The benefits of an open science approach in student research projects

Edward Wallace

Emma MacKenzie

Flic Anderson

Sophie Winterbourne

## Abstract

Open science is the idea “to allow scientific information, data and outputs to be more widely accessible (Open Access) and more reliably harnessed (Open Data) with the active engagement of all the stakeholders (Open to Society)”. Open science can also describe openness within a research group where all participants share their data, code, ideas, and feedback. These ideas can be applied to all aspects of science, from large research consortia to undergraduate projects. With great accessibility comes greater reproducibility, leading to better code quality and better research.

Here we describe what we have learned and gained from taking an open science approach in student research projects, from the perspective of the student, the day-to-day supervisor, and the PI or lab head. The undergraduate honours research project described in this article involved working with a data analysis pipeline for ribosome profiling data, riboviz, to analyse translation dynamics. riboviz is an example of open scientific software as all of the code required to run it is accessible on GitHub, along with information on errors and software developments documented in issue tickets. riboviz also takes an open science approach in that all team members can see and contribute to all work in progress.

# The undergraduate perspective

## Freedom to share information and anxiety.

Being students studying doing a Biotechnology Honours course, there was the potential to work on a variety of projects, with varying levels of openness, including projects working with trade secrets or compounds with therapeutic potential. In a talk prior to picking our projects, we were told a story of an undergraduate who had broken their disclosure agreement, leading to the name of the drug they were working on being released on Twitter. The lab’s competitors were able to find it, and this mistake cost the lab a lot of money. The idea of potentially leaking intellectual property caused a lot of anxiety. Working on an open science project removed this risk and this fear.

## How it differed from a traditional project

A traditional project, in theory, would involve designing and attempting an experiment, or scouring literature with the goal of answering a research question. This would potentially involve working with ‘trade secrets’ and would likely end with the submission of the final project. Any impact on the hosting lab, for example contributions leading to a publication, would not be visible to those outside the lab until a publication is made, which may be long after the student has left. Our project involved jumping into a team working on the riboviz data analysis pipeline, and contributing to the overall development of riboviz in the form of adding new datasets, testing the pipeline, and commenting on its usability from the perspective of complete novices. Our work is documented on GitHub and is publicly accessible, providing proof of our contributions to the overarching project in addition to our own projects. As our work is publicly accessible it provides concrete evidence of our abilities which can be accessed by potential employers. This instant visibility of our work would likely not be available had we completed a traditional project.

Compared to a traditional project, it is possible that contact hours were different. The contact hours of an undergraduate project depend heavily on the type of project – with a lab-based project requiring lots of contact hours, with students being in the lab with their supervisor. Alternatively for a literature-based research project, or a community outreach project, there would likely be fewer contact hours, as the student would require more guidance rather than constant supervision. We did not have any in-person contact hours. This was partially due to the COVID-19 pandemic, but also due to the project being completely computer based. However, we were able to attend regular meetings with multiple labs, or with our supervisors, and we posted regular updates on the lab [Slack](https://slack.com/intl/en-gb/) channel, using the Stand-Up Report method, to keep our supervisors updated on our progress. This was supplemented by issue tickets and commits on GitHub which our supervisors were able to see and comment on. However, the cost of a virtual computer-based project is that we potentially missed out on improving and gaining new practical lab skills.

## What was it like

We had the freedom to join a supportive lab where we were encouraged to take our projects in any direction of interest, while still being guided towards contributing useful material to the wider community. The open-sourced nature of the project allowed for wider collaboration with other members of the lab and cooperating with research groups across the world. It provided more variation and more potential for learning new skills as you were free to aid in different projects, in addition to your own, without the complication of the projects being confidential.

Completing an open-science project was a very exciting experience, though rather intimidating as it felt like we were contributing to a larger project rather than completing an isolated experiment or literature review. The idea that everything we did could be seen by others and had the potential to help them fuelled this feeling. Knowing that experts within the field could examine the code we had written or critique the conclusions we had reached created an additional layer of motivation in terms of maintaining high quality work that could be understood by others. Every aspect of the project being freely available also removed the anxiety that would have arisen if we had been dealing with confidential ideas or processes, for example testing potential antibiotics. Instead of hiding trade secrets, we were able to freely collaborate and learn from the work of others. Additionally, there was more room for dialogue with other students completing open-science projects. This was a valuable aid as it allowed us to bounce ideas off each other compared to the stress and isolation of a closed project that could not be discussed with peers.

## What was good or bad

The open-science nature of the project simplified communicating about it with other members of the lab when we were all working virtually. Though it was intimidating to be working with experts from around the world at such an early stage of our careers, it was valuable experience. Additionally, as everything was documented in the form of issues and commits on GitHub, our contributions to the overall project is available for any future employer to see, which might not have been the case if working out of a private lab book. When we first started the project, some of the documentation was confusing at times, as we had never worked with the software before, or anything like it. However, this meant we were able to provide feedback, and ultimately make the software more novice friendly by improving user documentation.

It did feel like we were dropped in the deep end as during our first official week there a ‘hackathon’ took place. This involved all individuals working on riboviz focusing solely on its development for that week. Having never worked in a lab before, let alone in a bioinformatics one, this was rather overwhelming. Everyday ended with a meeting, where everyone reported on their day, so there was pressure to learn quickly and be able to share progress. These meetings were like going on an intense language immersion course, and it felt like our brains were melting by the end of each day. There was a steep learning curve, but in the long run, this one week of ‘trial by fire’ made everything else seem more manageable in comparison. Additionally, we were able to see how a large group could operate efficiently through issue tickets and Git, which at the time were still intimidatingly new to us. The hackathon also meant our supervisors were both completely focused on this project while we were starting, so they could help us adjust to the new environment.

Overall, the project gave us some valuable skills for further study and working with open-science. We learned to prioritise tasks based on their importance within the group rather than for our individual needs. We also gained better debugging skills, knowing where to look in documentation and on google to help us solve problems.

Our communication skills were greatly improved by interacting with our supervisors and the lab group in the course of the project through our Stand-Up Reports and issue tickets on GitHub. The importance of communicating progress and obstacles frequently in a precise and informative manner, especially as all communication occurred virtually, quickly became obvious to avoid confusion and get help more quickly. This development is clearer now that we occasionally return to issue tickets written in the early stages of the project, and curse at our past selves for documenting the problem, and more importantly the solution, so poorly.

## How did we learn the tools

To complete our projects we needed to be able to confidently use the command line, Git, GitHub, the university supercomputer, and the data analysis pipeline riboviz. Early in the project we were given a tutorial on using Git and GitHub by our supervisor, Flic. This was extremely helpful and, as the tutorial materials were stored on GitHub, we were able to refer back to them any time we were feeling lost or confused. We were also pointed in the direction of learning materials produced by The [Carpentries team](https://carpentries.org/), including a course called ‘The Unix Shell’ which provided a solid understanding of using the command line to navigate and manipulate files. Within the first few weeks of the project, it became clear that we would be unable to run large datasets through the riboviz pipeline on a underpowered personal laptop, so we needed to learn how to remotely access the university’s supercomputer. Using resources provided by the university, we learned how to navigate the supercomputer, submit jobs and run interactive sessions.

The most important step of the project was learning how to use the data analysis pipeline [riboviz](https://github.com/riboviz/riboviz). Fortunately, the pipeline is reasonably well documented which made the experience slightly smoother. We were able to provide feedback on documentation while we were learning how to run the pipeline, and contributed to the improvement of user documentation. Time and practice meant that by the end of the project we were confidently running datasets through riboviz and writing our own visualisation code. The experience highlighted how important it is to document the use of scripts and code, which means we will be able to write better code in the future.

## Impact of COVID-19

We chose our projects in September 2020, then worked on them full time from January to May 2021, during the COVID-19 pandemic. All projects that were offered had to be adapted, as lab-based projects were not feasible during the constantly changing restrictions that came throughout 2020 and 2021. This meant disruptions to planned lab-based projects. Luckily the riboviz project was designed to be a computer based project, meaning we were able to work from home effectively, with as little disruption as possible. Sadly, during our project we never met anyone from the lab in person, which could have left us feeling extremely isolated. Fortunately, the lab was very welcoming, allowing us to attend all meetings and journal clubs. The open-science nature of the project came with the social benefit of being able to enthusiastically discuss the project with family and friends, which was extremely helpful in these times of working from home and lockdowns, as the project was the main focus of our lives for many months.

## Advice for undergrads selecting their projects

If you are an undergrad considering what direction to take your studies try to complete an undergraduate project in a field you are considering entering into, even if you have no prior experience. I was considering completing an MSc in bioinformatics and therefore picked a bioinformatics-based project. I found the experience of stepping outside of my comfort zone and into a field where I had little experience daunting but very valuable. The project provided a positive experience and a fantastic learning curve which confirmed my initial interest in the field and allowed me to gain confidence and competence within coding and software.

## Our top takeaways

* Being able to access other researchers work helped speed up the project. For example, when trying to create transcriptome annotation files for a new example species, I was able to look at pre-existing code and use it as a starting point for learning which functions and packages could be used. This saved me from starting from scratch and helped guide me.
* It is easier to learn when resources are available to refer back to.
* Open-Science is like a group project. Except the world is your group and they won’t affect your grade.
* Documenting your work helps you, aids others and helps them help you.

# The day-to-day supervisor perspective

As an early-stage researcher, it can often be difficult to find opportunities to develop and practice coaching, teaching and supervisory skills alongside making progress on your own workload and helping contribute to the lab’s goals. Likewise, as someone heavily involved on the riboviz project as a developer, it often seems like there are so many useful new features or improvements we’d like to add, and never enough time to get them done. Enter: undergraduate open science thesis projects!

## Overlapping goals

Supervising undergraduate students working on riboviz-related research projects in a day-to-day capacity along with my PI has presented an excellent opportunity for me to build key skills, as well as aligning the supervision with relevant work of my own, meaning that there’s not as great a ‘brain switch’ required when going between supervision tasks and my standard workload. It also means that I have been able to improve my own knowledge while helping support the students to learn the concepts, technical skills and organisational techniques required to complete their projects successfully.

Having two students joining to pursue projects with riboviz at the same time this year was particularly helpful, as a lot of the instruction and coaching were applicable to both, and meant that there was also an opportunity for peer learning. While not all labs might have capacity to offer multiple projects, I was surprised to find that the increase in supervision time required by two students was much less than expected. Many of the same problems were encountered during both projects and could be solved in one discussion.

Supervising two undergraduate projects at the same time was also unexpectedly valuable in highlighting areas where my instruction, knowledge, or communication skills could be improved: *“When one student doesn’t seem to understand, it could go either way. When two students don’t seem to understand, the problem probably lies with the communicator!”* The process has been really helpful in making me re-think how I communicate complex ideas, technical information and scientific concepts, and has contributed significantly to my personal development.

## Open Software Tools & Student Supervision

Introducing new tools and ideas at the beginning of a new project can always be time consuming initially. But many of the tools used in ‘open software’ projects such as riboviz lend themselves extremely well to use in a student supervision context, as well as making the task of developing research software easier. I have co-supervised undergraduate students previously where we’ve introduced concepts such as version control much later in the project timeline, with less success, but this year we took the ‘in at the deep end’ approach (with floatation aids available in the form of plenty of technical support and helpful and approachable colleagues).

With a little coaching early on in the project to demonstrate which tools we use to develop riboviz, as well as how we manage the riboviz project collaboratively, it was much easier to keep track of what tasks students were working on, answer questions, identify and resolve potential issues and deliver feedback on work - particularly any code. The fact that the rest of the riboviz team all use these tools definitely helped to make this a more ‘normal’ way of working, and meant we were able to answer any questions about how and when to use the tools.

After a short workshop on git version control and some early trouble-shooting, being able to ping code back and forth, refer to line numbers and particular snapshots of code, and ease of testing whether code would run on other machines proved a really powerful way to supervise students taking their first steps beyond using R within undergraduate courses and learning to become software developers.

Using zoom for remote ‘pair-programming’ sessions was also quite helpful, and possibly more convenient than doing the same exercise in-person (particularly if doing so with more than one student), as screen-sharing while live-coding ensures everyone can see the whole screen, rather than everyone crowding around one monitor.

GitHub issue tickets have been invaluable - they provide mutually accessible, backed-up spaces for making notes, keeping track of progress, linking ideas and concepts, asking questions, tagging in colleagues to get other opinions or help, and also help to store information which might be helpful for the overall riboviz project, not just the students’ projects. It’s very satisfying to be able to bat code, manuscripts and ideas back and forth using GitHub and issue tickets to quickly resolve a problem.

Using issue tickets also helps to get students thinking about the ways ideas can be broken down into tasks, and how best to prioritise these different steps to achieve larger goals, which is another key skill these projects help undergraduate students practice.

Using a chat-like service such as Slack was also helpful as it gave dedicated channels for asking questions, distributing meeting notes, sharing files and celebrating successes.

The Wallace lab runs a Slack-based [‘stand-up’ session](https://ewallace.github.io/manual/standup) three times a week (based on software development management practices), where members share in a couple of sentences: 1) What did I do or achieve recently? 2) What will I prioritize next? 3) What obstacles are impeding my progress? This practice was extremely helpful at staying up-to-date with what the students were working on currently, and what was working and what they might need extra support with.

The stand-up updates worked very well alongside weekly zoom meetings which helped get a more in-depth idea of progress and gave more opportunities to explore any issues in more depth, and share additional resources or demonstrate coding techniques (for example, these zoom sessions included demos of how to resolve real-world git merge conflicts, or how to debug R code using the command-line).

While the traditional weekly zoom supervision meetings helped to tackle larger issues and were always productive, I think that having the option of using these ‘less formal’ and always-open channels of communication such as slack channels and issue tickets helped make asking questions slightly less daunting. I feel that this was helpful in building a good mutual working relationship and definitely benefited the projects overall.

## Teamwork

Having undergraduate students joining our team to work on their own projects meant that they helped contribute more ideas, discovered more possibilities for improvement, and helped us progress towards the overall team goals faster.

Both Sophie and Emma have mentioned that introducing them to the project meetings and lab meetings early on was quite helpful in allowing them to see how the riboviz project and the Wallace lab were managed, meet colleagues, become familiar with the different areas of work and how those might interact with their own project work, and from a more social point of view, it helped them get a feel for the team’s dynamics - something that has been much harder to do in a remote-working context. Involving them in our regular meetings was another opportunity to check in with them and gauge progress from a supervision point of view, but it also helped integrate them within the wider group: we all had more awareness about what was being worked on, opportunities for collaboration, problem solving and a real feeling that these are skilled and engaged team members not just external undergrad students.

As their day-to-day supervisor I’ve probably benefited most from the experience of helping to train and guide such bright and engaged students, learning a lot about project organization and planning, communication, time management, as well as enjoying the small taste of power that comes from being able to delegate (relevant!) tasks to our new team members from time to time! More seriously, the prospect of supervising their thesis projects (not a stress-free task in itself) during COVID-19 and all of the difficulties it brought was certainly a little intimidating or challenging at times, as there’s no doubt that sometimes it’s easier to check everyone’s on the same page when we’re all in the same room. But the tools we’ve used made it feel much less of an impossible task and helped to make the process much smoother than it might otherwise have been.

As a result of hosting these open science undergraduate projects, riboviz has improved significantly across a whole range of measures, and our whole team has really enjoyed the experience of having students working with us. With each cohort I’ve been involved with, I’m more convinced that open science projects and open software tools make a powerful combination that benefit students and research groups equally.

## Day-to-day supervisor’s top takeaways

* Familiarise students with tools early & use them often: investing the time early on gives plenty of time for them (and you) to realise why these are time-savers (and often sanity-savers) later on in a project!
* Ensure that you’re visibly and frequently asking questions of colleagues, seeking help and sharing ideas: this makes it far less intimidating for your students to do the same with you, and will help avoid some misunderstandings, miscommunications and mistakes. Issue tickets and slack are great for this!
* Plan something to work on or read & discuss when the servers / vital equipment is down: this will happen, and possibly more than once. When our compute cluster went down early on, we ‘lost’ a week of project time, because we hadn’t planned any alternative project tasks and lots of subsequent tasks relied on the cluster. In this case, the students were able to work on their programming skills and read papers, but in future I’d plan for this eventuality!
* Identifying alternative project tasks is also great for when something isn’t working, but you don’t have any more supervision time to help troubleshoot until later in the week. This stops students getting stuck on one thing and not being able to progress at all without help.
* Remember that a research group environment is quite a jump from what students might be used to - project meetings, lab talk presentations and zoom interactions between PIs and research staff are probably a totally different world to lectures and tutorials. Include students in group and project meetings early on to help them acclimatise, and remember that it might be quite a culture shock to start with!

# The PI perspective

The ideal student research project benefits everyone involved: students learn and make their grades, while the research project moves forward and the research group learns from the students. Open science projects can approach this ideal, by treating students as collaborators throughout their projects, and having a clear structure set up for them to engage and contribute. The key opportunity is that doing open science takes time, and it is hard to find others willing to climb the “mountain of engagement” from learning about a project to participating, collaborating, and leading. Students whose research project intersects with a larger open science project have the time and incentive to engage.

Open-source software, like riboviz, needs users! Users test features, read documentation, find bugs, and engaged users suggest improvements. Project students are highly motivated users because they volunteered to do the project, they want to learn and to earn good grades. The mutual benefits are clearest when students, as users, test features that are both accessible to them and important to the project. For example, bioinformatics workflows have problems when adapted to new datasets with different quirks, such as different genome annotations differing in features (e.g. introns, UTRs) and format (e.g. attribution fields in a gff file). For riboviz, adapting to new datasets and organisms was a sticking point, and making this easier was important to the project’s success, so we learned to set running new datasets as an early objective for student projects.

## The importance of a structured pathway forward

We prepare undergraduates who are bioinformatics novices for expert data wrangling tasks by having a pathway from novice to expert, inspired by The Carpentries approach to data skills training. For riboviz, we’ve learned that students can start by installing the software and running a small built-in “vignette” dataset, overcoming installation problems. Then they can run on an existing full-size dataset, overcoming problems with data size and navigating directory structures. Next they work to analyse a dataset from an existing annotation, learning about common bioinformatics tasks such as adapter removal in sequencing analysis. Then they are ready to tackle adapting a dataset with a new genome annotation, an infuriatingly unpredictable process in real-world bioinformatics. Other projects can similarly develop expertise through a series of authentic research tasks of increasing complexity.

The documentation of the larger project is critical to students being able to get started, and students can also help to improve documentation. We have learned that this takes care and attention. Whatever state the existing documentation is in, students being confused about or stuck on something can indicate a need for better documentation. If students don’t find the relevant documentation, do you need a better contents page (that they can draft)? If students don’t understand an output file, do you need a better description (that they can draft)? If students don’t know how to contribute code, do you need a better developer guide (that they can draft)? Writing documentation may not “feel like research”, so you may need to explain its value so that students are motivated to contribute. Good documentation is essential to help researchers use and think clearly about using software, and the same is true of written experimental protocols. Writing documentation is excellent preparation for writing up a project report, especially the methods section, so benefits the student.

## Overcoming the “intimidation threshold” to working open

Working open means routinely sharing work, which can be difficult for students who may feel intimidated or anxious about being judged by others. These feelings do not reflect reality, as experienced bioinformaticians know that no-one else cares how you feel about your code, or even that you are the person who wrote it. Colleagues and users mostly care if the code works, is readable, and has documentation that explains what it does. Open science practices of collaboratively reviewing and improving work, such as single-function chunks of code and sections of documentation, can demonstrate this important life lesson. Students can learn to see their work how other people see it, in small steps, ultimately producing better quality work and lowering stress levels.

Sharing work can be made less threatening with clear expectations, structure, and role modeling. Be clear that you expect students to ask for help, for example that if they have been stuck on something for more than 2 hours then they should ask for help. Have a structure about how and where to ask for help, for example a dedicated “project help” Slack channel or the GitHub issue ticket for the particular task. Crucially, the expectations and structure must be set at the beginning of the project and their use modeled throughout. Be prepared to repeat yourself: in riboviz, we often ask “did you update the issue ticket?” When the conduct of the project is open, the whole team shares requests for help and reports on progress, and students experience other team members including the PI role-model asking and sharing. Openly discussing progress can lower the “intimidation threshold” for sharing technical contributions like code and data analysis. Emma and Sophie’s project start coincided with a week-long hackathon with our transatlantic collaborators, with daily meetings where we discussed progress and commented on each other’s code, which was an intense way to get started. We will do that again, because the very beginning of a project is the best time to demonstrate how open science works. Still, project students may be too intimidated by the PI’s status to ask openly, and the PI may be too busy to help promptly, so it’s important to have direct daily contact with another team member to triage problems and reassure the student. This team member needs to ask, when appropriate, “did you update the issue ticket?”

Explaining how credit is assigned is also crucial. Usually, the project report or write-up determines most of the student’s grade. The grade then doesn’t depend directly on any code they contribute or results that they obtain. Open science projects resemble other group projects, such as the IGEM-affiliated MSc projects run here in Edinburgh, where work is collaborative while reports are written and graded individually. By contrast, students’ work during the research project can focus on learning, contributing, and building towards results. Feedback in an open science project provides formative assessment of student progress in an authentic research setting.

Authentic contributions within the project can lead to authentic credit, including co-authorship on papers and code contributions on GitHub that are visible to potential employers. Our riboviz 2 preprint includes as co-authors 5 undergraduate or MSc students from Edinburgh, as well as 2 more from US collaborators. These co-authorships have been earned by adding new datasets, fixing bugs in code, adding new features, and improving documentation. Students also learn about the process of writing a collaborative paper by attending team manuscript meetings and contributing to the draft.

## Mistakes and benefits

We have made plenty of mistakes. Students have spent weeks stuck on problems that other team members know how to solve, have written large chunks of code that aren’t possible to incorporate into the larger project, and have cobbled together their own (undocumented) data structures. These sorts of mistakes aren’t unique to open science projects. Students doing their first experimental research projects can also get stuck, spend time on activities unimportant to the project, and drag their feet on asking for help. Most trainees need supervision on how to break a project down into small steps, in the new skills involved in executing those small steps, and in documentation and communication within a research group. So, lessons learned here transfer to other kinds of teaching and to supervision of trainees at all career stages. Open science projects such as riboviz emphasise the role of sharing and documentation, because that is an explicit outcome of the work rather than “just” a necessary stage of the pathway to results.

Finally, working on collaborative open science projects is fun. At Edinburgh, we are lucky to have excellent students who are eager to go from programming novices to making substantial contributions to an open-source scientific project, within only a few months. They can write excellent project reports and earn first-class grades. It’s satisfying to see them learn hard skills and to grow in confidence as scientists and programmers. I am amazed at what our students accomplish when treated as colleagues and given structured support to succeed.

## PI’s top takeaways

* Open science in student research projects can benefit everyone involved, as well as the wider research culture of the research group.
* Plan a path from novice to expert for the students, using authentic research tasks.
* Set clear expectations and structure for how to engage with the larger project, and role-model the behaviour you want.