Journal of Statistics and Data Science Education

ISSN: (Print) (Online) Journal homepage: https://amstat.tandfonline.com/loi/ujse21

Collaborative Writing Workflows in the Data-Driven Classroom: A Conversation Starter

Sara Stoudt

To cite this article: Sara Stoudt (2022) Collaborative Writing Workflows in the Data-Driven Classroom: A Conversation Starter, Journal of Statistics and Data Science Education, 30:3, 282-288, DOI: 10.1080/26939169.2022.2082602

To link to this article: https://doi.org/10.1080/26939169.2022.2082602

Published online: 23 Jun 2022.

JOURNAL OF STATISTICS AND DATA SCIENCE EDUCATION
2022, VOL. 30, NO. 3, 282–288
https://doi.org/10.1080/26939169.2021.2082602

Collaborative Writing Workflows in the Data-Driven Classroom: A Conversation Starter

Sara Stoudt

Department of Mathematics, College of Arts and Sciences, Bucknell University, Lewisburg, PA

ABSTRACT

To paraphrase John Tukey, the beauty of working with data is that you get to "play in everyone's backyard." A corollary to this statement is that working with data necessitates collaboration. Although students often learn technical workflows to wrangle and analyze data, these workflows may break down or require adjustment to accommodate the different stages of the writing process when it is time to face the communication phase of the project. In this article, I propose two writing workflows for use by students in a final-project setting. One workflow involves version control and aims to minimize the chance of a merge conflict throughout the writing process, and the other aims to add some level of reproducibility to a Google-Doc-heavy writing workflow (i.e., avoid manual copying and pasting). Both rely on a division of the labor, require a plan (and structure) to be created and followed by members of a team, and involve communication outside of the final report document itself. This article does not aim to solve all collaborative writing pain points but instead aims to start the conversation on how to explicitly teach students not only how to code collaboratively but to write collaboratively.

ARTICLE HISTORY
Received September 2021
Accepted May 2022

KEYWORDS
Communication skills; Data-intensive reports; Pedagogy; Teamwork; Version control

1. Introduction

Authentic values and workflows of statisticians and data scientists are increasingly being taught to students even as early as the introductory course (Carver et al. 2016; De Veaux et al. 2017). In particular, an emphasis on reproducibility has brought tools like R Markdown to introductory courses to allow code results and narrative to be interwoven (Baumer et al. 2014; Xie, Allaire, and Grolemund 2018). Tools like R Markdown and other computational-notebook structures appropriately blur the lines between the analysis workflow and the communication workflow (Rule, Tabard, and Hollan 2018). With both parts of a data-intensive pipeline on equal footing, students see that analysis and communication skills are both integral to finding meaning in data. Communication of findings cannot be an afterthought, and the National Academies of Sciences, Engineering, and Medicine agrees, identifying communication, along with workflow, reproducibility, and teamwork as part of "data acumen" (National Academies of Sciences, Engineering, and Medicine 2018).

Admittedly, we as instructors often focus more explicitly on the traditional content and computational material while ensuring that students have the opportunity to practice and strengthen the data acumen skills in service of mastering the content. However, we may teach a reproducible, computational workflow without explicitly talking about the *process* of communication, whether individual or collaborative. For example, when R Markdown is used for tasks like homework, students learn to work with this tool by themselves, without the distraction of coordinating with a partner or team. When used for tasks like in-class labs, students learn to work with this tool with a partner or small group, but this collaboration typically happens (at least mostly) synchronously. However, when students move on to a final project that requires a longer-term workflow and necessitates students working asynchronously or perhaps simultaneously yet in separate locations or on separate tasks, they may need to adapt their workflow.

In this article, I propose a couple of paths forward, hoping that they will start a conversation about reproducible workflows that go beyond code to include the important communication part of a data-analysis pipeline. I start with an outline of potential tools and approaches and then propose two workflow strategies that aim to be compromises between what is possible and what is easy to implement in classrooms. These strategies were devised especially with introductory courses that use R in mind (the focus on R is merely based on my own experience, not a commentary on what software should be used in these types of courses). Typically, there is much technical content to cover in these courses (so class time is precious), and students are not expected to have any experience with computing before joining the class (so learning a wide variety of tools may be a stretch).

2. What is Possible versus What is Easy

As part of an effort to give students a hands-on, authentic data experience as soon as possible, statistics and data science courses

CONTACT Sara Stoudt *sas072@bucknell.edu* Department of Mathematics, College of Arts and Sciences, Bucknell University, Lewisburg, PA, 17837-2010.

at the introductory level often conclude with a final project that involves data wrangling, data analysis, and presentation of results in both oral and written forms. By the time students start working on the project, they have become familiar with a particular computing workflow, but when faced with writing a report, this workflow may break down.

Consider a group of four students working together on a final project. In order to do the data wrangling and analysis necessary for the project, they may choose to "divide and conquer," contributing code and output to a central location. When it comes time to write the final report, they might want to divide the work amongst the members of the group in a similar way. Several different collaborative workflows may emerge.

Some courses may invest classroom time in teaching students to work with a version-control system like git or have a prerequisite that ensures students come in with this skill (Bryan 2018; Fiksel et al. 2019; Beckman et al. 2021). If students are already familiar with version control, then they may choose to store their report document within that system as well. The challenge then becomes stripping away the bad friction, that is, the stumbling blocks that come with tool usage that hinder the actual thinking or skill building, that distracts students from the writing process. For example, merge conflicts, when git requires a manual intervention to resolve differences between two versions of a file, arise as a major pain point both in anecdotes (Hart 2016) and in the literature investigating the use of version control and pedagogy (Brun et al. 2013; Haaranen and Lehtinen 2015; Case, Eloe, and Leopold 2016). We will have to keep this pain point in mind when proposing a collaborative writing workflow that involves version control.

Some statistics and data science courses will not cover a version-control workflow and do not have prerequisites in place to require students to come in with that knowledge, especially at the introductory level. For example, the instructor of a traditional introductory statistics course that not only serves statistics majors, but also the many other majors that require statistics to do their work, is faced with certain expectations of content to be covered that keeps the schedule pretty full. I have been in this situation, and anecdotally, I have seen students in an introductory statistics course write the main narrative of their final report in a Google Doc, copying and pasting the most recent figures and output as they go. Then they copy and paste the final text into their final R Markdown file before submitting.[1] Challenges here include who is in charge of the R Markdown file (usually one person is taking the driver's seat with respect to the actual coding), plots and results becoming stale in the Google Doc (leading to teammates writing about the wrong results), and formatting mishaps (when translating back and forth between Markdown and Google Doc syntax). I have seen this same collaborative writing workflow used by upper-level students who have used both R and R Markdown before and have a little experience with version control. They may be using version control for the actual code requisite for the project, but the final report is managed outside of that workflow.

[1] On May 23, 2022 RStudio Cloud actually did release a collaborative editing feature (in beta). *https://www.rstudio.com/blog/real-time-collaborative-editing-on-rstudio-cloud/*

These collaboration challenges faced in the classroom mirror those we may face in the workplace, namely different members of the team having different experience levels with the tools necessary for a completely reproducible workflow and idealized workflows breaking down under constraints. We want to encourage students to maintain a sustainable yet reproducible workflow, minimizing copying and pasting across multiple documents, at the final writing stage of a project. However, we need to do this while respecting the limitations of the classroom time and instructor expertise needed to teach new tools in service of this initial goal.

3. Envisioning a Compromise

Ideally, what we would want is a hybrid of Google Docs and R Markdown where multiple people can be editing an R Markdown file simultaneously. However, even advanced and specialized services like the RStudio Cloud do not yet have this feature (RStudio Community 2018, 2019, 2020). Can we piece together a collaborative writing workflow that approximates this desired approach without too much logistical overhead?

3.1. A Streamlined-Version-Control Compromise

This workflow combines a version-control workflow with some added guard rails to facilitate the writing process. The primary goal of this proposed workflow is to minimize the chance of a merge conflict throughout the writing process.

One way to avoid merge conflicts is to break up the final project report into stand-alone pieces that are stored in separate R Markdown files and can be worked on by different members of the team at the same time. With separate files for parts of a report like the introduction, data description, methods description, results, discussion, and conclusion, individuals on a team could then "push" their edits freely without fear of merge conflicts. No other team member should be touching their particular file during a given time period. These pieces of the written narrative can then be brought together into one report at any time. In addition, since the subfiles are R Markdown files, students can also break up the coding work as well. Logistically, it is possible to join multiple R Markdown files into one file, but it may be prudent to take advantage of new formats such as *bookdown* that specialize in structuring technical documents and provide added features like cross-referencing across subfiles (Xie 2021).

How can we make this divide-and-conquer process easy for students? Instructors could create a template repository on GitHub with a skeleton *bookdown* structure that has different report subsections contained in different files that are then brought together into the main document (typically index.Rmd in a *bookdown* approach) (GitHub Docs 2021a, 2021b; Robbins 2020). Instructors could even create templates that would make it straightforward to submit a class project to an undergraduate research journal or competition without the added friction of having to migrate to a particular structure or format (Journal of Undergraduate Research 2021; CauseWeb 2021). Instructors of more advanced students might want teams to create their own writing outline and associated subfiles. This would provide a good opportunity to teach students explicitly about the structure of a formal, data-intensive research report and the common practice of writing nonlinearly (Nolan and Stoudt 2021a).

This proposed collaborative writing workflow, using a version-control system like git, would look something like the following:

- Students in a team create a copy of a report structure repository within their own repository.
- Students split up the writing (and coding) tasks so that no more than one person is working on each subsection at a time. Students can "push" frequently without fear of merge conflicts.
- At any time, the full report can be recompiled in its entirety. Students should "pull" frequently to make sure the other subsections are up to date.

In this workflow, students get to practice their version-control workflow, but additional structure imposed in the beginning helps avoid technical stumbling blocks that can distract from the actual writing process.

3.2. A Version-Control-Free Compromise

This workflow adds some level of reproducibility to a Google-Doc-heavy writing workflow without relying on version control. The primary goal of this proposed workflow is to avoid manual copying and pasting across documents throughout the writing process.

There has been a lot of conversation among R developers about writing workflows, and there have been different approaches to try to ease the inherent challenges involved without relying on formal version control (rOpenSci 2016, 2018; Ross 2018). The *redoc* R package was an attempt to streamline the workflow between R Markdown and Microsoft Word users (Ross 2020). The *gdoc* package in R was an experiment to upload R Markdown files to Google Docs, and the *markdrive* package in R works to let users edit Google Docs in Markdown within R (Ross et al. 2016; McBain 2018).

There are some tools that come close to a Google-Doc-style user interface for R Markdown, most taking advantage of the Jupyter Notebook which can also run R code (Barba et al. 2019). For example, CoCalc allows for real-time, simultaneous editing in R Markdown files and in Jupyter Notebooks, with a user interface that mimics Google Docs, complete with other users' cursors. The free version has some major limitations though, including a restriction on using external data within the environment, the possibility of random restarts, and overall lower-quality server access. The CoCalc team advocates for a subscription to continue the service as a whole (Sagemath, Inc. 2022b), and there are academic licenses available (Sagemath, Inc. 2022a).

If analysis does not need to be done in a programming language like R or Python, the Integrated Statistical Learning Environment (ISLE) provides a more point-and-click alternative (Philipp 2021). With proper setup ahead of time by the instructor, teams have access to a joint document in the same interface as their data exploration pane. They can move output, like tables and plots, into a report document in real time without changing windows or browser tabs. However, because this is not a code-based analysis software, there is not the same level of reproducibility (e.g., a student can move a histogram into the report, but their teammate cannot reproduce that histogram automatically or make edits to elements like the labels).

Another code-enabled alternative is Google Colab which is integrated with Google Drive so that a user can share and collaborate within a Jupyter Notebook (Google n.d.-b). This resource is free, with some capacity limitations (memory, speed, runtime), but capacity is not guaranteed (Google n.d.-a). This resource is almost real-time, but there is a short delay when one person makes a change. For example, if two people edit the document simultaneously, one edit will be overridden when the page is refreshed (AguaClara Cornell 2018). Other limitations come with the use of R in this tool; R packages need to be installed each time the notebook is loaded, and this process can be slow (Watson 2020). Kaggle Kernels work similarly to Google Colab, using Jupyter Notebooks in addition to R Markdown and R and Python scripts, but have more packages installed by default (Tatman 2019). However, the collaborative workflow is a little more stilted. There is no automatic merging, so a user has to keep track of their collaborator's versions and avoid overwriting their own progress in their working draft (Plotts 2018).

These tools and approaches start to face down the problem, and in some use cases, may even serve as a complete (paid) solution. However, a recently developed, free tool, the R package *trackdown*, also provides a way to bridge the gap between workflows using Google Docs and R Markdown (Zandonella 2021a). This R package facilitates writing in Google Docs with automatic transfer of content to an R Markdown file, no copy and paste necessary (Zandonella 2021b). Zandonella's *trackdown* package provides an interface for uploading an initial R Markdown document to Google Drive, from R itself, such that a Google Doc is created for easier editing of the narrative around the code. This Google Doc has extra headers and guidance that encourages writers to avoid editing portions of the document that involve code or output. The document can thus be edited live in terms of the narrative text while not affecting the code. The package also facilitates grabbing the most up-to-date version of the Google Doc within R itself and updating the R Markdown version of the document stored in a shared Google Drive at any time. There is also a mechanism for rendering the R Markdown file and sharing the output to Google Drive automatically so that writers can see the freshest output (albeit in a different file than the one they are editing).

Now the collaborative writing workflow looks something like this:

- One member of a team creates an initial R Markdown file and makes it available as a Google Doc to the rest of the team.
- The team members edit the Google Doc for the written portion of the project.
- One team member is in charge of the coding parts of the R Markdown file, making sure to make the most recent rendered report available via Google Drive to the rest of the team while they are writing and incorporating the final narrative into the R Markdown document at the end of the writing process.

This still relies on one person being in charge of the actual code chunks in the R Markdown file, but we can imagine a workflow where students each take a turn being "in charge" of the coding. In addition, this package is dependent on the Google Docs API which may change over time. However, what appeals to me about this approach is that it connects a commonly-used writing tool with a commonly used coding tool and models

a solution for the realistic scenario faced by data-intensive researchers where they may be the main person in a collaborative project who is doing the coding yet writing with a team of individuals who are not as comfortable with tools like R Markdown.

3.3. Team Delegation, Communication, and Integration

Both of these proposed collaborative writing workflows require a plan to be created and followed by members of a team as well as communication outside of the final report document itself. Navigating different roles and responsibilities throughout the timeline of a project introduces students to a communication skill of a different sort. This extra layer of coordination is not unlike team-based workflows found in non-academic data science environments (Wang et al. 2019; Chattopadhyay et al. 2020).

In the streamlined-version-control compromise, teams must decide who is working on which sections at what time and decide on checkpoints where they may be reviewing one another's work or switching responsibilities. These roles and tasks can be made explicit and transparent by using GitHub issues to track to-do items and the review of pull requests to track feedback on isolated parts of the project (Timbers, Campbell, and Lee 2022; GitHub 2022b; Timbers 2022b).

In the version-control-free compromise, suggestions must be resolved in the Google Doc before they can be brought back into the R Markdown file, and students must decide on a rotating schedule of who is in charge of the R Markdown document and code chunks within. To match the transparency of a version-control approach, teams could be encouraged to create a document that lays out the schedule of who has "checked out" the R Markdown document and when. This would signal that no one else should be changing that document, and code output including figures, tables, and results may be in flux. Similarly a record of when the R Markdown document has been "checked in" would signal that code output is up to date and ready to be written about. This check out and check in framework has the added benefit of priming students for "pushing" and "pulling" if they later learn about a formal version-control system. Writing tasks can be assigned using the action-item feature in Google Docs (Google 2022b). Google Docs also has its own sense of version control, so it is possible to look back at previous comments to track feedback and see the document's version history to track progress (Google 2022a).

Since both approaches rely on a division of the labor to avoid overlapping and conflicting efforts, peer-review and "smoothing" steps will be required before the final report is submitted. Peer-review is necessary to make sure individual pieces are accurate, consistent with the rest of the narrative, and clearly explained. A "smoothing" step ensures that the tone of the whole written work is consistent, there are transitions between subsections written by different people, and there are no other style inconsistencies (e.g., in the coding, visualizations, or explanations) across authors. A "smoothing" step can be much more challenging for code if members of a team follow different naming and style conventions, motivating the establishment of a consistent style before starting the project (Nolan and Stoudt 2021b). Again, peer-review and "smoothing" steps mirror real-world processes in both academic and non-academic collaborations, and it can be beneficial for students to get experience with them early on in their careers.

Students may have the instinct to "divide and conquer" in group projects but may have a harder time identifying appropriate subtasks, especially for the revision and "smoothing" stages of a project rather than the initial-draft stage. For example, I saw this when we recently piloted the use of ISLE, discussed above, in our introductory statistics classes. Although our students did not have the coding element in the collaborative workflow, they still experienced the divide-and-conquer approach in labs and throughout the final project which they built up from exploratory data analysis and formal inference conducted on a real-world dataset throughout the semester.

The final project report is a combination of revised work done throughout the semester, making it literally a divided-and-conquered product. However, the challenge is to create a final product that does not read as a disjointed piece. After students received feedback on all of the initial pieces of the final report, I gave them a checklist of concrete tasks that would help them synthesize these pieces and create their final report. I sensed that students could identify subtasks when there were concrete to-do items, as in lab questions or when following a rubric for the exploratory analysis, but when revision and synthesis was required, tasks became more abstract and harder to delegate. This was also the part of the semester where tensions were highest for groups who faced an imbalance of workload amongst its members. The checklist was therefore also framed as an opportunity for students who had been taking a more passive role up until this point to take a leadership role before the end of the semester. Subtasks included some that could be done in isolation, like revising a subsection based on the feedback received from peers and the instructor. Other subtasks broke down the "smoothing" process including prompts like "arrange revised pieces into one document," "smooth transitions between pieces written or revised by different people," "ensure consistent style of figures and axes labels," "cut redundancies in exploratory data analysis now that formal inference is included," "read final draft with an eye toward the final rubric," and "read final draft out loud to catch awkward phrasing and transitions." By making these steps in the writing workflow explicit, I signaled my expectations as a reader, trying to distill some of the writing-style properties that I read for that are harder to articulate on a project rubric.

4. Discussion

Much of my description above has focused on the technological tools or the communication needed to juggle the many moving parts of technical reports within the collaborative writing process, but tool builders, tool users, and educators alike may also want to learn from studies of the collaborative writing process outside of technical contexts. The following discussion is just scratching the surface of the broad literature, and we as writers in data-related contexts have much to learn.

Many researchers have outlined different stages, modes, and roles in the writing process. For example, a variety of taxonomies

of the writing process boil down to three main phases: pre-writing, writing, and re-writing (Flower and Hayes 1981; Kellogg 1988). Some expand those roles to include more concrete steps along the way such as brainstorming, planning, and researching (Posner and Baecker 1992). Others advocate for explicit creation of subtasks within these phases as a way to support the collaborative writing process, prompting us to further develop checklists or templates to guide students (Teevan, Iqbal, and von Veh 2016). This subtask approach is also in the spirit of parallel-partitioned writing approaches where members of a writing team can work on the same document but are focused on different parts at any given time (Lowry and Nunamaker 2003).

Collaborative writing groups may work along spectrums of synchronicity (when teammates are working) and proximity (where teammates are working) at any stage of a project; no matter where on the spectrums groups are, adding structure to the writing process by explicitly defining tasks can improve production quality and quantity, especially for new writers (Lowry et al. 2005). This divide-and-conquer style of writing has been studied and found to produce high quality text when compared to the work of groups who have one person serving in the main-writer role. Balancing both the writing and editing workloads also was associated with an increase in the quality of content, although the ability to work synchronously and in the same physical location was found to be an important factor in improving the organization of written work, perhaps suggesting that we encourage synchronous meetings for "smoothing" tasks (Yim et al. 2017). A related work style, cooperative revision, divides the tasks amongst group members but allows each team member freedom to edit others' work in later stages of the writing process. This is a blend of parallel (working on parts of one draft at the same time) and sequential writing (handing off drafts to the next person) (Posner and Baecker 1992).

In early calls for the creation of collaborative writing tools, desired features included ways to communicate about the writing process itself, make collaborators aware of one another's contributions, easy ways to combine multiple contributions, and balancing flexibility with constraints (Jones 1995). User testing of collaborative writing software continues to reveal preferences for tracking changes and commenting capabilities (Noël and Robert 2004). These features of even nominally version-control-free writing tools have the added benefit of building in a fail-safe for failure modes in the writing workflow. For example, suppose a student made a breaking change in the R Markdown and wanted to return to the work done by the previous student in charge of this document. That version would still be "checked in" to Google Drive and could be recovered. If there were check-in/check-out steps in between, there would still be a record of the document uploaded previously as Google Drive keeps track of version histories. Similarly, if the team wanted to revert to a piece of writing that they had overwritten, they could see previous versions and recover the text from there. When using explicit version-control software, previous work can be recovered at any time. Instead, the worst-case scenario is a breakdown in communication that results in a merge conflict despite the writing structure's design to minimize the chance of entering this scenario. Students can practice in advance for this or pick up the skill as needed in an isolated environment (Timbers 2022a).

When choosing between writing workflows and tools, it may also be helpful to consider how the choice might affect the reproducibility of the final product. Some crucial steps in an analysis workflow where reproducibility can break down include initial data inputs, data pre-processing, data analysis, and migration of results into a manuscript. A proposed writing workflow can be assessed for potential pitfalls at any of these points.

In both the version-control-free compromise and the streamlined-version-control compromise, teams must agree on a standard data file to read in that each teammate likely stores locally. Even when using version control, data is not as easily shared back and forth due to memory constraints (GitHub 2022a). If a teammate unknowingly overwrites their raw data locally, teammates may start to obtain differing outputs.

For both scenarios, data pre-processing and analysis happen in the R Markdown file, so we must consider how these two writing workflows handle the document. In the version-control-free compromise there should only be one person working on the R Markdown document at one time. As long as the check-in/check-out process is followed, there should not be discrepancies in the pre-processing and analysis code across teammates. In the streamlined-version-control compromise, one person's portion of the overall R Markdown file might need to change if another person's portion changes (e.g., a team member in charge of pre-processing cleans and renames a column of the data, affecting the variable name that a team member in charge of initial data exploration must reference). To anticipate this potential pitfall, teams could be encouraged to map out the dependencies amongst their individual subsections ahead of time.

The migration of results to a final manuscript is automatic for the streamlined-version-control compromise and requires interaction with a Google Doc in the version-control-free compromise. Again, in the latter case, if the check-in/check-out process is respected, teammates should be writing about the most up-to-date results. However, it is possible for teams using version control to have problems with teammates writing about stale figures or results if upstream analyses are changing as they are writing. Having teams communicate when various inputs are stable throughout the process could alleviate this tension. Overall, both workflows can be reproducible with proper planning and communication.

5. Conclusion

In this article, I have outlined some approaches to blend the computational and writing processes required in collaborative, data-intensive work with the goal of keeping products reproducible while acknowledging pain points that come with collaboration. I have outlined these approaches with introductory statistics and data science students in mind. Ideally these workflows are as authentic as possible while avoiding logistical barriers that distract students from the content they are working to master in the context of a dataset of interest and the narrative they are trying to form about that dataset.

Both approaches involve common themes of a preliminary report structure, a division of labor, external communication about roles and timelines, and a final peer-review and

"smoothing" process regardless of whether version control is involved. These proposed workflows are not the only ways forward, but I hope they start the conversation about how to address writing workflows in the data-driven classroom and prompt classroom tests of strategies that fit within the context of differing student bodies and instructor constraints. This conversation starter may have the added benefit of causing us to reckon with our own collaborative writing workflows as well to ensure that we are practicing what we preach.

Disclosure Statement

The author of this manuscript has no competing interests to disclose.

References

AguaClara Cornell (2018), "Collaboration in Colab." Available at *https://aguaclara.github.io/aguaclara_tutorial/colab/colab-collaboration.html*.

Barba, L. A., Barker, L. J., Blank, D. S., Brown, J., Downey, A. B., George, T., Heagy, L. J., Mandli, K. T., Moore, J. K., Lippert, D., Niemeyer, K. E., Watkins, R. R., West, R. H., Wickes, E., Willing, C., and Zingale, M. (2019), *Teaching and Learning with Juputer*, Self Published. *https://jupyter4edu.github.io/jupyter-edu-book/*.

Baumer, B., Cetinkaya-Rundel, M., Bray, A., Loi, L., and Horton, N. J. (2014), "R Markdown: Integrating A Reproducible Analysis Tool into Introductory Statistics," *Technology Innovations in Statistics Education*, 8, 1–29. *https://escholarship.org/uc/item/90b2f5xh*.

Beckman, M. D., Çetinkaya-Rundel, M., Horton, N. J., Rundel, C. W., Sullivan, A. J., and Tackett, M. (2021), "Implementing Version Control With Git and GitHub as a Learning Objective in Statistics and Data Science Courses," *Journal of Statistics and Data Science Education*, 29, S132–S144. *https://www.tandfonline.com/doi/full/10.1080/10691898.2020.1848485*

Brun, Y., Holmes, R., Ernst, M. D., and Notkin, D. (2013), "Early Detection of Collaboration Conflicts and Risks," *IEEE Transactions on Software Engineering*, 39, 1358–1375. *http://ieeexplore.ieee.org/document/6520859/*

Bryan, J. (2018), "Excuse Me, Do You Have a Moment to Talk About Version Control?," *The American Statistician*, 72, 20–27. *https://www.tandfonline.com/doi/full/10.1080/00031305.2017.1399928*

Carver, R., Everson, M., Gabrosek, J., Horton, N., Lock, R., Mocko, M., Rossman, A., Roswell, G. H., Velleman, P., Witmer, J., and Wood, B. (2016), "Guidelines for Assessment and Instruction in Statistics Education (GAISE) College Report 2016," p. 143.

Case, D. M., Eloe, N. W., and Leopold, J. L. (2016), "Scaffolding Version Control into the Computer Science Curriculum," pp. 175–183. *http://ksiresearchorg.ipage.com/seke/dms16paper/dms16paper_36.pdf*

CauseWeb (2021), "USCLAP Competition," *https://www.causeweb.org/usproc/usclap*.

Chattopadhyay, S., Prasad, I., Henley, A. Z., Sarma, A., and Barik, T. (2020)," What's Wrong with Computational Notebooks? Pain Points, Needs, and Design Opportunities," in Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, ACM, Honolulu, HI, pp. 1–12. *https://dl.acm.org/doi/10.1145/3313831.3376729*

De Veaux, R. D., Agarwal, M., Averett, M., Baumer, B. S., Bray, A., Bressoud, T. C., Bryant, L., Cheng, L. Z., Francis, A., Gould, R., Kim, A. Y., Kretchmar, M., Lu, Q., Moskol, A., Nolan, D., Pelayo, R., Raleigh, S., Sethi, R. J., Sondjaja, M., Tiruviluamala, N., Uhlig, P. X., Washington, T. M., Wesley, C. L., White, D., and Ye, P. (2017), "Curriculum Guidelines for Undergraduate Programs in Data Science," *Annual Review of Statistics and Its Application*, 4, 15–30. *http://www.annualreviews.org/doi/10.1146/annurev-statistics-060116-053930*

Fiksel, J., Jager, L. R., Hardin, J. S., and Taub, M. A. (2019), "Using GitHub Classroom To Teach Statistics," *Journal of Statistics Education*, 27, 110–119. *https://www.tandfonline.com/doi/full/10.1080/10691898.2019.1617089*

Flower, L., and Hayes, J. R. (1981), "A Cognitive Process Theory of Writing," *College Composition and Communication*, 32, 365. *https://www.jstor.org/stable/356600?origin=crossref*

GitHub (2022a), *About Git Large File Storage*. *https://docs.github.com/en/repositories/working-with-files/managing-large-files/about-git-large-file-storage*.

——— (2022b), *About Pull Request Reviews*. *https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/reviewing-changes-in-pull-requests/about-pull-request-reviews*.

GitHub Docs (2021a), "Creating a Repository from a Template," *https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/creating-a-repository-on-github/creating-a-repository-from-a-template*.

——— (2021b), "Creating a Template Repository," *https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/creating-a-repository-on-github/creating-a-template-repository*.

Google (2022a), *Check or Revert to Earlier File Versions*. *https://support.google.com/a/users/answer/9308971?hl=en*.

——— (2022b), *Use Comments & Action Items*. *https://support.google.com/docs/answer/65129?hl=en&co=GENIE.Platform%3DDesktop*.

——— (n.d.-a), *Colaboratory Frequently Asked Questions*. *https://research.google.com/colaboratory/faq.html#resource-limits*.

——— (n.d.-b), *Overview of Colaboratory Features*. *https://colab.research.google.com/notebooks/basic_features_overview.ipynb#scrollTo=aro-UJgUQSH1*.

Haaranen, L., and Lehtinen, T. (2015), "Teaching Git on the Side: Version Control System as a Course Platform," in Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, ACM, Vilnius Lithuania, pp. 87–92. *https://dl.acm.org/doi/10.1145/2729094.2742608*

Hart, T. (2016), "Tweet: 'Writing a Multi-Author Paper on Github...'," *https://twitter.com/emhrt_/status/740777537547173889*.

Jones, S. (1995), "Identification and Use of Guidelines for the Design of Computer Supported Collaborative Writing Tools," *Computer Supported Cooperative Work (CSCW)*, 3, 379–404. *http://link.springer.com/10.1007/BF00750747*

Journal of Undergraduate Research (2021), "Submission Guidelines," *http://www.jurpress.org/submissionguidelines*.

Kellogg, R. T. (1988), "Attentional Overload and Writing Performance: Effects of Rough Draft and Outline Strategies," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 14, 355–365. *https://doi.org/10.1037/0278-7393.14.2.355*.

Lowry, P., and Nunamaker, J. (2003), "Using Internet-Based, Distributed Collaborative Writing Tools to Improve Coordination and Group Awareness in Writing Teams," *IEEE Transactions on Professional Communication*, 46, 277–297. *http://ieeexplore.ieee.org/document/1255526/*

Lowry, P., Nunamaker, J., Curtis, A., and Lowry, M. (2005), "The Impact of Process Structure on Novice, Virtual Collaborative Writing Teams," *IEEE Transactions on Professional Communication*, 48, 341–364. *http://ieeexplore.ieee.org/document/1546292/*

McBain, M. (2018), "Markdrive," *https://github.com/milesmcbain/markdrive*.

National Academies of Sciences, Engineering, and Medicine. (2018), *Data Science for Undergraduates: Opportunities and Options*. Washington, DC: The National Academies Press. *https://doi.org/10.17226/25104*.

Noël, S., and Robert, J.-M. (2004), "Empirical Study on Collaborative Writing: What Do Co-authors Do, Use, and Like?," *Computer Supported Cooperative Work (CSCW)*, 13, 63–89. *http://link.springer.com/10.1023/B:COSU.0000014876.96003.be*

Nolan, D., and Stoudt, S. (2021a), *Communicating with Data: The Art of Writing for Data Science*. Oxford: Oxford University Press, Chapter Writing the First Draft.

——— (2021b), *Communicating with Data: The Art of Writing for Data Science*. Oxford: Oxford University Press, Chapter Communicating Through Code.

Philipp, B. (2021), "Teaching Statistical Concepts and Modern Data Analysis With a Computing-Integrated Learning Environment," *Journal of Statistics and Data Science Education*, 29, S61–S73. *https://doi.org/10.1080/10691898.2020.1854637*.

Plotts, J. (2018), *Feature Launch: Kernels Collaboration*, Kaggle. *https://www.kaggle.com/product-feedback/54980*.

Posner, I. R., and Baecker, R. M. (1992), "How People Write Together," *Proceedings 25th Hawaii International Conference on System Sciences*, 9, 127–138. *https://www.dgp.toronto.edu/public_user/RMB/papers_old/p23.pdf*.

Robbins, J. (2020), "Bookdown Template," *https://github.com/jtr13/bookdown-template*.

rOpenSci (2016), "Google docs/drive Workflow and Package Ecosystem," *https://github.com/ropensci/unconf16/issues/9*.

——— (2018), "Workflow for Publications Using rmarkdown with Users that Won't Get Passed Word/Google Docs," *https://github.com/ropensci/unconf18/issues/42*.

Ross, N. (2018), "Rmd Rant," *https://github.com/noamross/rmd-rant*.

——— (2020), "Redoc – Reversible Reproducible Documents," *https://github.com/noamross/redoc*.

Ross, N., White, N., Salmon, M., and Ushizima, D. (2016), "gdoc," *https://github.com/ropensci-archive/gdoc*.

RStudio Community (2018), "Live Collaborative Editing on RStudio Cloud?," *https://community.rstudio.com/t/live-collaborative-editing-on-rstudio-cloud/18898*.

——— (2019), "Why Isn't There a Free Collaborative Editor for R?," *https://community.rstudio.com/t/why-isnt-there-a-free-collaborative-editor-for-r/33573*.

——— (2020), "How can Students Collaborate with RStudio.Cloud?," *https://community.rstudio.com/t/how-can-students-collaborate-with-rstudio-cloud/58066*.

Rule, A., Tabard, A., and Hollan, J. D. (2018), Exploration and Explanation in Computational Notebooks, in Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, ACM, Montreal, QC, Canada, pp. 1–12. *https://doi.org/10.1145/3173574.3173606*.

Sagemath, Inc. (2022a), *Licenses*. *https://doc.cocalc.com/account/licenses.html#buying-licenses*.

——— (2022b), *Trial Projects*. *https://doc.cocalc.com/trial.html*.

Tatman, R. (2019), *Getting Started with Kaggle: Workshop in a Box*. *https://www.kaggle.com/rtatman/getting-started-with-kaggle-workshop-in-a-box*.

Teevan, J., Iqbal, S. T., and von Veh, C. (2016), "Supporting Collaborative Writing with Microtasks," in Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, ACM, San Jose, CA, pp. 2657–2668. *https://dl.acm.org/doi/10.1145/2858036.2858108*

Timbers, T. (2022a), *r-Cube-Notebook*. *https://github.com/ttimbers/r-cube-notebook*.

——— (2022b), *Review My Pull Request*. *https://github.com/ttimbers/review-my-pull-request*.

Timbers, T., Campbell, T., and Lee, M. (2022), *Data Science: A First Introduction*, Self Published, chapter Communicating using GitHub issues. *https://datasciencebook.ca/Getting-started-with-version-control.html#communicating-using-github-issues*.

Wang, A. Y., Mittal, A., Brooks, C., and Oney, S. (2019), "How Data Scientists Use Computational Notebooks for Real-Time Collaboration," *Proceedings of the ACM on Human-Computer Interaction*, 3, 1–30. *https://dl.acm.org/doi/10.1145/3359141*

Watson, O. (2020), *Using Colab Notebooks for teaching R*. *https://colab.research.google.com/github/OJWatson/jhcolab/blob/master/JH_Demonstration.ipynb*.

Xie, Y. (2021), *bookdown: Authoring Books and Technical Documents with R Markdown*. NY: CRC Press.

Xie, Y., Allaire, J., and Grolemund, G. (2018), *R Markdown: The Definitive Guide*. NY: CRC Press.

Yim, S., Wang, D., Olson, J., Vu, V., and Warschauer, M. (2017), "Synchronous Collaborative Writing in the Classroom: Undergraduates' Collaboration Practices and their Impact on Writing Style, Quality, and Quantity," in Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, ACM, Portland, OR, pp. 468–479. *https://dl.acm.org/doi/10.1145/2998181.2998356*

Zandonella, C. (2021a), "Trackdown – R Package for Improving Collaborative Writing," *https://github.com/ClaudioZandonella/trackdown*.

——— (2021b), "The Trackdown Workflow," *https://claudiozandonella.github.io/trackdown/articles/trackdown-workflow.html*.