The benefits of an open-science approach in student research projects

Emma MacKenzie, **Sophie Winterbourne**, **Felicity Anderson** and **Edward Wallace**
(University of Edinburgh, Edinburgh, UK)

Open science is a movement to allow scientific information, data and outputs to be more widely accessible and reusable, with the active engagement of all the stakeholders. Open science can also describe openness within a research group where all participants share their data, analysis code, ideas and feedback. These ideas can be applied to all aspects of science, from large research consortia to student 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 undergraduate and masters student research projects, from the perspective of the student, the day-to-day supervisor, and the principal investigator (PI) or research group leader. We argue for the importance of clear expectations, communication, documentation, and of modelling collaborative behaviour. To design a good student project, we recommend planning the project outcomes so that everybody wins, and planning a pathway from novice to expert within the project.

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 is openly available. Riboviz also takes an open-science approach in that all team members can see and contribute to all work in progress, collaborating across research groups in the UK and USA. We host riboviz on GitHub, a website for software, code, documentation and collaboration. Files can be downloaded, edited and returned to GitHub using Git, which passes information between a local machine and GitHub while allowing for version control. Discussions on errors, new features and other specific issues for riboviz take place in small forums on GitHub called 'issue tickets'.

The student perspective (Emma MacKenzie and Sophie Winterbourne)

There are many kinds of student research projects

Students on our undergraduate biotechnology honours course had access to a wide variety of projects. Potential projects had varying levels of openness. A traditional project might involve designing and attempting an experiment (in the absence of a pandemic) or a computational analysis or scouring literature with the goal of answering a research question. The immediate outputs of the project would usually be talks to the hosting lab and the degree program, the written report and any results and reagents generated by the student. Any impact beyond the hosting lab, for example contributions leading to a publication, might not be visible until long after the student has left.

Some other available projects involved working with trade secrets or compounds with therapeutic potential, with appropriate legal restrictions on confidentiality. The idea of navigating an unfamiliar set of rules for protecting intellectual property caused us a lot of anxiety, which we avoided by working on an open-science project.

What our project was like

Our project involved jumping into a team working on the riboviz data analysis pipeline. We contributed to the overall development of riboviz in the form of adding new datasets, testing the pipeline and commenting on its usability. Our work is documented on GitHub and is publicly accessible, providing proof of our contributions to the overarching project, in addition to our own reports. Sharing our contributions instantly and openly provides concrete evidence of our capabilities, which is accessible to potential employers.

Due to the COVID-19 pandemic, but also as the projects were completely computer based, we did not have any in-person contact hours. However, we were able to attend regular video meetings with our supervisors and with collaborators, and we posted frequent progress updates on the lab Slack message board. This contact was supplemented by issue ticket updates and commits as we

sent work from our local machines to GitHub, which our supervisors were able to see and comment on. We gained practical collaboration and computing skills; however, we missed out on improving our wet-lab or bench skills.

What it felt like to work in open science

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-science nature of the project allowed for wider collaboration with other members of the lab and cooperating with research groups across the world. It provided potential for learning a variety of skills as we were free to aid in different sub-projects, in addition to our own.

Completing an open-science project was a very exciting experience, as we were contributing to a larger project rather than completing an isolated experiment or literature review. We initially felt intimidated by the idea that everything we did could be seen by others and had the potential to help or hinder them. The project team supported us by emphasizing that sharing our day-to-day work was both normal and helpful. As the project continued, we learned to openly collaborate and to learn from the work of others in the group, and we also enjoyed bouncing ideas off peers working on other open projects. 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.

Benefits and challenges of working open

The open-science nature of the project simplified communicating about it with other members of the lab when we were all working virtually. Working with experts from around the world at such an early stage of our careers was valuable experience, and their support helped to overcome our initial intimidation. When we first started the project, some of the documentation was confusing, as we had never worked with the software before, or with 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 during our first official week with the lab when a 'hackathon' took place. This involved all individuals working on riboviz focusing solely on its development for that week, across three labs in three time zones. Having never worked in a lab before, let alone in a bioinformatics one, this was rather overwhelming. Every day 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 1 week of 'trial by fire' made everything else seem more manageable in comparison. We were able to see how a large group could operate efficiently using GitHub, which was 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 valuable skills for further study and working with open science. We learned to prioritize tasks based on their importance within the group rather than just for our individual needs. We also gained better debugging skills, knowing where to look in documentation and on the internet 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 Slack conversations and issue tickets on GitHub. We soon learned the importance of frequently communicating progress and obstacles in a precise and informative manner, both to avoid confusion and to get help more quickly, especially as all communication occurred virtually. For example, a prolonged delay was caused by a failed dataset run which could have been avoided if the issue ticket had been updated in sufficient detail. This personal development is clear when 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 we learned the many tools needed for coding and collaboration

To complete our projects we needed to be able to confidently use the command line, Git, GitHub, the university computing cluster 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, 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 an underpowered personal laptop, so we needed to learn how to remotely access the university's computing cluster. Using resources provided by the university, we learned how to navigate the cluster, submit jobs and run interactive sessions.

The most important step of the project was learning how to use the data analysis pipeline 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 visualization 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.

Coping with COVID-19 disruption

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 students selecting their projects

If you are an undergraduate or masters student considering what direction to take your studies, try to choose a project in a field you are considering entering into, even if you have no prior experience. Both of us were, coincidentally, considering completing an MSc in bioinformatics and therefore picked a bioinformatics-based project. We each found the experience of stepping outside of our comfort zone and into an unfamiliar field daunting but very valuable. The project provided a positive experience and a fantastic learning curve which confirmed our initial interest in the field and allowed us to gain confidence and competence within coding and software.

Students' top takeaways

- Being able to access other researchers' work can speed up projects. Pre-existing code could be used as a starting point to learn which functions and packages could be used for a specific purpose.
- It is easier to learn when resources are available to refer back to.
- Documenting your work helps you, aids others and helps them help you.
- Open science is like a group project, where the other members of your group are researchers and the world can see your work.

The day-to-day supervisor perspective (Felicity Anderson)

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: student open-science projects!

Overlapping goals

Supervising 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. It has also made it possible to align the supervision with relevant work of my own, meaning that there's not as much context-switching required when going between supervision tasks and my standard workload. This means that I have been able to improve my own knowledge while helping support the students to learn the concepts, technical skills and organizational 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 student 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 and 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 students previously where we've introduced concepts such as version control much later in the project timeline, with less success. 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. We could also 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, it proved a really powerful way to supervise students taking their first steps beyond using R within student courses and learning to become software developers. We could quickly and easily 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.

Using video calls 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). With screen-sharing while live-coding, 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 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 prioritize these different steps to achieve larger goals, which is another key skill these projects help students to 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 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 supervision meetings by video call. These weekly meetings reviewed overall progress and gave more opportunities to explore any issues in more depth and share additional resources or demonstrate coding techniques. For example, I led demos of how to resolve real-world git merge conflicts and how to debug R code using the command-line.

While the weekly 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 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 overall lab goals faster.

Both Sophie and Emma have mentioned that introducing them to the project meetings and lab meetings early on was 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. 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 also helped integrate them within the wider group. As a result, 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 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 and 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 during COVID-19, and all of the difficulties it brought with it, was challenging at times. 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. However, the tools we 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 student 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

- Familiarize students with tools early and use them often. Investing time early on gives plenty of time for them (and you) to realize why these are time-savers (and often sanity-savers) later on in a project!
- Include students in group and project meetings early on to help them acclimatize to the research group environment, and remember that it might be quite a culture shock!
- Visibly and frequently ask questions, seek help and share ideas. This makes it less intimidating for students to do the same and will help avoid misunderstandings, miscommunications and mistakes.
- Plan something to work on or read and discuss when the servers/vital equipment is down or something isn't working: this will happen, and possibly more than once! Have alternative tasks in mind.

The PI perspective (Edward Wallace)

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 (Figure 1a). 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 and 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

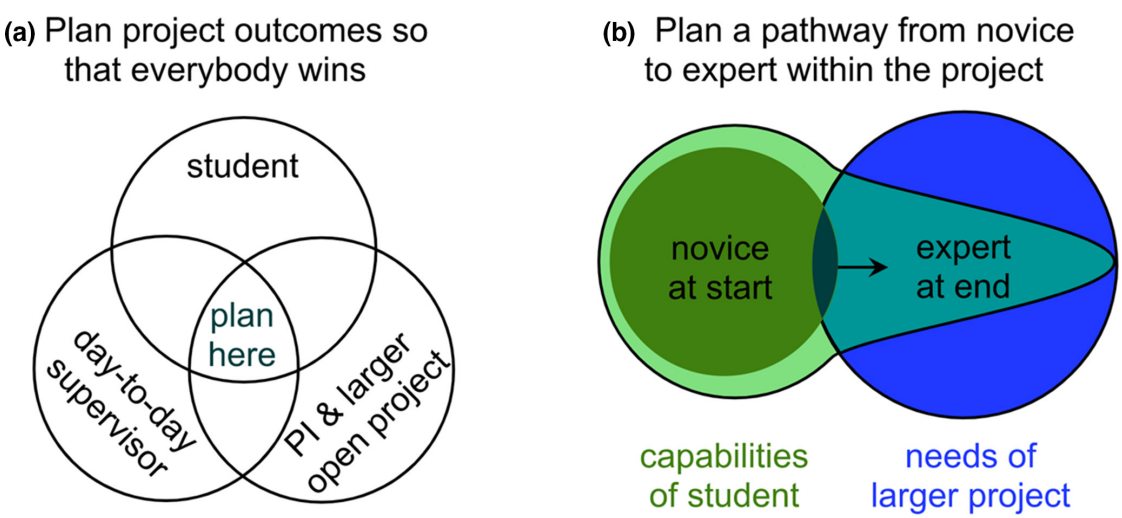

Figure 1. How to design a good student project. a, Plan project outcomes so that everybody wins. b, Plan a pathway from novice to expert within the project.

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 students who are bioinformatics novices for expert data-wrangling tasks by having a pathway from novice to expert (Figure 1b), 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 most people care about the code, not about its author(s). 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 the same way as other people see it, in small steps. This more detached perspective ultimately produces better quality work and lower stress levels.

Sharing work is made easier by clear expectations, structure and role modelling. Be clear that you expect students to ask for help, e.g., 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, e.g., 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 modelled throughout. Be prepared to repeat yourself: in riboviz, we often encourage open communication by asking "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. Despite our best efforts, project students may still 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"?

Explain the benefits

It's important to explain how credit is assigned and the different benefits of the student's project final grade and of their contributions to a larger project. 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 six undergraduate or MSc students from Edinburgh, as well as two 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, lessons learned, and rewards

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. I, as supervisor, have asked students to do tasks that make sense to me, yet which are unclear or unachievable for novices, or missing from our documentation. Offering related projects over subsequent years helped us to learn from our previous mistakes and to progress to new ones instead.

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. Lessons learned about project planning and communication transfer to other kinds of teaching and to supervision of trainees at all career stages. Open-science projects such as riboviz emphasize 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 rewarding 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. Project students have both time and motivation to contribute and can get credit for their contributions to code, documentation, and publications.
- Plan a path from novice to expert for the students, using authentic research tasks. Structure this path so the next task is achievable, to build confidence and skills.
- Overcome the 'intimidation threshold' for collaborative work by having a structure that makes it easy to share work, setting expectations that sharing is non-optional and modelling the collaborative behaviour that you want.
- Offer related projects over subsequent years to build on previous work and experience. ■

Acknowledgments

This work was supported by the Biotechnology and Biological Sciences Research Council (BB/S018506/1); the Wellcome Trust (208779/Z/17/Z); and through a Wellcome-University of Edinburgh ISSF3 award. We thank Christopher Jacoby, Kostas Kavoussanakis and Liana Lareau for comments on the manuscript.

Further Reading

- UNESCO on open science - https://en.unesco.org/science-sustainable-future/open-science
- Markowetz, F. (2015) Five selfish reasons to work reproducibly. *Genome Biol.* **16**, 274. DOI: 10.1186/s13059-015-0850-7
- Cope, A., Anderson, F., Favate, J. et al. (2021) riboviz 2: A flexible and robust ribosome profiling data analysis and visualization workflow. *bioRxiv* DOI: 10.1101/2021.05.14.443910
- Wallace, E., Anderson, F., Kavoussanakis, K. et al. (2021) riboviz: software for analysis and visualization of ribosome profiling datasets. *figshare*. Software. DOI: 10.6084/m9.figshare.12624200
- For iGEM MSc projects - https://igem.org/Competition
- The Carpentries, organisation and materials teaching coding skills to researchers - https://carpentries.org/
- For "Mountain of Engagement" - Sharan, M., Ye, H., Yehudi, Y. et al. (2021) OLS-3 Cohort Talks and Transcripts (v1.0). Zenodo. DOI: 10.5281/zenodo.5071349
- Ten arguments against Open Science that you can win - https://www.software.ac.uk/blog/2020-12-17-ten-arguments-against-open-science-you-can-win
- Introduction to git and GitHub - https://guides.github.com/introduction/git-handbook/

Emma MacKenzie is currently studying for an MSc in bioinformatics at the University of Edinburgh. She received a first-class honours degree in biological sciences (biotechnology) from the University of Edinburgh, after spending her honours project using ribosome profiling data to analyse translation in Schizosaccharomyces pombe *using data analysis pipeline 'riboviz'. She worked as a research assistant in bioinformatics with the Wallace Lab over the summer between her honours and masters degrees. She plans on using her bioinformatics skills to study human health and genetics in the future.*

Sophie Winterbourne is currently an MSc Bioinformatics student at the University of Edinburgh. She received a first-class honours degree in biological sciences (biotechnology) from the University of Edinburgh. Her honours project focused on studying inhibitory features of translation in Saccharomyces cerevisiae *using the open source software package 'riboviz'. Sophie worked in the Wallace Lab as a research assistant in bioinformatics where she helped improve documentation and provided new functionality. She would like to continue her studies within the field of translational biology.*

Felicity 'Flic' Anderson is a research assistant in bioinformatics in the Wallace Lab. Flic is a developer for the 'riboviz' open source software package for processing and analysis of ribosome profiling data, working to make the codebase more robust and sustainable. She is a member of Edinburgh Carpentries and a certified Carpentries instructor in foundational coding and data science skills. Flic is also involved in outreach through the 4273pi project, which designs and delivers bioinformatics training in Scottish schools. Flic begins a PhD in the use of software engineering techniques in research software projects in January 2022 with EPCC. @Flic_Anderson on Twitter.

Edward Wallace is a Sir Henry Dale Fellow (group leader) in the Institute of Cell Biology at the University of Edinburgh, funded by Wellcome and The Royal Society. The group studies how organisms respond to their environment, focusing on molecular mechanisms used by fungi. The riboviz project is funded by BBSRC and NSF-BIO, to develop better software tools for understanding protein synthesis and its regulation. Alongside his research, Dr Wallace is an open science advocate and teaches data literacy to scientists, working with The Carpentries and Edinburgh Carpentries. @ewjwallace on Twitter. Image credit: Tom Humberstone.