DevOps for DataVis: A Survey and Provocation for Teaching Deployment of Data Visualizations

Jane L. Adams

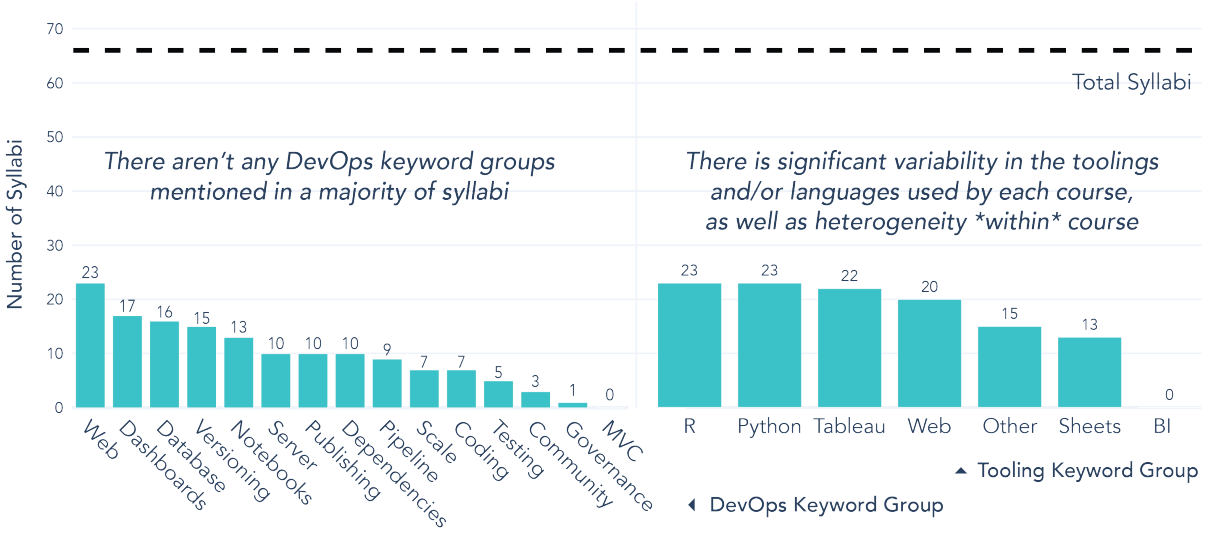

Fig. 1: Syllabi mentioning devops and tooling keywords. No DevOps keyword groups were mentioned in even close to a majority of syllabi. Is this because they are assumed as a prerequisite? Not taught? or taught but not considered noteworthy? Regarding tooling keywords, note the non-dominated front: no one tooling group was mentioned on much more than a third of all syllabi. Some courses mention multiple toolings – does this improve breadth of understanding, or introduce confusion?

Abstract—We present a provocation towards teaching development operations ("DevOps") and other infrastructure concepts in the course of collegiate data visualization instruction. We survey 65 syllabi from semester-long, college-level data visualization courses, with an eye toward languages and platforms used, as well as mentions of deployment related terms. Results convey significant variability in language and tooling used in curricula. We identify a distinct lack of discussions around 'DevOps for DataVis' scaffolding concepts such as version control, package management, server infrastructure, high-performance computing, and machine learning data pipelines. We acknowledge the challenges of adding supplemental information to already dense curricula, and the expectation that prior or concurrent classes should provide this computer science background. We propose a group community effort to create one free 'course' or 'wiki' as a living reference on the ways these broader DevOps concepts relate directly to data visualization specifically. A free copy of this paper and all supplemental materials are available at `https://osf.io/bxaqz/`.

Index Terms—Computing, infrastructure, deployment, software engineering, education.

1 Introduction

There exists significant heterogeneity in the content of collegiate data visualization curricula, both with regard to content and tooling. Some of these differences can be explained by the programs in which these courses are housed, which may range from social science to machine learning— the inherent symptoms of a highly interdisciplinary field of study. Likewise, there is tremendous variability in the existing familiarity students have with the technologies and languages used in these data visualization courses. The result of this diversity can be productive, as courses can theoretically cater more narrowly to the direct needs of students in a particular program; but they can also create problems. Students may complete a course feeling confident in their ability to code interactive visualizations, only to face confusing and complex battles in deploying these visualizations for use in a portfolio or in the context of building a dashboard for an employer. In these latter cases, it may have been beneficial for the student to have encountered educational scaffolding related to deployment and infrastructure – development operations, or "DevOps" – during their coursework.

This is a symptom also of the 'gap' between academic research and industry practice, as described by Velt et al. [19], investigated by Parsons through interviews with practitioners [17], and discussed in the VisGap workshops of 2021-2023 [5, 7, 11]. As the proportion of PhD graduates heading to industry surpassed academia for the first time in 2020, and continues to rise, educational aims necessarily should consider the needs of industry positions [13]. Concurrently, as visualization researchers increasingly encourage one another to consider the long term reusability of research prototypes, the value of lessons in these concepts extends beyond the classroom [11]. A search of "data visualization" and "deployment" yields a wealth of examples where visualization has been employed to track or triage DevOps infrastructure ("DataVis for DevOps"). However, research, and academic instruction, are scant in describing how DevOps concepts and methods apply to the

- *Jane Adams is with Northeastern University. E-mail: adams.jan@northeastern.edu*
- *Conflict of Interest (COI) Disclosure: Jane Adams is on the steering committee of alt.VIS, and was an organizer in 2021 and 2022.*

structure and multi-modal dissemination of data visualizations ("DevOps for DataVis"). Additionally, data visualization educators may see more complex computer science topics as outside their purview or pedagogical responsibility. Likewise, pre- and co-requisite classes may not consider data visualization applications of DevOps concepts relevant enough to their students to teach in, for example, an 'Intro to CS' course.

To better understand the diversity of tooling and DevOps content in college-level data visualization courses, we conduct a convenience sampled survey of recent syllabi, looking for keyword groups that would identify specific language(s) (e.g., "Python", "R", "Web", "Tableau") and might indicate deployment considerations (e.g., "Publishing", "Databases", "Versioning", "Servers", and "Data Governance"). Overall, we find that there is notable heterogeneity in tooling used across curricula, with a nearly four-way tie in mentions between R, Python, Tableau, and the web stack (HTML/CSS/JavaScript, e.g. D3.js). Additionally, one quarter of all syllabi surveyed contain no mention of *any* keywords related to DevOps. These results suggest a contributing factor to the gap between academia and industry in visualization, and may indicate a potential 'hidden curriculum' that is currently missing from visualization education, à la "Missing Semester of Your CS Education" [2].

We do not presume to provoke changes to the core structure of data visualization curricula: these courses are diverse in tooling and subject matter by design, and fulfill different needs for students with different expertise levels, publication needs for their visualizations, and career objectives [3]. They may be offered at different points in students' program curriculum, and have varying prerequisites. Additionally, unlike other components of a visualization curriculum (such as perception and cognition, marks and channels, and considerations to embellishment versus clutter), DevOps tooling changes at a rapid pace, which risks the addition of specific tools or technology stacks quickly becoming obsolete as lessons are recycled in subsequent years. Instead, we recommend a 'living document', such as a wiki or open web course, to house these dynamic learning resources, shared amongst the visualization education community, from which educators or students might select a subset of relevant concepts. In particular, we posit that there are fundamentally unique concepts from DevOps that apply in unusual or nuanced ways to Data Visualization, such that other basic computer science scaffolding resources may be irrelevant, overly complex, or incomplete.

2 Background

To understand how development operations hold relevance for teaching data visualization, we introduce some background on DevOps as a field and its relation to data visualization; and the need for scaffolding in computer science education broadly, along with existing models of remedies for 'missing' parts of that scaffolding.

2.1 DevOps

A survey of papers about "DevOps" by Leite et al. reviews concepts and challenges of the field [14]. They organize the field of DevOps into Dev (development-related) and Ops (operations-related). Within development, the engineering perspective focuses on delivery, while the management perspective is on process. On the operations side, the engineering focus is runtime, and the management perspective is toward people. A summary of this conceptual framework, with key terms from each category, is included in Table 1 [14]. Much like data visualization, the field of DevOps is deeply collaborative and interdisciplinary as it attempts to bridge the gap between developers and operators in the context of production environments. We include some ideas of concepts specific to the data visualization process in each category, and show how across development and operations, there are different foci. The 'Management' title of the Process/People section of the mapping may require a more flexible name: in the data visualization development process, the roles of 'manager' and 'engineer' are less distinct.

There are two facets to the concern of reproducibility as it relates to deployment in data visualization. The first is on the academic side: in order for scientific research to be reproducible, or even usable, students and faculty alike often require knowledge of systems such as version control and package management, to ensure that code compiles consistently over time and environments [6, 11]. Second, when academics collaborate with industry, there is a concern about "matters of care", as elucidated by Akbaba et al. [1]. For example, they identify maintenance as one critical area of neglect when visualization academics work with industry collaborators on a tool. Similarly, Walny et al. in their review of several large data visualization projects identify specific gaps in the data visualization design 'handoff' to collaborators: Adapting to data changes, anticipating edge cases, understanding technical challenges, articulating data-dependent interactions, communicating data mappings, and preserving data mapping integrity across integrations [20]. All of these steps happen between project conceptualization and deployment and use, and are iterative steps of the visualization design process itself. In their 2020 meta-analysis, Battle and Scheidegger highlight the need for collaboration between the visualization and data management communities to address the conflict between computational constraints and user-centric design concerns, proposing optimization standards, redefining visualization contributions, and fostering interdisciplinary collaborations as potential solutions [4]. All of these considerations overlap with the field of DevOps in some way, and we expect that implementing these recommendations will necessitate discussions of development operations within visualization education.

	Engineering	**Management**
Development	Delivery: Complexity, automation, architecture, versioning DV: Interactivity, state management, modularity / reuse, data infrastructure	Process: Product cycles, compliance, measurement DV: Task analysis, user studies, personas
Operations	Runtime: Security, stability, A/B testing, scalability DV: Data privacy, load-on-filter, map tilings, labeling, JIT visualizations, caching views	People: Collaboration, teams, knowledge, autonomy DV: Adoption, extension, graphical communication, visual hypothesis generation; dashboard-as-a-service

Table 1: A table summarizing the conceptual map described by Leite et al. in their survey of DevOps concepts and challenges [14], along with suggestions we have made for how these conceptual areas might map to data visualization ("DV") development ideas.

2.2 Education

It has been well-documented that "scaffolding" can improve learning outcomes for students. One meta-analysis, specifically of computer-based scaffolding studies in STEM education, found that scaffolding had a significant effect on the cognitive outcomes of problem-based learning ("PBL") [12]. Scaffolding can be divided into two types: "conceptual scaffolding" helps students connect pieces of evidence into a cohesive mental model, understanding interactions between elements; "strategic scaffolding" helps students navigate a problem space with a targeted approach to interacting with evidence. In the context of deployment considerations for data visualization, conceptual scaffolding might involve teaching students a mental model of the relationships between e.g., data sets, automated analysis pipelines, APIs, caching, modules, packages, compilation, repository, server, and client. A strategic scaffolding approach might focus directly on a targeted deployment stack: e.g., how to get data from a public health resource into an interactive visualization embedded in a blog post about epidemiology, using Excel, DataWrapper, and Substack. The "Zone of Proximal Development" (ZPD) exists between what a student can accomplish independently, and what a student is unable to do. It consists of tasks that a student can accomplish *with help*. The purpose of scaffolding is to enable students to move into this learning zone, such that scaffolding can be removed ("fading") as mental models form and students are more able to solve problems independently [15, 21]. In the context of data visualization education, this might consist of early assignments being presented with a schematic of the mental model for where data lives, how it flows, and what components are implicated (server, client, memory, database, etc.), while later assignments might leave the drawing of these schematics to the student.

Especially in rapidly changing fields like computer science, the challenges of percolating down new information to curricula is significant. *Data engineering*—a distinct task from the upstream of modeling and analysis of data science—continues to be a source of frustration and significant time expenditure for practitioners, yet under-represented in educational material [10]. For non-majors taking introductory CS courses, fears can range from perceptions of STEM as difficult, to more concrete challenges like coding and preparation [9]. Data visualization is a notably interdisciplinary field, with many newcomers from non-CS backgrounds, so it is not surprising that questions about how to teach development in data visualization education are growing areas of consideration [3]. Courses such as MIT's "The Missing Semester of Your CS Education" [2] and a similar course developed at UC Davis [8] show promise as examples of supplemental online course materials to onboard students to more mundane or 'meta' CS concepts.

Happily, we do see some great examples of scaffolding efforts in some data visualization curricula. Two courses that we identified as particularly effectual at covering a wide variety of concepts included Columbia University's 2016 'Metrics + Data Visualization I' course, taught by Aurelia Moser [16]; and its 2017 course 'Data Visualization for Architecture, Urbanism and the Humanities", taught by Juan Francisco Saldarriaga [18], which cover concepts such as version control, scraping, and APIs.

3 Methods

To conduct our study on the integration of DevOps concepts in collegiate data visualization instruction, we employed the following methods:

3.1 Collection

We included syllabi from college-level data visualization courses using a convenience sampling approach. We conducted Google searches using the keywords "data visualization syllabus", "data visualization course", "data visualization filetype:pdf", and "data visualization site:github.com". We also reached out on vis.social, a federated social media server which caters toward visualization academics and practitioners, using the #DataVisualization, #HCI, and related hashtags, asking for links to syllabi that met our criteria. We excluded non-English syllabi, workshops, short courses, and non-college sources. We did not discriminate between computer science and non-computer science courses, or between graduate and non-graduate courses, because many courses appeared to be offered to both undergraduates and graduates, or simultaneously across multiple academic programs, and at minimum, introduction to these mental models are useful for students at all levels. Additionally, our intention was to source a wide sample, so as to highlight the variability in curricula. We do not break out our analyses by these sub-groupings, but future analysis might find distinctions in material between course levels or parent disciplines.

3.2 Coding

We used a keyword-driven, thematic open coding approach [22] to analyze the syllabi obtained from various data visualization courses. Each syllabus was carefully reviewed via close reading in order to tag these categories, which were initially formulated following literature review of DevOps considerations, but added to, adapted, merged, or divided during the open coding process. We chose a manual coding approach over Natural Language Processing ("NLP"), because of concerns related to lemmatization, namespace collisions, or overlapping usage. We did not code requirements for completing coursework (e.g. LaTeX for write-ups; hardware system requirements), privacy policies, software subscription costs, or other themes and topics that were not directly related to DevOps concepts. We left out Geospatial terms (such as ArcGIS, MapBox, Leaflet), as it was often difficult to ascertain the extent to which these were used as tooling, data sources, packages within other tools, or for deployment. 'Web' appears twice because as a tooling keyword group, this encapsulates terms such as HTML, CSS, and Javascript, whereas in the DevOps keyword group, 'web' refers to web terms, such as "web-based", "web apps", internet, and browser.

Table 3: A table of tooling keyword groups and associated syllabi counts.

Tooling	Keywords	Syllabi
R	R, ggplot, Plotly R, Shiny, dplyr, tidyverse	23
Python	Python, Plotly Python, Bokeh, Vega-Altair, SciPy, Jupyter notebooks, Google Colab, Flask, Streamlit, bqplot	23
Web	JavaScript, D3.js, Vega-lite, SVG, HTML, CSS, React, Node, JS	20
Tableau	Tableau Public, Tableau Desktop	22
Sheets	Excel, Google Sheets, spreadsheets	13
BI	PowerBI, Qlikview, Qlik, Business Intelligence	0
Other	DataWrapper, MATLAB, Processing, Adobe Illustrator, Google Charts	15

Table 4: A table of DevOps keyword groups and associated data visualization syllabi counts.

DevOps		
Server	Server, host(ed/ing), cloud, deploy(ment, ing, ed), Node, Heroku, Vercel, AWS, GCP, Netlify, Github Pages, nginx, WSGI, Flask	10
Publishing	Publish(ed/ing), embed(ded), posting (excluding to LMS)	10
Dashboards	Dashboard(s/ing), Streamlit, Shiny, Dash	17
Versioning	Git, GitHub, GitLab, BitBucket, version control, versioning	15
Dependencies	Dependencies, package management, Anaconda, Conda, Pip, PyPI, CRAN, npm, virtual environment, libraries	10
Database	database, API, (R)DBMS, (No/My)SQL, MongoDB, 'big data', data management	16
Web	web (-page, -based, application, apps), internet, browser, developer, client-side	23
Pipeline	Pipeline(s), data flows, ETL, automat (-ion, -ed) (excludes data cleaning unless automated)	9
Scale	scal(e/ing), latency, speed, load balancing, HPC	7
Community	Stack Overflow, Reddit, Discord, Slack (public, excluding course chat)	3
Governance	data governance, access, passwords, PII, OAuth, security, credentials, .env, authentication, SSO	1
Notebooks	Notebooks, Observable, Jupyter Notebooks, R Markdown, .ipynb, .Rmd, Google Colab, JupyterHub, IPython, Iodide	13
Coding	IDEs, VSCode, IntelliJ, PyCharm, Processing, Terminal, Vim, Vi, Emacs, bash, console	7
Testing	Unit tests, QA testing, quality assurance, bug fixes, red teaming	5
MVC	MVC, Django, Ruby on Rails, AngularJS	0

4 Results

The most significant finding of this analysis is 1) the heterogeneity of tooling across curricula, and 2) the lack of DevOps terms mentioned on syllabi, which may serve as a proxy for classroom time expended teaching these concepts related to deployment. Features of each technology are present in the co-occurrence: for example, Python keywords may co-occur with Notebooks keywords, likely because syllabi that mention Python might also mention Jupyter Notebooks or Google Colab. Likewise, syllabi that mention Tableau might also mention the deployment of dashboards.

4.1 Tooling

There was significant heterogeneity in the tooling keyword groups mentioned across the syllabi surveyed, as shown in Fig. **??**. We found "R", "Python", "Tableau", and "Web" to each be found in about one third of syllabi. "Other" and "Sheets" were found in about a quarter of all syllabi, and there were no mentions of the "BI" category. Note that the proportions do not add up to 100% because many syllabi mentioned multiple tool sets. Interestingly, several syllabi mentioned both R- and Python-related keywords. Syllabi with mentions of sheets-related terms, such as Excel or Google sheets, co-occurred most often with the Tableau keyword group. These co-occurrences are shown in Fig. 2. Using these co-occurrence summaries, we can see where certain tools might complement one another, or be offered as a choice to students. In the case of JavaScript and Python co-occurring, for example, we saw the languages being offered as a binary choice, depending on students' prior familiarity with each. Conversely, in the case of Tableau and sheets tools like Excel, these tools were taught in conjunction with one another as discrete steps of a data visualization design workflow.

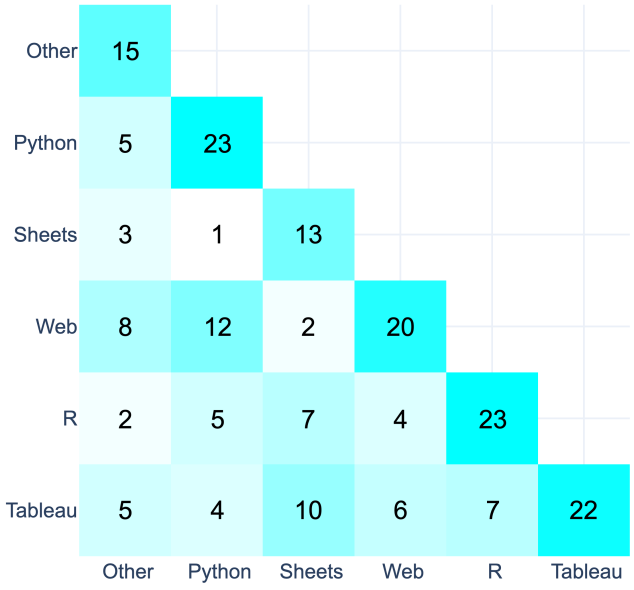

Fig. 2: Co-occurrence of tooling keywords. Of the one third of the total syllabi that mentioned Python keywords, half *also* mentioned web tool keywords (e.g. HTML, CSS, Javascript).

4.2 DevOps

Overall, we found that three quarters of syllabi surveyed contained mention of at least one of our DevOps keywords, and one quarter did not make any mention of DevOps keywords. The most common keyword groups included the 'Web' category, followed by 'Dashboards', 'Database', and 'Versioning', as shown in Fig. **??**. Rarely mentioned keyword groups included concepts such as 'Model-View-Controller' (MVC), e.g. Ruby on Rails and Django (never mentioned); 'Governance', which included security and privacy considerations; 'Community', in reference to resources for connecting with other coders, such as Stack Overflow; 'Testing', encapsulating unit tests, QA testing, and debugging; and 'Coding', with concepts such as IDEs and terminal commands.

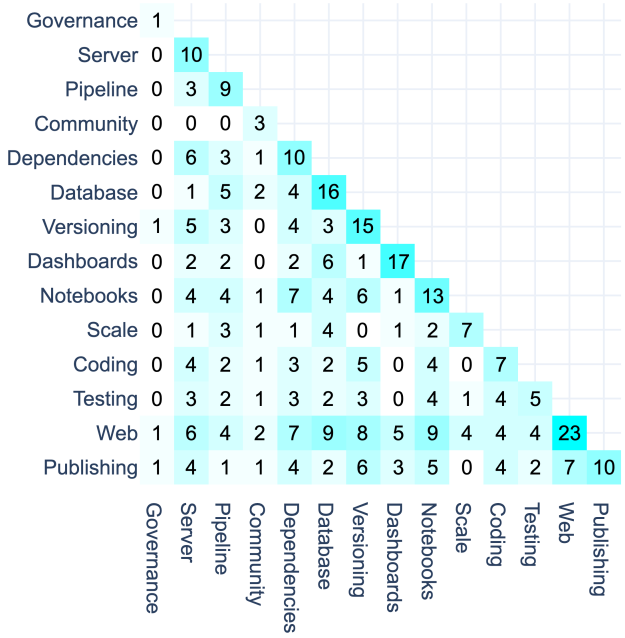

Fig. 3: Co-occurrence of DevOps keywords. Of the one third of all syllabi that mentioned notebook-related terms (e.g. Jupyter Notebooks, Google Colab, Observable), nearly half *also* mentioned web terms. Notebooks and dependency considerations (e.g. package management) are often mentioned together, perhaps because notebooks can provide a means of more easily syncing environments for different users.

5 Discussion

We found significant variations in tooling from course to course, which may not necessarily inhibit learning, but can make for a rather complex landscape of platforms, languages, and acronyms for newcomers to navigate. For courses that allow students to create visualizations in one choice language of many, it is not clear whether the curriculum includes a discussion about the affordances of each, such as: compute time, syntax, data structures, readability, financial cost, or real-world adoption. We posit that in some cases, within-course heterogeneity of toolings may be useful for providing varied exposure to visualization methods, or could prove overwhelming or irrelevant. Likewise, between-course heterogeneity may indicate frameworks that are catered toward the disciplines in which the courses are housed; or, could hint at indecisiveness or confusion in choosing optimal toolings when developing curricula. A more comprehensive survey might break out these courses by audience to identify which toolings are most popular in each discipline.

Overall, we find a dearth of information for students in visualization courses related to DevOps. In some cases, the absence of this content may be due to instructors seeing many of these concepts as 'table stakes': an assumed baseline for entry to begin a course. Indeed, some syllabi explicitly outlined prerequisite coursework or requirements for existing familiarity with concepts like git, or programming in general. However, the technologies and mental models associated with specifically the deployment of data visualizations may be more nuanced or complex than students' baseline familiarity with, e.g., building a shopping cart app, participating in a hackathon, or contributing to an open

source project, and these assumptions about students' familiarity could often be off-base. Additionally, the greater the number of prerequisites for a data visualization course, the greater the barrier to entry is for students from non-CS backgrounds, whether newly entering computer science or looking to enhance a non-CS course load with visualization experience. Finally, we notice particular lack of education around DevOps considerations which may have been a non-issue just five years ago, but are of increasing importance in real-world visualization applications, such as: Scaling, including High-Performance Computing (HPC), latency considerations, and load balancing; Databases; and automated pipelines, e.g. machine learning workflows. These topics have grown in relevance recently due to advances in compute power, data collection efforts, and artificial intelligence developments. Of course, these topics have their own courses and entire programs, but we saw a number of syllabi that mentioned concepts from these domains, such as dimensionality reduction and modeling. However, while the methods were alluded to, there remains a notable lack of discussion around the DevOps that these methods impel. We also expected to find that there might be a clear difference or grouping of 'CS' versus 'Non-CS' courses in terms of prerequisites, DevOps keywords, or even course titles, but we did not see a clearly identifiable pattern. It seems that non-CS courses varied in devoting more or less syllabus space to DevOps concepts due to their unfamilarity for students; likewise, some CS courses may have devoted more or less time based on the assumption that students were already familiar with the concepts.

5.1 Note on Motivation

We would be remiss if not also acknowledging personal exasperation with and desire for better preparatory coursework in the realm of DevOps for data visualization. Direct experience with these frustrations by the author of this paper in part motivated the data collection effort and recommendations made. Outside of the careful coding process described here, this topic was motivated also by spontaneous conversations among fellow PhD students surrounding confusion and feelings of being 'lost' in a sea of new web technologies and complex data management structures. While these sentiments have not been fully quantified as yet, the visualization research community may find that in developing these resources, thereby acknowledging this knowledge gap, more students feel comfortable admitting their sense of aimlessness or overwhelm on these facets of data visualization creation.

Many data visualization courses in Python implicitly, but not explicitly, teach a simple model of data and interaction, such as a locally hosted data file being parsed in a Jupyter notebook. However, in a 'real world' scenario, the considerations that must be made to existing institutional infrastructure and various stakeholders exceeds simply performing task analysis or interviews, and necessitates a fundamental technical understanding of the components at play in the deployment of the final visualization. This additional knowledge could include concepts such as CI/CD, virtualization and containerization, authentication, and other such DevOps terms as outlined in Table 1.

5.2 Recommendations

We propose that the community set to task developing a living document that can serve as a resource to unite these disparate curricula, and offer students a more complete picture of 1) the tooling options within visualization design, and 2) mental models associated with DevOps, which include considerations for deployment, testing, collaboration, and maintenance. We suggest MIT's "The Missing Semester of Your CS Education" as an example of a simple online course that can provide quick navigation to basic concepts that scaffold a visualization education. In Table 2.1, we sketch out the beginnings of a Venn diagram showing some overlapping concepts between the MIT course and our open-coded DevOps keyword categories. This overlap demonstrates how, in some areas, efforts may not need to be reduplicated, and references to existing resources can be easily linked. However, in other areas, such as publishing dashboards, or the specifics of how notebooks render visualizations, we suspect that more careful and nuanced input from the visualization community (both in academia and industry) may be necessary to sufficiently onboard or supplement students' understanding.

5.3 Limitations & Future Work

It is important to note that our study had limitations, such as potential biases in the selection of syllabi due to convenience sampling and the exclusion of non-English sources. The syllabi sampled over-represents the United States and Europe. We observed a higher ratio of code-based technologies (e.g. Python, R, Javascript) as compared to low- or no-code technologies (e.g. Tableau, BI, spreadsheets) in syllabi from Github as compared to syllabi found via Google Search: prior to searching GitHub, the *majority* of syllabi did not contain DevOps keywords. Additionally, syllabi do not contain the entirety of course material. It is possible, and even likely, that mention was made of these keyword groups in individual lesson plans, or on additional resource links. We do not presume that syllabi are full representative accountings of classroom material. A more robust collection effort might have 'bombed' all links on a course website, or performed Open Character Recognition (OCR) on class slides or lecture recordings to pull additional text. Professors may devote significant time in-class to teaching these tools, but not see value or significance in documenting them on the syllabus due to their auxiliary (albeit essential) nature. Despite these shortcomings, these results reflect our findings in reading these syllabi: that conversations about deployment and infrastructure rarely factor significantly or comprehensively into data visualization course material as it is shared in the semester lesson plan. Further surveys should be conducted which more completely sample the visualization higher education space, and to quantitatively compare the abundance of these terms in curricula with job descriptions and documentation of practice in industry. Nonetheless, we believe that our findings provide valuable insights into the current state of data visualization education and the need to supplement curricula with a dynamic compendium of DevOps learning resources.

6 Conclusion

In conclusion, we put forth a provocation towards developing new community resources to teach students about DevOps for DataVis, and support this provocation with coded readings of syllabi from college courses in data visualization. We identify and discuss the reasons for, and challenges of, the wide variety of toolings employed by different courses. In searching specifically for DevOps related terminology, we identify conceptual areas and concrete applications that may be omitted from existing courses, which may be necessary teachings for students preparing for careers in industry. We make a recommendation for a low-stakes, dynamic resource that can be shared amongst the community to aid in scaffolding a more complete education in data visualization. Such a resource should be easily editable and discussed, and rely on outside resources to stay current, such as by interfacing with existing documentation for various toolings. Other content, such as conceptual models (thinking beyond the internet as 'a series of tubes', for example) may remain current for longer due to their tooling-agnostic focus.

Acknowledgments

Thanks to Andrew McNutt for the term 'table stakes', links to some syllabi, early discussion of some of these ideas, and editing and feedback. Thanks also to Andrew Heiss, Eduardo Graells-Garrido, Jack Dougherty, Tamara Munzner, Marian Dörk, Alexander Lex, and T. from Data Rocks for sharing syllabi and other resources. Thanks to Michael Davinroy also for editing help.

References

[1] D. Akbaba, D. Lange, M. Correll, A. Lex, and M. Meyer. Troubling collaboration: Matters of care for visualization design study. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–15, 2023. 2

[2] A. Athalye, J. Gjengset, and J. J. Gonzalez. The missing semester of your cs education, January 2020. 2, 3

[3] B. Bach, S. Carpendale, U. Hinrichs, and S. Huron. Visualization empowerment: How to teach and learn data visualization (dagstuhl seminar

22261). In *Dagstuhl Reports*, vol. 12. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023. 2, 3

[4] L. Battle and C. Scheidegger. A structured review of data management technology for interactive visualization and analysis. *IEEE transactions on visualization and computer graphics*, 27(2):1128–1138, 2020. 2

[5] T. Dang, N. Nguyen, J. Hass, J. Li, Y. Chen, and A. Sill. The gap between visualization research and visualization software in high-performance computing center. *VisGap2021-The Gap between Visualization Research and Visualization Software*, 2021. 1

[6] J.-D. Fekete and J. Freire. Exploring reproducibility in visualization. *IEEE Computer Graphics and Applications*, 40(5):108–119, 2020. doi: 10.1109/MCG.2020.3006412 2

[7] C. Gillmann, M. Krone, G. Reina, and T. Wischgoll. VisGap 2022: Frontmatter. In C. Gillmann, M. Krone, G. Reina, and T. Wischgoll, eds., *VisGap - The Gap between Visualization Research and Visualization Software*. The Eurographics Association, 2022. doi: 10.2312/visgap.20222006 1

[8] G. Gilson, S. Ott, N. Rose Ledesma, A. Prabhu, and J. Porquet-Lupine. Design and evaluation of "the missing cs class," a student-led undergraduate course to reduce the academia-industry gap. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1*, SIGCSE 2022, p. 467–473. Association for Computing Machinery, New York, NY, USA, 2022. doi: 10.1145/3478431.3499422 3

[9] E. Hogan, R. Li, and A. G. Soosai Raj. Cs0 vs. cs1: Understanding fears and confidence amongst non-majors in introductory cs courses. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, pp. 25–31, 2023. 3

[10] B. Howe, M. Franklin, L. Haas, T. Kraska, and J. Ullman. Data science education: We're missing the boat, again. In *2017 IEEE 33rd international conference on data engineering (ICDE)*, pp. 1473–1474. IEEE, 2017. 3

[11] T. Isenberg. Personal experiences of providing and using research prototypes. In *Proceedings of VisGap – The Gap between Visualization Research and Visualization Software (Workshop at EuroVis)*, pp. 17–22. Eurographics Association, Goslar, Germany, 2022. doi: 10.2312/visgap.20221059 1, 2

[12] N. J. Kim, B. R. Belland, and A. E. Walker. Effectiveness of computer-based scaffolding in the context of problem-based learning for stem education: Bayesian meta-analysis. *Educational Psychology Review*, 30:397–429, 2018. 2

[13] F. Kreier. How mixing academia and industry opens doors in graduate school and beyond. *Nature News*, May 2023. 1

[14] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles. A survey of devops concepts and challenges. *ACM Comput. Surv.*, 52(6), nov 2019. doi: 10.1145/3359981 2

[15] N. D. Martin, C. Dornfeld Tissenbaum, D. Gnesdilow, and S. Puntambekar. Fading distributed scaffolds: The importance of complementarity between teacher and material scaffolds. *Instructional Science*, 47:69–98, 2019. 2

[16] A. Moser. Metrics + data visualization i - sva-dsi syllabus. https://mzl.la/sva-vis-1, Sept. 2016. Accessed: 2023-07-07. 3

[17] P. Parsons. Understanding data visualization design practice. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):665–675, 2021. 1

[18] J. F. Saldarriaga, B. Wanner, and M. Madera. Data visualization for architecture, urbanism and the humanities syllabus. https://github.com/juanfrans-courses/dataViz_arch_hum/blob/master/Spring_2017/Syllabus.md, Spring 2017. Accessed: 2023-07-07. 3

[19] R. Velt, S. Benford, and S. Reeves. Translations and boundaries in the gap between hci theory and design practice. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 27(4):1–28, 2020. 1

[20] J. Walny, C. Frisson, M. West, D. Kosminsky, S. Knudsen, S. Carpendale, and W. Willett. Data changes everything: Challenges and opportunities in data visualization design handoff. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):12–22, 2020. doi: 10.1109/TVCG.2019.2934538 2

[21] A. West, J. Swanson, and L. Lipscomb. Ch. 11 scaffolding. *Instructional methods, strategies and technologies to meet the needs of all learners*, 2019. 2

[22] M. Williams and T. Moser. The art of coding and thematic exploration in qualitative research. *International Management Review*, 15(1):45–55, 2019. 3