of data models, computational time considerations, and convergence results	
Mathematical Foundations	Knowledge (MF)	Probability	Probability: the use of probability axioms, counting methods, random variables, and probability distributions to understand data modeling concepts	

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_subtopic	dspt_competency_concept	dspt_learning_outcome	vil-lanova_stat_2700	vil-lanova_stat_2710	vil-lanova_stat_2715
Modeling Methods	Classification	Algorithmic	Algorithmic: relating to detailed procedural implementation of data modeling methods for classification problems (categorical outcomes)			
Modeling Methods	Classification	Empirical	Empirical			
Modeling Methods	Classification	Logistic Regression	Logistic Regression: modeling the logit-transformed conditional mean of a binary outcome using a linear combination of predictors			
Modeling Methods	Regression	Algorithmic	Algorithmic: relating to detailed procedural implementation of data modeling methods for regression problems (quantitative outcomes)			
Modeling Methods	Regression	Linear Regression	Linear Regression: modeling the conditional mean of a quantiative outcome as a linear combination of predictors			

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_topic_sub	dspt_competency_content	learning_outcome	st_[illegible]	vil-[illegible]	[illegible]h270	[illegible]h271
Modeling Methods	Regression	Tree-based	Tree-based: methods that use sequential splitting of the data using the values of predictors to create data subsets that have homogenous outcomes				
Modeling Methods	Unsupervised	Clustering	Clustering: methods that form groups of cases in which intragroup variation is smaller than intergroup variation				
Modeling Methods	Unsupervised	Dimension Reduction	Dimension Reduction: methods that group variables or combine variables into new ones in ways that retain key information in the original data				
Modeling Methods	Unsupervised	Pattern Mining	Pattern Mining: the use of methods that detect recurring motifs in data in ways that are different from clustering cases (standard clustering) or clustering variables (dimension reduction)				
Modeling Methods	Unsupervised	Text Analysis	conduct text analyses with tidytext				

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_subtopic	dspt_competency_learning_outcome	dspt_concept	vil-lanova_course_number
Modeling Methods	Unsupervised	Text Analysis	conduct a sentiment analysis	8040
Modeling Methods	Unsupervised	Text Analysis	use tf-idf to identify terms important to one document	8040
Modeling Methods	Unsupervised	Text Analysis	identify stop words in a text	8040
Modeling Methods	Unsupervised	Text Analysis	generate word clouds	8040
Modeling Methods	Unsupervised	Text Analysis	consider n-grams	8040
Modeling Methods	Unsupervised	Text Analysis	correlate pairs of words	8040

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_subtopic	dspt_competency_concept	dspt_competency_concept_topic	dspt_learning_outcome	vil-lanova_course_number	vil-lanova_course_name	smith270	smith271
Modeling Methods	Unsupervised	Text Analysis		use LDA for topic modeling				
Modeling Methods	Unsupervised	Text Analysis		create network graphs of texts				
Modeling Principles	Diagnostics	Consistency		Consistency (across visualizations, models, etc.)				
Modeling Principles	Diagnostics	Cross-validation		Crossvalidation: the use of sequential within-sample splitting to assess the accuracy of a predictive model				
Modeling Principles	Diagnostics	Precision / Accuracy		Precision / Accuracy: conditional and marginal metrics that measure the quality of a predictive classification model				

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_sub_topic	dspt_competency_concept_pair	learning_outcome	st_oafcalestonsmith270h271 vil-
Modeling Principles	Diagnostics	Sensitivity	Sensitivity: conditional metric for measuring the accuracy of a predictive classification model (generally for the "positive" class)	
Modeling Principles	Goal	Algorithmic	Algorithmic (black box): is the goal of modeling black box prediction?	
Modeling Principles	Goal	Correlation	Correlation: the goal to identify factors that are related (but not necessarily causally) to an outcome	
Modeling Principles	Goal	Empirical	Empirical (non parametric)	
Modeling Principles	Goal	Estimation	Estimation: the goal to obtain a value for a parameter of a data model	

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_sub_topic	dspt_competency_concept	learning_outcome	st_olaf	villanova	blessman	smith270	smith271
Modeling Principles	Goal	Prediction	Prediction: the goal to make accurate forecasts of an outcome variable using a data model					
Modeling Principles	Structure	Correlated	Correlated: the quality of data values being systematically related due to the presence of common underlying factors					
Modeling Principles	Structure	Hierarchical	Hierarchical: the quality of data units being nested within each other (e.g., students within classrooms within schools within school districts)					
Modeling Principles	Structure	Uncorrelated	Uncorrelated: the quality of data values being unrelated due to a lack of common underlying factors					
Modeling Principles	Tuning	Bagging, Boosting	Bagging, Boosting: techniques used to combine the performance of many predictive models to form an overall better prediction					

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_topic_sub	dspt_competency_concept	learning_outcome	st_olaf	charleston	villanova	smith270	smith271
Modeling Principles	Tuning	Bias-Variance Trade-Off	Bias-Variance Trade-Off: the principle that the bias of a statistical model is inversely related to its variance across training datasets					
Modeling Principles	Tuning	Feature (Engineering)	Feature (Engineering): the practice of creating new predictor variables from existing ones to enhance the predictive accuracy of a model					
Modeling Principles	Tuning	Model Packaging	Model Packaging (ensemble methods)					
Statistical Foundations	Knowledge (SF)	Case	identify case or unit, reshape data as needed					
Statistical Foundations	Knowledge (SF)	Statistical Thinking	identify bias from data processing choices					

Table 5: Comparison of the treatment of various topics across the different courses.

dspt_topic	dspt_competency_subtopic	dspt_competency_subtopic_topic	dspt_competency_subtopic_topic_learning_outcome	vil-lanova_stat_2701	vil-lanova_stat_2711
Statistical Foundations	Knowledge (SF)	Uncertainty	Uncertainty: the principle that the products of data analysis will vary depending on the sample collected		
Statistical Foundations	Method (SF)	Data Collection	Data Collection: the organization of information into a form amenable for data analysis		
Statistical Foundations	Method (SF)	Inference	Inference: the process of quantifying uncertainty in modeling results by accounting for variation across samples		
Statistical Foundations	Method (SF)	Summarization			
Aggregate Measure	Summarization				

Table 5: Comparison of the treatment of various topics across the different courses.

	dspt_competency_concept_topic	dspt_competency_concept_topic_learning_outcome	village_classification_270	village_classification_271
Aggregate Measure: the process of simplifying the representation of data by creating a summary measure				

Source: Article Notebook