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 self-learn how to drive. Although that might be doable, it is preferable and safer to learn good practices. One way to achieve this is to build local communities of practice by bringing together scientists that perform code-intensive research to spread know-how and good practices.

Here, we indicate important challenges and issues that stand in the way of establishing these local communities of practice. For a given researcher working for an academic institution, their capacity to conduct data-intensive research will be arbitrarily relying on the presence of well-trained bioinformaticians in their neighborhood.

In this paper, we propose a model to build a local community of practice for scientific programmers. First, Software/Data Carpentry (SWC) programming workshops designed for researchers new to computational biology can be organized. However, while they provide an immediate solution for learning, more regular long-term assistance is also needed. Researchers need persisting, local support to continue learning and to solve programming issues that hamper their research progress. The solution we describe here is to implement a study group where researchers can meet-up and help each other in a "safe-learning atmosphere". Based on our experience, we describe two 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.

The current challenge is to make these local communities self-sustainable despite the high turnover of researchers at any institution and the lack of academic reward (e.g. publication). Here, we present some lessons learned from our experience. We believe that our local communities of practice will prove useful for other scientists that want to set up similar structures of 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, some 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, after the release 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]. In addition to genome and gene expression, sequencing has also profound consequences for Ecology and Microbial Ecology [2]. In addition to development of NGS technologies, application of mass spectrometry to metabolomics have 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]. **Altogether, the researcher in the Life Sciences now faces challenges in "Big Data" analysis coupled with the need to adopt good practices in scientific programming.**

Good practices and skills are needed in scientific programming

Consequently, good programming skills are becoming essential in Life Sciences but, usually, training lags behind. First, as past education of current life scientist did not comprise bioinformatic courses, new PhD students are most often devoid of any background in bioinformatics, data analysis or statistics whereas they are well trained for wet-lab matters. In addition, at international institutions like Universities, the staff originates from a variety of background so that programming and data analysis levels are also highly variable. Overall, the vast majority of wet-lab researchers need tailor-made practical training to learn programming and data analysis.
Current efforts in Bioinformatics and Data Science training for Life Scientists present 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 have been formed [6]. Several foundations such as the Software and Data Carpentry (SWC) Foundations provide periodic training workshops to researchers to instruct basic and robust software development skills [7]. The two organisations have joined efforts very recently across Europe [8]. Expensive one-time training courses are regularly offered throughout the year but, in most cases, researchers need more regular support to debug their code, make progress and adopt good practices. Additionally, institutional support for teaching skills and good practices in scientific programming to Life Scientists is usually lacking. Vital skills for bioinformaticians do not only include hard-skills in programming and data science as management, leadership and project organisation skills are also required [9]. New challenges have also arisen as researchers will increasingly need to comply with the FAIR (Findable, Accessible, Interoperable and Reusable) principles that have become mandatory for funding agencies and publishing [10]. The increasing generation of large amounts of data will certainly push for more data integration and re-usability. **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

Fueled by Etienne Wenger's idea that learning is usually a social activity, we propose to build a community of practice in scientific programming for life scientists [15]. 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 its members are practioners *i.e.* they are currently engaged in research that involves scientific programming. This community of practice acts as a "one-stop shop" to get technical assistance, assist to an one-hour lesson, hear about the next training workshop and experience live software demos. While short-term immediate issues ("help me now to debug my code") can be solved, the community has also the capacity to steward solutions for long-term data-related problems ("how do I comply with the FAIR guidelines"). **Overall, we believe that building these local communities of practice in scientific programming will considerably support scientific research and help to tackle the data deluge in the Life Sciences.** In this paper, we provide an overview of the problems they solve, propose a field guide to implement such communities and confront our guide with reality using two real-world examples.

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 will face when trying to analyze its data using the computer.

Isolation

It is becoming increasingly common that a wet lab biologist is asked by his supervisor to analyse a set of pre-existing data. For instance, a PhD student receives transcriptomic mRNA-Seq data and his/her supervisor asks 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 face 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 re-inventing the wheel, lack of version control and irreproducible results. While some compiled easy-to-use softwares such as samtools [11] can help to get around, usually, a researcher will need to build its own collection of tools and scripts. Version control is often overlooked by researchers as non-critical and can lead to cryptic file nomenclature. We believe that version control with git[1] and github[2] for instance can be seen as mandatory good practices just like accurate pipetting in the molecular biology 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 an "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

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

bioinformatics. This also presents a challenge for future 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 for code-related science.

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 will not hire an experienced bioinformatician. Therefore, the wet lab researcher is often confronted with the ”what do I need to learn?” conundrum. Having experts around is then mandatory for novices. Even in the best scenario where an expert bioinformatician is available, it might not be optimal to get all the knowledge in one field from one person. Instead, we propose that building a community of expert 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. These different levels and the progression from one learning stage to another are illustrated in Fig 1. Here, it is important to note that although champions often lead the local community of practice, it also happens that beginners and competent practitioners set up a session where they invite experts to discuss a particular topic. Thus, rather than a rigid hierarchical structure, the local community is meant to be horizontal and welcoming.

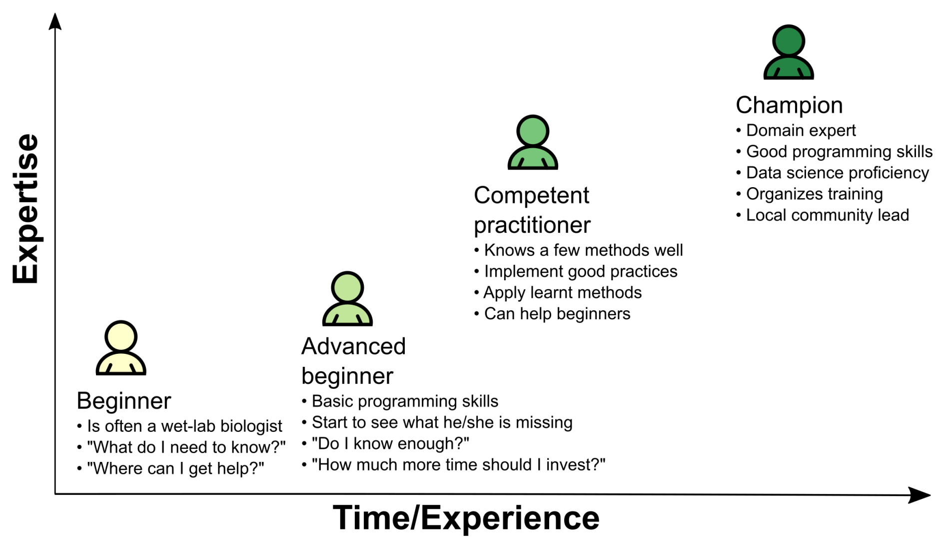

Fig 1. Different learning stages in scientific programming. This figure displays the different stages of learning encountered by experimental biologists.

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

Hereafter, we describe two real-life examples of community building at two Universities. These two communities were built by local scientists and aimed at Life Science researchers.

Learning from these experience, a practical model to build a community of practice at a given research institution is suggested.

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, it felt that a regular meet-up group was needed so that researchers with different programming levels could help and support each other. Hence, in April 2017, we started up the **Amsterdam Science Park Study Group**[3] following the Mozilla Study Group guidelines. Briefly, Mozilla Study Groups are informal and regular meet-ups of researchers that want to improve their coding practice, discover new softwares and tools applicable to their research and work together to solve code-related issues. Study Groups have been implemented at various academical institutions around the globe[4]. We felt that a local Amsterdam Science Park Study Group was necessary to follow-up on the Software Carpentry workshop and the goal of building a local community. We quickly decided to stick to the guidelines suggested by the Mozilla Science Lab[5]. Originally, we started with one scientist from the University (Marc Galland) and two engineers in software engineering (Mateusz Kuzak and Carlos Martinez). But after 5 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 (sequencing, statistics, ecology). Most Study Group members came from two different institutes (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[6] 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". It was started as a place to discuss Python programming, address scientific issues in computational biology, such as metagenomics, and help other graduate students to learn scientific coding. The main ComBEE group meets once a month to talk about 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 webpage from google sites to using the template put together by the Mozilla Science Lab[7]. Early in the development of ComBEE, the facilitating of the

[3] `https://scienceparkstudygroup.github.io/studyGroup/`
[4] `https://science.mozilla.org/programs/studygroups`
[5] `https://mozillascience.github.io/study-group-orientation/`
[6] `https://github.com/mozillascience/studyGroup`
[7] `https://combee-uw-madison.github.io`

language-specific study groups was delegated on a semester by semester basis, this in turn 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 just 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 continues to contribute back to the group, lending his expertise and experience to the latest study group discussions. The ComBEE Study Group is now three years old and acts as a stable resource center for new graduate students and employees.

A suggested model to build a community of practice

Based on these two experiences, we propose a working model (Fig 2) to create a local community of practice 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 coupling them to formation of a Study Group. In our experience, SWC workshops work well since they provide workshops aimed at researchers and these organizations possess a long history of teaching programming to scientists [7,12]. Yet, other formats for programming workshops should also work well in practice and running SWC workshops is not mandatory. These programming workshops serve as a starting point for learning and gathering researchers together in one room. When absolute beginners join these workshops, they become "advanced beginners" since they have some notions 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). This local community needs to regularly meet 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 can occur very frequently. Therefore, a local co-working group should be set-up with a regular meeting schedule. Mozilla Study Group are well documented in the form of the on-line guide[8] and even comes with a template to create a Study Group website[9]. There are nearly 100 Mozilla Study Groups around the world and the Mozilla Foundation[10] facilitates the exchange of experience between the leaders of these groups. The University of Toronto Coders Study Group[11], 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 co-working group can be used but we believe that Mozilla Study Groups offer a range of online material and support such that there is no need to "re-invent the wheel".

In stage 2, the Study Group acts as 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 will also welcome new novice members as they join the research institution or as they hear about the existence of a local co-working 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

[8] http://mozillascience.github.io/studyGroupHandbook/
[9] https://github.com/mozillascience/studyGroup
[10] https://www.mozilla.org/en-US/
[11] https://uoftcoders.github.io/studyGroup/

sessions where they invite experts and discuss a particular topic: leading a lesson is not restricted to community leads. At the end of this stage, most advanced beginners should have become competent practitioners (Fig 2).

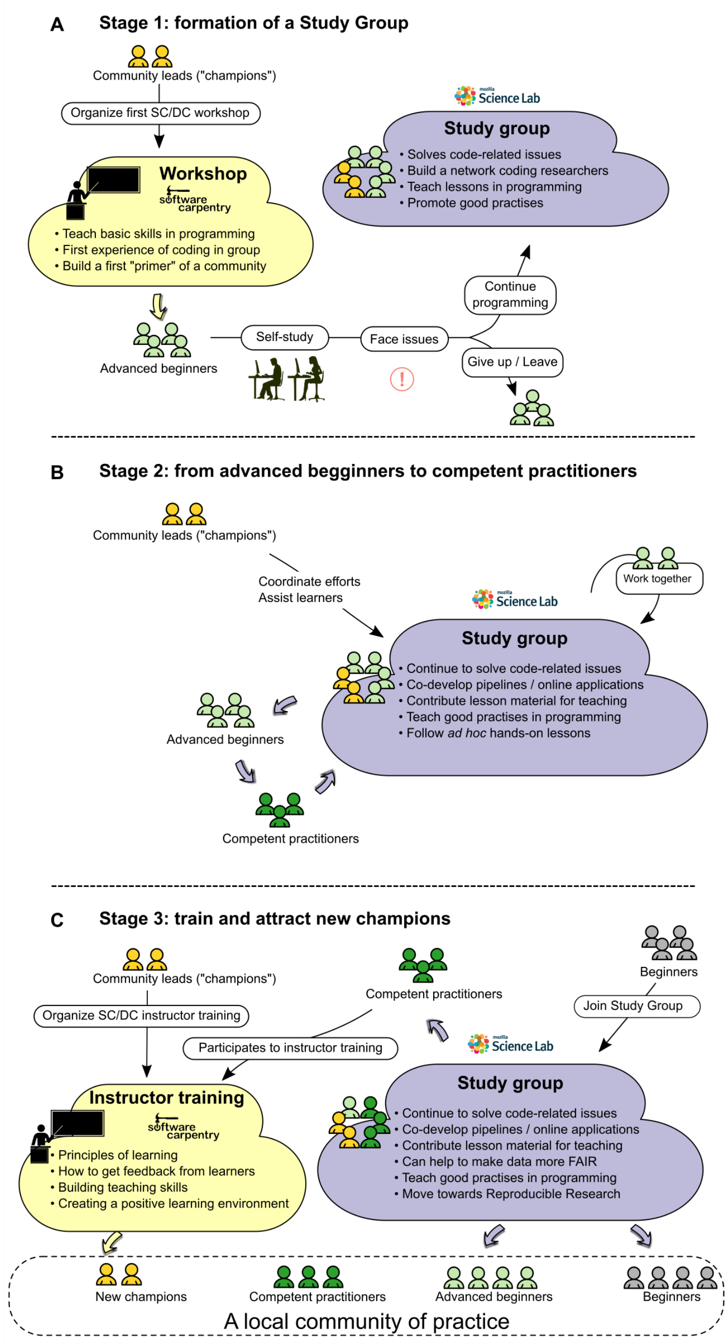

Fig 2. (Legend on next page)

Fig 2. (Previous page.)A two-step model to build a local community of practice. This figure describes the two 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 SWC 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 learn new programming skills. Therefore, a local 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" follow dedicated SWC instructor training sessions, new community leads ("champions") can be trained. In addition, the local Study Group keeps on attracting new beginners. Study Group sessions together with optional SWC 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.

In stage 3, a subset of the competent practitioners from the local community will become community leads ("champions", Fig 2). To become champions, competent practitioners need to increase their teaching skills and be able to visualize the level of their audience (Fig 1). That can be done through by becoming SWC instructors through a specific instructor training event. This specific occasion can be organized by initial community champions since they usually dispose of both the network and know-how to set-up these specific workshops. Once again, it is not mandatory to rely on the SWC organization as long as competent practitioners get a deeper knowledge of teaching techniques where they improve their own skills. Yet, we now have a good perspective on the long-term experience of the SWC foundation with over 500 workshops organized and 16,000 attendees [7]. Moreover, major bioinformatic programs such as ELIXIR also rely on the SWC model to educate experimental biologists to scientific programming, computation, data management and data science skills [7, 8].

In practice, stages 2 and 3 can occur simultaneously because an active Study Group acts as an attraction pole for experienced computational biologists and other scientific programmers ("champions").
There are many models for building such local communities. The one we chose is to have regular fortnight Mozilla Science Lab Study Group complemented by occasional SWC workshops. Mozilla Study Group are well-documented in the form of an online guide[12] and this even comes coupled with a template to create a Study Group website[13]. There are nearly 100 Mozilla Study Groups round the world and the Mozilla Foundation[14] facilitates the exchange of experience between the leaders of these groups. The University of Toronto Coders Study Group[15], which is a mix of work-along and co-working sessions is a good Study Group example. In addition to Mozilla Study Groups, SWC *ad hoc* workshops help members of the community to make progress and embrace good programming habits. We propose a positive-feedback model (Fig 2) on how to build a

[12] http://mozillascience.github.io/studyGroupHandbook/
[13] https://github.com/mozillascience/studyGroup
[14] https://www.mozilla.org/en-US/
[15] https://uoftcoders.github.io/studyGroup/

local community of practice.

Room for improvement: how to make these communities self-sustainable?

After a local community has been established, the community as a whole face a different challenge: keeping the community alive. Here, we address a few key points for making a community sustainable.

Practical considerations

- The community needs a critical mass: this refers to the number of people who come to study sessions on a regular (or semi-regular) basis. Based on our experience from the examples above, 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 critical mass ensures there is always enough people in attendance to make the session useful.

- Constant refresh: universities are dynamic environments where people come and go. This has an impact on the local community of practice as some of its members are bound to disappear after some time. However such a dynamic environment should also help to bring in new people both eager to learn and with relevant knowledge to share in the group. Use the turnover of people to your advantage, making sure to continue to recruit new members.

- Meeting notifications and frequency: for most people it becomes easier to attend an event when they know well-in-advance when it will take place. For instance, more people will be able to plan on attending the Study Group sessions if they know they take place on Tuesdays, every two weeks. Schedule the meetings well-in-advance and keep a consistent day of the week and time. 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 fortnightly meetings (every two weeks) as a good starting frequency

- Meet-up place: to keep an informal and welcoming atmosphere, sessions should take place in a relatively quiet environment with a good Internet connection. As such, a campus café outside of busy hours can be a good place to start up.

Community composition: what type of scientists should constitute a Study Group?

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 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 to be in the community to ensure that they will continue to be interested in belonging to the community. In our experience, we have identified the following types of community members, each with their motivations to be in 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. Usually, novices lack the overview of tools and software necessary and sufficient to perform their work. Usually, they have questions related to immediate research issues. No time for overview!

- 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 are also the most willing to teach novices, being able to easily relate to the beginner state of mind. In turn, this increase their learning skills, a key aptitude for a successful community of practice.

- Champions: these are people with the highest experience level on a particular skill in the community. It is often challenging to engage them to become part of the community. Yet, in our experience, champions usually reinforce their knowledge by 'going back to basics': it is useful for them to understand what are the usual *gotchas* for novices. Also, champions 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 champions 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, we think that one competent practitioner for three beginners is a good ratio as he/she can solve the problems of the three novices almost simultaneously. Finally, from time to time, experts ("champions") may be invited to speak about their area of expertise, for instance a new software or technique. Although here we have described three type of community members, we would like to emphasize that these categories are just guidelines and rules. In reality people will not fall neatly one of these categories, but rather be in an intermediate state between two of these categories.

Conclusion

The next challenge for our local communities of practice will be to move from an "emergent community" to a "mature community" as defined by the Community Roundtable organization[16]. Today, our emergent communities generate some user-defined content (lesson notebooks), have a rather informal community management method and most decisions affecting the community are taken by consensus. An effort is being made to assign clear and specific roles to administration members of the local community based on their expertise and aptitude. For instance, some people are competent in building websites so they 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. Another challenge is to secure funding and support from the local institution as this can boost *ad hoc* 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). As stated by Wilson and co-authors, "progress will not happen by itself" [12]. As major scientific journals and funding agencies require a Research Data Management plan together with FAIR datasets [10], it becomes more and more important to enable wet-lab researchers to conduct good scientific programming, data science and research data management. Eventually, these local communities of practice should speed up code-intensive biological research, promote Open Science and Reproducibility and spread good practices among life scientists.

[16]The community round-table: state of community management 2016

Acknowledgments

We are thankful to the Data and Software Carpentry organization 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 and not in the author list is fully acknowledged and consist of Dr Emiel van Loon (University of Amsterdam, IBED), Pietro Marchesi ((University of Amsterdam, SILS), Joeri Jongbloets (University of Amsterdam, SILS), Dr Like Fokkens (University of Amsterdam, SILS), Zsofia Koma (University of Amsterdam, IBED) and Susanne Wilkens (University of Amsterdam, IBED). We are grateful to Dr Anita Schürch (UMC Utrecht) for training researchers through several Software and Data Carpentry workshops.

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. 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

7. Wilson G. Software Carpentry: lessons learned. F1000Research 2016, 3:62

8. 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.

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.

10. 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

11. 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.

12. 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. https://doi.org/10.1371/journal.pcbi.1005510

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. 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

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

16. 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

17. 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.