How Computer Science and Statistics Instructors Approach Data Science Pedagogy Differently: Three Case Studies

Sam Lau
UC San Diego
La Jolla, California, USA
lau@ucsd.edu

Deborah Nolan
UC Berkeley
Berkeley, California, USA
deborah_nolan@berkeley.edu

Joseph Gonzalez
UC Berkeley
Berkeley, California, USA
jegonzal@cs.berkeley.edu

Philip Guo
UC San Diego
La Jolla, California, USA
pg@ucsd.edu

ABSTRACT

Over the past decade, data science courses have been growing more popular across university campuses. These courses often involve a mix of programming and statistics and are taught by instructors from diverse backgrounds. In our experiences launching a data science program at a large public U.S. university over the past four years, we noticed one central tension within many such courses: instructors must finely balance how much computing versus statistics to teach in the limited available time. In this experience report, we provide a detailed firsthand reflection on how we have personally balanced these two major topic areas within several offerings of a large introductory data science course that we taught and wrote an accompanying textbook for; our course has served several thousand students over the past four years. We present three case studies from our experiences to illustrate how computer science and statistics instructors approach data science differently on topics ranging from algorithmic depth to modeling to data acquisition. We then draw connections to deeper tradeoffs in data science to help guide instructors who design interdisciplinary courses. We conclude by suggesting ways that instructors can incorporate both computer science and statistics perspectives to improve data science teaching.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

data science, case studies, programming education, statistics

ACM Reference Format:
Sam Lau, Deborah Nolan, Joseph Gonzalez, and Philip Guo. 2022. How Computer Science and Statistics Instructors Approach Data Science Pedagogy Differently: Three Case Studies. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2022), March 3–5, 2022, Providence, RI, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3478431.3499384

SIGCSE 2022, March 3–5, 2022, Providence, RI, USA

ACM ISBN 978-1-4503-9070-5/22/03.
https://doi.org/10.1145/3478431.3499384

1 INTRODUCTION

It is widely acknowledged that both computer science and statistics are fundamental to data science [13]. The computer science perspective generally focuses on "how"—how to do data analysis in practical real-world settings. Data scientists write programs using languages like Python and use software packages like `pandas` [28] to load, manipulate, and clean data. Computer science also provides tools to analyze the runtime and memory usage of code. These tools have led to the development of systems for working with large datasets, including relational databases and distributed computing (e.g. MapReduce, Hadoop, Spark). Finally, data scientists rely on algorithms for numeric optimization to fit their models. Without computer science, data scientists are limited to small datasets and cannot make use of the vast amounts of data being generated today.

On the other hand, the statistics perspective generally focuses on "why"—for instance, why an analysis on a relatively small sample can generalize to a larger population. Statistics provides fundamental tools for inference like hypothesis testing and confidence intervals. These statistical tools let data scientists say precisely *why* they have (or don't have) confidence in the results of an analysis. And statistical learning gives basis to the models that data scientists use to make predictions. Without statistics, data scientists can only draw conclusions about their samples—they can't use their samples to make inferences or predictions about the unobserved population.

Domain experts acknowledge that data science courses should demonstrate how both computer science and statistics can be applied to do useful data analyses [14]. However, data science instructors then face the practical challenge of providing students with an effective balance of these two fields, given the limited time available in a quarter- or semester-long course. In practice, different instructors strike different balances between computing and statistics content. This tension is reflected in data science programs as a whole as well—some programs have more of an emphasis on computing, and others have more statistics, as discussed in past SIGCSE panels and reports [2, 4, 7, 20, 22, 33, 35, 36].

This experience report is, to our knowledge, the first to reflect on why and how computer science and statistics instructors approach teaching data science differently. The authors of this paper come from both computer science and statistics departments at a large public U.S. university. We have worked together over the past four years to design curriculum and write a textbook for an undergraduate data science course that serves over

2,500 students a year. This paper presents three case studies of where we made critical pedagogical changes along the way.

On the surface, these discussions were about changes to a specific course. However, we found that our negotiations actually revealed fundamentally different approaches to data science. Each case study explains how our approaches differed, and how we eventually integrated both perspectives. In the first case study, we discuss programming abstractions—should our students implement gradient descent themselves or use a Python package? In the second, we discuss how to approach statistical modeling for different levels of mathematical maturity. In the third, we discuss the tension between working with clean scientific data versus messy data in the wild.

We reflect on our case studies as a whole to show how the different approaches of computer science and statistics also mirror deeper tradeoffs in data science education: the tradeoff between emphasizing inference and prediction, and the tradeoff between theory and practice. We conclude by making suggestions for data science instructors who bridge the gap between the two disciplines. This experience report's contributions to computing education are:

- Three case studies from our experiences that highlight the different approaches that computer science and statistics instructors take to teaching data science.
- A discussion of fundamental tradeoffs for data science pedagogy, and suggestions for other topics that can benefit from a close coupling of computer science and statistics.

2 RELATED WORK

Data science as a discipline was established in part because of the need for people who can combine computing and statistical knowledge for data analyses. This need is acknowledged by scientific research [23], industry [17], statisticians [13, 18], and computer scientists [8]. To this end, a prominent National Academy of Sciences report calls for undergraduate data science programs to include topics from both computer science and statistics [14]. However, developing data science curricula is challenging for instructors who need to decide what topics to include from these two disciplines.

This challenge of balancing disciplinary focus is reflected in SIGCSE papers on data science education from the past five years. One theme from this past work on data science curricula design is that each instructor strikes a different balance between computing and statistics content. For instance, computer science instructors might add data science topics to existing computing courses, thus focusing on computing more than statistics. Instructors have taken this approach in introductory courses for computing majors [16], computing courses for non-majors [5], and data systems courses [11]. Similarly, statistics instructors include data science topics into statistics courses that might otherwise not include computing, including introductory courses [6, 27] and more advanced courses [21, 30]. Finally, instructors have designed brand new courses that try to focus equally on both computing and statistics [3, 19, 34]. These courses tend to be listed under a data science department rather than a computer science or statistics department. As a whole, this set of prior work provides examples of successful course designs. In contrast, we take a broader view in this paper by examining how computer science and statistics instructors approach data science topics differently rather than arguing for a specific course design.

Beyond single courses, data science degree programs also differ in how they balance computer science and statistics content. Some programs place more course hours in computing [20, 33], some programs place more in statistics [4, 7, 35, 36], and other programs have equal course hours in computer science and statistics [2, 22]. Rather than debating what the goals of a program should be, this paper describes the thinking that computer science and statistics instructors go through when setting learning goals in the first place.

3 SETTING: A LARGE UNDERGRADUATE DATA SCIENCE COURSE AND TEXTBOOK

Three authors of this paper helped design and instruct an upper-division undergraduate data science course at a large public U.S. university; two authors are from computer science and one is from statistics. This course was launched in Spring 2017. It is now a requirement for both data science majors and minors at our institution, serving over 2,500 students during the 2020–2021 academic year. In this course, students learn standard computational tools for working with data, including relational databases, Python-based programming tools (e.g., `pandas` [28], `scikit-learn` [31]), and algorithmic techniques such as gradient descent. Students also learn statistical techniques for modeling, including statistical techniques for visualization, regression, and bias-variance tradeoff. Our course prepares students for real-world analyses by showing how *both computer science and statistics* are used throughout data science.

Since our course includes topics from both computer science and statistics, it is co-taught each term by one instructor from the computer science department and one from the statistics department. We are not the only instructors involved with the course; other faculty from our departments have also instructed and made helpful contributions to the curriculum. Although we do not speak for all the instructors of the course, we have directly instructed in seven out of the eleven offerings so far. We are also the authors of a textbook used in this course [24].

In the past four years of working together, we have made many changes to the course and its accompanying textbook. Most of these changes needed discussion amongst this paper's co-authors before implementing. In the following sections, we share three case studies where these discussions highlighted how our backgrounds approach data science differently, and how we reached agreement by acknowledging the perspectives of colleagues in different fields.

Caveats: We acknowledge that other institutions have different arrangements for teaching—for instance, some courses alternate instructors between computer science and statistics, and data science programs have students take separate courses from both computer science and statistics. To provide experiences that remain relevant across different teaching arrangements, in this paper we focus on more foundational data science concepts rather than course-specific logistical details. Although our case studies happened to take place in one course, we expect that they provide insight for instructors who must strike their own balance between computer science and statistics content when designing and iterating on their courses.

4 THREE CASE STUDIES

This section reports on three case studies from our course and textbook design experiences. For each case study, we summarize

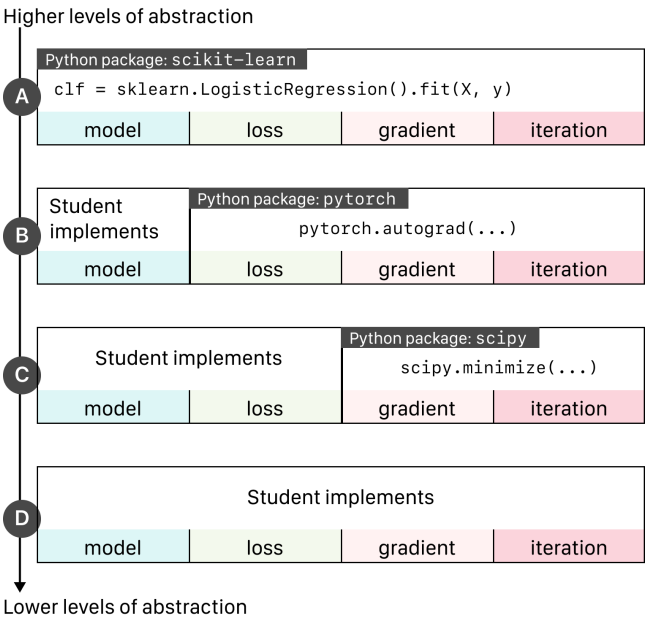

Figure 1: We considered multiple ways to teach students to fit models using Python. Each approach works at a different level of abstraction, and our debate centered around how much implementation students should do themselves.

what actually happened by presenting a back-and-forth 'simulated conversation' between a computer science (CompSci) instructor and a statistics (Stat) instructor. These conversations are meant to capture where each side started from, how each presented the benefits and tradeoffs of their pedagogical approach, and eventually how we came to a resolution[1]. Since our reported conversations are 'simulated', they do not directly quote individuals, although we consulted our old meeting notes where possible. Instead they are meant to capture the high-level points-of-view that emerged from us jointly reflecting on actual pedagogical negotiations we had with each other throughout the past four years of working together.

4.1 Do We Need to Delve into Algorithms?

One of the most foundational algorithms in data science is *gradient descent*, which is used to fit all kinds of models ranging from simple (e.g., linear regression) to complex (e.g., neural networks). The gradient descent algorithm can be taught at multiple levels of abstraction. But, which level of abstraction is best for teaching data science? Or, do we even need to teach it at all? We tried several approaches over different iterations of our course and its textbook, summarized in Figure 1.

Starting Points.

Stat instructor: The main goal of our course is to teach students how to do sound work with data. It's key that our students learn the importance of using well-tested and well-implemented code when they optimize and fit models. The Python `scikit-learn` package implements all the models we teach, and our students will likely only use the `fit()` method from `scikit-learn` once they start working on real data problems (Figure 1a). Teaching them how to implement models and gradient descent *from scratch* (Figure 1d) is not necessary since packages are well-tested and optimized.

CompSci instructor: Well, I think numeric optimization is an important topic in data science. Even students who only use packages like `scikit-learn` will still need to understand gradient descent in order to fully use its API. There are interesting teaching possibilities here that connect the theory of optimization with its implementation! For example, we can use automatic differentiation software like the `autograd` module from `pytorch` (Figure 1b). Or, we could have students calculate and implement their own loss functions, then pass those into gradient descent optimizers like `scipy.minimize` (Figure 1c).

Discussion.

Stat instructor: Teaching students to implement gradient descent seems great for a specialized course on numeric optimization. But it's less clear how implementing gradient descent from scratch helps them understand and draw conclusions from real-world data. How do you see it fitting into the larger data science picture?

CompSci instructor: Optimization is an important topic in data science because it lets us determine what models are feasible to use given real-world constraints on time and memory usage. For instance, neural networks could only be applied to realistic problems, like image recognition, after we had more powerful computers and better gradient descent algorithms. Implementing gradient descent lets us teach students about the runtime and memory needed to fit models. For example, we can have students implement both batch and stochastic gradient descent, ask them to fit models using both algorithms, and then have them explain why one might converge faster than the other. It's important that our students can say, *"I can still fit this model if the training data doubles in size, but I'll need to change my approach if the training data is ten times larger."*

Stat: Ok, I agree that runtime is an important tradeoff for data scientists to consider. Runtime is especially important for large, complex models like neural networks. But, teaching gradient descent at levels of abstraction lower than `scikit-learn` (Figure 1a) has drawbacks, especially if students need to learn other complex Python packages to do so. We've seen that students already find it hard to learn a package specifically for gradient descent—although they wrote fewer lines of code, each package has specific idioms that they had to learn. That's a lot of cognitive overhead! It's useful to teach students how to work with multiple Python packages, but we only have so much time in one course.

CompSci: Point taken. Though there's another aspect to what I mentioned earlier: to fully use the `scikit-learn` API, our students will need to have a deep understanding of gradient descent. For instance, `scikit-learn` models let data scientists change the stopping criteria and the maximum number of iterations, and they need to experiment with these parameters every time they try to fit a model. I want our students to identify when they need to change these parameters and what the tradeoffs are—for example, if their model doesn't converge, they should know they can increase the cap on the number of iterations if they're willing to run for longer.

[1]This explanatory approach of a simulated dialogue is inspired by business school case studies that show how stakeholders discuss and make business decisions [37].

Stat: I agree that's important, but can't we just teach the gradient descent algorithm using pseudocode to get those concepts across?

CompSci: I think students develop better intuition about gradient descent when they implement and debug a basic version of it themselves. For example, they can add custom logging statements into their own implementations to look at their program's behavior at a fine-grained level. `scikit-learn` also implements other optimization algorithms, including second-order methods like BFGS [26]. Even though we don't have time in our course to explain these algorithms in detail, implementing gradient descent from scratch gives students a basic framework to start understanding when other algorithms can be useful.

Resolution.

Our discussion eventually converged, and we agreed that gradient descent was important for our students to understand well. Data scientists make many tradeoffs when choosing and fitting models—they consider how much data they have, whether the model might underfit or overfit the data, and how accurate and interpretable the model is. At each step in modeling, data scientists must also think about the computational resources that they have access to. We found gradient descent to be a natural place to introduce these central considerations in modeling.

Thus, we resolved to take a hybrid approach. We teach students to use `scikit-learn` (Figure 1a) but also teach them to implement a basic version of gradient descent completely from scratch in Python (Figure 1d). Since our students have some programming background, we found they could implement basic versions of gradient descent after one lecture's worth of explanation through pseudocode, which we felt was reasonable for our course's time constraints. We also used this hybrid approach in our textbook, and took care to point out that the book contains simple implementations for teaching purposes only, not for production use.

4.2 How to Approach Statistical Modeling?

Data scientists use statistical models to draw inferences and make predictions using data. In this case study, we discuss various approaches for teaching the foundations of modeling.

Starting Points.

Stat: The likelihood principle offers a unifying approach to modeling that extends to many settings. In a traditional statistics course, we define a simple linear model as $Y = \theta_0 + \theta_1 x + \epsilon$, where the errors, ϵ, are independent normally distributed with mean 0 and constant variance. From this perspective, fitting a model means finding the model parameters that maximize the likelihood of observing the data, i.e. maximum likelihood estimation (MLE) (Figure 2a).

When we work through MLE derivations, we motivate different loss functions. For instance, the simple linear model described above naturally connects to squared loss (Figure 2b). And, at the end of the derivation, we can get closed-form solutions for model coefficients, which have meaningful interpretations. For instance, the 'regression effect' clearly appears in the slope of the regression line—the sample correlation scaled by the standard deviations of x and y (Figure 2c). The beauty of the likelihood principle is that it shows how a data-generating model leads to both loss functions

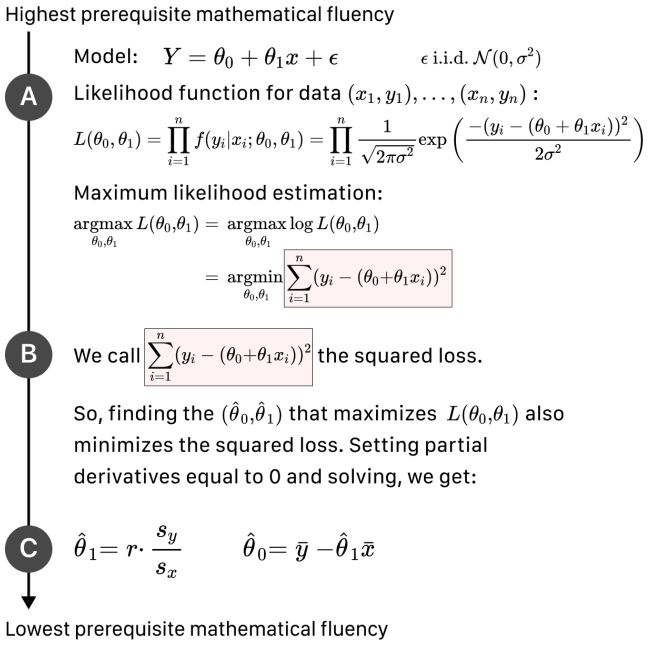

Figure 2: This sketch of a derivation for the simple linear model parameters highlights several possible pedagogical approaches. Instructors can start with (a) likelihood principle, (b) loss minimization, or (c) a closed-form solution.

and parameter estimates. We can also use this approach to make inferences for model parameters.

CompSci: OK, but my focus is on prediction, not inference—I'm not as concerned about the exact form of the model and its parameters compared to getting a predictor with high accuracy. Although the likelihood principle leads to useful interpretations of model parameters, I often have so many parameters in my models that there isn't a useful interpretation for the parameters anyway. An over-parameterized model isn't a big concern for predictive accuracy since we can control over-fitting through regularization. Moreover, since we can fit models using loss minimization via gradient descent, we don't need to find closed-form solutions for the MLE. And, the likelihood principle for widely used models like logistic regression gets complicated quickly. The advantage of using the MLE in this case is not clear.

Discussion.

CompSci: When students go into the world, it's unlikely that they will need to consider the likelihood principle when they fit models. Most models we use in this course have closed-form MLE solutions because they are relatively simple, but more complicated models like neural networks don't have nice analytic solutions. When faced with more complex problems, students might get easily stumped because they can't cast the problem in terms of likelihood.

Stat: MLE shows explicitly how a data scientist's assumptions affect how we fit the model. Important extensions are well-motivated by this approach, such as weighted regression and the generalized linear model. The likelihood approach I'm proposing will require

us to state model assumptions upfront (Figure 2a). And when assumptions are violated, the likelihood approach can provide a path forward. Without MLE, I'm concerned the course sends the message that data scientists can pick any arbitrary model and loss function they want without considering the assumptions behind them.

CompSci: I agree that it's important for students to understand whether a model is appropriate for their data. However, to understand MLE, students need to draw on knowledge of random variables, expectation, probability distributions, and calculus. We've tried to teach MLE before, but students felt unprepared, and they fixated on the math rather than the underlying concepts we really wanted to teach. In hindsight I think we underestimated how hard it is to develop fluency with probability.

Stat: I agree the technical machinery used with MLE can become a burden for students, especially when they first see multivariate modeling. But even if we don't cover MLE in this course, it's important to help students build mathematical maturity. The stepping stones to MLE—expectation, variance, and bias—are useful when we teach general modeling principles. For example, in our course we make a distinction between empirical loss (or training set average error) and expected loss (based on model error). When we teach that a model's test set error is a statistical estimator for the expected loss, we reinforce statistical ideas that they can use later.

Resolution.

We converged to agree that the likelihood principle is an important concept in modeling. However, we could only realistically cover modeling from a loss minimization approach based on the time constraints of our course and prerequisite student knowledge (Figure 2b). Thus we postponed the likelihood principle and MLE to a more advanced theory course. Then in our course and its textbook, we included the conceptual stepping stones to the likelihood principle by teaching model fitting through empirical loss minimization.

We now teach a broad array of loss functions—e.g., squared, absolute, and cross-entropy loss—but do not show their derivations from a likelihood standpoint. We also encourage students to consider how prediction errors should be penalized based on domain knowledge. For instance, if over-medication in surgery can lead to serious health problems, then an asymmetric loss function that penalizes overestimates more than underestimates is preferable [29]. By taking the middle ground and focusing on loss and optimization, we give a balanced perspective that covers both inference for model parameters and prediction of future observations.

4.3 How to Approach Real-World Data?

It takes a lot of work to find useful data for teaching, so we had many discussions on what data to use in our course and textbook.

Starting Points.

Stat: Data analyses begin by asking how the data were collected. One important distinction is whether or not the data were collected according to a chance mechanism. With probability sampling, we can teach the notion of *representative data*. For instance, with a simple random sample (SRS) we know that every potential sample has the same chance of being selected. More broadly, in the real world, alternative probability methods (e.g., stratified, cluster, systematic sampling) are used in scientific surveys. If we ask students to carry out probability calculations for small populations, then they can better understand the importance of random samples.

CompSci: These sampling methods are a nice-to-have rather than a necessity. The SRS is all we need to get the point across to students. I'm not sure why we need to discuss multiple random sampling methods when nearly all of the datasets in the course are *not* collected using random sampling. For example, we work with datasets of social media posts (e.g., tweets), housing prices, and emails. All of these are examples of *found data*: data scientists usually "find" data that are already available online rather than collecting data themselves. And none of these are probability samples. This also reflects a current trend in data analysis: Traditional scientists start with a research question and then collect data *specifically* to answer that question. Today, data are more plentiful and accessible, so data scientists often start with available *found data* and explore interesting questions that data might be able to answer.

Discussion.

CompSci: Probability calculations with various sampling schemes can get complex quickly, and I don't see the point of teaching counting arguments and combinatorics. It's also easy to spend too much time on fun probability problems with dice and cards. While this develops intuition for probability, it can feel very disconnected from real-world data analysis problems.

Stat: I agree it's tempting to teach too many fun probability problems and that counting arguments have little value in data analysis. But, sampling methods *do* have important real-world implications, especially in recent times: for instance, treating margins of error in election polling as if the data were from a SRS winds up under-estimating the true margin of error [38]. And more importantly, for nonrandom samples it doesn't make sense to compute p-values or confidence intervals—these methods assume random draws from a population, but *found data* break that assumption.

CompSci: Yes, ignoring the sample design can lead to major mistakes. There are many famous examples where ignoring this led to major problems with the analysis, like the wrong call in the Dewey-Truman election [1]. Still, even the most careful surveys aren't true probability samples because of issues like non-response [15]. Should we teach students random samples when in reality there are many caveats? Few samples are truly random in practice.

Stat: That is indeed the case, but the ideal scenario is worth knowing about and it gives students a useful comparison point against the data they do have. And, there are statistical methods for adjusting for, say, non-response [39]. Even if we don't have time in this course to talk about specific methods, data scientists should know that these methods exist. More importantly, data scientists need to be aware of common ways that their samples may not be representative. That way, when they do assume that their sample is representative, they should be able to justify it.

CompSci: I do think what you are saying is important; our students should acknowledge when they make these assumptions and point out potential sources of bias for their analyses. But, I want to return to an earlier point because I think it is an important one. So much of today's data are large administrative data, such as digital traces. It doesn't make sense to ignore these data sources.

Stat: I agree we need to expand beyond an idealized view. But we also want our students to avoid "big data hubris" [25]—thinking that

having more data automatically leads to sound analyses. They need to leverage big data where it makes sense. They also need to understand how to analyze scientifically-collected data. We shouldn't only teach one or the other, because they will use both on the job.

Resolution.

Although our starting points were about sampling methods, the discussion quickly went to the core of how to treat representative and non-representative data. We decided to reduce the emphasis on specific sampling methods, but kept the stratified sample as a comparison to the SRS (simple random sample). The main change we made to the course and textbook was to develop a framework to help data scientists reason about whether or not their data are representative. We extended the traditional way of teaching survey sampling to introduce the target population, access frame, and sample. This framework gives us the tools to discuss accuracy, including various sources of bias (e.g., coverage, selection, non-response), and has a natural extension for introducing measurement error. The framework also allows us to situate and link together different kinds of data, including survey samples, controlled experiments, administrative data, and instrumental measurements.

5 DISCUSSION

In this section, we draw common themes from our three case studies, focusing on the places where computer science and statistics approach data science differently. These case studies show how these two academic disciplines developed with distinct goals—these goals motivate pedagogical tradeoffs that instructors make when designing data science courses. We discuss these tradeoffs and suggest ways that instructors might acknowledge and incorporate both perspectives into data science teaching.

5.1 Inference and Prediction

A central tradeoff in data science pedagogy is balancing the emphasis on inference with emphasis on prediction. This is illustrated in the second case study of this paper (Section 4.2). In our experience, statistics focuses more on inference while computer science focuses more on prediction. For instance, a data scientist can construct confidence intervals to infer the slope of a regression line in a model. In contrast, when data scientists focus on prediction, their goal is to find a model with high predictive accuracy. This tradeoff is analogous to the contrast between the data modeling and algorithmic modeling cultures in statistics [10]. An overly extreme focus on inference might only allow relatively simple models. On the other hand, an overly extreme focus on prediction might select models without regard for interpretability or robustness.

The experiences from our case studies have led us to the view that teaching both inference and prediction together helps students appreciate the difference and learn when to emphasize one over the other. For instance, one well-known economic study uses an inferential view to determine which social factors (e.g., segregation, income inequality) are significantly correlated with economic mobility [12]. Another study uses a predictive view to estimate income from phone call records—the significance of each covariate is less relevant than the model's overall predictive ability [9]. We recommend that data science instructors bridge the gap between traditional computer science and statistics views by including both predictive and inferential views for statistical modeling.

5.2 Theory and Practice

Both computer science and statistics contribute methods and theory for data science. For instance, computer science teaches practical software tools and theoretical analysis of runtimes. Statistics teaches practical modeling techniques and bias-variance theory that applies across models. When we first launched our data science course, each of our lessons tended to focus heavily on either theory or practice. This divide sparked pedagogical discussions amongst this paper's co-authors as we developed our course and textbook. For instance, in our third case study (Section 4.3) we debated how much to emphasize the theory for random sampling versus the practical reality of using found data in-the-wild. We moved individual lessons towards a closer balance of both theory and practice as we discussed and iterated on our course. We expect that other instructors can anticipate similar discussions based on the fact that theory and practice both have clear roles in data science—for instance, a lesson that strictly focuses on one may be improved by incorporating other perspective.

5.3 Both Perspectives in Data Science Pedagogy

In our experience, many topics in data science can be approached through both computer science and statistics perspectives. Although our three case studies cover three specific examples, we constantly discuss how we might better balance these two disciplines across multiple courses.

For example, data scientists often work with tabular data using the concept of a dataframe. The dataframe was originally designed by statisticians for exploratory data analysis but also serves as a computational data structure. Instructors can contrast the dataframe with its database analog, the *relation* (e.g. [32]).

Relatedly, there is a distinction between the term *data type* as used in programming versus in statistics. For instance, survey responses can be recorded using numbers (e.g., 0 for "no", 1 for "yes", and 2 for "maybe") or as strings. A statistician would describe this data type abstractly as categorical, and would not compute the average even when the data are stored as numbers in a program. Instructors can discuss how computer science and statistics view data types differently.

6 CONCLUSION

This paper shares three case studies from our time spent working closely together on a data science course and textbook. The recurring themes we present highlight the different approaches that computer science and statistics take in teaching data science. These differences are captured in tradeoffs between explaining the "how" and "why" of each concept and reflect ongoing debates amongst experts in the field. We advise data science instructors to seek interdisciplinary views but also understand that it's difficult to please everyone—rather than searching for an immediately "perfect" balance of computing and statistics, instructors can instead welcome different points of view and improve their courses over time.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. NSF IIS-1845900 and Grant No. NSF DGE-1735234

REFERENCES

[1] 2021. Dewey Defeats Truman. *Wikipedia* (July 2021).

[2] Joel C. Adams. 2020. Creating a Balanced Data Science Program. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 185–191.

[3] Genevera I. Allen. 2021. Experiential Learning in Data Science: Developing an Interdisciplinary, Client-Sponsored Capstone Program. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. Association for Computing Machinery, New York, NY, USA, 516–522.

[4] Paul Anderson, James Bowring, Renée McCauley, George Pothering, and Christopher Starr. 2014. An Undergraduate Degree in Data Science: Curriculum and a Decade of Implementation Experience. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. Association for Computing Machinery, New York, NY, USA, 145–150. https://doi.org/10.1145/2538862.2538936

[5] Austin Cory Bart, Dennis Kafura, Clifford A. Shaffer, and Eli Tilevich. 2018. Reconciling the Promise and Pragmatics of Enhancing Computing Pedagogy with Data Science. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 1029–1034. https://doi.org/10.1145/3159450.3159465

[6] Ben Baumer. 2015. A Data Science Course for Undergraduates: Thinking with Data. *The American Statistician* 69, 4 (2015), 334–342.

[7] Ismail Bile Hassan, Thanaa Ghanem, David Jacobson, Simon Jin, Katherine Johnson, Dalia Sulieman, and Wei Wei. 2021. Data Science Curriculum Design: A Case Study. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. Association for Computing Machinery, New York, NY, USA, 529–534.

[8] Avrim Blum, John Hopcroft, and Ravindran Kannan. 2020. *Foundations of Data Science*. Cambridge University Press.

[9] Joshua Blumenstock, Gabriel Cadamuro, and Robert On. 2015. Predicting Poverty and Wealth from Mobile Phone Metadata. *Science* 350, 6264 (Nov. 2015), 1073–1076. https://doi.org/10.1126/science.aac4420

[10] Leo Breiman. 2001. Statistical Modeling: The Two Cultures (with Comments and a Rejoinder by the Author). *Statistical science* 16, 3 (2001), 199–231.

[11] Thomas C. Bressoud and Gavin Thomas. 2019. A Novel Course in Data Systems with Minimal Prerequisites. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 15–21. https://doi.org/10.1145/3287324.3287425

[12] Raj Chetty, Nathaniel Hendren, Patrick Kline, and Emmanuel Saez. 2014. Where Is the Land of Opportunity? The Geography of Intergenerational Mobility in the United States. *The Quarterly Journal of Economics* 129, 4 (2014), 1553–1623.

[13] William S. Cleveland. 2001. Data Science: An Action Plan for Expanding the Technical Areas of the Field of Statistics. *International statistical review* 69, 1 (2001), 21–26.

[14] Committee on Envisioning the Data Science Discipline: The Undergraduate Perspective, Computer Science and Telecommunications Board, Board on Mathematical Sciences and Analytics, Committee on Applied and Theoretical Statistics, Division on Engineering and Physical Sciences, Board on Science Education, Division of Behavioral and Social Sciences and Education, and National Academies of Sciences, Engineering, and Medicine. 2018. *Envisioning the Data Science Discipline: The Undergraduate Perspective: Interim Report*. National Academies Press, Washington, D.C. https://doi.org/10.17226/24886

[15] National Research Council. 2013. *Nonresponse in Social Science Surveys: A Research Agenda*. National Academies Press.

[16] Sarah Dahlby Albright, Titus H. Klinge, and Samuel A. Rebelsky. 2018. A Functional Approach to Data Science in CS1. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 1035–1040. https://doi.org/10.1145/3159450.3159550

[17] Thomas H. Davenport and D. J. Patil. 2012. Data Scientist: The Sexiest Job of the 21st Century. *Harvard Business Review* (Oct. 2012).

[18] Richard D. De Veaux, Mahesh Agarwal, Maia Averett, Benjamin S. Baumer, Andrew Bray, Thomas C. Bressoud, Lance Bryant, Lei Z. Cheng, Amanda Francis, and Robert Gould. 2017. Curriculum Guidelines for Undergraduate Programs in Data Science. *Annual Review of Statistics and Its Application* 4 (2017), 15–30.

[19] John DeNero. [n. d.]. Data 8. http://data8.org/.

[20] Alan Fekete, Judy Kay, and Uwe Röhm. 2021. A Data-Centric Computing Curriculum for a Data Science Major. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. Association for Computing Machinery, New York, NY, USA, 865–871.

[21] J. Hardin, R. Hoerl, Nicholas J. Horton, D. Nolan, B. Baumer, O. Hall-Holt, P. Murrell, R. Peng, P. Roback, D. Temple Lang, and M. D. Ward. 2015. Data Science in Statistics Curricula: Preparing Students to "Think with Data". *The American Statistician* 69, 4 (Oct. 2015), 343–353. https://doi.org/10.1080/00031305.2015.1077729

[22] Jessen Havill. 2019. Embracing the Liberal Arts in an Interdisciplinary Data Analytics Program. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 9–14. https://doi.org/10.1145/3287324.3287436

[23] Tony Hey, Stewart Tansley, and Kristin Tolle. 2009. *The Fourth Paradigm: Data-Intensive Scientific Discovery*.

[24] Samuel Lau, Joseph Gonzalez, and Deborah Nolan. 2018. *Principles and Techniques of Data Science*.

[25] David Lazer, Ryan Kennedy, Gary King, and Alessandro Vespignani. 2014. The Parable of Google Flu: Traps in Big Data Analysis. *Science* 343, 6176 (2014), 1203–1205.

[26] Dong C. Liu and Jorge Nocedal. 1989. On the Limited Memory BFGS Method for Large Scale Optimization. *Mathematical programming* 45, 1 (1989), 503–528.

[27] Adam Loy, Shonda Kuiper, and Laura Chihara. 2019. Supporting Data Science in the Statistics Curriculum. *Journal of Statistics Education* 27, 1 (2019), 2–11.

[28] Wes McKinney. 2011. Pandas: A Foundational Python Library for Data Analysis and Statistics. *Python for high performance and scientific computing* 14, 9 (2011), 1–9.

[29] Kate Milner and Jonathan Rougier. 2014. How to weigh a donkey in the Kenyan countryside. *Significance* 11, 4 (2014), 40–43.

[30] Deborah Nolan and Duncan Temple Lang. 2010. Computing in the Statistics Curricula. *The American Statistician* 64, 2 (2010), 97–107.

[31] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, and Vincent Dubourg. 2011. Scikit-Learn: Machine Learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[32] Devin Petersohn, Stephen Macke, Doris Xin, William Ma, Doris Lee, Xiangxi Mo, Joseph E. Gonzalez, Joseph M. Hellerstein, Anthony D. Joseph, and Aditya Parameswaran. 2020. Towards Scalable Dataframe Systems. *arXiv preprint arXiv:2001.00888* (2020). arXiv:2001.00888

[33] Bina Ramamurthy. 2016. A Practical and Sustainable Model for Learning and Teaching Data Science. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. Association for Computing Machinery, New York, NY, USA, 169–174. https://doi.org/10.1145/2839509.2844603

[34] Suraj Rampure, Allen Shen, and Josh Hug. 2021. Experiences Teaching a Large Upper-Division Data Science Course Remotely. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. Association for Computing Machinery, New York, NY, USA, 523–528.

[35] Stephanie Rosenthal and Tingting Chung. 2020. A Data Science Major: Building Skills and Confidence. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 178–184.

[36] Mariam Salloum, Daniel Jeske, Wenxiu Ma, Vagelis Papalexakis, Christian Shelton, Vassilis Tsotras, and Shuheng Zhou. 2021. Developing an Interdisciplinary Data Science Program. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 509–515.

[37] Harvard Business School. [n. d.]. What is the Case Study Method? https://www.exed.hbs.edu/hbs-experience/learning-experience/case-study-method. Accessed: 2021-08-13.

[38] Houshmand Shirani-Mehr, David Rothschild, Sharad Goel, and Andrew Gelman. 2018. Disentangling Bias and Variance in Election Polls. *J. Amer. Statist. Assoc.* 113, 522 (2018), 607–614.

[39] Wei Wang, David Rothschild, Sharad Goel, and Andrew Gelman. 2015. Forecasting Elections with Non-Representative Polls. *International Journal of Forecasting* 31, 3 (2015), 980–991.