Improving Content Analysis: Tools for Working with Undergraduate Research Assistants

Benjamin Goehring, *University of Michigan, USA*

ABSTRACT Undergraduate research assistants (URAs) perform important roles in many political scientists' research projects. They serve as coauthors, survey respondents, and data collectors. Despite these roles, there is relatively little discussion about how best to train and manage URAs who are working on a common task: content coding. Drawing on insights from psychology, text analysis, and business management, as well as my own experience in managing a team of nine URAs, this article argues that supervisors should train URAs by pushing them to engage with their own mistakes. Via a series of simulation exercises, I also argue that supervisors—especially supervisors of small teams—should be concerned about the effects of errant post-training coding on data quality. Therefore, I contend that supervisors should utilize computational tools to monitor URA reliability in real time. I provide researchers with a new `R` package, `ura`, and a web-based application to implement these suggestions.

Undergraduate research assistants (URAs) perform key roles in many research projects. They serve as coauthors, data collectors, and survey respondents. They also classify and code data—a task that has become increasingly common with the growing popularity of supervised machine-learning models and text analysis (Grimmer, Roberts, and Stewart 2022; Grimmer and Stewart 2013). Students and researchers both stand to gain from working with one another. In addition to course credit or compensation, students accrue valuable skills and experience. They meet faculty members who can serve as mentors, learn how to overcome the challenges of the research process, and refine their own interests (Hakim 1998; Hunter, Laursen, and Seymour 2007; Lopatto 2004, 2007; Starke 1985). Faculty members, postdocs, and graduate students also stand to benefit. In addition to receiving assistance on a project, researchers get to know their students and influence their career trajectories—which faculty members and graduate students both believe is a rewarding part of their job (Dolan and Johnson 2009; Zydney et al. 2002).

Benjamin Goehring *is a PhD candidate in political science and public policy at the University of Michigan. He can be reached at bengoehr@umich.edu.*

A sizable literature examines best practices for developing undergraduate research programs that improve learning outcomes (Corwin, Graham, and Dolan 2015; Druckman 2015; Shanahan et al. 2015); aid faculty research (Chopin 2002; Gillies and Marsh 2013); and assist underrepresented students (Gándara 1999; Jones, Barlow, and Villarejo 2010; Ovink and Veazey 2011). Fewer studies, however, delve into the "nuts and bolts" of integrating URAs into political science research. URAs must be onboarded, trained, and supervised. Nevertheless, PhD programs provide little formal instruction in pedagogy and personnel management. Although many researchers have developed their own system for training and supervising, there is scant pedagogical discussion in the discipline regarding how best to manage URAs.

URAs are a unique group within academia whose training and management deserve special attention. Compared to graduate student research assistants, URAs often lack the same technical skills and focused academic interests. Working with URAs requires investing more time in their development and contending with the possibility of them leaving a project due to changing preferences (Dolan and Johnson 2009; Gillies and Marsh 2013). Moreover, URAs are not simply employees. A successful experience with URAs means that they come away from the project with

doi:10.1017/S1049096523000744

a new set of skills and a greater appreciation for the research process. Consequently, training and management must prioritize both completing the project efficiently and offering URAs a valuable learning experience.

This article offers suggestions for integrating URAs into common data-classification tasks in which raw data are coded for future analysis. Drawing on insights from business management, psychology, and text classification, I argue that the extant training method of error management training (EMT) provides a helpful theoretical lens for training URAs to perform content-coding tasks. According to EMT, the errors that occur naturally during a classification task are assets that can be harnessed to improve training outcomes. Drawing on my own experience in managing a group of nine URAs on a content-coding task, I describe how to infuse EMT into URA training and offer anecdotal evidence that it encourages URAs to critically engage with the task at hand. Of course, errors also can arise after URAs have completed training and begun work on an actual project. Using a simulation exercise to frame the stakes of a single URA performing poorly relative to other team members, I also argue that supervisors should utilize computational tools to monitor URA reliability in real time. If done sparingly and thoughtfully, URA monitoring can recognize potentially expensive mistakes without seriously compromising future reproducibility. I provide examples of URA monitoring from my own experience as a URA supervisor and provide other researchers with a set of open-source tools—the `R` package `ura` and a web-based application—to efficiently monitor URA progress and reliability.

URAs must be onboarded, trained, and supervised. Nevertheless, PhD programs provide little formal instruction in pedagogy and personnel management.

URA TRAINING AND MANAGEMENT

The increasing popularity of machine learning and text analysis in the social sciences (e.g., Grimmer 2015; Grimmer and Stewart 2013) has resulted in a small literature on training and managing research assistants.[1] Classifying unlabeled data (e.g., newspaper articles) into groups is a common application of machine-learning models. Supervised models, which organize unlabeled data into groups using predictive methods honed on labeled data, comprise a robust method for these types of classification tasks (Barberá et al. 2021; Grimmer and Stewart 2013). However, creating the labeled dataset is a time-intensive process that, due to reliability concerns, is often conducted by a team of research assistants.

Creating labeled datasets requires careful planning and considerable upfront work. Research assistants should possess similar capabilities and have the necessary skills; codebooks should be exhaustive and detailed; and the type and amount of data in the labeled set must be optimized (Barberá et al. 2021; Grimmer, Roberts, and Stewart 2022; Krippendorff 2018; Neuendorf 2016). Training research assistants is another well-recognized component in generating quality data. Neuendorf (2016, 158), for example, writes that "three words describe good coder preparation: train, train, and train." However, this literature largely omits the pedagogical details of training. For instance, in a recently published textbook on text analysis, Grimmer, Roberts, and Stewart (2022, 192), note that training "involves having the coders carefully read the codebook and ask any questions. It often also involves asking the coders to label a sample of texts and evaluating whether they have understood the instructions or whether the instructions need to be revised." Neuendorf (2016) provides similar guidance, treating training as an iterative process of coding, discussion, and codebook revision until the group reaches acceptable reliability levels.

An important exception is Krippendorff (2018, 134), who offers a specific example of a program to train coders for a content-analysis task. Interested in television violence, he provided initial guidance to his research assistants before having them code a practice set of television programs. After coding the violent acts in one program, the coders compared their results to those of an expert panel. The process was repeated with additional programs until the supervisors deemed the research assistants ready to begin working on the actual task. Krippendorff (2018, 134) states that this "self-teaching program" encouraged coders to "reevaluate their conceptions" and become proficient.

Although it is not framed in such a way, the training system described by Krippendorff (2018) shares features with EMT, a training methodology that emphasizes the pedagogical purpose of making mistakes. According to EMT, requiring trainees to fail and learn from their mistakes encourages them to consider why they erred and to reassess their approach to the task (Brown and Others 1982; Ivancic and Hesketh 2000; Keith and Frese 2008). Errors are not merely an indication that something went awry but rather a "basis to think ahead and try something new" (Keith and Frese 2008, 60). In this way, EMT differs from other types of training that neither encourage errors nor provide trainees with strict instructions that prevent mistakes.

As with EMT, Krippendorff's (2018) training program treats errors as more than a signal that additional didactic training is warranted. Instead, errors serve as a jumping-off point for trainees to learn more about a task. In this example, it likely meant returning to the codebook and television program to determine where the error occurred. In other cases, as Krippendorff (2018) notes, it might implicate the expert panelists' findings, resulting in codebook changes. However, there are differences between EMT and this example from Krippendorff (2018). Most notably, EMT usually involves placing trainees in a situation where they must complete a task with minimal guidance (e.g., replacing an automobile tire without instructions). This freewheeling, exploratory approach is inappropriate to a classification task in which the consistent application of conceptual definitions is of the utmost importance. Nevertheless, there is space within these confines to embrace errors as a tool for learning.

Although Krippendorff's (2018) program offers a blueprint for how errors can aid in training URAs, it is lacking implementation details and examples of how students who engaged with their mistakes improved training outcomes. The following sections describe a detailed example of how I implemented EMT for a

https://doi.org/10.1017/S1049096523000744 Published online by Cambridge University Press

classification project and provide anecdotal evidence of it encouraging URAs to reflect critically on a task. A discussion follows about effectively monitoring URAs after training is completed to minimize errors while maintaining reproducibility.

BACKGROUND

In 2021, my colleague Kenneth Lowande and I hired nine URAs to search the ProQuest database for newspaper articles covering unilateral actions (e.g., executive orders) issued by the President of the United States (Goehring and Lowande 2022; Goehring, Lowande, and Shiraito 2023). For each action, a URA used criteria that we set to search the archives of 54 US newspapers. We intentionally constructed the search criteria to cast a wide net and return articles that were false-positive matches. Consequently, after finding all articles that possibly could be covering an action, the URA had to examine the text of each article. An article was deemed relevant if it mentioned a unilateral action and attributed it to the president.[2]

Our team collected coverage for a sample of approximately 1,200 unilateral actions issued between 1989 and 2021. Overall, the URAs performed very well, agreeing almost 94% of the time on whether an action received coverage from at least one article. We cannot take all of the credit for this high level of quality; however, we believe that how we trained and managed the team contributed to their success.

ERRORS AS A TRAINING TOOL

Properly classifying an article required more than simply searching the text for keywords. The URAs needed a strong grasp of our conceptualization of “relevant coverage” to know whether an article covered an action. Each URA attended an initial one-hour training session and then spent an average of four hours practicing classifying articles. This practice task, visualized in figure 1, was a facsimile of the project: the instructions were to use predefined search criteria to find all newspaper articles that might cover the action and then go through each one and determine whether it provided relevant coverage. However, unlike the actual task, we developed an “answer key” for the practice set by completing the search procedures ourselves.

After completing the practice actions, every student checked their work against the results that Lowande and I had found. For each discrepancy between their findings and ours, the URAs went back to the definition of “relevant coverage” and either described where they went wrong or argued why we were the ones who had erred.[3] In some cases, this meant describing why they denoted an article as covering the given action when, in fact, it should have been excluded. In other cases, it meant justifying why they omitted an article when it should have been included.

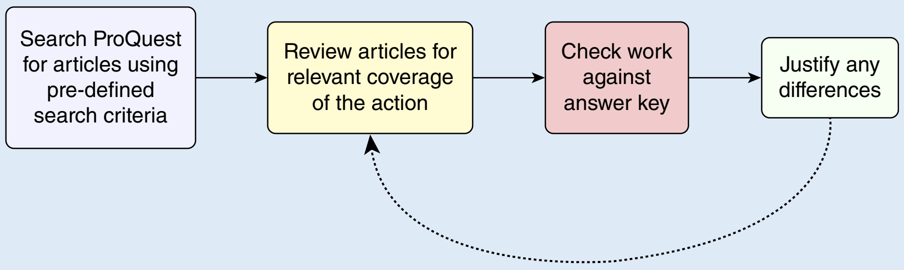
Asking URAs to check their own work encouraged them to think critically about their mistakes. Errors served a key pedagogical purpose. Although we provided an answer key containing what we believed to be the correct answers, we did not indicate why we thought a newspaper article did or did not cover the action. The students had to figure this out for themselves by referencing the article, the definition of “relevant coverage,” and the codebook examples. In this way, we struck a balance between EMT and the consistency necessary in any classification task. We could not ask URAs to figure out through trial and error what constitutes “relevant coverage” because that would yield different operationalizations across coders. Yet, we could encourage them to grapple with difficult cases within the confines of the prescribed definition.

The training task seemingly led URAs to critically assess their work. Table 1 lists five verbatim examples of URAs explaining in their own words why their answers from the training exercise differed from the answer key. In each example, the URA incorrectly classified a newspaper article. The first two examples are straightforward instances of the justification process reminding the URA that we were interested in non-opinion pieces about executive rather than legislative actions. The last three examples, conversely, provide more detailed critical reflections. The third justification points directly to the piece of text that should have led the URA to discount the article as not providing relevant coverage. Likewise, the fourth example shows that the URA realized that a crucial detail was missed: the article was issued before the action was undertaken. The fifth example describes the URA offering evidence for why the article provided relevant coverage before correctly noting that it never mentioned the president taking concrete action.

Lowande and I were encouraged by the ways in which URAs described and justified their responses. More than once, their answers required us to review the codebook to clarify language and tweak some of our examples. Often, however, the justifications served only to reinforce core concepts from the codebook.

Figure 1

Training Task Diagram


Search ProQuest for articles using pre-defined search criteria
Review articles for relevant coverage of the action
Check work against answer key
Justify any differences

This figure displays the training steps that Lowande and I asked our URAs to complete. For every unilateral action, the URA first searched through the ProQuest database for articles that match criteria defined by us (shown in blue). The URAs then reviewed each possible match returned by the search criteria to determine whether it met our definition of “relevant coverage” (shown in yellow). After working through all of the unilateral actions in the practice set, the URAs checked their results against an answer key (shown in red). If the URAs discovered that they erred, they went back to the article and codebook to either describe why they erred or to argue why their original coding decision was correct (shown in green).

https://doi.org/10.1017/S1049096523000744 Published online by Cambridge University Press

Table 1
URA Justification Examples

I shouldn't have included this one because it seems to be an opinion piece.
This article references legislation, not executive action.
I included this article for the following language: "After weeks of negotiations, House Republican leaders agreed to language that would weaken the economic embargo against Cuba for the first time in four decades. The bill, pushed by Rep. George Nethercutt, R-Washington, would permit the direct sale of food to Cuba, similar to a policy allowing the sale of medicine." But I'm now realizing that it says "House Republicans."
I put this article as relevant coverage because it clearly stated the name of the executive order and the publication of the article was after the order was issued. However, after a more thorough read, I realized that the sentence in which the order is mentioned says that Bush started "Preserve America" in 2001 (In 2001, President Bush started "Preserve America" to help increase historic tourism.). The order was issued in 2003, so this article should not be considered relevant coverage.
Looking back on it, I probably should not have included this article. I interpreted the following sentence as an example of unilateral action in the works: "The president vowed retaliation against the global terrorist group, and he gave a full-throated defense of his administration's anti-terrorism efforts in the face of Republican criticism." However, I am not sure that this warrants inclusion due to the vague nature of the President's statement.

Note: Each entry in this table is an example of URAs justifying or explaining any differences between their coding of newspaper articles and our own. Other than minor spelling errors, the entries are verbatim.

Requiring the URAs to work through why they erred prompted them to engage more deeply with the nuances of the task, which we believe translated into a more reliable and a higher-quality dataset.

URA MONITORING

Although this process provides a method for effectively training URAs, it does not prevent errors from occurring during the actual task. Errors can arise for several reasons. As the semester progresses, URAs may become more focused on competing priorities (e.g., extracurricular activities or studying for a test). Likewise, family or health emergencies could affect a URA's ability to perform the task with the same attention to detail demonstrated during training.

Requiring the URAs to work through why they erred prompted them to engage more deeply with the nuances of the task, which we believe translated into a more reliable and a higher-quality dataset.

Errors by even a single URA seriously affects data quality. Consider this more generalized example of the coding task that we conducted. Each member on a team of URAs is randomly assigned to code 100 unilateral actions from a population of 200 actions. Each action can be assigned to more than one URA, thereby making a subset of the actions suitable for testing inter-rater reliability (IRR). Generally, the URAs are very reliable: if an action is coded by more than one URA, then they all agree about whether the media provides coverage. However, there is one URA, labeled *i*, who is not very precise. Whereas the other URAs always agree on a given action's coding, URA *i* diverges from the others with some probability.

Figure 2 uses simulations to demonstrate the implications of URA *i*'s unreliability for IRR. Within each of the four facets, coding data were generated according to the process outlined previously. The only difference among the facets is the size of the URA teams. For a team composed of three, six, 10, or 20 URAs, the black line in each facet shows how changing the probability that URA *i* disagrees with the other coders impacts the reliability of the dataset, as measured by Krippendorf's Alpha. A robust metric for calculating IRR, Krippendorf's Alpha calculates reliability on a scale from -1 to 1. The blue line in figure 2 marks $\alpha = 0.8$, above which often is used as an indicator of high reliability (Krippendorff 2018).

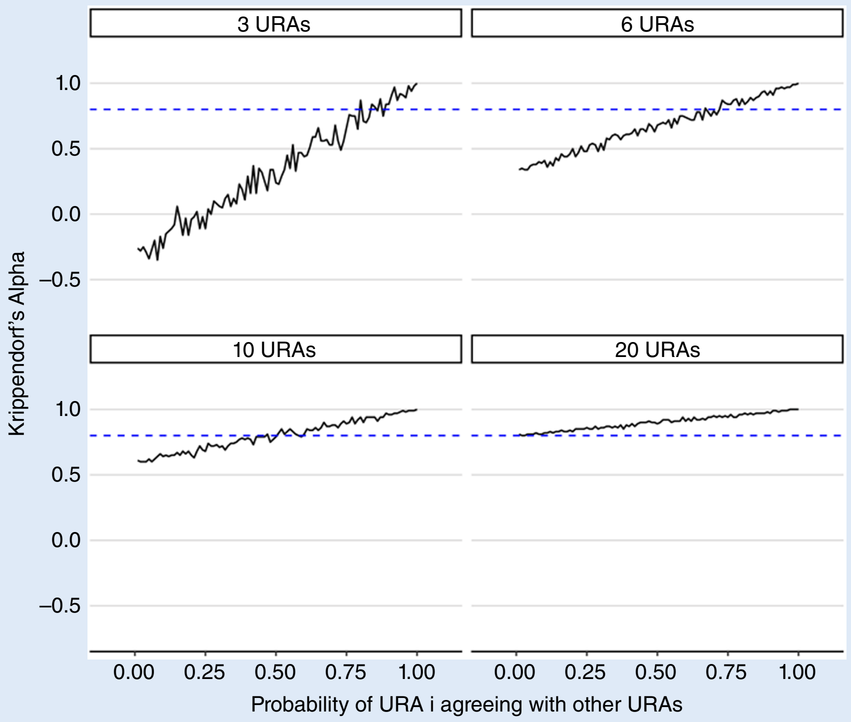
Overall, figure 2 shows that one URA making systematic errors can seriously affect reliability. As URA *i* becomes less precise (i.e., moving to the left on the horizontal axis within a given facet), α decreases. The severity of this decline in reliability varies significantly with the size of the team performing the content-coding task. Whereas using a relatively large team (i.e., bottom-right quadrant) can compensate for the errors of one URA, the reliability of a smaller group (i.e., top-left quadrant) is especially vulnerable to one poorly performing coder.

This simulation is robust to alternative specifications. The online appendix includes additional simulations, in which I vary the number of actions coded by each URA and the share of actions that are coded by more than one URA. Decreasing either of these parameters increases the variability of α but does not affect the main finding that a single poorly performing URA can significantly affect IRR, especially on a team with fewer members.[4]

We tried to reduce the likelihood of endemic errors by monitoring URA reliability in real time. Each student worked in a Microsoft Excel workbook located in a Dropbox folder synced with our machines.[5] The workbooks and file structures were formatted consistently and the URAs recorded their progress in the same way using identical column names and values. Using R, the data from all of the URA workbooks could be compiled instantly, and I could calculate the reliability of each coder relative to other members of the team. I measured the reliability of individual coders using the straightforward metric of percent agreement, calculating for every coder the share of their actions that were coded the same by the other coders.

Figure 2

The Effects of One Poorly Performing URA on IRR



This figure shows how one poorly performing URA affects the IRR of a dataset. For a task conducted by a given number of URAs, each facet shows how the (im)precision of one URA affects the Krippendorf's Alpha of the final dataset. The blue dashed line at 0.8 represents the conventional level of reliability for Krippendorf's Alpha. The data for each facet are generated by randomly assigning each URA 100 actions from a set of 200 actions (with replacement). Therefore, some but not all of the actions were coded by more than one URA and suitable for IRR testing. The URAs always agreed on actions that were coded by more than one URA, except for URA *i*, who agreed with the others with some probability (shown on the horizontal axis). If the probability of URA *i* agreeing with the other URAs was 1, then all of the URAs assigned the same coding to actions. If the probability of URA *i* agreeing with the other URAs was 0, then URA *i* assigned the opposite coding of the other URAs. The online appendix includes the simulation code and additional plots that vary the number of actions assigned to each URA. As shown in those plots, varying the number of actions assigned to each URA and the number of actions sampled does not affect the findings.

Table 2

Percent Agreement, by Coder

Coder ID	Percent Agreement
4	80
2	77
3	76
5	76
1	74
6	51

To further illustrate this process, consider a hypothetical scenario using data drawn from the previous simulation exercise, in which we are monitoring a team of six URAs who are coding actions for whether they received media coverage. So far, each member of the team has coded 100 actions—a subset of which also was coded by another member of the team. We plan to assign the team members more actions to code but want first to check how well they have performed up to this point. Looking at measures of IRR, they appear to be performing satisfactorily but not as well as recommended, agreeing with one another 73.8% of the time ($\alpha = 0.67$). However, these statistics measure group-level reliability, masking any variation in the performance of individual coders. Therefore, we also calculate each coder's percent agreement with other coders. The results of that coder-level reliability measure are presented in table 2.

There is a clear outlier among the six coders. Whereas coders one through five agree with one another a roughly similar share of the time, coder six agrees with his peers much less frequently. As shown in figure 2, one errant coder working on a relatively small team can have a significant effect on IRR—and that is exactly the case here. In fact, similar to the simulation exercise, it is only coder six who is miscoding actions relative to his peers.[6] Although this decreases everyone's agreement rate—because, in this example, each member of the team coded at least some actions coded by everyone else—it has a much more significant effect on the agreement rate of coder six.

This process of monitoring coder-level agreement rates required a variety of computational tools. In addition to standardized URA workbooks and data dictionaries, I wrote a considerable amount of `R` code to compile coding results and calculate measures of relative reliability. To complement this article, I repackaged the code I used to monitor the URAs into two open-source applications for other researchers. The first is an `R` package named `ura` that provides a simple, programmatic interface for calculating overall IRR diagnostics and examining the reliability of one URA relative to other members of a team. Although other packages exist for calculating IRR diagnostics, `ura` is unique in its ability to examine a coder's performance relative to the other team members. It also was designed with accessibility in mind, performing most of the preprocessing data-cleaning steps for analysts. The online appendix includes an example of the package in action; download instructions and additional documentation are available on GitHub[7] and the Comprehensive R Archive Network.

The second open-source software is a web-based application that implements `ura` on a web browser.[8] This point-and-click interface provides all researchers, regardless of their programming experience, with the ability to quickly examine the reliability of individual URAs. A researcher need only upload a comma-separated values file containing the coding dataset into the application and select the necessary column names. The website and online appendix both provide additional documentation and worked-through examples, and the application's underlying code is available on GitHub.[9]

Of course, monitoring is only part of the process; it also is important to consider how and when to intervene when learning that a coder is underperforming. If we found any URA to be noticeably underperforming relative to the other team members, we checked in to see if a personal issue was affecting the student's performance. Otherwise, we reinforced key points from the training and worked through examples from the codebook before having the URA resume work. Crucially, so that we did not compromise the future reproducibility of the project, we refrained from introducing any new material that was not accessible to other coders. We reinforced only the training materials and the information contained in the codebook; classifying data in a reproducible manner requires coders to work independently using only the codebook (Neuendorf 2016). If outside resources are relied on, future URAs will be unable to replicate the original results. We also only intervened sparingly to correct significant problems. Monitoring provides a safety net to detect errors before they become endemic. It is neither a tool for continuous fine-tuning of coder behavior nor a substitute for high-quality training.

Monitoring provides a safety net to detect errors before they become endemic. It is neither a tool for continuous fine-tuning of coder behavior nor a substitute for high-quality training.

DISCUSSION

This article provides researchers with two main takeaways. First, researchers should treat URA training as a pedagogical enterprise. There often are better ways to teach students a topic than assigning a relevant book to read and asking if they have any questions; likewise, there are more creative options available to supervisors than asking if URAs understand a codebook until reliability is reached. One method is described in this article, which leans on insights from EMT to encourage URAs to think critically about a task. Second, supervisors should consider monitoring URA reliability in real time. Used sparingly and thoughtfully, monitoring can protect data quality without significantly compromising future reproducibility.

Ultimately, however, suggestions for improving workflows also must be time efficient for researchers. I believe these tools save the most time by reducing the likelihood of systemic errors that require tasks to be repeated, but my proposals also should increase day-to-day efficiency. Lowande and I saved considerable time by not having to grade training datasets because we had our URAs review their own work. In terms of monitoring, the `ura` package and the associated web application greatly simplify workflows, especially for supervisors who lack a programming background. If URAs record their data in a standardized way, these tools make it easy to recognize coding errors before they become widespread.

There are numerous opportunities for future work on training and managing URAs. Perhaps most important, scholars should compare more rigorously the effects of various URA training programs on student and project outcomes. It is not difficult to envision experimental designs that give URAs a content-coding task and randomly assign a training procedure. In addition to examining how different training procedures affect the ability of URAs to accurately complete the task, scholars could determine whether training affects feelings of engagement and interest in conducting further study on the topic.

Although they are left implicit here, other practical suggestions are worth mentioning. First, incorporating URAs into a task requires careful planning. In addition to creating a detailed, exhaustive codebook, training systems must be developed and established up front. Without a detailed plan, URAs likely will neither produce good work nor receive the most benefit from the experience. Planning also is necessary for monitoring to succeed because file structures and coding rules are difficult to change after they are in place.

Second, technology should be used to one's advantage. Even when hiring URAs to conduct qualitative data collection, think through how file-sharing, `ura`, and other software can improve workflows. Whereas our project was focused on the specific case of using URAs for content-coding tasks, most of these suggestions are applicable to a wide range of different tasks often conducted by URAs. Finally, always treat URAs as colleagues. Keep them updated on the task's progress and show them early results from the data they collected. These actions, although seemingly insignificant, keep students engaged and reinforce that they are considered partners in a scholarly enterprise.

ACKNOWLEDGMENTS

I thank Ayse Eldes, Charles Shipan, Mika LaVaque-Manty, Eugenia Quintanilla, and Jade Burt for helpful comments and suggestions. Special thanks to Kenneth Lowande for encouraging me to write this article and providing considerable support along the way. I am solely responsible for any errors.

DATA AVAILABILITY STATEMENT

Research documentation and data that support the findings of this study are openly available at the *PS: Political Science & Politics* Harvard Dataverse at https://doi.org/10.7910/DVN/QDWDFQ.

SUPPLEMENTARY MATERIAL

To view supplementary material for this article, please visit http://doi.org/10.1017/S1049096523000744.

CONFLICTS OF INTEREST

The author declares that there are no ethical issues or conflicts of interest in this research. ■

NOTES

1. This literature does not differentiate among undergraduate, graduate, and other types of research assistants.
2. Data collection began during the COVID-19 pandemic. It did not have any noticeable effects on the project.
3. My idea to ask trainees to check their own work and justify any differences came from my experience of being trained as a research assistant on the Welfare Rules Database at the Urban Institute.
4. The code used to generate figure 2 and the alternative simulations are available in the online appendix.
5. To retain as much independence among coders as possible, we did not give students' access to their peers' folders (Grimmer, Roberts, and Stewart 2022).
6. More precisely, the data were generated so that coders one through five would always agree with one another on whether an action received coverage, whereas coder six agreed with the others with a probability of 0.5.
7. See https://github.com/bengoehring/ura.
8. The application is available at https://evaluate-uras.shinyapps.io/ura-shiny.
9. See https://github.com/bengoehring/ura-shiny.

REFERENCES

Barberá, Pablo, Amber E. Boydstun, Suzanna Linn, Ryan McMahon, and Jonathan Nagler. 2021. "Automated Text Classification of News Articles: A Practical Guide." *Political Analysis* 29 (1): 19–42.

Brown, Ann L., and Others. 1982. *Learning, Remembering, and Understanding.* Technical Report No. 244. Cambridge, MA: Bolt Beranek and Newman, Inc. http://hdl.handle.net/2142/17511.

Chopin, Suzzette F. 2002. "Undergraduate Research Experiences: The Translation of Science Education from Reading to Doing." *The Anatomical Record* 269 (1): 3–10.

Corwin, Lisa A., Mark J. Graham, and Erin L. Dolan. 2015. "Modeling Course-Based Undergraduate Research Experiences: An Agenda for Future Research and Evaluation." *CBE—Life Sciences Education* 14 (1): 1–13.

Dolan, Erin, and Deborah Johnson. 2009. "Toward a Holistic View of Undergraduate Research Experiences: An Exploratory Study of Impact on Graduate/Postdoctoral Mentors." *Journal of Science Education and Technology* 18 (6): 487–500.

Druckman, James N. 2015. "Merging Research and Undergraduate Teaching in Political Behavior Research." *PS: Political Science & Politics* 48 (1): 53–57.

Gándara, Patricia. 1999. *Priming the Pump: Strategies for Increasing the Achievement of Underrepresented Minority Undergraduates.* New York: College Entrance Examination Board. https://eric.ed.gov/?id=ED562803.

Gillies, Sharon L., and Steven Marsh. 2013. "Doing Science Research at an Undergraduate University." *International Journal of Arts & Sciences* 6 (4): 379–90.

Goehring, Benjamin. 2023. "*Replication Data for 'Improving Content Analysis: Tools for Working with Undergraduate Research Assistants.'*" PS: Political Science & Politics. Harvard Dataverse DOI:10.7910/DVN/QDWDFQ.

Goehring, Benjamin, and Kenneth Lowande. 2022. "*Behavioral Foundations of Presidential Accountability.*" Ann Arbor: University of Michigan. Working Paper. http://lowande.polisci.lsa.umich.edu/lg-presidential-pandering.pdf.

Goehring, Benjamin, Kenneth Lowande, and Yuki Shiraito. 2023. "*Letter: Theories and Evidence of Policy Significance.*" Ann Arbor: University of Michigan. Working Paper.

Grimmer, Justin. 2015. "We Are All Social Scientists Now: How Big Data, Machine Learning, and Causal Inference Work Together." *PS: Political Science & Politics* 48 (1): 80–83.

Grimmer, Justin, Margaret E. Roberts, and Brandon M. Stewart. 2022. *Text as Data: A New Framework for Machine Learning and the Social Sciences.* Princeton, NJ: Princeton University Press.

Grimmer, Justin, and Brandon M. Stewart. 2013. "Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts." *Political Analysis* 21 (3): 267–97.

Hakim, Toufic. 1998. "Soft Assessment of Undergraduate Research: Reactions and Student Perspectives." *Council on Undergraduate Research Quarterly* 18 (June): 189–92.

Hunter, Anne-Barrie, Sandra L. Laursen, and Elaine Seymour. 2007. "Becoming a Scientist: The Role of Undergraduate Research in Students' Cognitive, Personal, and Professional Development." *Science Education* 91 (1): 36–74.

Ivancic, Karolina, and Beryl Hesketh. 2000. "Learning from Errors in a Driving Simulation: Effects on Driving Skill and Self-Confidence." *Ergonomics* 43:1966–84.

Jones, Melanie T., Amy E. L. Barlow, and Merna Villarejo. 2010. "Importance of Undergraduate Research for Minority Persistence and Achievement in Biology." *Journal of Higher Education* 81 (1): 82–115.

Keith, Nina, and Michael Frese. 2008. "Effectiveness of Error Management Training: A MetaAnalysis." *Journal of Applied Psychology* 93 (1): 59–69.

Krippendorff, Klaus. 2018. *Content Analysis: An Introduction to Its Methodology.* Fourth edition. Los Angeles: SAGE Publications, Inc.

Lopatto, David. 2004. "Survey of Undergraduate Research Experiences (SURE): First Findings." *Cell Biology Education* 3 (4): 270–77.

Lopatto, David. 2007. "Undergraduate Research Experiences Support Science Career Decisions and Active Learning." *CBE—Life Sciences Education* 6 (4): 297–306.

Neuendorf, Kimberly A. 2016. *The Content Analysis Guidebook.* Second edition. Los Angeles: SAGE Publications, Inc.

Ovink, Sarah M., and Brian D. Veazey. 2011. "More Than 'Getting Us Through': A Case Study in Cultural Capital Enrichment of Underrepresented Minority Undergraduates." *Research in Higher Education* 52 (4): 370–94.

Shanahan, Jenny Olin, Elizabeth Ackley-Holbrook, Eric Hall, Kearsley Stewart, and Helen Walkington. 2015. "Ten Salient Practices of Undergraduate Research Mentors: A Review of the Literature." *Mentoring & Tutoring: Partnership in Learning* 23 (5): 359–76.

Starke, Mary C. 1985. "A Research Practicum: Undergraduates as Assistants in Psychological Research." *Teaching of Psychology* 12 (3): 158–60.

Zydney, Andrew L., Joan S. Bennett, Abdus Shahid, and Karen W. Bauer. 2002. "Faculty Perspectives Regarding the Undergraduate Research Experience in Science and Engineering." *Journal of Engineering Education* 91 (3): 291–97.