Building a local community of practice in scientific programming for Life Scientists

Sarah L. R. Stevens[1] Mateusz Kuzak[2], Carlos Martinez[3], Aurelia Moser[4], Petra Bleeker[5] and Marc Galland[5*]

1 Department of Bacteriology, University of Wisconsin–Madison, Microbial Sciences Building, 1550 Linden Drive, Madison, WI 53706, USA.
2 Dutch Techcentre for Life Sciences (foundation office), Catharijnesingel 54, 3511 GC Utrecht, Netherlands.
3 Netherlands eScience Center, Science Park 140, Amsterdam, 1098 XG, Netherlands.
4 Mozilla Foundation, 331 East Evelyn Avenue, Mountain View, CA 94041, USA.
5 Department of Plant Physiology, Swammerdam Institute for Life Sciences, University of Amsterdam, 1098 XH Amsterdam, Netherlands

* Corresponding author: m.galland@uva.nl

Abstract

For most experimental biologists, handling the avalanche of data generated is similar to learning how to drive on your own. Although that might be doable, it is preferable and safer to learn programming and good practices together and preferably from experienced scientists. One way to achieve this is to build local communities of practice in scientific programming by bringing together life scientists that perform code-intensive research to spread know-how and good practices. The community will make sure that, for a given researcher working for an academic institution, their capacity to conduct data-intensive research will not be arbitrarily reliant on the presence of well-trained bioinformaticians in their vicinity.

In this paper, we propose a three-step field guide for building a local community of practice in scientific programming for life scientists. The first step is organizing a Carpentries programming workshop designed for researchers new to computational biology. However, while such workshops provide an immediate solution for learning, more regular long-term assistance is also needed. Researchers need persisting, local support to continue learning and solving programming issues that may hamper their research progress. The solution we describe here is to implement a Mozilla Study Group where researchers can meet-up and help each other in a "safe-learning atmosphere". We describe two real-life examples of building local communities of practice: one in the Netherlands at the Amsterdam Science Park and one in the United States at the University of Wisconsin-Madison. In order to help future researchers to build their own community of practice in scientific programming we discuss challenges and implemented solutions.

We believe that our local communities of practice will prove useful for other life scientists who want to set up similar support groups for researchers involved in scientific programming and data science.

Author summary

In this paper, we describe why and how to build a community of practice for life scientists that start to make more use of computers and programming in their research. A community of practice is a small group of scientists that meet regularly to help each other and promote good practices in scientific computing. While most life scientists are well-trained in the laboratory to conduct experiments, good practices with (big) datasets and their analysis are often missing. This paper proposes a field-guide on how to build such a community of practice at a local academic institution. Based on two real-life examples, challenges and related recommendations are provided. We believe that the current data deluge that life scientists will increasingly face can benefit from the implementation of these small communities. Good practices spread among experimental scientists will foster open, transparent, sound scientific results beneficial to society.

Introduction

Life Sciences is becoming a data-driven field

In the last ten years, since the advent of the first next-generation sequencing (NGS) technologies, DNA and RNA sequencing costs have plunged to levels that make genome sequencing an affordable reality for every life scientist [1]. The affordability and technological advances in NGS have profound consequences for ecology and microbial ecology [2]. In addition, application of mass spectrometry to metabolomics has boomed over the last decade for both simple and complex metabolites [3]. Finally, imaging coupled with artificial intelligence methods is also revolutionizing the plant sciences [4] and medical fields [5]. Consequently, better programming skills are becoming essential in life sciences. Yet, training often lags behind. Improved coding capacities are especially needed for researchers working in genomics, bioinformatics, data science (e.g. Machine Learning) and statistical analyses. As most undergraduate education in biology does not comprise any bioinformatic courses, new PhD students are often devoid of any background in genomic analyses whereas they are well trained for wet lab matters. In addition, at international institutions such as universities, the staff originates from a variety of backgrounds and their programming and data analysis levels are highly variable. Overall, the vast majority of wet lab biologists need tailor-made, practical training to learn scientific programming and data analysis. Consequently, rather than needing additional computing resources or sequencing, more and more biologists are faced with a training bottleneck [6–9, 13, 14]. Current efforts in bioinformatics and data science training for life scientists exist in several formats. In Europe, ELIXIR coordinates bioinformatic resources for researchers. In addition to that, ELIXIR nodes have started a "train-the-trainer" program where sixty instructors (May 2018) have been educated [10]. Several foundations such as the Carpentries Foundation provide periodic training workshops to researchers to teach basic and robust software development skills [11]. The two organizations, ELIXIR and the Carpentries Foundation, have joined efforts very recently across Europe [12]. Organisations such as GOBLET (the Global Organisation for Bioinformatics Learning, Education and Training) or the Bioinformatics Training Network are also offering extensive online training material [13, 14].

Altogether, all life scientists are now in need of training in scientific programming to support research involving bioinformatics, data science and statistics. This training should help researchers to stay up to date in scientific programming which biological research increasingly relies upon.

Good practices in scientific programming are needed to increase research reproducibility

Recent papers have shown that modern biology is facing issues regarding reproducibility, termed the "reproducibility crisis" [15]. While evidence suggests this might not be as bad as it sounds [16], there is clearly a need for more reproducible practices regarding code. Indeed, even in a code-intensive field such as artificial intelligence, it has been shown that out of 400 algorithms presented at two conferences, only 6% had published their corresponding code [17]. Thus, most code used in research remains a "black box" [18]. Yet, programming has become an important tool in biological research, such that as many as 92% of academic researchers stated that they use research software in their daily work [19]. In addition, 56% of researchers develop their own software, of which 21% indicate that they have never received any form of training in software development [19]. While the use of laboratory notebooks for documentation of lab work is widely taught as part of undergraduate education in biology, such practices are not

widely emphasized for coding. When learning to code on their own, often researchers only learn the skills needed to write code and analyze their datasets and in doing so they do not learn good code documentation practices. With computational work, it is also easy to fall prey to the 'I can easily rerun this' fallacy. However good documentation and data stewardship are vital for anyone, including the original researcher, to be able to reuse and understand code or data analyses at a later date. Part of the solution lies in dedicated training to researchers promoting good practices in programming [20]. In addition to proper code documentation, better practices in data management are also needed. Poorly accessible and poorly documented datasets have also contributed to the reproducibility crisis. Therefore, better data stewardship is necessary. One of the recent initiatives to foster improved data management is the FAIR (Findable, Accessible, Interoperable and Reusable) principles initiative which provides guidelines to boost reproducibility and reuse of datasets [21]. **Therefore, the long term goal of any programming scientist should be to steward good practices in code-intensive research by promoting open science, reproducible research and sustainable software development.**

Part of the solution: building a local community of practice

We want to improve the efficiency of training workshops in bioinformatics, data analysis and other code-related topics by supplementing them with a support structure for long-term learning. While one-time training courses are regularly offered throughout the year, in most cases, researchers will need a more regular support to debug their code, make progress in their research, and adopt good practices. How do we provide continuous and local support for life scientists?

Fueled by Etienne Wenger's idea that learning is usually a social activity [22, 23], we propose to build a local community of practice in scientific programming for life scientists. This community fulfills the three requirements of Wenger's definition: it has a specific domain i.e. bioinformatics and data science, its members engage in common activities e.g. training events, and they are practitioners i.e. researchers currently engaged in research that involves scientific programming. It also complies with a more recently revised definition of a community of practice such that production and exchange of knowledge between members and a sense of shared identity between participants have also been acknowledged as crucial [24]. Community building and organization is a field in itself that has been considerably reviewed [24–29]. Such a local community of practice would require a few motivated leaders and would provide a safe environment where participants can experiment with their new knowledge [26]. This local community of practice in scientific programming would act as a "one-stop shop" to get technical assistance, follow in one-hour lessons, hear about the next training workshop, and experience live software demos. As stated by Wenger and Snyder [30], communities of practice "help to solve problems quickly", "transfer best practices" and "develop professional skills". While short-term immediate issues ("help me now to debug my code") can be solved, the community also has the capacity to steward solutions for long-term data-related problems ("how do I comply with the FAIR guidelines?"). A community of practice that brings different life scientists together can also be part of the solution to reproducibility issues by bringing together software scientists and wet lab biologists. Communities of practice can also foster the adoption of good practices as exemplified in health sciences [31]. By co-working with their peers, scientists are probably more likely to compare their methods and embrace best practices. Finally, this local community will be able to network with other coding researchers e.g. those located in the same regional student group [32].

This paper will explicitly describe why and how to build a community of practice in scientific programming. We propose a model of how to build such a community that we exemplify in two case studies. Finally, we discuss the challenges and possible solutions

that we encountered when building these communities in the hope that others will find these useful to set-up their own community of practice. **Overall, we believe that building these local communities of practice in scientific programming will support and speed-up scientific research, spread good practices and, ultimately, help to tackle the data deluge in the life sciences.**

Why do we need to build up local community of practice in scientific programming?

Local communities of practice are meant to solve several issues that any wet lab life scientist would face when trying to analyze their data using the computer.

Isolation

It is becoming increasingly common for wet lab biologists to be asked by their supervisors to analyse a set of pre-existing data. For instance, a PhD student receives transcriptomic mRNA-Seq data and the supervisor asks her to retrieve gene expression levels and extract differentially expressed genes. While this problem might sound trivial for a well-trained bioinformatician, the PhD student first faces a so-called "isolation issue". The PhD student's peers and fellow lab mates are also wet lab scientists with little to no coding experience. Therefore the PhD student lacks access to an experienced bioinformatician from whom they can learn. This can lead to a sentiment of isolation deleterious to their work.

Self-learning and adoption of bad practices

In such a scenario, most researchers tend to invent a custom solution from scratch. This can lead to the adoption of bad practices such as reinventing the wheel, lack of version control and irreproducible results. While some compiled easy-to-use software such as samtools [33] can help to get started, typically researchers need to build their own collection of tools and scripts. Version control is often overlooked by researchers as non-critical and can lead to cryptic file nomenclature and irreproducible results. We believe that version control with git[1] and github[2] for instance should be considered a mandatory, good practice just like accurate pipetting in the molecular lab.

Apprehension

In addition to feeling isolated, a researcher who is starting to code may be afraid of the breadth of knowledge that needs to be grasped before achieving anything. Indeed, bioinformatics is a fast-evolving field of research and staying up-to-date can feel like an overwhelming task, even for an experienced bioinformatician. Eventually, this fear may lead to "impostor syndrome", where the researcher feels like he will be exposed as a fraud and someone more competent will unveil his lack of knowledge of coding and bioinformatics. This also presents a challenge for continued learning since the researcher is then afraid to ask for help when available. Indeed, "impostor syndrome" is likely to affect those that embrace a new challenge such as learning to program in order to analyze data.

[1] `https://git-scm.com/`
[2] `https://github.com/`

The issue of how to get started

Learning to code in a research team is akin to an apprenticeship. The 'apprentice' will benefit from the experience and knowledge of more experienced team members. For instance, a researcher working on mRNA-Seq for several years will be able to demonstrate the use of basic QC tools, short-read aligners, differential gene expression calls, etc. Yet, many research teams do not have an experienced bioinformatician on staff. Therefore, the wet lab researcher is left wondering "what do I need to learn?". Having experienced individuals around is therefore necessary for novices to learn. Even in the best case where an expert bioinformatician is available, it may be problematic for beginners to get all their knowledge in one field from one person. Instead, we propose that building a community of experienced bioinformaticians will spread good practices, such as using version control. Building a local study group is then a solution to connect bioinformatics novices and experts. Ideally, a novice should make progress toward increased skill levels, as illustrated in Fig 1 [34]. Here, it is important to note that while champions often lead a local community of practice, it is also possible for beginners and competent practitioners to set up a session and invite experts to discuss a particular topic. Thus, rather than a rigid hierarchical structure, the local community is meant to be horizontal and welcoming.

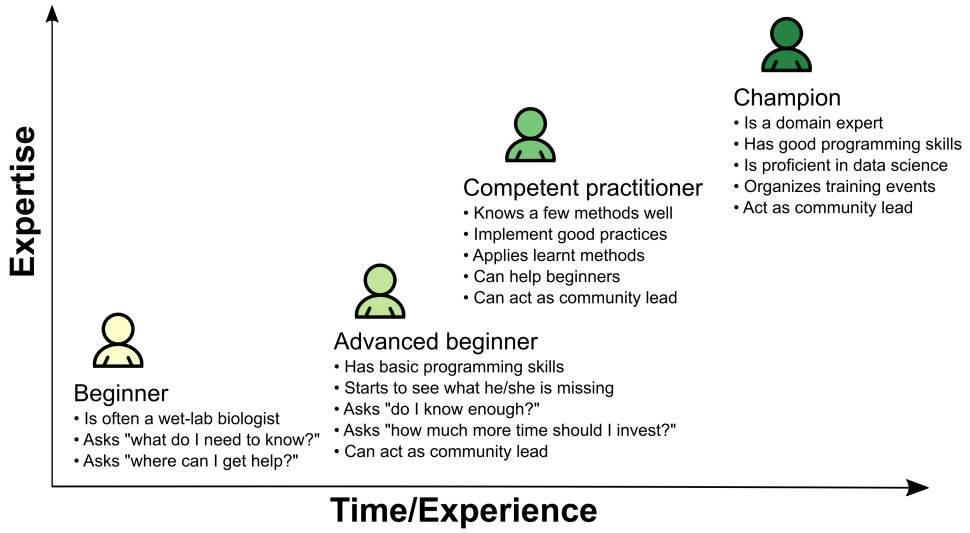

Fig 1. Different learning stages in scientific programming. This figure displays the different stages of learning encountered by experimental biologists.

How do we build local communities in scientific programming? A model inspired by experience

Here, we propose a three-stage working model (Fig 2) to create a local community of practice in scientific programming composed of life scientists at any given institution without any prior community structure. **In stage 1, we form the "primer" of a**

local community of practice by first running basic programming workshops organized by local community leads (defined as "champions") and then coupling them to formation of a study group. Champions do not necessarily have to be experts themselves. In our experience, Carpentries workshops work well since they provide training aimed at researchers and possess a long history of teaching programming to scientists [11, 20]. Yet, other formats for programming workshops would also work well in practice and running Carpentries workshops is not required. These programming

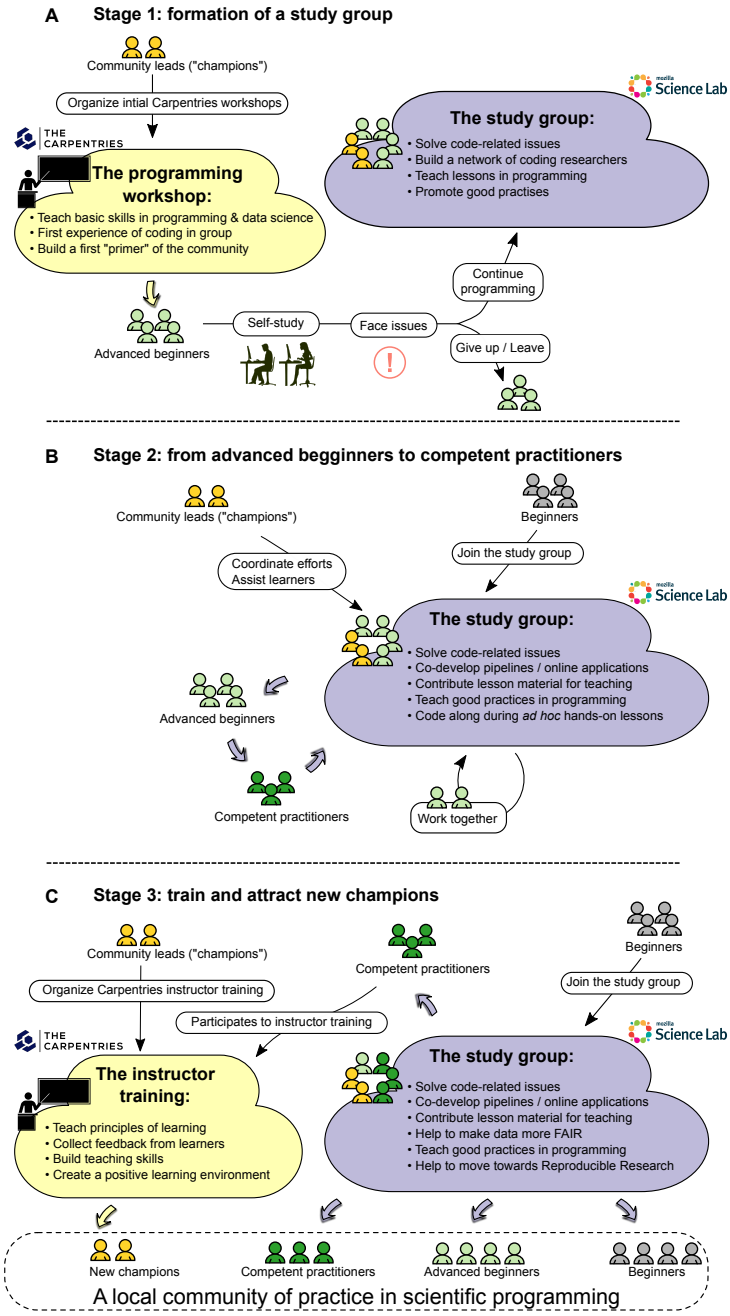

Fig 2. (Legend on next page)

Fig 2. (Previous page.)A three-step model to build a local community of practice in scientific programming. This figure describes the three steps to build a community of practice for life scientists that use programming in their research.
(A) First, a few scientists acting as community leads set up one or more Carpentries workshops to impart basic programming and data science skills to wet lab life scientists. After completion of the workshop, the novices will often face programming issues that need to be solved frequently. Furthermore, they need to continue to learn new programming skills. Therefore, a local study group such as a Mozilla Study Group can be formed by community leads ("champions") and "advanced beginners" to foster a regular meeting place for solving programming issues together and discovering new tools.
(B) By attending a regularly scheduled study group, advanced beginners start to work together and make progress. Together with additional guidance and *ad hoc* assistance by community leads, some advanced beginners become "competent practitioners".
(C) Finally, as some "competent practitioners" attend the Carpentries' instructor training sessions, new community leads ("champions") are trained. In addition, the local study group keeps attracting new beginners. Study group sessions together with optional Carpentries events help to educate community members and help them to become "advanced beginners" and "competent practitioners". As "competent practitioners" become community "champions", this closes the loop and help the local community of practice become fully mature with all categories of learners present.

workshops serve as a starting point for both learning and gathering researchers together in one room. During the workshop, people are actively paired with each other and they are invited to learn about each other. Often beginners and bioinformaticians who might have never met despite working at the same institution will connect and engage afterwards. When absolute beginners join these workshops, they become "advanced beginners" since they have some notion of the command-line, Python and/or R programming, version control, etc. Together with community "champions", these "advanced beginners" can "seed" a local community of practice (Fig 2). Once formed the local community needs to meet regularly to continue practicing the skills they learned at these programming workshops. During their daily work, "advanced beginners" often lack the support needed to face programming issues that they may encounter frequently. Therefore, a local co-working group should be set-up with a regular meeting schedule. The Mozilla Study Group provides a well documented Handbook for starting and maintaining study groups[3] and includes a template to create a Study Group website. Mozilla Study Groups are informal and regular meetings of researchers that want to improve their coding practice, discover new software and tools applicable to their research and work together to solve code-related issues. There are nearly 100 Mozilla Study Groups around the world and the Mozilla Foundation[4] facilitates the exchange of experiences between the leaders of these groups. The University of Toronto Coders Study Group[5], which is a mix of work-along and co-working sessions is a good study group example. Therefore, a Mozilla Study Group can be started to form a local group of scientific coders. Again, other forms of a co-working group can be used but we believe that Mozilla Study Groups offers a range of online material and support such that there is no need to "reinvent the wheel".

In stage 2, the study group becomes a regular practice for advanced beginners where they progressively become competent practitioners thanks to mutual help and guidance from champions (Fig 2). This study group also welcomes new novice members as they join the research institution or as they hear about

[3] http://mozillascience.github.io/studyGroupHandbook/
[4] https://www.mozilla.org/en-US/
[5] https://uoftcoders.github.io/studyGroup/

the existence of the group. The community leads will provide guidance, specific lessons, and assistance during hands-on practicals which will nurture the community and raise the global scientific programming level of all local community members. It should be duly noted that absolute and advanced beginners can also lead sessions: leading is not restricted to champions and any motivated individual can become a community lead. Also, champions do not necessarily have to be experts themselves: instead they can invite experts and start the discussion on a particular topic of interest. At the end of this stage, most advanced beginners will likely have become competent practitioners.

In stage 3, a subset of the competent practitioners from the local community will become community leads ("champions", Fig 2). To become champions, advanced beginners or competent practitioners need to increase their teaching and facilitating skills and be able to recognize the skill level of their audience (Fig 1). These skills can be attained by becoming a Carpentries instructor which requires attending an instructor training event: these sessions can be organized by initial community champions since they usually have both the network and know-how to set-up these specific workshops. Once again, it is not mandatory to rely on the Carpentries Foundation organization as long as competent practitioners get a deeper knowledge of teaching techniques where they improve their own skills. However, we now have a good perspective on the long-term experience and success of the Carpentries Foundation with over 500 workshops organized and 16,000 attendees present [11]. Moreover, major bioinformatic programs such as ELIXIR also rely on this model to educate experimental biologists to scientific programming, computation, data management and data science skills [11,12].

In practice, stages 2 and 3 can occur simultaneously because an active study group attracts both experienced computational biologists and other scientific programmers.

Case studies

Hereafter, we describe two case studies of community built by local scientists and aimed at life science researchers.

The Amsterdam Science Park example

In Amsterdam, Mateusz Kuzak, Carlos Martinez and Marc Galland started to build a local community of practice by first organizing a two-day Software Carpentry workshop in October 2016. The goal of the workshop was to teach basic programming skills (shell, version control and Python) to a group of 26 wet lab biologists. This started a dialog about the skills needed by life scientists to help them in their daily work. After a few months, a subset of the workshop attendees made progress but most of them did not continue to program either because (i) they did not need it at the time, (ii) they felt isolated and could not get support from their peers or (iii) they did not make time for practice alongside regular lab work. Thus, a regular meetup group was needed so that researchers with different programming levels could help and support each other. Hence, in March 2017, we started up the Amsterdam Science Park Study Group following the Mozilla Study Group guidelines. These Mozilla Study Groups have been implemented at various academic institutions around the globe[6]. We felt that a local study group was necessary to follow-up on the Carpentries workshop and the goal of building a local community. We quickly decided to stick to the guidelines suggested by the Mozilla Science Lab[7]. Originally, we started with one scientist from the University of Amsterdam

[6] https://science.mozilla.org/programs/studygroups
[7] https://mozillascience.github.io/study-group-orientation/

(Marc Galland) and two engineers in software engineering (Mateusz Kuzak and Carlos Martinez). But after five months, we decided to gather more scientists, taking advantage of the beginning of a new academic year to build up a first community with enough expertise in R and Python programming, bioinformatics and data science as well as in various scientific fields (genomics, statistics, ecology). Most study group members came from two different institutes (the Swammerdam Institute for Life Sciences and the Institute for Biodiversity and Ecosystem Dynamics) which helped the group to be more multidisciplinary. At the same time, a proper website, forked from the Mozilla Study Group[8], was set-up to streamline communication and advertise events.

The University of Wisconsin-Madison example

At the University of Wisconsin-Madison, Sarah Stevens started a community of practice in the fall of 2014. The community theme is centered around Computational Biology, Ecology and Evolution and is called "ComBEE" for short. It was started as a place to help other graduate students to learn scientific coding, such as Python and discuss scientific issues in computational biology, such as metagenomics. The main ComBEE group meets once a month to discuss computational biology in ecology and evolution. Under the ComBEE umbrella, there are also two spin-off study groups, which alternate each week so that attendees can focus on their favorite programming language. Later in ComBEE's development, Sarah transitioned to being a part of the Mozilla Study Group community, taking advantage of the existing resources. Most recently, the current group leaders converted the web page from google sites to using the template put together by the Mozilla Science Lab[9].

Early in the development of ComBEE, the facilitating of the language-specific study groups was delegated on a semester by semester basis: this helped to keep more members involved in the growth and maturation of the local community. One of the early members of ComBEE was a life sciences graduate student who had recently attended a Software Carpentry Workshop and had no other experience doing bioinformatics. He wanted to continue his development and was working on a very computationally intensive project. He has since run the Python Study Group for several semesters and is an exceedingly competent computational biologist. He continued to contribute back to the group through the end of his PhD, lending his expertise and experience to the latest study group discussions. The ComBEE study sroup is now more than three years old and acts as a stable resource center for new graduate students and employees.

Room for improvement: challenges and solutions learned from experience

Community building and organization has been well documented [25–27, 29]. In this section, each point emphasizes a critical component of a successful community of practice, describes it in our local context, explicitly states the challenge we faced and the corresponding implemented solutions.

Gather a core group of motivated individuals

One of the first tasks for setting up a community of practice is to gather a team of motivated individuals that will act as leaders of the community [26, 27]. In our case, our leaders or "champions" are individuals ready to promote good practices in scientific

[8] https://github.com/mozillascience/studyGroup
[9] https://combee-uw-madison.github.io

programming, support beginners during co-working sessions, organize training activities regularly, and act as "control tower" to point to the right expert if not available within the team. While these highly motivated individuals do not necessarily have to be technical experts themselves, these leaders should abide by the values of the community of practice: being a collaborative person, being willing to help beginners, respecting differences, promoting open science, ensuring scientific integrity, embracing diversity, etc. While gathering a "core team" of 7 members in Amsterdam and 2-3 at Madison-Wisconsin was relatively easy, this might be more challenging for smaller institutions. Relying on existing smaller communities is a good way to recruit a core team [26]: think about recruiting the members of an informal R coders group for instance.

Keeping participants coming and engaging into the community

At a given institution, it is very likely that experts will be within reach. For someone who is part of the "core team" of a study group, the challenge is to attract these experts and ensure that they regularly participate in activities in the form of lessons, support in co-working session, and attend organizational meetings. There is not a one solution fits them all but, rather, a combination of practical fixes to retain champions into the community. We found that skill development as stated by others

In Amsterdam, we found out that giving the ability to reach out to a wider diverse audience through one-hour lessons is very effective. Previously, in the Amsterdam community, we have had experts teaching one-hour sessions on genomics, data visualization, or version control with git. While not all of these experts remained in the community, most experts already involved in teaching persist and regularly participate in Study Group sessions or Carpentry workshops. Furthermore, the possibility of becoming a Carpentries Foundation instructor is often also tempting for these types of "teaching champions". Altogether, participation in community events brings them visibility, reputation, and enhances their teaching abilities. In addition to teaching, some members of the Amsterdam community of practice are motivated by the ability to solve basic questions from beginners in a more "high-throughput" way at lessons, workshops, or during co-working sessions. In Madison, we motivate and retain competent practitioners and experts by asking them to help with the organization, usually leading a semester of one of the study groups. This way the semester leader has a little more control over the topics discussed for that semester and can tailor it to their interests for that semester. This takes some weight off of the main leaders' shoulders and gives the semester leader a greater investment in the group's success. These people usually go on to continue to participate heavily in the group long after the semester ends. They spread the word about how useful the group is to new students and continue to participate by coming to meetings and helping answer questions from novices. Last but not least, we found that a strong motivation for experts is to enhance valuable skills for their career such as group management, communication capacities and networking. Being part of a community of practice that helps its members to build valuable scientific and career skills is a strong incentive to keep its participants motivated [26, 27]. Furthermore, the ability to network and start new scientific collaborations is also a valuable asset for community members [29].

How to deal with the ever-ongoing turnover at academic institutions

At academic institutions, the constant turnover of students and temporary staff remains a continual challenge. Keeping the local community ongoing requires a critical mass both for the core team and for the audience. As stated above, assembling a core team of motivated individuals is critical since they will maintain a structure and regularly

invite new members to attend and participate. Usually, the core team individuals have a fixed-term contract of several years which helps to keep them in the community for some time. We also found it helpful to have a sufficient number of staff with permanent positions into the core team and as regular contributors or helpers. As for the audience, the key is advertisement as, often, new people at a given institution are not always aware of the existence of a local support group. In our experience, word to mouth conveyed by inspirational core team members is the most effective way to recruit new attendees for the local community of practice as shown before [27]. It must be said here that the high turnover of students also has its positive sides: a dynamic environment brings in new people both eager to learn and with relevant knowledge to share in the group. We recommend using the turnover of people to your advantage by making an effort to recruit both new members and champions.

Dealing with the impostor syndrome

Creating a safe learning environment is one of the requirement for a thriving community of practice [26]. To encourage beginners to ask questions, request assistance and feel welcomed, we have enforced a Code of Conduct at each of our two local study groups to promote a safe-learning environment following the Mozilla Foundation Code of Conduct [10]. Other templates are available as well such as the one provided by the Carpentries Foundation[11]. This Code of Conduct sets up expectations about the environment so that it can promote a welcoming and supportive atmosphere for all people regardless of background or identity. Some good advice for promoting a supportive environment includes promoting all questions and not acting surprised when a beginner asks a simple question, e.g. "What is the Shell?" should not be answered by "Oh you don't know that?". Also, in-depth technical discussions are strongly discouraged since they tend to alienate novices. Similar environments have been established elsewhere that also provide learners with a safe learning supportive [12].

Community leadership and institutional support

An effort should be made to assign clear and specific roles to administration members of the local community based on their expertise and interest. For instance, some people are experienced or interested in building websites should be given priority when it comes to updates and modifications of the group website. This role-definition will empower members and help to create a mature community [27]. At the University of Wisconsin-Madison, a student has been made responsible for running the Python section of the ComBEE study group while in Amsterdam, a web administrator has been appointed from the regular community member. In Amsterdam, the community leadership, coordination and practical arrangements is mostly done by one person (Marc Galland) while this is more distributed at the Madison ComBEE community.
Another challenge is to secure funding and people support from the local institution [26,27] as this can boost the number of organized *ad hoc* training events, and further support PhD/post-docs/staff involved by freeing their time from other activities (e.g. teaching). In turn, this will enable staff and students to engage further into the community.

Community composition

Another important aspect to consider is the composition of the community. Who are the members of this community? Why are they interested in being part of it? Are

[10] `https://science.mozilla.org/code-of-conduct`
[11] `https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html`
[12] `https://www.recurse.com/about`

they getting what they expected out of the community? It is important to know who is part of your community as well as their reasons for participating to ensure that they will continue to be interested in be a part of the community. In terms of domain, the initial community leads in Amsterdam and Madison were life scientists themselves which benefited the community for both content definition and tailor-made support. In terms of expertise, we have identified the following types of community members (Fig 1), each with their own motivations, to be part of the community.

- Absolute and advanced beginners: these are people with the most basic level of knowledge. For them, the motivation to be part of a community is obvious: they want to learn programming and often need rapid assistance to complete their research. Usually, novices lack the overview of tools and software necessary and sufficient to perform their work.

- Competent practitioners: these are people who already competent (at least to some extent) in a particular bioinformatics/data science skill. They may have started as beginners or they may have joined the community having acquired their knowledge somewhere else. For them, contributing to the community is a good way to reinforce their set of talents. Often, competent practitioners make excellent teachers, as they are able to easily relate to the beginner state of mind. In turn, this increases their learning and teaching skills, which are key for a successful community of practice.

- Experts: these are people with the highest experience level on a particular skill in the community. Experts usually reinforce their knowledge by 'going back to basics': it is useful for them to understand what are the usual *gotchas for novices*. Also, experts are people who are in a position where they would need to mentor / provide support to novices anyway. Building a local community of practice provides them with an opportunity to help novices in a more structural way instead of helping them individually in an *ad hoc* fashion.

Regarding the ratio of these types of community members, though not a hard rule we recommend one competent practitioner to three beginners as a good ratio because the competent practitioner can solve the problems of the three novices almost simultaneously. Although we have described three type of community members here, we would like to emphasize that these categories are just guidelines and rules. In reality people will not fall neatly into one of these categories, but rather be in an intermediate state between two of these categories.

Practical considerations

First, the community needs a critical mass of people that come to study sessions on a regular (or semi-regular) basis. Based on our experience, at least 10 recurrent community members is a good number. In any given community session, there will be a number of people who are not able to attend, but having a large enough audience ensures there is always enough people in attendance to make the session useful. Second, it is important to send meeting notifications in advance and frequently enough: for most people it becomes easier to attend an event when they know well-in-advance when and where it will take place. For instance, more people will be able to attend sessions if they know that they take place on Tuesdays at 10am, every two weeks. Schedule the meetings well-in-advance and keep a consistent day, time and place to help people remember them. The meeting frequency should be a compromise between researchers' schedules and community maintenance. On the one hand, researchers should not be required to attend too many meetings. On the other hand, regular meet-ups help to keep the community

structured and active. Consequently, we suggest weekly or fortnightly meetings (every two weeks) as a good starting frequency. Finally, to keep an informal and welcoming atmosphere, sessions should take place in a relatively quiet environment with a good Internet connection. Places such as a campus café outside of busy hours or a small conference room can be good places to start.

Conclusion

This paper will hopefully serve as a guide for others and encourage them to start their own local community of practice in scientific programming at their institution. We hope that our field guide inspired model and the lessons learned from our experience will save time and effort for future community leads. Building a community of practice is far from trivial and we, as scientists, are perhaps not the most aware of everything that has been done on the community building and organization subjects [24–28]. Since "progress will not happen by itself" [20], a community of practice in scientific programming will bring many benefits to its members and to their institution: it fosters the development of new skills for its members, breaks down "mental borders" between departments, networks domain experts at a local site and helps to retain knowledge that would otherwise be lost with the departure of temporary staff and students.
The convergence of the "big data" avalanche in biology (particularly in genomics) and new FAIR requirements for data management [21] makes it more and more important to enable wet lab researchers to conduct good scientific programming, efficient data analysis, and proper research data management. **Eventually, these local communities of practice in scientific programming should speed up code-intensive biological analyses, promote open science, research reproducibility and spread good practices among life scientists at a given institution.**

Acknowledgments

We are thankful to the Carpentries Foundation for assistance in workshop organization. We kindly acknowledge the Mozilla Foundation for assistance in starting and maintaining the Amsterdam Science Park and the Computational Biology, Ecology, & Evolution (ComBEE) Study Groups. The local community of the Amsterdam Science Park Study Group not in the author list is fully acknowledged and consist of Dr Emiel van Loon (UvA-IBED), Pietro Marchesi (UvA-SILS), Joeri Jongbloets (UvA-SILS), Dr Like Fokkens (UvA-SILS), Zsofia Koma (UvA-IBED) and Dr Susanne Wilkens (UvA-IBED). We are grateful to Dr Anita Schürch (UMC Utrecht) for training researchers through several Software and Data Carpentry workshops. We would also like to thank the members and leaders of the Computational Biology, Ecology and Evolution (ComBEE) Study Group and the Carpentry community at the University of Wisconsin-Madison.

References

1. Hayden EC. Technology: The $1,000 genome. Nature. 2014 507:294–295
2. Hiraoka S, Yang CC, Iwasaki W. Metagenomics and Bioinformatics in Microbial Ecology: Current Status and Beyond. Microbes Environ. 2016 31(3):204-212.
3. Gowda GA, Djukovic D. Overview of mass spectrometry-based metabolomics: opportunities and challenges. Methods Mol Biol. 2014 1198:3-12.

4. Singh A, Ganapathysubramanian B, Singh AK and Sarkar S. Machine Learning for High-Throughput Stress Phenotyping in Plants. Trends Plant Sci. 2016 21(2):110-24. doi: 10.1016/j.tplants.2015.10.015

5. Poldrack RA and Gorgolewski KJ. Making big data open: data sharing in neuroimaging. Nat Neurosci. 2014 17(11):1510-7. doi: 10.1038/nn.3818

6. Batut B, Hiltemann S, Bagnacani A, Baker D, Bhardwaj V, Blank C, Bretaudeau A, Guéguen L, Čech M, Chilton J, Clements D, Doppelt-Azeroual O, Erxleben A, Freeberg M, Gladman S, Hoogstrate Y, Hotz H-R, Houwaart T, Jagtap P, Lariviere D, Le Corguillé G, Manke T, Mareuil F, Ramírez F, Ryan D, Sigloch F, Soranzo N, Wolff J, Videm P, Wolfien M, Wubuli A, Yusuf D, Backofen R, Taylor J, Nekrutenko A, and Grüning B. Community-driven data analysis training for biology bioRxiv 225680; doi: https://doi.org/10.1101/225680

7. Watson-Haigh N, Shang C, Haimel M, Kostadima M, Loos R, Deshpande N, Duesing K, Li X, McGrath A, McWilliam S, Michnowicz S, Moolhuijzen P, Quenette S, De Leon Revote J, Tyagi S and Schneider M., Next-generation sequencing: a challenge to meet the increasing demand for training workshops in Australia. Briefings in Bioinformatics 2013 14(5):563-574 doi: http://dx.doi.org/10.1093/bib/bbt022

8. Friesner J, Assmann S, Bastow R, Bailey-Serres J, Beynon J, Brendel V, Buell C, Bucksch A, Busch W, Demura T, Dinneny J, Doherty C, Eveland A, Falter-Braun P, Gehan M, Gonzales M, Grotewold E, Gutierrez R, Kramer U, Krouk G, Ma S, Markelz C, Megraw M, Meyers B, Murray J, Provart N, Rhee S, Smith R, Spalding E, Taylor C, Teal T, Torii K, Town C, Vaughn M, Vierstra R, Ware D, Wilkins O, Williams C and Brady S. The Next Generation of Training for Arabidopsis Researchers: Bioinformatics and Quantitative Biology. Plant Physiology. 2017 175(4):1499-1509. doi: https://doi.org/10.1104/pp.17.01490

9. Welch L, Lewitter F, Schwartz R, Brooksbank C, Radivojac P, Gaeta B and Schneider M. Bioinformatics Curriculum Guidelines: Toward a Definition of Core Competencies. PLoS Comp Biol 2014 10(3): e1003496. doi: 10.1371/journal.pcbi.1003496

10. Morgan SL, Palagi PM, Fernandes PL, Koperlainen E, Dimec J, Marek D, Larcombe L, Rustici G, Attwood TK, Via A. The ELIXIR-EXCELERATE Train-the-Trainer pilot programme: empower researchers to deliver high-quality training. F1000Res. 2017 6:1557. doi: 10.12688/f1000research.12332.1

11. Wilson G. Software Carpentry: lessons learned. F1000Research 2016, 3:62 doi: 10.12688/f1000research.3-62.v2

12. Pawlik A, van Gelder CWG, Nenadic A, Palagi P, Korpelainen E, Lijnzaad P, Marek D, Sansone SA, Hancock J, Goble C. Developing a strategy for computational lab skills training through Software and Data Carpentry: Experiences from the ELIXIR Pilot action. F1000Research 2017, 6:1040. doi: 10.12688/f1000research.11718.1

13. Corpas M, Jimenez RC, Bongcam-Rudloff E, Budd A, Brazas MD, Fernandes PL, Gaeta B, van Gelder C, Korpelainen E, Lewitter F, McGrath A, MacLean D, Palagi PM, Rother K, Taylor J, Via A, Watson M, Schneider MV, Attwood TK. The GOBLET training portal: a global repository of bioinformatics training materials, courses and trainers. Bioinformatics 2015 31(1):140-2. doi: 10.1093/bioinformatics/btu601.

14. Schneider MV1, Walter P, Blatter MC, Watson J, Brazas MD, Rother K, Budd A, Via A, van Gelder CW, Jacob J, Fernandes P, Nyrönen TH, De Las Rivas J, Blicher T, Jimenez RC, Loveland J, McDowall J, Jones P, Vaughan BW, Lopez R, Attwood TK, Brooksbank C. Bioinformatics Training Network (BTN): a community resource for bioinformatics trainers. Brief Bioinform 2012 13(3):383–389 doi: 10.1093/bib/bbr064.

15. Baker M. 1,500 scientists lift the lid on reproducibility Nature 533(7604):452-454.

16. Fanelli D. Opinion: Is science really facing a reproducibility crisis, and do we need it to? Proc. Natl. Acad. Sci. U.S.A. 2018 doi: 10.1073/pnas.1708272114

17. Hutson M. Artificial intelligence faces reproducibility crisis. Science 2018 359(6377):725-726. doi: 10.1126/science.359.6377.725.

18. Morin A, Urban J, Adams PD, Foster I, Sali A, Baker D and Sliz P. Shining Light into Black Boxes Science 2012, 336(6078):159-160. doi: 10.1126/science.1218263

19. Hettrick S, Antonioletti M, Carr L, Chue Hong N, Crouch S, De Roure D, Emsley I, Goble C, Hay A, Inupakutika D, Jackson M, Nenadic A, Parkinson T, Parsons M, Pawlik A, Peru G, Proeme A, Robinson J and Sufi S. UK Research Software Survey 2014 [Data set]. Zenodo. http://doi.org/10.5281/zenodo.14809

20. Wilson G, Bryan J, Cranston K, Kitzes J, Nederbragt L and Teal TK Good enough practices in scientific computing. PLoS Comput Biol 2017 13(6):e1005510. doi: 10.1371/journal.pcbi.1005510

21. Wilkinson M, Dumontier M, Aalbersberg J, Appleton G, Axton M, Baak A et al. The FAIR Guiding Principles for scientific data management and stewardship. Sci. Data 2016 3 http://dx.doi.org/10.1038/sdata.2016.18

22. Lave J and Wenger E Situated Learning: Legitimate Peripheral Participation Cambridge University Press, 1991.

23. Wenger E. Communities of Practice: Learning, Meaning, and Identity. Cambridge University Press, 1998.

24. Li LC, Grimshaw JM, Nielsen C, Judd M, Coyte PC and Graham ID. Evolution of Wenger's concept of community of practice. Implement Sci. 2009 4:11. doi: 10.1186/1748-5908-4-11.

25. Wenger E. Communities of practice: A brief introduction National Science Foundation (US), 2011

26. Webber E. Building successful communities of practice TACIT, 2018.

27. Brown M.J. Building powerful community organizations Long Haul Press, 2006.

28. Wenger E, McDermott RA and Snyder W. Cultivating Communities of Practice. Harvard Business School Press, 2002.

29. Budd A, Corpas M, Brazas MD, Fuller JC, Goecks J, Mulder NJ, Michaut M, Ouellette BF, Pawlik A, Blomberg N. A quick guide for building a successful bioinformatics community. Plos Comp Biol 2015 Feb 5;11(2):e1003972. doi: 10.1371/journal.pcbi.1003972

30. Wenger E and Snyder W. Communities of practice: the organizational frontier. Harvard business review, 2000 78:139-146

31. Bauer MS, Damschroder L, Hagedorn H, Smith J and Kilbourne AM. An introduction to implementation science for the non-specialist. BMC Psychology, 2015 3(1):32. doi:10.1186/s40359-015-0089-9.

32. Macintyre G, Michault M and Abeel T. The regional student group program of the ISCB student council: stories from the road. PLoS Comp Biol 2013 9(9):e1003241. doi: 10.1371/journal.pcbi.1003241.

33. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R and 1000 Genome Project Data Processing Subgroup (2009) The Sequence alignment/map (SAM) format and SAMtools. Bioinformatics, 25, 2078-9. doi: 10.1093/bioinformatics/btp352.

34. Dreyfus S.E. and Dreyfus H.L. A Five-Stage Model of the Mental Activities Involved in Directed Skill Acquisition. Washington, DC: Storming Media. 1980