Using GitHub Actions for Computational Communication Research*

Inessa De Angelis†

October 28, 2025

Communication researchers increasingly use computational methods to collect and analyze large datasets. Part of the promise of these methods is their ability to increase the reproducibility and transparency of workflows. Taking up calls for more reproducible workflows and better recognition of the tools used in communication research, this paper introduces GitHub Actions for automated data collection. I describe the components required to set up and run a GitHub Actions workflow and illustrate its utility by discussing how I collected Bluesky posts tagged with "#CdnPoli" during the 2025 Canadian federal election. I conclude by offering best practices for other researchers, contributing to the development and documentation of reproducible workflows for computational communication research.

Keywords: Computational methods; open science; GitHub Actions; reproducible workflows; digital trace data

*__PRELIMINARY VERSION - Please do not cite or circulate without permission__

Code to support this workflow is available: https://osf.io/w43z5/?view_only=ec79c48ba5f74a7d94645e6b45129466

†Faculty of Information, University of Toronto, Inessa.DeAngelis@mail.utoronto.ca

Introduction

Communication researchers increasingly employing computational methods to collect and analyze large datasets. Part of the promise of these new computational methods is their ability to streamline data collection, storage, and management (Marwick et al., 2018), while enhancing reproducibility (Chan et al., 2024). The National Academies of Sciences, Engineering, and Medicine define reproducibility as "... obtaining consistent results using the same input data; computational steps, methods, and code; and conditions of analysis" (2019, p. 46). However, these promises of reproducibility do not always translate to tangible practices. While nuance is needed in certain circumstances, reproducibility in communication research continues to be hindered by the growing complexity of computational workflows (Marwick et al., 2018; van Atteveldt et al., 2019), insufficient documentation (Waldherr et al., 2025), and limited incentives for the sharing of code and data (Chan et al., 2024; van Atteveldt et al., 2019). Without greater emphasis on reproducibility and open science practices, the validity of findings may be undermined, public research funds may be wasted, and incorrect conclusions may be drawn (Dienlin et al., 2021; Haim & Puschmann, 2023).

The open science movement aims to address some of these aforementioned issues to improve the transparency, rigor, and validity of scholarly research. Open science seeks to make "... explicit the often implicit decision-tree scientists use to arrive at their conclusions" (Lewis, 2020, p. 74). In communication research, tangible open science practices include making code, data, and other materials publicly available, alongside preregistration, registered reports, or preprints (Perreault & Dienlin, 2025). Dienlin et al. (2021) also advocate for more collaboration, discussing open science methods during classroom lectures, and implementing Transparency and Openness Promotion (TOP) guidelines.

Several researchers note that communication research often lacks the infrastructure, incentives, and standardized practices needed to support reproducible computational data collection and analysis (van Atteveldt et al., 2019). Lawrence et al. (2023) argue that as data collection and analysis pipelines become more complex, better documentation and transparency is needed to ensure the validity of findings. Likewise, Haim & Puschmann (2023) contend that open science practices should not end with the sharing of code and data. Instead, they argue that "... computing power and programming skills are also crucial resources" and researchers should endeavor to share other materials, tools, and even standardized analytical procedures (Haim & Puschmann, 2023, p. 248). Taken together, these points emphasize that well-intentioned open science practices may fall short without sharing fully documented workflows and research infrastructures.

Responding to Lawrence et al. (2023), Haim & Puschmann (2023), and van Atteveldt et al.'s (2019) calls for more reproducible workflows and recognition of tools used in computational communication research, this paper focuses on documenting reproducible workflows. Specifically, I show how to set up reproducible data collection workflows using GitHub Actions. Using Bluesky posts tagged with the popular Canadian political hashtag, "#CdnPoli" from the 2025 Canadian federal election as a case study, I illustrate how researchers can systematically

collect digital trace data in reproducible and transparent ways. This paper contributes to the development of best practices in open science and reproducibility by documenting and explaining how to use GitHub Actions for data collection.

The organization of this paper is as follows. First, I provide background on GitHub and GitHub Actions, before reviewing reproducibility concerns in computational communication research. I then provide some brief context on the political uses of social media in Canadian politics to emphasize the relevance of collecting Bluesky posts tagged with the popular political hashtag, #CdnPoli as an example workflow. I then walk through, step-by-step, how to configure your repository settings and write a YAML file to run GitHub Actions workflows. Finally, I conclude with a discussion of best practices and recommendations for future workflow development.

GitHub and GitHub Actions background

Git, the version control system that underlies GitHub, was originally developed by Linus Torvalds in 2005 to manage changes in code across large software development teams (Evans & DeBacker, 2020). Building on Git's underlying system, Chris Wanstrath, P. J. Hyett, Tom Preston-Werner, and Scott Chacon launched GitHub in February 2008 (Evans & DeBacker, 2020). Since then, GitHub has become one of the most widely used platforms for collaborative software development, version control, continuous integration (CI), continuous deployment (CD), and the sharing of code and data, supporting open science and reproducibility practices in the natural and social sciences (Alexander, 2023).

Following Microsoft's acquisition of GitHub in 2018, the platform expanded the development of new tools and services. One such tool is GitHub Actions, which launched in 2019. GitHub Actions allows users to automate tasks such as testing, building, and deploying code (Valenzuela-Toledo & Bergel, 2022). Among developers and researchers in computer science and engineering, GitHub Actions is a standard tool for automating workflows (Kinsman et al., 2021). More recently, researchers in fields such as ecology and evolutionary biology recognized its potential to systematically collect data and collaborate (Braga et al., 2023; Kim et al., 2022).

Central to GitHub Actions is the concept of a workflow. A workflow is a clearly defined series of steps that are executed to achieve a specific goal. As van Atteveldt et al. (2019, p. 3940) argue, "documenting and sharing a workflow accelerates and improves scientific progress," as it enables other researchers to understand, evaluate, and replicate the exact procedures used in studies. GitHub Actions supports this goal by providing infrastructure to clearly define, execute, and document workflows in a transparent and reproducible way (Saroar & Nayebi, 2023).

GitHub Actions workflows use YAML syntax, which is described as a "... human friendly data serialization standard for all programming languages" (Heller, 2021, p. 36). YAML syntax is similar to XML and JSON, but requires indentations, similar to Python to define structure and hierarchy (Heller, 2021). YAML files serve as workflow configuration files for GitHub Actions, explicating when and how automated workflows should be executed. They outline

the environment setup, inputs and outputs, and the exact order in which scripts should run, ensuring that the workflow can be replicated across computing environments. Because YAML files are more declarative than imperative, they describe the desired end goal of the workflow, instead of detailing each procedural step (Laster, 2023).

Importantly, GitHub Actions is language agnostic, allowing researchers to run code written in nearly any programming language. This flexibility inherently supports open science practices by lowering technical barriers and enabling the development of reproducible workflows across disciplinary boundaries. Kinsman et al. (2021) found that software developers most commonly use Python, Java, and Ruby in their automated workflows. Communication researchers most frequently write code using R and Python (Chan et al., 2024), highlighting that existing disciplinary coding strengths are compatible with GitHub Actions. Despite this compatibility and GitHub's broader use for code and data sharing among communication researchers (van Atteveldt et al., 2019), its Actions features remain underused in the field. GitHub Actions has potential, particularly in the area of data collection. For example, GitHub Actions can be used to systematically collect social media comments during an election campaign, enabling continuous and reproducible data collection. Consequently, this paper addresses this methodological and technical gap by introducing how GitHub Actions can be applied to digital trace data collection, while promoting reproducible workflows and open science practices.

Computational Methods and Reproducibility in Communication

Computational social science methods enable researchers to collect and analyze digital trace data at an unprecedented scale (Lazer et al., 2009). Digital trace data can be defined "as records of activity (trace data) undertaken through an online information system (thus, digital)" (Howison et al., 2011, p. 769). These data enable researchers to examine user behaviour, online discourse, information flows, and user-generated textual and audio-visual content at a previously unavailable granular level using computational methods (Lazer et al., 2009; Ohme et al., 2024). However, the growing use of these data and computational tools also raises concerns about transparency, replicability, and reproducibility (van Atteveldt & Peng, 2018).

Reproducibility and replicability are growing concerns for communications researchers, as reflected in special issues devoted to these subjects (for example, Dienlin et al., 2021; McEwan et al., 2018; Shaw et al., 2021), the launch of *Computational Communication Research* (the journal), and the formation of the International Communication Association's (ICA) Computational Methods Division (Waldherr et al., 2025). These developments suggest growing institutional support for promoting reproducibility and replicability in research and fostering communities that foster best practices and the sharing of knowledge and tools. Despite these developments, reproducibility can be difficult to achieve in individual research projects. Researchers often write their own custom code even when they could reuse code written by others (van Atteveldt et al., 2019), some data cleaning and preprocessing steps go undocumented (van Atteveldt & Peng, 2018), and some social media platforms' Terms of Services (TOS) restrict or prohibit

data sharing (Freelon, 2018). Furthermore, researchers may face structural barriers, including limited incentives to share workflows and insufficient training in open science practices, which can hinder efforts to undertake reproducible research (Chan et al., 2024).

Other social science disciplines, including psychology, economics, and political science likewise confront questions about reproducibility and replicability in quantitative and computational research (Balafoutas et al., 2025; Breuer & Haim, 2024; Brodeur et al., 2024). In response to the replication crisis, psychology has more widely adopted preregistration and open data norms, along with making space for more publications of replication studies (Balafoutas et al., 2025; Breuer & Haim, 2024). Economics and political science are institutionalizing reproducibility through practices such as data and code sharing and the appointment of Data Editors at journals such as *Political Communication* (Brodeur et al., 2024; Lawrence et al., 2023). Although these fields are further along in developing best practices and tools, their approaches can offer valuable lessons for communication researchers seeking to improve transparency and reproducibility.

This paper contributes to these ongoing discussions in computational communication research and the broader social sciences by documenting how GitHub Actions can be used to automate and transparently record data collection processes. By offering a concrete example of a reproducible workflow, this paper adds to emerging efforts to improve the openness and reusability of computational tools in communication scholarship.

Canadian Politics on Social Media

A growing body of Canadian scholarship examines the political uses of Twitter (now X), Facebook, and other social media platforms during federal, provincial, and municipal election campaigns, as well as periods of governance (for example, Giasson et al., 2019; Raynauld & Greenberg, 2014; Small, 2011). These studies largely conclude that politicians and their staff use social media to broadcast partisan messaging, instead of engaging in real-time, two-way flows of communication with the public (Small, 2010).

Small's (2011) analysis of how Canadian politicians, journalists, citizens, and other actors use the popular Canadian political hashtag, #CdnPoli on Twitter (X) found that less than 10 percent of posts using the hashtag involved genuine conversation. Instead, users who employ this hashtag use it to discuss a variety of matters related to federal politics, including salient issues, policy announcements, politician and candidate character assessments, and discussions of "who won" and/or "who lost" during televised leaders' debates (Elmer, 2013). Small (2011) suggests that using the hashtag may be a way to cut through the "noisy environment" of Twitter (X) rather than fostering deliberative conversations between users.

Concerns over Twitter's (X) new ownership and political leaning prompted many Canadian Members of Parliament (MPs) affiliated with centre and centre-left leaning political parties, political practitioners, and journalists to migrate to Bluesky as an alternative platform to discuss Canadian politics (De Angelis & Alexander, 2025). Despite Bluesky's decentralized

network structure, De Angelis & Alexander (2025) found that MPs applied their understanding of Twitter's (X) affordances and logic directly to Bluesky, using the platform to post about policy issues, the Ottawa bubble, and their constituencies.

Another notable aspect of Bluesky is its accessible Application Programming Interface (API), which contrasts with the restrictive data access policies of other social media platforms in the "post-API" age (Freelon, 2018). This openness not only allows communication researchers to collect and analyze data to further understand political discourse in the fragmented media environment, but also supports open science practices and reproducibility by enabling the sharing of code, full datasets, and workflows.

Recognizing the previous research studying Canadian electioneering on social media and Bluesky's potential to support more deliberative political conversations, I systematically collected all Bluesky posts tagged with #CdnPoli during the 2025 Canadian federal election campaign. This GitHub Actions workflow enabled me to automatically collect and save all posts throughout the 37 day writ period (running from March 23 until April 28, 2025). The main goal in examining the uses of #CdnPoli on Bluesky is to demonstrate the practical utility of GitHub Actions workflows for computational communication research. These resulting datasets could be analyzed to better understand who uses the #CdnPoli hashtag, how they use it, and what this reveals about the state of deliberative conversations about Canadian politics during a federal election campaign.

Methods: How to Use GitHub Actions

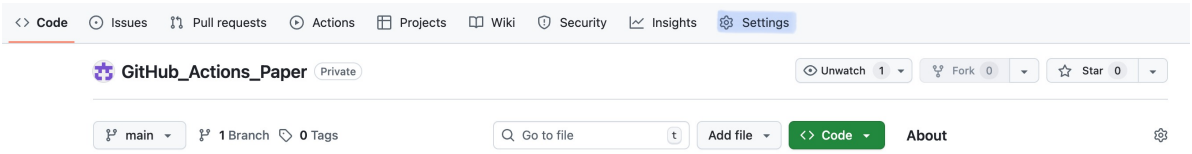
This section explains how to set up a GitHub repository and configure a YAML workflow file to successfully run GitHub Actions workflows. Snippets of code are shown below and all code is available on the Open Science Framework (OSF): https://osf.io/w43z5/?view_only=ec79c48ba5f74a7d94645e6b45129466, instead of the author's GitHub repository, to preserve anonymity for the peer-review process.

Configuring your repository

Although GitHub Actions can be run from any existing repository, the recommended first step is to create a new repository (either public or private) specifically for data collection purposes. The choice between a public or private repository should be made based on personal preference, project parameters, and data sharing restrictions set by the platform or source from where the data is collected (Freelon, 2018). While GitHub offers different tiers of paid plans that increase the data processing and storage capabilities, especially in private repositories, automated data collection using GitHub Actions works with a free account, provided that the usage does not surpass the free monthly minutes and storage quotas (Heller, 2021). Large-scale or long-term projects that are computationally intensive or exceed the free quota will incur

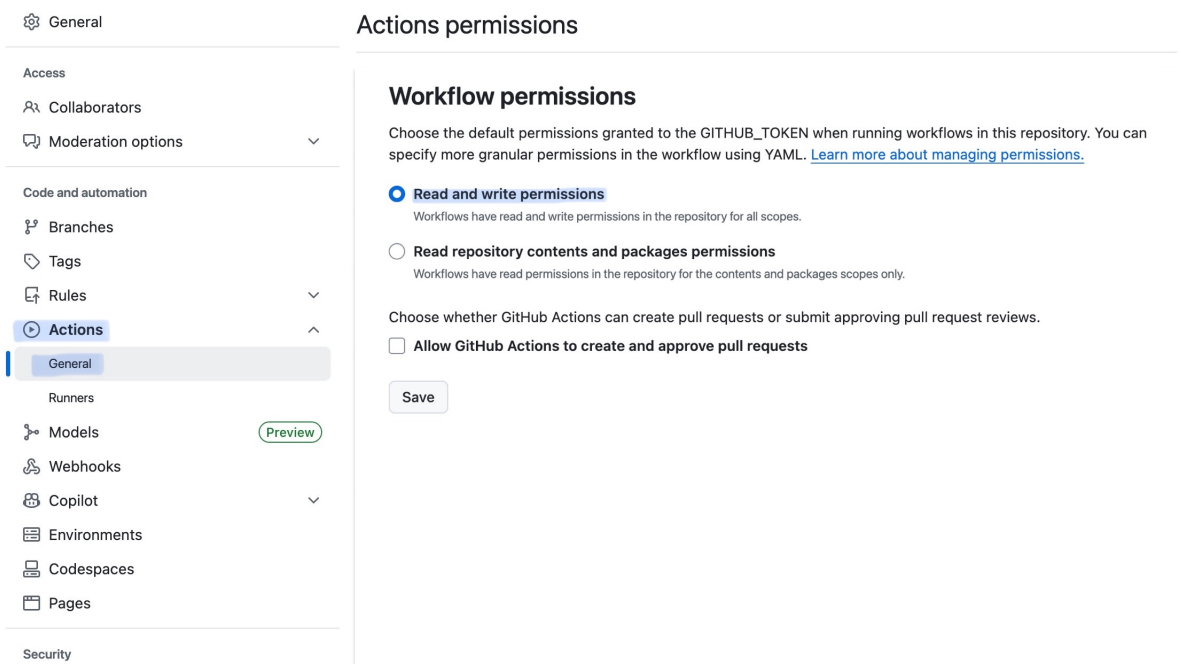
additional costs. Storage is billed based on hourly usage, while minutes are calculated based on the total processing time used throughout the month (GitHub, 2025a).

Figure 1: Locating the "Settings" tab in your repository



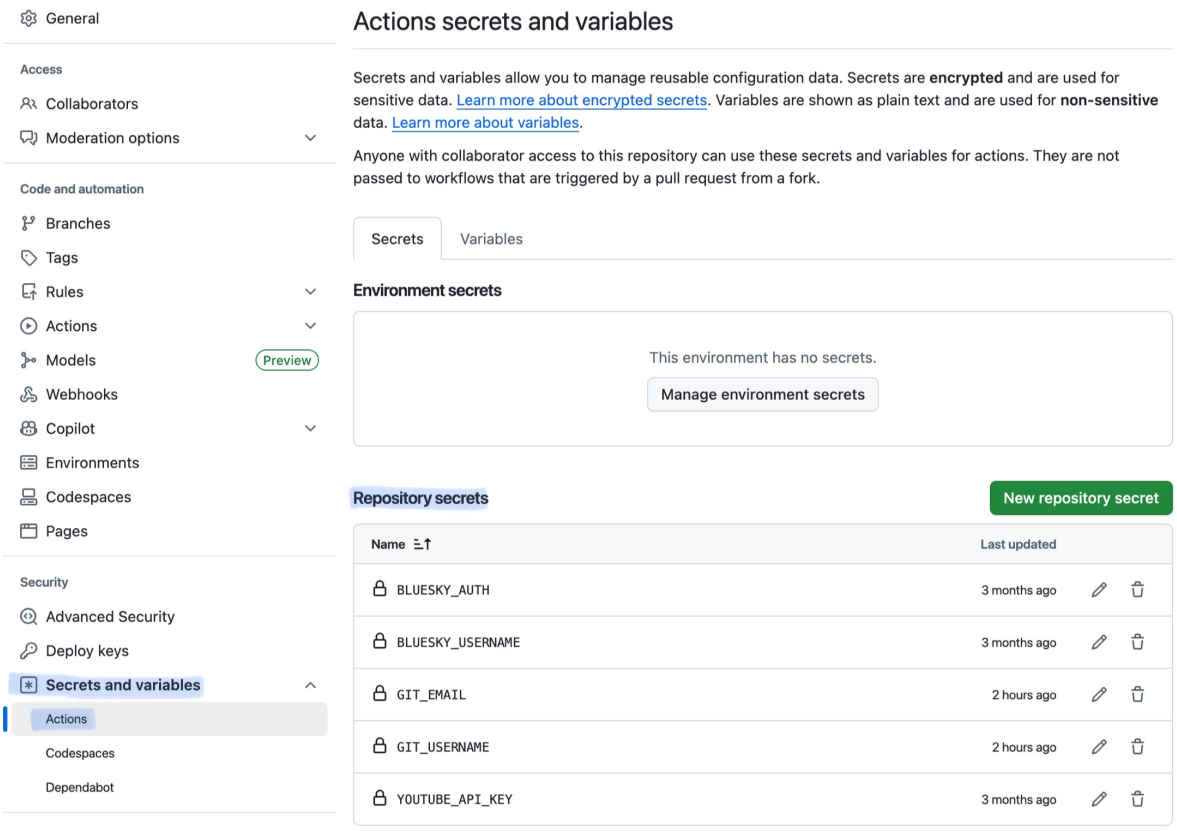
After selecting or creating your repository, certain settings need to be amended before any workflows can successfully run (see Figure 1). Within your repository settings, navigate to the "Actions" tab and then to the "General" sub-tab. Under the "Workflow permissions" section, click "Read and write permissions" (see Figure 2). Enabling this setting ensures that the `GITHUB_TOKEN` has the correct permissions to execute workflows and commit any newly created or modified files back to your repository. Committing to GitHub is part of version control, meaning you save changes to your files and maintain a record of the modifications (Alexander, 2023). The `GITHUB_TOKEN` is an automatically generated authentication token provided by GitHub, allowing workflows to securely access the GitHub API on behalf of GitHub Actions (Laster, 2023). Without this configuration, workflows may fail even if the scripts ran correctly and data collection succeeded, because the results cannot be committed.

Figure 2: Changing your workflow permissions



In this specific use case of collecting Bluesky posts, it is not necessary to manually add the `GITHUB_TOKEN` to the YAML file. The token will be automatically functional as long as the appropriate permissions are enabled. However, for other workflows, explicit usage of the `GITHUB_TOKEN` may be needed. For other use cases, users should consult the relevant GitHub Actions documentation for guidance on authentication and permission requirements.

Figure 3: Setting up your repository secrets



Next, you need to set up your "secrets" to securely store any sensitive information needed to run your workflow. This includes information such as usernames, API keys, and passwords for both GitHub and external platforms like Bluesky or YouTube. GitHub offers both repository and environment secrets (GitHub, 2025b). Repository secrets belong to a single repository and grant access to all workflows and environments inside it. In contrast, environment secrets apply to specific environments (such as "staging" or "production"), allowing you to run different configurations for jobs within the same repository.

To set up your repository secrets, go to your repository settings, navigate to the "security" section, then click on "Secrets and Variables," followed by "Actions" (see Figure 3). Once on the "Actions" page, click the green "New Repository Secret" button. You will be prompted to add a "name" for your secret and then the "secret" itself. It is recommended to use a short but

descriptive name (such as `bluesky_key`) when setting up your secrets, as the secret name will be used in place of the actual key or password in your R or Python data collection scripts. The value of the secret will be your API key, password, or username, ensuring that your sensitive information remains private and is not accidentally exposed directly in the code.

Setting up your workflow

After configuring your repository settings and adding secrets, you can create your workflow using a YAML file. There are two ways to set this up. The first is through the GitHub interface in your web browser, where you navigate to the "Actions" tab in your repository and click on hyperlinked "set up a workflow yourself" text (see Figure 4). This will take you to a new page with a blank YAML file where you can begin writing your workflow (see Figure 5).

Figure 4: Getting started with your GitHub Actions workflow

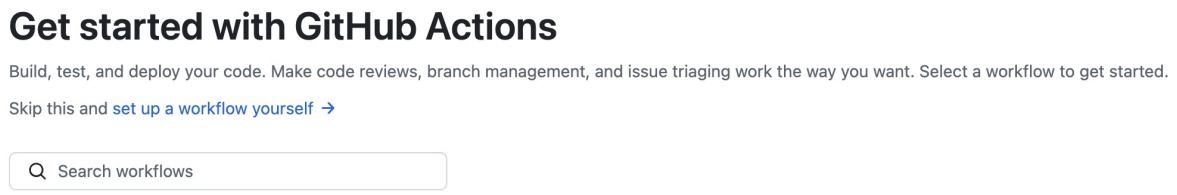
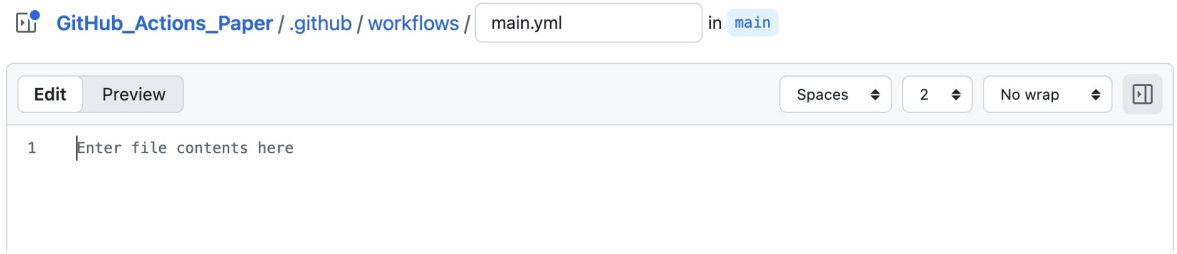


Figure 5: Setting up and writing your YAML file



Alternatively, you can initiate your workflow locally using a code editor such as RStudio or VS Code. To do this, create a new directory with the following structure: `.github/workflows/`. Inside this directory, add a new YAML file which ends with a ".YML" or ".YAML" extension (such as "Bluesky.YAML"). There is no functional difference between files with a ".YML" or ".YAML" extension, however ".YAML" is the preferred version (Heller, 2021). Regardless of how the file is created and which extension you use, the purpose remains the same: to define the configuration of your GitHub Actions workflow.

For the specific purpose of automating Bluesky data collection, your YAML file must include the following core components in order to successfully run: `Events`, `Jobs`, `Steps`, `Actions`,

and `Runners`. In the next section, I will discuss each component in more detail, accompanied by example YAML syntax.

Main elements of the workflow

Events: The `on` field in a YAML file specifies the event(s) that trigger the workflow. Common triggers include `push`, `pull_request`, or `schedule`. Multiple events, such as `push` and `fork` can also trigger the workflow. For workflows designed to automate daily or weekly social media data collection, the `schedule` trigger is used in conjunction with a `cron` expression, which defines the required frequency and time. The cron expression is in Coordinated Universal Time (UTC), not your local time zone.

The cron expression is made up of five fields. The first is minute, followed by hour, day of the month, month, and day of the week. The following YAML snippet demonstrates how to schedule the cron to run every day at 7:00 UTC.

```yaml
on:
  schedule:
    - cron: '0 7 * * *' # run every day at 7am UTC
```

Jobs: This section defines the core tasks that will be executed as part of the workflow. Each job is self-contained within a specific virtual environment called the "runner." The virtual environment refers to the isolated execution environment where workflow jobs run, while runners serve as the physical or virtual computers where the workflow code is executed (Laster, 2023). Runners are either provided and hosted by GitHub or self-hosted by the user. Within each job, there are a series of steps that need to be executed in order. These can include installing dependencies, executing scripts, and/or committing files to the repository. Jobs can run independently (in parallel) or sequentially. The steps required as part of this job will be outlined in more detail below.

Collect_posts: refers to the unique identifier or job ID that each job is assigned. This ID is employed internally within the workflow to reference or organize jobs. This job ID does not appear in your "Actions" tab of your GitHub Repository, only the name specified below. The job ID can be named anything, but it is best to choose something short, descriptive, and meaningful. I selected "collect_posts" because my workflow is designed to collect Bluesky posts or other social media data.

```yaml
jobs:
  collect_posts:
    name: Collect CdnPoli posts from Bluesky
    runs-on: macOS-latest
```

Name: This is the human-readable name that will also serve as the title for your workflow in the "Actions" tab of your repository. Ensure that it is something short, descriptive, and meaningful. As shown in the YAML snippet above, my workflow name is "Collect CdnPoli posts from Bluesky."

Runs on: This field specifies the runner environment in which each job is executed (GitHub, 2025b). This environment acts as a virtual machine provided by GitHub and determines the operating system and software environment available for the workflow. When configuring their environment, users specify the latest version (for example, macOS-latest) or a specific version (for example, windows-2019) based on the script or software they plan to run.

Steps: Each job in a GitHub Actions workflow is composed of a series of steps, which need to be executed in order by the runner. These steps are written as list items under the `steps` field. Most workflows contain multiple steps that must be executed in a specific sequence. Steps may use predefined GitHub Actions (indicated by the uses: keyword) or custom shell commands (indicated by the run: keyword). Each step will be further explained below.

The first command uses GitHub's official "actions/checkout@v2" action, which clones the content of your repository into the runner's environment. This step ensures that your workflow has access to all the files and scripts in your repository.

```
    steps:
      - uses: actions/checkout@v2
```

The next step prepares the R environment in the virtual machine, which allows the runner to install the relevant R packages and execute the R scripts during the workflow.

```
      - uses: r-lib/actions/setup-r@v2
```

After R (R Core Team, 2024) is installed, you must install the necessary R packages to collect and manipulate data from Bluesky using `Rscript`. The two required packages for this workflow are `tidyverse` (Wickham et al., 2019) for data wrangling and `atrrr` (Gruber et al., 2024) for interacting with the Bluesky API. A third package, `dotenv` (Csárdi, 2021) for loading environment variables, could also be used as part of this workflow as an alternative to GitHub secrets. Alternatively, dependencies can be listed in a .txt file (for example, requirements.txt) depending on the programming language and workflow structure.

```
      - name: Install dependencies
        run: Rscript -e 'install.packages(c("tidyverse", "atrrr"))'
```

The next step accesses your secret environment variables, including your Bluesky username and password, securely stored in the repository settings. It also runs your custom R script (`01-download_Bluesky_data.R`) to authenticate the Bluesky API, collect the latest #CdnPoli posts, do any basic data cleaning, and save the Bluesky posts into a CSV file.

```
      - name: Collect CdnPoli posts from Bluesky
        env:
          BLUESKY_USERNAME: ${{ secrets.BLUESKY_USERNAME }}
          BLUESKY_AUTH: ${{ secrets.BLUESKY_AUTH }}
        run: |
          Rscript "01-download_Bluesky_data.R"
```

Furthermore, to enable GitHub to automatically push any new or modified files back to your repository, you must configure the Git user credentials using the secrets previously stored. This step ensures that the runner is properly authenticated as a GitHub user or bot, allowing it to make commits on your behalf. GitHub bots are automated accounts that can undertake tasks like committing code or merging pull requests on your behalf (Heller, 2021; Kinsman et al., 2021). Bots do not use your personal Git credentials, but still require authentication to securely interact with the repository and execute tasks. For more information on how to authenticate a GitHub Actions bot to commit on your behalf, see Yanjingzhu (2019).

```
      - name: Configure git user
        run: |
          git config --global user.name "${{ secrets.GIT_USERNAME }}"
          git config --global user.email "${{ secrets.GIT_EMAIL }}"
```

The workflow attempts to commit and push all changes to the repository. This includes committing the newly collected dataset and updated "last_run_time.txt" file to the repository. If no changes exist (for example, there are no new #CdnPoli posts), the script prints a "no changes to commit" message instead of failing. For scheduled workflows like this one focusing on daily or weekly social media data collection, there is typically at least one small change in each run, such as an updated last_run_time.txt file, which records the time of the most recent workflow execution. The successful commit message will say "New data!."

```
      - name: Commit results
        run: |
          git add -A
          git commit -m 'New data!' || echo "No changes to commit"
          git push origin || echo "No changes to commit"
```

Finally, although this step is not specified in our YAML workflow file, GitHub Actions automatically concludes the workflow by performing behind-the-scenes cleanup tasks after our job finishes. This step is labeled "Post Run actions/checkout@v2" in our workflow log. During this step, the cleanup procedures typically include removing temporary credentials and tidying up any environment settings used during the repository checkout.

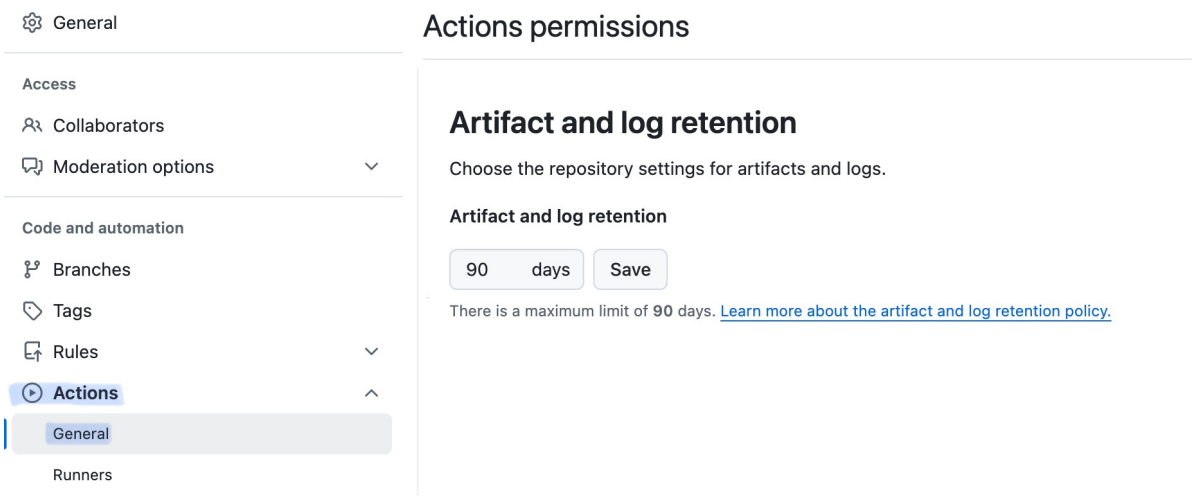
```
Post Run actions/checkout@v2
```

Reviewing your Workflow Logs

After your workflow has run, you can review the detailed execution logs to verify success or identify the step where errors occurred. To review this log, go back to the "Action" tab in your repository, followed by clicking on the title of your workflow. Then you will be prompted to click on the workflow title, listed by date. In this view, each step defined in your YAML file is displayed, showing exactly where the process succeeded or failed and how long each step took to run. If a workflow fails to execute at any stage, this log will indicate the step where the failure occurred, along with error messages that can help diagnose the issue. Reviewing this information is essential for debugging and maintaining a reproducible data collection workflow.

GitHub also allows you to enable email or mobile push notifications (via the GitHub app) to alert you when a workflow fails. Workflow logs store information for up to 90 days by default, with the option to customize this period in your repository settings. To adjust the amount of time, go to your repository settings, then click on "Actions," followed by "General," and then look under the "Artifact and log retention" section (see Figure 6).

Figure 6: Changing the number of days for artifact and log retention



Lastly, it is good practice to regularly review your datasets collected via GitHub Actions to ensure that no data is missing or incorrect. Even if the workflow successfully executes, downstream issues such as hitting API limitations or incomplete responses may affect the completeness or accuracy of your dataset. Once the data is verified, you can proceed to the data cleaning and analysis stages of your project.

Discussion and Conclusion

This paper provides a step-by-step guide for setting up a GitHub Actions workflow for computational communication research, using the collection of Bluesky posts tagged with the hashtag #CdnPoli as a case study. In doing so, this paper addresses gaps in current computational communication research by documenting how to set up transparent and automated workflows for data collection.

One of the benefits of using GitHub Actions is that it embeds reproducibility into the research process from the outset. As Chan et al. (2024) argue, reproducibility should be intentional, not an afterthought. Rather than retroactively cleaning up or re-writing code before making it public for replication purposes, GitHub Actions compels researchers to proactively document and continually maintain their workflows, operationalizing Chan et al.'s argument.

While GitHub is already used by communication researchers for code and data sharing, its automation features remain underused in the field (van Atteveldt et al., 2019). Integrating GitHub Actions into research pipelines enables researchers to implement CI and CD, while transparently and systematically collecting data. This approach also aligns with calls for greater transparency, documentation, and code sharing to bolster open science and reproducibility in communication research (Dienlin et al., 2021; Haim & Puschmann, 2023; Waldherr et al., 2025).

Beyond data collection, GitHub Actions could be further employed to support other stages of the computational workflow, including data cleaning, preprocessing, and analysis (van Atteveldt et al., 2019). As open science practices gain traction across communication and the broader social sciences, researchers need to standardize and document the processes that produce research outputs. This paper contributes to these efforts by offering a practical and scalable approach for automating computational workflows in communication research.

Competing Interests

The author declares no competing interests.

Acknowledgements and Funding

I am grateful to Rohan Alexander for encouraging me to incorporate GitHub Actions into my workflow and for providing valuable feedback on earlier drafts of this paper.

This research was supported in part by the Data Sciences Institute at the University of Toronto and the Social Sciences and Humanities Research Council of Canada (grant ID: 767-2025-1588).

References

Alexander, R. (2023). *Telling Stories with Data: With Applications in R*. Chapman; Hall/CRC.

Balafoutas, L., Celse, J., Karakostas, A., & Umashev, N. (2025). Incentives and the Replication Crisis in Social Sciences: A Critical Review of Open Science Practices. *Journal of Behavioral and Experimental Economics*, *114*, 102327. https://doi.org/10.1016/j.socec.2024.102327

Braga, P. H. P., Hébert, K., Hudgins, E. J., Scott, E. R., Edwards, B. P., Sanchez Reyes, L. L., Grainger, M. J., Foroughirad, V., Hillemann, F., Binley, A. D., et al. (2023). Not Just for Programmers: How GitHub can Accelerate Collaborative and Reproducible Research in Ecology and Evolution. *Methods in Ecology and Evolution*, *14*(6), 1364–1380. https://doi.org/10.1111/2041-210X.14108

Breuer, J., & Haim, M. (2024). Are We Replicating Yet? Reproduction and Replication in Communication Research. *Media and Communication*, *12*. https://doi.org/10.17645/mac.8382

Brodeur, A., Esterling, K., Ankel-Peters, J., Bueno, N. S., Desposato, S., Dreber, A., Genovese, F., Green, D. P., Hepplewhite, M., Hoces de la Guardia, F., Johannesson, M., Kotsadam, A., Miguel, E., Velez, Y. R., & Young, L. (2024). Promoting Reproducibility and Replicability in Political Science. *Research & Politics*, *11*(1), 1–8. https://doi.org/10.1177/20531680241233439

Chan, C., Schatto-Eckrodt, T., & Gruber, J. (2024). What Makes Computational Communication Science (Ir)Reproducible? *Computational Communication Research*, *6*(1), 1–30. https://doi.org/10.5117/CCR2024.1.5.CHAN

Csárdi, G. (2021). *dotenv: Load Environment Variables from '.env'*. https://CRAN.R-project.org/package=dotenv

De Angelis, I., & Alexander, R. (2025). What are Canadian Members of Parliament Doing on Bluesky? *Canadian Journal of Political Science/Revue Canadienne de Science Politique*, 1–22. https://doi.org/10.1017/s0008423925100619

Dienlin, T., Johannes, N., Bowman, N. D., Masur, P. K., Engesser, S., Kümpel, A. S., Lukito, J., Bier, L. M., Zhang, R., Johnson, B. K., Huskey, R., Schneider, F. M., Breuer, J., Parry, D. A., Vermeulen, I., Fisher, J. T., Banks, J., Weber, R., Ellis, D. A., … De Vreese, C. (2021). An Agenda for Open Science in Communication. *Journal of Communication*, *71*(1), 1–26. https://doi.org/10.1093/joc/jqz052

Elmer, G. (2013). Live Research: Twittering an Election Debate. *New Media & Society*, *15*(1), 18–30. https://doi.org/10.1177/1461444812457328

Evans, R. W., & DeBacker, J. (2020). *Git and GitHub Tutorial: Use, Collaboration, and Workflow*. https://pslmodels.github.io/Git-Tutorial/

Freelon, D. (2018). Computational Research in the Post-API Age. *Political Communication*, *35*(4), 665–668. https://doi.org/10.1080/10584609.2018.1477506

Giasson, T., Le Bars, G., & Dubois, P. (2019). Is Social Media Transforming Canadian Electioneering? Hybridity and Online Partisan Strategies in the 2012 Quebec Election. *Canadian Journal of Political Science/Revue Canadienne de Science Politique*, *52*(2), 323–341. https://doi.org/10.1017/S0008423918000902

GitHub. (2025a). *About Billing for GitHub Actions*. https://docs.github.com/en/billing/concepts/product-billing/github-actions

GitHub. (2025b). *Understanding GitHub Actions*. https://docs.github.com/en/actions/get-started/understand-github-actions.

Gruber, J. B., Guinaudeau, B., & Votta, F. (2024). *atrrr: Wrapper for the "AT" Protocol Behind "Bluesky"*. https://jbgruber.github.io/atrrr/

Haim, M., & Puschmann, C. (2023). Opening up Data, Tools, and Practices: Collaborating with the Future: Introduction to the Special Issue Analytical Advances through Open Science: Employing a Reference Dataset to Foster Best-Practice Data Validation, Analysis, and Reporting. *Digital Journalism*, *11*(2), 247–254. https://doi.org/10.1080/21670811.2023.2174894

Heller, P. (2021). *Automating Workflows with GitHub Actions: Automate Software Development Workflows and Seamlessly Deploy your Applications using GitHub Actions*. Packt Publishing Ltd.

Howison, J., Wiggins, A., & Crowston, K. (2011). Validity Issues in the Use of Social Network Analysis with Digital Trace Data. *Journal of the Association for Information Systems*, *12*(12), 2. https://doi.org/10.17705/1jais.00282

Kim, A. Y., Herrmann, V., Barreto, R., Calkins, B., Gonzalez-Akre, E., Johnson, D. J., Jordan, J. A., Magee, L., McGregor, I. R., Montero, N., et al. (2022). Implementing GitHub Actions Continuous Integration to Reduce Error Rates in Ecological Data Collection. *Methods in Ecology and Evolution*, *13*(11), 2572–2585. https://doi.org/10.1111/2041-210X.13982

Kinsman, T., Wessel, M., Gerosa, M. A., & Treude, C. (2021). How do Software Developers use GitHub Actions to Automate their Workflows? *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, 420–431. https://doi.org/10.1109/MSR52588.2021.00054

Laster, B. (2023). *Learning GitHub Actions: Automation and Integration of CI/CD with GitHub* (First edition). O'Reilly Media, Inc.

Lawrence, R. G., Arceneaux, K., Clemm Von Hohenberg, B., Dunaway, J., Esser, F., Kreiss, D., Rinke, E. M., & Thorson, K. (2023). *New Methods, Old Methods: Emerging Trends and Challenges in Political Communication Research*. 1–8. https://politicalcommunication.org/article/new-methods-old-methods-emerging-trends-and-challenges-in-political-communication-research/

Lazer, D., Pentland, A. (Sandy)., Adamic, L., Aral, S., Barabási, A.-L., Brewer, D., Christakis, N., Contractor, N., Fowler, J., Gutmann, M., Jebara, T., King, G., Macy, M., Roy, D., & Van Alstyne, M. (2009). Life in the Network: The Coming Age of Computational Social Science. *Science*, *323*(5915), 721–723. https://doi.org/10.1126/science.1167742

Lewis, N. A., Jr. (2020). Open Communication Science: A Primer on Why and Some Recommendations for How. *Communication Methods and Measures*, *14*(2), 71–82. https://doi.org/10.1080/19312458.2019.1685660

Marwick, B., Boettiger, C., & Mullen, L. (2018). Packaging Data Analytical Work Reproducibly Using R (and Friends). *The American Statistician*, *72*(1), 80–88. https://doi.org/10.1080/00031305.2017.1375986

McEwan, B., Carpenter, C. J., & Westerman, D. (2018). On Replication in Communication

Science. *Communication Studies*, *69*(3), 235–241. https://doi.org/10.1080/10510974.2018.1464938

National Academies of Sciences, Engineering, and Medicine. (2019). *Reproducibility and Replicability in Science* (1st ed.). National Academies Press. https://doi.org/10.17226/25303

Ohme, J., Araujo, T., Boeschoten, L., Freelon, D., Ram, N., Reeves, B. B., & Robinson, T. N. (2024). Digital Trace Data Collection for Social Media Effects Research: APIs, Data Donation, and (Screen) Tracking. *Communication Methods and Measures*, *18*(2), 124–141. https://doi.org/10.1080/19312458.2023.2181319

Perreault, G. P., & Dienlin, T. (2025). Normalizing Open Science Practice: Understandings, Evaluations, and Implementations of Open Science Practices in the Field of Communication. *Journalism & Mass Communication Quarterly*, *102*(1), 274–299. https://doi.org/10.1177/10776990241262346

R Core Team. (2024). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Raynauld, V., & Greenberg, J. (2014). Tweet, Click, Vote: Twitter and the 2010 Ottawa Municipal Election. *Journal of Information Technology & Politics*, *11*(4), 412–434. https://doi.org/10.1080/19331681.2014.935840

Saroar, S. G., & Nayebi, M. (2023). Developers' Perception of GitHub Actions: A Survey Analysis. *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, 121–130. https://doi.org/10.1145/3593434.3593475

Shaw, A., Scharkow, M., & Wang, Z. J. (2021). Opening a Conversation on Open Communication Research. *Journal of Communication*, *71*(5), 677–685. https://doi.org/10.1093/joc/jqab033

Small, T. A. (2010). Canadian Politics in 140 Characters: Party Politics in the Twitterverse. *Canadian Parliamentary Review*, *33*(3). http://www.revparl.ca/33/3/33n3_10e_Small.pdf

Small, T. A. (2011). What the Hashtag? A Content Analysis of Canadian Politics on Twitter. *Information, Communication & Society*, *14*(6), 872–895. https://doi.org/10.1080/1369118X.2011.554572

Valenzuela-Toledo, P., & Bergel, A. (2022). Evolution of GitHub Action Workflows. *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 123–127. https://doi.org/10.1109/SANER53432.2022.00026

van Atteveldt, W., & Peng, T.-Q. (2018). When Communication Meets Computation: Opportunities, Challenges, and Pitfalls in Computational Communication Science. *Communication Methods and Measures*, *12*(2-3), 81–92. https://doi.org/10.1080/19312458.2018.1458084

van Atteveldt, W., Strycharz, J., Trilling, D., & Welbers, K. (2019). Toward Open Computational Communication Science: A Practical Road Map for Reusable Data and Code. *International Journal of Communication*, *13*, 20. https://ijoc.org/index.php/ijoc/article/viewFile/10631/2765

Waldherr, A., Weber, M., Wu, S., Haim, M., Velden, M. A. van der, & Chen, K. (2025). Between Innovation and Standardization: Best Practices and Inclusive Guidelines in Computational Communication Science. *Journalism & Mass Communication Quarterly*, *102*(1), 13–36.

https://doi.org/10.1177/10776990241301676

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., ... Yutani, H. (2019). Welcome to the Tidyverse. *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686

Yanjingzhu. (2019). *GitHub Actions bot email address?* [Online forum post]. GitHub Community Discussion. https://github.com/orgs/community/discussions/26560#discussioncomment-3531273