

GigaScience

Streamlining Data-Intensive Biology with Workflow Systems

--Manuscript Draft--

Manuscript Number:	GIGA-D-20-00245R1	
Full Title:	Streamlining Data-Intensive Biology with Workflow Systems	
Article Type:	Review	
Funding Information:	Gordon and Betty Moore Foundation (GBMF4551)	Dr. C. Titus Brown
	STATE AND FEDERAL CONTRACTORS WATER AGENCY (A19- 1844)	Not applicable
	National Science Foundation (1711984)	Dr. N. Tessa Pierce
Abstract:	<p>As the scale of biological data generation has increased, the bottleneck of research has shifted from data generation to analysis. Researchers commonly need to build computational workflows that include multiple analytic tools and require incremental development as experimental insights demand tool and parameter modifications. These workflows can produce hundreds to thousands of intermediate files and results that must be integrated for biological insight. Data-centric workflow systems that internally manage computational resources, software, and conditional execution of analysis steps are reshaping the landscape of biological data analysis, and empowering researchers to conduct reproducible analyses at scale. Adoption of these tools can facilitate and expedite robust data analysis, but knowledge of these techniques is still lacking. Here, we provide a series of practices and strategies for leveraging workflow systems with structured project, data, and resource management to streamline large-scale biological analysis. We present these strategies in the context of high-throughput sequencing data analysis, but the principles are broadly applicable to biologists working beyond this field.</p>	
Corresponding Author:	Nuri Teresa Pierce, Ph.D University of California Davis Davis, CA UNITED STATES	
Corresponding Author Secondary Information:		
Corresponding Author's Institution:	University of California Davis	
Corresponding Author's Secondary Institution:		
First Author:	Taylor Reiter	
First Author Secondary Information:		
Order of Authors:	Taylor Reiter	
	Phillip T. Brooks	
	Luiz Irber	
	Shannon E.K. Joslin	
	Charles M. Reid	
	Camille Scott	
	C. Titus Brown	
	N. Tessa Pierce	
Order of Authors Secondary Information:		
Response to Reviewers:	Dear Editors,	

Thank you for the opportunity to submit a revised draft of our manuscript "Streamlining Data-Intensive Biology with Workflow Systems" by Reiter et al. We appreciate the effort you and the reviewers have dedicated to providing valuable feedback for improving the manuscript. In particular, we thank the reviewers for their strong points on managing reproducible scientific software installations. We hope that the edited manuscript better describes the important compromises we see between usability, installability, and reproducibility.

Below, please find a point-by-point response to each review. The manuscript changes are also available on github, perhaps most easily viewed here: <https://github.com/dib-lab/2020-workflows-paper/pull/58/files>.

Reviewer #1:

>This manuscript is a review that provides insights into creating workflows and executing them using workflow engines. The authors also provide extensive practical recommendations on performing data-intensive biology. Overall it is easy to understand, and the figures are nice. Below are some changes that I suggest.

>

>1. The Abstract states, "These workflows commonly produce hundreds to thousands of intermediate files." Can you provide an example of a workflow that would require such a large number of intermediate files? That would help to make this need more concrete.

We have better described large workflows and linked to a larger directed acyclic graph (DAG) within the Figure 5 caption.

>2. Line 76: "These features ensure that the steps for data analysis are minimally documented..." Consider rephrasing. To me it reads that minimal documentation is desirable. But I think you mean that things should be documented at least to a minimum requirement.

Yes - we have rephrased this sentence to better reflect the point.

>3. The paper briefly emphasizes software containers. It puts more emphasize on software-management systems like conda. However, in my opinion, containers are a critical tool and should be emphasized more. Software-management systems sometimes cannot install all of the required dependencies, and not all tools are included in these systems. However, containers provide more flexibility for these types of situations and are supported by all or nearly all workflow management systems.

We have reworked the software management section to better describe the range of software management solutions, including both limitations of conda and benefits and limitations of container-based software installation.

>4. Type-o on line 138: "devloping"

fixed - thanks!

>5. Line 205: "Using software without learning management systems." I'm not sure that I understand the wording on this. This term has a very specific meaning in education administration (https://en.wikipedia.org/wiki/Learning_management_system), but I think you mean something different. Also, that who section feels a bit disjointed. It was difficult to wrap my mind around exactly what it is trying to say.

We have reworded this section for clarity. We hope it now reflects the scope of software usage and installation options and provides some useful context for how to choose between them.

>6. Line 268: Readers may be unclear what "seqanswers" is. Please provide more context.

We removed this unnecessary reference. SeqAnswers is described instead in a later portion of the manuscript.

>7. Lines 410-411: This sentence seems unnecessary. Same with lines 420-421.

Removed.

>8. Line 474: consider using italics rather than bold text to emphasize this point.

Agreed - changed.

>9. Line 502: You reference these repositories in the next paragraph, but it would be better to do so the first time you mention them.

We moved the references here, thanks.

>10. Line 509: This statement is somewhat subjective. Consider removing it or providing a more detailed justification.

We have pared down this section.

>11. Table 3: Some users may be unfamiliar with what bash is.

Removed reference to bash.

>12. Line 638: Should be "Principal Component Analysis"

fixed!

>13. The manuscript starts with a focus on using workflow systems. Later it provides a wide range of general recommendations for doing data-intensive biology. Maybe it's just me, but I felt like it got too long and a bit unfocused in the second half. It's up to you, but you might consider splitting it into two manuscripts: one on workflow systems and one on data-intensive biology. Or maybe consider removing/simplifying some of the sections so that it is not so long and is a bit more focused.

We recognize that the manuscript is long, but feel that it provides a comprehensive set of recommendations for researchers to get started using data-intensive workflows for working with sequencing data. In particular, we think that the quality control and data management recommendations we discuss in the latter half of the manuscript are critical for building high-quality, reproducible sequence analysis workflows.

Reviewer #2:

>

>Reiter et al. provides a review on the current state of the data-centric workflows for data analysis tasks in biology. The authors touch all aspects of data analysis tasks, not only the workflow systems but also the whole ecosystem that contains everything from workflow management systems to data integrity and managing computational resources. I generally found the recommendations and the review very useful for the community except one point detailed below.

>

>I understand this manuscript is based on personal experiences as authors acknowledged in their summary. However, in my opinion, this review misses important developments in reproducibility that are compatible with the main recommendations of the review.

>

>The authors mention software wrangling as a crucial part in scientific reproducibility assuming that the initial data is intact and available. They mention Conda, Singularity, and Docker as methods to manage software for reproducibility. However, we see reproducibility as a spectrum. A fully reproducible workflow would have the following ingredients assuming the data is available: 1) code and usage transparency, 2)

installability and 3) reproduction of runtime environment. The authors give reasonable recommendations for the code and usage transparency which is mainly making sure that the code is documented to the highest quality, and available publicly. Conda, Singularity and Docker remedies the installation problem. They make dependencies easily installable in most cases. However, authors do not mention short-comings of these solutions. The main short-coming of those tools is that they don't fully satisfy reproduction of the runtime environment if they do so in the case of docker/singularity they do this opaquely and are not transparent. It is hard to verify exactly what are the contents of a container. Docker recommends use of docker files but they do not necessarily have the version of the software that is in the container. It is mostly a collection of commands installing software from package managers. Which brings us to the same problems I describe for conda below. These commands often include invocations of package managers like Apt (in the case of Debian-derived foundations). A package installed via apt today will *not* be identical to the same package installed a year ago. Containers are also harder to maintain if you do not analyze data and develop code on dockerized environments exclusively.

>

>Conda packages, on the other hand, do not provide reproducibility at all. You can get different binaries for the same name+version query at different times and there is no way to track which source files of dependencies produced that binary. The system environment where the software is built is not isolated. During the build, processes have access to other libraries that are not in the package recipe and also conda assumes certain low-level packages to be available in all environments.

>

>In essence, the authors don't mention that *all* the tools they recommend (conda, docker and singularity) are completely time-dependent. Some Conda channels provide archives of pre-built binaries, yes, but since not all dependencies (beyond the kernel) are taken into account at build time you will *not* be able to run these binaries without changes on a different system.

>

>There are package managers like GNU Guix remedies most of these problems and provide the state-of-the-art reproducibility exemplified by the recent "Ten Years Reproducibility Challenge" (<https://www.nature.com/articles/d41586-020-02462-7>). This package management system is also incorporated in snakeMake-based pipelines providing gold-standard reproducibility for multiple NGS analysis pipelines (<https://academic.oup.com/gigascience/article/7/12/giy123/5114263>).

>

>Altuna Akalin & Ricardo Wurmus

We strongly agree that reproducibility is a spectrum, and thank the reviewers for their thoughtful discussion and references. In response, we've extensively revised the software management portion of the manuscript, detailing the reproducibility limitations of each approach and adding information on functional package managers like GNU guix and Nix. Additionally, we have added some information on choosing between these systems that will hopefully convey the merits of each.

In our experience, software management systems currently present significant trade-offs between usability, installability, and reproducibility. In providing recommendations to the field, we feel it is important to prioritize usability, which is a critical determinant for widespread adoption of computational techniques. Systems that provide 100% reproducibility but are not straightforward to implement are not a viable solution for researchers with limited computational training or experience.

While conda is not currently the best system for producing long-term (10-year) reproducibility, we feel that it provides a reasonable balance between usability and reproducibility, particularly when used according to the provided recommendations (small, isolated, version-pinned environments are critical for both usability and reproducibility). Furthermore, broad community adoption has resulted in proliferation of conda-installable tools and helps ensure that the ecosystem around these tools will be maintained and improved over the coming years.

The expanded text on software management systems is reproduced here:

	<p>> "On the lightweight end, the conda package manager has emerged as a leading software management solution for research workflows (Figure 2). Conda handles both cluster permission and version conflict issues with a user-based software environment system, and features a straightforward "recipe" system which simplifies the process of making new software installable (including simple management of versions and updates). These features have led to widespread adoption within the bioinformatics community: packages for new software become quickly available, and can be installed easily across platforms. However, conda does not completely isolate software installations and aims neither for bitwise reproducibility nor long-term archiving of install packages, meaning installations will not be completely reproducible over time. Heavyweight software management systems package not only the software of interest, but also the runtime environment information, with the goal of ensuring perfect reproducibility in software installation over time. Tools such as singularity and docker [3,11,37,38] wrap software environments in "containers" that capture and reproduce the runtime environment information. Container-based management is particularly useful for systems where some dependencies may not be installable by lightweight managers. However, software installation within these containers can be limited by similar reproducibility issues, including changes in dependency installations over time. "Functional package managers" such as GNU Guix and Nix strictly require all dependency and configuration details be encoded within each software package, providing the most comprehensively reproducible installations. These have begun to be integrated into some bioinformatic tools [16], but have a steeper learning curve for independent use. In addition, standard installation of these managers requires system-wide installation permissions, requiring assistance from system administrators on most high-performance computing systems."</p> <p>All changes can be most easily seen here: https://github.com/dib-lab/2020-workflows-paper/pull/58/files.</p> <p>Overall, we hope the revised manuscript reflects the important points the reviewers have raised while still emphasizing accessible techniques that will move researchers towards fully automated, reproducible analyses.</p> <p>Sincerely,</p> <p>N. Tessa Pierce, corresponding author</p>
Additional Information:	
Question	Response
Are you submitting this manuscript to a special series or article collection?	No
<p>Experimental design and statistics</p> <p>Full details of the experimental design and statistical methods used should be given in the Methods section, as detailed in our Minimum Standards Reporting Checklist. Information essential to interpreting the data presented should be made available in the figure legends.</p> <p>Have you included all the information requested in your manuscript?</p>	Yes

<p>Resources</p> <p>A description of all resources used, including antibodies, cell lines, animals and software tools, with enough information to allow them to be uniquely identified, should be included in the Methods section. Authors are strongly encouraged to cite Research Resource Identifiers (RRIDs) for antibodies, model organisms and tools, where possible.</p> <p>Have you included the information requested as detailed in our Minimum Standards Reporting Checklist?</p>	<p>Yes</p>
<p>Availability of data and materials</p> <p>All datasets and code on which the conclusions of the paper rely must be either included in your submission or deposited in publicly available repositories (where available and ethically appropriate), referencing such data using a unique identifier in the references and in the “Availability of Data and Materials” section of your manuscript.</p> <p>Have you have met the above requirement as detailed in our Minimum Standards Reporting Checklist?</p>	<p>Yes</p>

Streamlining Data-Intensive Biology With Workflow Systems

This manuscript ([permalink](#)) was automatically generated from [dib-lab/2020-workflows-paper@09910b4](#) on November 6, 2020.

Authors

- **Taylor Reiter** [0000-0002-7388-421X](#) · [taylorreiter](#) · [ReiterTaylor](#) Department of Population Health and Reproduction, University of California, Davis · Funded by Moore Foundation GBMF4551
- **Phillip T. Brooks** [0000-0003-3987-244X](#) · [brooksph](#) · [brooksph](#) Department of Population Health and Reproduction, University of California, Davis
- **Luiz Irber** [0000-0003-4371-9659](#) · [luizirber](#) · [luizirber](#) Department of Population Health and Reproduction, University of California, Davis · Funded by Moore Foundation GBMF4551
- **Shannon E.K. Joslin** [0000-0001-5470-1193](#) · [shannonekj](#) · [IntrprtnGnmes](#) Department of Animal Science, University of California, Davis · Funded by State and Federal Water Contractors A19-1844
- **Charles M. Reid** [0000-0002-7337-8894](#) · [charlesreid1](#) Department of Population Health and Reproduction, University of California, Davis · Funded by Moore Foundation GBMF4551
- **Camille Scott** [0000-0001-8822-8779](#) · [camillescott](#) · [camille codon](#) Department of Population Health and Reproduction, University of California, Davis · Funded by Moore Foundation GBMF4551
- **C. Titus Brown** [0000-0001-6001-2677](#) · [ctb](#) · [ctitusbrown](#) Department of Population Health and Reproduction, University of California, Davis · Funded by Moore Foundation GBMF4551
- **N. Tessa Pierce** [0000-0002-2942-5331](#) · [bluegenes](#) · [saltyscientist](#) Department of Population Health and Reproduction, University of California, Davis · Funded by NSF 1711984

1 **Abstract**

2 As the scale of biological data generation has increased, the bottleneck of research has shifted from data
3 generation to analysis. Researchers commonly need to build computational workflows that include
4 multiple analytic tools and require incremental development as experimental insights demand tool and
5 parameter modifications. These workflows can produce hundreds to thousands of intermediate files and
6 results that must be integrated for biological insight. Data-centric workflow systems that internally
7 manage computational resources, software, and conditional execution of analysis steps are reshaping the
8 landscape of biological data analysis, and empowering researchers to conduct reproducible analyses at
9 scale. Adoption of these tools can facilitate and expedite robust data analysis, but knowledge of these
10 techniques is still lacking. Here, we provide a series of practices and strategies for leveraging workflow
11 systems with structured project, data, and resource management to streamline large-scale biological
12 analysis. We present these strategies in the context of high-throughput sequencing data analysis, but the
13 principles are broadly applicable to biologists working beyond this field.

14 **Author Summary**

15 We present a guide for workflow-enabled biological sequence data analysis, developed through our own
16 teaching, training and analysis projects. We recognize that this is based on our own use cases and
17 experiences, but we hope that our guide will contribute to a larger discussion within the open source and
18 open science communities and lead to more comprehensive resources. Our main goal is to accelerate the
19 research of scientists conducting sequence analyses by introducing them to organized workflow practices
20 that not only benefit their own research but also facilitate open and reproducible science.

21

22 **Introduction**

23 Biological research has become increasingly computational. In particular, genomics has experienced a
24 deluge of high-throughput sequencing data that has already reshaped our understanding of the diversity
25 and function of organisms and communities, building basic understanding from ecosystems to human
26 health. The analysis workflows used to produce these insights often integrate hundreds of steps and
27 involve a myriad of decisions ranging from small-scale tool and parameter choices to larger-scale design
28 decisions around data processing and statistical analyses. Each step relies not just on analysis code written
29 by the researcher, but on third-party software, its dependencies, and the compute infrastructure and
30 operating system on which the code is executed. Historically, this has led to the patchwork availability of
31 underlying code for analyses as well as a lack of interoperability of the resulting software and analysis
32 pipelines across compute systems [1]. Combined with unmet training needs in biological data analysis,
33 these conditions undermine the reuse of data and the reproducibility of biological research, vastly limiting
34 the value of our generated data [2].

35 The biological research community is strongly committed to addressing these issues, recently formalizing
36 the FAIR practices: the idea that all life sciences research (including data and analysis workflows) should
37 be Findable, Accessible, Interoperable, and Reusable [3]. For computational analyses, these ideals are
38 readily achievable with current technologies, but implementing them in practice has proven difficult,
39 particularly for biologists with little training in computing [3]. However, the recent maturation of data-
40 centric workflow systems designed to automate and facilitate computational workflows is expanding our
41 capacity to conduct end-to-end FAIR analyses [5]. These workflow systems are designed to handle some
42 aspects of computational workflows internally: namely, the interactions with software and computing
43 infrastructure, and the ordered execution of each step of an analysis. By reducing the manual input and
44 monitoring required at each analysis juncture, these integrated systems ensure that analyses are repeatable
45 and can be executed at much larger scales. In concert, the standardized information and syntax required

46 for rule-based workflow specification makes code inherently modular and more easily transferable
47 between projects [5,6]. For these reasons, workflow systems are rapidly becoming the workhorses of
48 modern bioinformatics.

49 Adopting workflow systems requires some level of up-front investment, first to understand the structure
50 of the system, and then to learn the workflow-specific syntax. These challenges can preclude adoption,
51 particularly for researchers without significant computational experience [4]. In our experiences with both
52 research and training, these initial learning costs are similar to those required for learning more traditional
53 analysis strategies, but then provide a myriad of additional benefits that both facilitate and accelerate
54 research. Furthermore, online communities for sharing reusable workflow code have proliferated,
55 meaning the initial cost of encoding a workflow in a system is mitigated via use and re-use of common
56 steps, leading to faster time-to-insight [5,7].

57 Building upon the rich literature of “best” and “good enough” practices for computational biology
58 [8,9,10], we present a series of strategies and practices for adopting workflow systems to streamline data-
59 intensive biology research. This manuscript is designed to help guide biologists towards project, data, and
60 resource management strategies that facilitate and expedite reproducible data analysis in their research.
61 We present these strategies in the context of our own experiences working with high-throughput
62 sequencing data, but many are broadly applicable to biologists working beyond this field.

63 **Workflows facilitate data-intensive biology**

64 Data-intensive biology typically requires that researchers execute computational workflows using
65 multiple analytic tools and apply them to many experimental samples in a systematic manner. These
66 workflows commonly produce hundreds to thousands of intermediate files and require incremental
67 changes as experimental insights demand tool and parameter modifications. Many intermediate steps are
68 central to the biological analysis, but others, such as converting between file formats, are rote
69 computational tasks required to passage data from one tool to the next. Some of these steps can fail

70 silently, producing incomplete intermediate files that imperceptively invalidate downstream results and
71 biological inferences. Properly managing and executing all of these steps is vital, but can be both time-
72 consuming and error-prone, even when automated with scripting languages such as bash.

73 The emergence and maturation of workflow systems designed with bioinformatic challenges in mind has
74 revolutionized computing in data intensive biology [11]. Workflow systems contain powerful
75 infrastructure for workflow management that can coordinate runtime behavior, self-monitor progress and
76 resource usage, and compile reports documenting the results of a workflow (**Figure 1**). These features
77 ensure that the steps for data analysis are repeatable and at least minimally described from start to finish.
78 When paired with proper software management, fully-contained workflows are scalable, robust to
79 software updates, and executable across platforms, meaning they will likely still execute the same set of
80 commands with little investment by the user after weeks, months, or years.

81

82 *Figure 1: **Workflow Systems:** Bioinformatic workflow systems have built-in functionality that facilitates*
83 *and simplifies running analysis pipelines. **A. Samples:** Workflow systems enable you to use the same code*
84 *to run each step on each sample. Samples can be easily added if the analysis expands. **B. Software***
85 ***Management:** Integration with software management tools (e.g. conda, singularity, docker) can automate*
86 *software installation for each step. **C. Branching, D. Parallelization, and E. Ordering:** Workflow*
87 *systems handle conditional execution, ensuring that tasks are executed in the correct order for each*
88 *sample file, including executing independent steps in parallel if possible given the resources provided. **F.***
89 ***Standard Steps:** Many steps are now considered “standard” (e.g. quality control). Workflow languages*
90 *keep all information for a step together and can be written to enable you to remix and reuse individual*
91 *steps across pipelines. **G. Rerun as necessary:** Workflow systems keep track of which steps executed*
92 *properly and on which samples, and allow you to rerun failed steps (or additional steps) rather than re-*
93 *executing the entire workflow. **H. Reporting:** Workflow languages enable comprehensive reporting on*

94 *workflow execution and resource utilization by each tool. **I. Portability:** Analyses written in workflow*
95 *languages (with integrated software management) can be run across computing systems without changes*
96 *to code.*

97 To properly direct an analysis, workflow systems need to encode information about the relationships
98 between every workflow step. In practice, this means that each analysis step must specify the input (or
99 types of inputs) needed for that step, and the output (or types of outputs) being produced. This structure
100 provides several additional benefits. First, workflows become minimally self-documented, as the directed
101 graph produced by workflow systems can be exported and visualized, producing a graphical
102 representation of the relationships between all steps in a pipeline (see **Figure 5**). Next, workflows are
103 more likely to be fully enclosed without undocumented steps that are executed by hand, meaning analyses
104 are more likely to be reproducible. Finally, each step becomes a self-contained unit that can be used and
105 re-used across multiple analysis workflows, so scientists can spend less time implementing standard steps,
106 and more time on their specific research questions. In sum, the internal scaffolding provided by workflow
107 systems helps build analyses that are generally better documented, repeatable, transferable, and scalable.

108 **Getting started with workflows**

109 The workflow system you choose will be largely dependent on your analysis needs. Here, we draw a
110 distinction between two types of workflows: “research” workflows that are under iterative development to
111 answer novel scientific questions, and “production” workflows, which have reached maturity and are
112 primarily used to run a standard analysis on new samples. In particular, research workflows require
113 flexibility and assessment at every step: outliers and edge cases may reveal interesting biological
114 differences, rather than sample processing or technical errors. Many workflow systems can be used for
115 either type, but we note cases where their properties facilitate one of these types over the other.

116 **Using workflows without learning workflow syntax** While the benefits of executing an analysis within
117 a data-centric workflow system are immense, the learning curve associated with command-line systems

118 can be daunting. It is possible to obtain the benefits of workflow systems without learning new syntax.
119 Websites like Galaxy, Cavatica, and EMBL-EBI MGnify offer online portals in which users build
120 workflows around publicly-available or user-uploaded data [12,13,14]. On the command line, many
121 research groups have used workflow systems to wrap one or many analysis steps (specified in an
122 underlying workflow language) in a more user-friendly command-line application that accepts user input
123 and executes the analysis. These pipeline applications allow users to take advantage of workflow software
124 without needing to write the workflow syntax or manage software installation for each analysis step.
125 Some examples include the nf-core RNA-seq pipeline [1,15], the PiGx genomic analysis toolkit [16], the
126 ATLAS metagenome assembly and binning pipeline [17,18], the Sunbeam metagenome analysis pipeline
127 [19,20], and two from our own lab, the dammit eukaryotic transcriptome annotation pipeline [21] and the
128 elvers *de novo* transcriptome pipeline [22]. These pipeline applications typically execute a series of
129 standard steps, but many provide varying degrees of customizability ranging from tool choice to
130 parameter specification.

131 **Choosing a workflow system** If your use case extends beyond these tools, there are several scriptable
132 workflow systems that offer comparable benefits for carrying out your own data-intensive analyses. Each
133 has its own strengths, meaning each workflow software will meet an individual's computing goals
134 differently (see **Table 1**). Our lab has adopted Snakemake [23], in part due to its integration with Python,
135 its flexibility for building and testing new analyses in different languages, and its intuitive integration
136 with software management tools (described below). Snakemake and Nextflow [25] are commonly used
137 for developing new research pipelines, where flexibility and iterative, branching development is a key
138 feature. Common Workflow Language (CWL) and Workflow Description Language (WDL) are
139 workflow specification formats that are more geared towards scalability, making them ideal for
140 production-level pipelines with hundreds of thousands of samples [26]. WDL and CWL are commonly
141 executed on platforms such as Terra [27] or Seven Bridges Platform [28]. Language-specific workflow
142 systems, such as ROpenSci's Drake [29], can take full advantage of the language's internal data

143 structures, and provide automation and reproducibility benefits for workflows executed primarily within
 144 the language ecosystem.

145 *Table 1: Four of the most widely used bioinformatics workflow systems (2020), with links to*
 146 *documentation, example workflows, and general tutorials. In many cases, there may be tutorials online*
 147 *that are tailored for use cases in your field. All of these systems can interact with tools or tasks written in*
 148 *other languages and can function across cloud computing systems and high-performance computing*
 149 *clusters. Some can also import full workflows from other specification languages.*

Workfl ow Syste m	Documentation	Example Workflow	Tutorial
Snake make	https://snakemake.readthedocs.io/	https://github.com/snakemake-workflows/chipseq	https://snakemake.readthedocs.io/en/stable/tutorial/tutorial.html
Nextflo w	https://www.nextflow.io/	https://github.com/nf-core/sarek	https://www.nextflow.io/docs/latest/getstarted.html
Comm on workflo w langua ge	https://www.commonwl.org/	https://github.com/EBI-Metagenomics/pipeline-v5	https://www.commonwl.org/user_guide/02-1st-example/index.html
Workfl ow descrip tion langua ge	https://openwdl.org/	https://github.com/gatk-workflows/gatk4-data-processing	https://support.terra.bio/hc/en-us/articles/360037127992-1-howto-Write-your-first-WDL-script-running-GATK-HaplotypeCaller

150 The best workflow system to choose may be the one with a strong and accessible local or online
 151 community in your field, somewhat independent of your computational needs. The availability of field-
 152 specific data analysis code for reuse and modification can facilitate the adoption process, as can
 153 community support for new users. Fortunately, the standardized syntax required by workflow systems,
 154 combined with widespread adoption in the open science community, has resulted in a proliferation of

155 open access workflow-system code for routine analysis steps [30,31]. At the same time, consensus
156 approaches for data analysis are emerging, further encouraging reuse of existing code [32,33,34,35,36].
157 The [Getting started developing workflows](#) section contains strategies for modifying and developing
158 workflows for your own analyses.

159 **Wrangling Scientific Software**

160 Analysis workflows commonly rely on multiple software packages to generate final results. These tools
161 are heterogeneous in nature: they are written by researchers working in different coding languages, with
162 varied approaches to software design and optimization, and often for specific analysis goals. Each
163 program has a number of other programs it depends upon to function (“dependencies”), and as software
164 changes over time to meet research needs, the results may change, even when run with identical
165 parameters. As a result, it is critical to take an organized approach to installing, managing, and keeping
166 track of software and software versions. On many compute systems, system-wide software management
167 is overseen by system administrators, who ensure commonly-used and requested software is installed into
168 a “module” system available to all users. Unfortunately, this system limits software version transparency
169 and does not lend itself well to exploring new workflows and software, as researchers do not have
170 permission to install software themselves. To meet this need, most workflow managers integrate with
171 software management systems that handle software installation, management, and packaging, alleviating
172 problems that arise from complex dependencies and facilitating documentation of software versions.
173 Software management systems range from lightweight systems that manage only the software and its
174 dependencies, to heavyweight systems that control for all aspects of the runtime and operating system,
175 ensuring 100% reproducibility of results across computational platforms and time.

176 On the lightweight end, the conda package manager has emerged as a leading software management
177 solution for research workflows (**Figure 2**). Conda handles both cluster permission and version conflict
178 issues with a user-based software environment system, and features a straightforward “recipe” system

179 which simplifies the process of making new software installable (including simple management of
180 versions and updates). These features have led to widespread adoption within the bioinformatics
181 community: packages for new software become quickly available, and can be installed easily across
182 platforms. However, conda does not completely isolate software installations and aims neither for bitwise
183 reproducibility nor long-term archiving of install packages, meaning installations will not be completely
184 reproducible over time. Heavyweight software management systems package not only the software of
185 interest, but also the runtime environment information, with the goal of ensuring perfect reproducibility in
186 software installation over time. Tools such as singularity and docker [3,11,37,38] wrap software
187 environments in “containers” that capture and reproduce the runtime environment information. Container-
188 based management is particularly useful for systems where some dependencies may not be installable by
189 lightweight managers. However, software installation within these containers can be limited by similar
190 reproducibility issues, including changes in dependency installations over time. “Functional package
191 managers” such as GNU Guix and Nix strictly require all dependency and configuration details be
192 encoded within each software package, providing the most comprehensively reproducible installations.
193 These have begun to be integrated into some bioinformatic tools [16], but have a steeper learning curve
194 for independent use. In addition, standard installation of these managers requires system-wide installation
195 permissions, requiring assistance from system administrators on most high-performance computing
196 systems.

197

198 *Figure 2: The conda package and environment manager simplifies software installation and*
199 *management. A. Conda Recipe Repositories: Each program distributed via Conda has a “recipe”*
200 *describing all software dependencies needed for installation using Conda (each of which must also be*
201 *installable via Conda). Recipes are stored and managed in the cloud in separate “channels”, some of*
202 *which specialize in particular fields or languages (e.g. the “bioconda” channel specializes in*
203 *bioinformatic software, while the “conda-forge” channel is a more general effort to provide and maintain*

204 *standardized conda packages for a wide range of software) [11].* **B. Use Conda Environments to Avoid**
205 **Installation Conflicts:** *Conda does not require root privileges for software installation, thus enabling use*
206 *by researchers working on shared cluster systems. However, even user-based software installation can*
207 *encounter dependency conflicts. For example, you might need to use python2 to install and run a*
208 *program (e.g. older scripts written by members of your lab), while also using snakemake to execute your*
209 *workflows (requires python>=3.5). By installing each program into an isolated “environment” that*
210 *contains only the software required to run that program, you can ensure all programs used throughout*
211 *your analysis will run without issue. Using small, separate environments for your software, specifying the*
212 *desired software version, and building many simple environments to accommodate different steps in your*
213 *workflow is critical for reducing the amount of time it takes conda to resolve dependency conflicts*
214 *between different software tools (“solve” an environment). Conda virtual environments can be created*
215 *and installed either on the command line, or via an environment YAML file, as shown. In this case, the*
216 *environment file also specifies which conda channels to search and download programs from. When*
217 *specified in a YAML file, conda environments are easily transferable between computers and operating*
218 *systems. Broad community adoption has resulted in a proliferation of both conda-installable scientific*
219 *software and tools that leverage conda installation specifications. For example, the Mamba package*
220 *manager is an open source reimplementation of the conda manager that can install conda-style*
221 *environments with increased efficiency [39]. The BioContainers Registry is a project that automatically*
222 *builds and distributes docker and singularity containers for bioinformatics software packages using each*
223 *package’s conda installation recipe [40].*

224 **Getting started with software management**

225 **Using software without learning software management systems** First, there are a number of ways to
226 test software before needing to worry about installation. Some software packages are available as web-
227 based tools and through a series of data upload and parameter specifications, allow the user to interact
228 with a tool that is running on a back-end server. Integrated development environments (IDE) like

229 PyCharm and RStudio can manage software installation for language-specific tools, and can be very
230 helpful when writing analysis code. While these approaches do not integrate into reproducible workflows,
231 they may be ideal for testing a tool to determine whether it is useful for your data before integration in
232 your analysis.

233 **Choosing a software management system** It is important to balance the time needed to learn to properly
234 use a software management system with the needs of both the project and the researchers. Software
235 management systems with large learning curves are less likely to be widely adopted among researchers
236 with a mix of biological and computational backgrounds. In our experience, software management with
237 conda nicely balances reproducibility with flexibility and ease of use. These trade-offs are best for
238 research workflows under active development, where flexible software installation solutions that enable
239 new analysis explorations or regular tool updates are critical. For production workflows that require
240 maximal reproducibility, it is worth the larger investment required to use heavyweight systems. This is
241 particularly true for advanced users who can more easily navigate the steps required for utilizing these
242 tools. Container-based software installation via docker and singularity are common for production-level
243 workflows, and Guix and Nix-based solutions are gaining traction. Importantly, the needs and constraints
244 of a project can evolve over time, as may the system of choice.

245 **Integrating software management within workflows** Workflow systems provide seamless integration
246 with a number of software management tools. Each workflow system requires different specification for
247 initiation of software management, but typically requires about one additional line of code per step that
248 requires the use of software. If the software management tool is installed locally, the workflow will
249 automatically download and install the specified environment or container and use it for specified step.

250 In our experience, the complete solution for using scientific software involves a combination of
251 approaches. Interactive and exploratory analyses conducted in IDEs and jupyter notebooks (usually with
252 local software installation with conda) are useful for developing an analysis strategy and creating an

253 initial workflow. This is then followed by workflow-integrated software management via conda,
254 singularity, or nixOS for executing the resulting workflow on many samples. This process not linear: we
255 often cycle between exploratory testing and automation as we iteratively extend our analyses.

256 **Workflow-Based Project Management**

257 Project management, the strategies and decisions used to keep a project organized, documented,
258 functional, and shareable, is foundational to any research program. Clear organization and management is
259 a learned skill that takes time to implement. Workflow systems simplify and improve computational
260 project management, but even workflows that are fully specified in workflow systems require additional
261 investment to stay organized, documented, and backed up.

262 **Systematically document your workflows**

263 Pervasive documentation provides indispensable context for biological insights derived from an analysis,
264 facilitates transparency in research, and increases reusability of the analysis code. Good documentation
265 covers all aspects of a project, including file and results organization, clear and commented code, and
266 accompanying explanatory documents for design decisions and metadata. Workflow systems facilitate
267 building this documentation, as each analysis step (with chosen parameters) and the links between those
268 steps are completely specified within the workflow syntax. This feature streamlines code documentation,
269 particularly if you include as much of the analysis as possible within the automated workflow framework.
270 Outside of the analysis itself, applying consistent organizational design can capitalize on the structure and
271 automation provided by workflows to simplify the generation of quality documentation for all aspects of
272 your project. Below, we discuss project management strategies for building reproducible workflow-
273 enabled biological analyses.

274 **Use consistent, self-documenting names**

275 Using consistent and descriptive identifiers for your files, scripts, variables, workflows, projects, and even
276 manuscripts helps keep your projects organized and interpretable for yourself and collaborators. For
277 workflow systems, this strategy can be implemented by tagging output files with a descriptive identifier
278 for each analysis step, either in the filename or by placing output files within a descriptive output folder.
279 For example, the file shown in **Figure 3** has been preprocessed with a quality control trimming step. For
280 large workflows, placing results from each step of your analysis in isolated, descriptive folders can be
281 essential for keeping your project workspace clean and organized.

282

283 *Figure 3: Consistent and informative file naming improves organization and interpretability. For ease of*
284 *grouping and referring to input files, it is useful to keep unique sample identification in the filename,*
285 *often with a metadata file explaining the meaning of each unique descriptor. For analysis scripts, it can*
286 *help to implement a numbering scheme, where the name of first file in the analysis begins with “00”, the*
287 *next with “01”, etc. For output files, it can help to add a short, unique identifier to output files processed*
288 *with each analysis step. This particular file is a RAD sequencing fastq file of a fish species that has been*
289 *preprocessed with a fastq quality trimming tool.*

290 **Store workflow metadata with the workflow**

291 Developing biological analysis workflows can involve hundreds of small decisions: What parameters
292 work best for each step? Why did you use a certain reference file for annotation as compared with other
293 available files? How did you finally manage to get around the program or installation error? All of these
294 pieces of information contextualize your results and may be helpful when sharing your findings. Keeping
295 information about these decisions in an intuitive and easily accessible place helps you find it when you
296 need it. To capitalize on the utility of version control systems described below, it is most useful to store
297 this information in plain text files. Each main directory of a project should include notes on the data or

298 scripts contained within, so that a collaborator could look into the directory and understand what to find
299 there (especially since that “collaborator” is likely to be you, a few months from now!). Code itself can
300 contain documentation - you can include comments with the reasoning behind algorithm choice or include
301 a link to online documentation or solution that helped you decide how to shape your differential
302 expression analysis. Larger pieces of information can be kept in “README” or notes documents kept
303 alongside your code and other documents. For example, a GitHub repository documenting the reanalysis
304 of the Marine Microbial Eukaryote Transcriptome Sequencing Project uses a README alongside the
305 code to document the workflow and digital object identifiers for data products [41,42]. While this
306 particular strategy cannot be automated, it is critical for interpreting the final results of your workflow.

307 **Document data and analysis exploration using computational notebooks**

308 Computational notebooks allow users to combine narrative, code, and code output (e.g. visualizations) in
309 a single location, enabling the user to conduct analysis and visually assess the results in a single file (see
310 **Figure 4**). These notebooks allow for fully documented iterative analysis development, and are
311 particularly useful for data exploration and developing visualizations prior to integration into a workflow
312 or as a report generated by a workflow that can be shared with collaborators.

313

314 *Figure 4: Examples of computational notebooks. Computational notebooks allow the user to mix text,*
315 *code, and results in one document. **Panel A.** shows an RMarkdown document viewed in the RStudio*
316 *integrated development environment, while **Panel B.** shows a rendered HTML file produced by knitting*
317 *the RMarkdown document [43]. **Panel C.** shows a Jupyter Notebook, where code, text, and results are*
318 *rendered inline as each code chunk is executed [44]. The second grey chunk is a raw Markdown chunk*
319 *with text that will be rendered inline when executed. Both notebooks generate a histogram of a metadata*
320 *feature, number of generations, from a long-term evolution experiment with Escherichia coli [45].*
321 *Computational notebooks facilitate sharing by packaging narrative, code, and visualizations together.*

322 *Sharing can be enhanced further by packaging computational notebooks with tools like Binder [46].*
323 *Binder builds an executable environment (capable of running RStudio and Jupyter notebooks) out of a*
324 *GitHub repository using package management systems and docker to build reproducible and executable*
325 *software environments as specified in the repository. Binders can be shared with collaborators (or*
326 *students in a classroom setting), and analysis and visualization can be ephemerally reproduced or altered*
327 *from the code provided in computational notebooks.*

328

329 **Visualize your workflow**

330 Visual representations can help illustrate the connections in a workflow and improve the readability and
331 reproducibility of your project. At the highest level, flowcharts that detail relationships between steps of a
332 workflow can help provide big-picture clarification, especially when the pipeline is complicated. For
333 individual steps, a graphical representation of the output can show the status of the project or provide
334 insight on additional analyses that should be added. For example, **Figure 5** exhibits a modified
335 Snakemake workflow visualization from an RNA-seq quantification pipeline [47].

336

337 *Figure 5: A directed acyclic graph (DAG) that illustrates connections between all steps of a sequencing*
338 *data analysis workflow. Each box represents a step in the workflow, while lines connect sequential steps.*
339 *The DAG shown in this figure illustrates a real bioinformatics workflow for RNA-seq quantification was*
340 *generated by modifying the default Snakemake workflow DAG. This example of an initial workflow used*
341 *only to quality control and then quantify one FASTQ file against a transcriptome more than doubles the*
342 *amount of files in a project. When the number of steps are expanded to carry out a full research analysis*
343 *and the number of initial input files are increased, a workflow can generate hundreds to thousands of*
344 *intermediate files. Fortunately, workflow system coordination alleviates the need for a user to directly*
345 *manage file interdependencies. For a larger analysis DAG, see [48]*

346 **Version control your project**

347 As your project develops, version control allows you to keep track of changes over time. You may
348 already do this in some ways, perhaps with frequent hard drive backups or by manually saving different
349 versions of the same file - e.g. by appending the date to a script name or appending “version_1” or
350 “version_FINAL” to a manuscript draft. For computational workflows, using version control systems
351 such as Git or Mercurial can be used to keep track of all changes over time, even across multiple systems,
352 scripting languages, and project contributors (see **Figure 6**). If a key piece of a workflow inexplicably
353 stops working, consistent version control can allow you to rewind in time and identify differences from
354 when the pipeline worked to when it stopped working. Backing up your version controlled analysis in an
355 online repository such as GitHub, GitLab, or Bitbucket provides critical insurance as you iteratively
356 modify and develop your workflow.

357

358 *Figure 6: **Version Control** Version control systems (e.g. Git, Mercurial) work by storing incremental*
359 *differences in files from one saved version (“commit”) to the next. To visualize the differences between*
360 *each version, text editors such as Atom and online services such as GitHub, GitLab and Bitbucket use red*
361 *highlighting to denote deletions, and green highlighting to denote additions. In this trivial example, a*
362 *typo in version 1 (in red) was corrected in version 2 (in green). These systems are extremely useful for*
363 *code and manuscript development, as it is possible to return to the snapshot of any saved version. This*
364 *means that version control systems save you from accidental deletions, preserve code you thought you no*
365 *longer needed and preserve a record of project changes over time.*

366 When combined with online backups, version control systems also facilitate code and data availability
367 and reproducibility for publication. For example, to preserve the version of code that produced published
368 results, you can create a “release”: a snapshot of the current code and files in a GitHub repository. You

369 can then generate a digital object identifier (DOI) for that release using a permanent documentation
370 service such as Zenodo ([49]) and make it available to reviewers and beyond (see “sharing” section,
371 below).

372 **Share your workflow and analysis code**

373 Sharing your workflow code with collaborators, peer reviewers, and scientists seeking to use a similar
374 method can foster discussion and review of your analysis. Sticking to a clear documentation strategy,
375 using a version control system, and packaging your code in notebooks or as a workflow prepare them to
376 be easily shared with others. To go one step further, you can package your code with tools like Binder,
377 ReproZip, or Whole Tale, or make interactive visualizations with tools like Shiny apps or Plotly. These
378 approaches let others run the code on cloud computers in environments identical to those in which the
379 original computation was performed (**Figure 4, Figure 7**) [46,50,51]. These tools substantially reduce
380 overhead associated with interacting with code and data, and in doing so, make it fast and easy to rerun
381 portions of the analysis, check accuracy, or even tweak the analysis to produce new results. If you also
382 share your code and workflows publicly, you will also help contribute to the growing resources for open
383 workflow-enabled biological research.

384

385 *Figure 7: Interactive visualizations facilitate sharing and repeatability. A. Interactive visualization*
386 *dashboard in the Pavian Shiny app for metagenomic analysis [52,53]. Shiny allows you to build*
387 *interactive web pages using R code. Data is manipulated by R code in real-time in a web page, producing*
388 *analysis and visualizations of a data set. Shiny apps can contain user-specifiable parameters, allowing a*
389 *user to control visualizations or analyses. As seen above, sample “PT1” is selected, and taxonomic ranks*
390 *class and order are excluded. Shiny apps allow collaborators who may or may not know R to modify R*
391 *visualizations to fit their interests. B. Plotly heatmap of transcriptional profiling in human brain samples*

392 [\[54\]](#). Hovering over a cell in the heatmap displays the sample names from the x and y axis, as well as the
393 intensity value. Plotting tools like plotly and vega-lite produce single interactive plots that can be shared
394 with collaborators or integrated into websites [\[55,56\]](#). Interactive visualizations are also helpful in
395 exploratory data analysis.

396 **Getting started developing workflows**

397 In our experience, the best way to have your workflow system work *for* you is to include as much of your
398 analysis as possible within the automated workflow framework, use self-documenting names, include
399 analysis visualizations, and keep rigorous documentation alongside your workflow that enables you to
400 understand each decision and entirely reproduce any manual steps. Some of the tools discussed above will
401 inevitably change over time, but these principles apply broadly and will help you design clear, well-
402 documented, and reproducible analyses. Ultimately, you will need to experiment with strategies that work
403 for you – what is most important is to develop a clear set of strategies and implement them tenaciously.
404 Below, we provide a few practical strategies to try as you begin developing your own workflows.

405 **Start with working code** When building a workflow for the first time, start from working examples
406 provided as part of the tool documentation or otherwise available online. This functioning example code
407 then provides a reliable workflow framework free of syntax errors which you can customize for your data
408 without the overhead of generating correct workflow syntax from scratch. Be sure to run this analysis on
409 provided test data, if available, to ensure the tools, and command line syntax function at a basic level.

410 **Table 1** provides links to official repositories containing tutorials and example biological analysis
411 workflows, and workflow tutorials and code sharing websites like GitHub, GitLab, and Bitbucket have
412 many publicly available workflows for other analyses. If a workflow is available through Binder, you can
413 test and experiment with workflow modification on Binder’s cloud system without needing to install a
414 workflow manager or software management tool on your local compute system [\[46\]](#).

415 **Test with subsampled data** Once you have working workflow syntax, test the step on your own data or
416 public data related to your species or condition of interest. First, create a subsampled dataset that you can
417 use to test your entire analysis workflow. This set will save time, energy, and computational resources
418 throughout workflow development. If working with FASTQ data, a straightforward way to generate a
419 small test set is to subsample the first million lines of a file (first 250k reads):

```
420 head -n 1000000 FASTQ_FILE.fq > test_fastq.fq
```

421 While there are many more sophisticated ways to subsample reads, this technique should be sufficient for
422 testing each step of a most workflows prior to running your full dataset. In specific cases, such as
423 eukaryotic genome assembly, you may need to be more intentional with how you subsample reads and
424 how much sample data you use as a test set.

425 **Document your process** Document your changes, explorations, and errors as you develop. We
426 recommend using the Markdown language so your documentation is in plain text (to facilitate version
427 control), but can still include helpful visual headings, code formatting, and embedded images. Markdown
428 editors with visual previewing, such as HackMD, can greatly facilitate notetaking, and Markdown
429 documents are visually rendered properly within your online version control backups on services such as
430 GitHub [\[57\]](#).

431 **Develop your workflow** From your working code, iteratively modify and add workflow steps to meet
432 your data analysis needs. This strategy allows you to find and fix mistakes on small sections of the
433 workflow. Periodically clean your output directory and rerun the entire workflow, to ensure all steps are
434 fully interoperable (using small test data will improve the efficiency of this step!). If possible, using mock
435 or control datasets can help you verify that the analysis you are building actually returns correct biological
436 results. Tutorials and tool documentation are useful companions during development; as with any
437 language, remembering workflow-specific syntax takes time and practice.

438 **Assess your results** Evaluate your workflow results as you go. Consider what aspects (e.g. tool choice,
439 program parameters) can be evaluated rigorously, and assess each step for expected behavior. Other
440 aspects (e.g. filtering metadata, joining results across programs or analysis, software and workflow bugs)
441 will be more difficult to evaluate. Wherever possible, set up positive and negative controls to ensure your
442 analysis is performing the desired analysis properly. Once you're certain an analysis is executing as
443 designed, tracking down unusual results may reveal interesting biological differences.

444 **Back up early and often** As you write new code, back up your changes in an online repository such as
445 GitHub, GitLab, or Bitbucket. These services support both drag-and-drop and command line interaction.

446 **Scale up your workflow** Bioinformatic tools vary in the resources they require: some analysis steps are
447 compute-intensive, other steps are memory intensive, and still others will have large intermediate storage
448 needs. If using high-performance computing system or the cloud, you will need to request resources for
449 running your pipeline, often provided as a simultaneous execution limit or purchased by your research
450 group on a cost-per-compute basis. Workflow systems provide built-in tools to monitor resource usage for
451 each step. Running a complete workflow on a single sample with resource monitoring enabled generates
452 an estimate of computational resources needed for each step. These estimates can be used to set
453 appropriate resource limits for each step when executing the workflow on your remaining samples.

454 **Find a community and ask for help when you need it** Local and online users groups are helpful
455 communities when learning a workflow language. When you are first learning, help from more advanced
456 users can save you hours of frustration. After you've progressed, providing that same help to new users
457 can help you cement the syntax in your mind and tackle more advanced uses. Data-centric workflow
458 systems have been enthusiastically adopted by the open science community, and as a consequence, there
459 is a critical mass of tutorials and open access code, as well as code discussion on forums and via social
460 media, particularly Twitter. Post in the relevant workflow forums when you have hit a stopping point you

461 are unable to work through. Be respectful of people's time and energy and be sure to include appropriate
462 details important to your problem (see [Strategic troubleshooting](#) section).

463 **Data and resource management for workflow-enabled** 464 **biology**

465 Advancements in sequencing technologies have greatly increased the volume of data available for
466 biological query [58]. Workflow systems, by virtue of automating many of the time-intensive project
467 management steps traditionally required for data-intensive biology, can increase our capacity for data
468 analysis. However, conducting biological analyses at this scale requires a coordinated approach to data
469 and computational resource management. Below, we provide recommendations for data acquisition,
470 management, and quality control that have become especially important as the volume of data has
471 increased. Finally, we discuss securing and managing appropriate computational resources for the scale of
472 your project.

473 **Managing large-scale datasets**

474 Experimental design, finding or generating data, and quality control are quintessential parts of data
475 intensive biology. There is no substitute for taking the time to properly design your analysis, identify
476 appropriate data, and conduct sanity checks on your files. While these tasks are not automatable, many
477 tools and databases can aid in these processes.

478 **Look for appropriate publicly-available data**

479 With vast amounts of sequencing data already available in public repositories, it is often possible to begin
480 investigating your research question by seeking out publicly available data. In some cases, these data will
481 be sufficient to conduct your entire analysis. In others cases, particularly for biologists conducting novel

482 experiments, these data can inform decisions about sequencing type, depth, and replication, and can help
483 uncover potential pitfalls before they cost valuable time and resources.

484 Most journals now require data for all manuscripts to be made accessible, either at publication or after a
485 short moratorium. Further, the FAIR (findable, accessible, interoperable, reusable) data movement has
486 improved the data sharing ecosystem for data-intensive biology [[59](#),[60](#),[61](#),[62](#),[63](#),[64](#),[64](#),[65](#)]. You can find
487 relevant sequencing data either by starting from the “data accessibility” sections of papers relevant to
488 your research or by directly searching for your organism, environment, or treatment of choice in public
489 data portals and repositories. The International Nucleotide Sequence Database Collaboration (INSDC),
490 which includes the Sequence Read Archive (SRA), European Nucleotide Archive (ENA), and DataBank
491 of Japan (DDBJ) is the largest repository for raw sequencing data, but no longer accepts sequencing data
492 from large consortia projects [[66](#)]. These data are instead hosted in consortia-specific databases, which
493 may require some domain-specific knowledge for identifying relevant datasets and have unique download
494 and authentication protocols. For example, raw data from the Tara Oceans expedition is hosted by the
495 Tara Ocean Foundation [[67](#)]. Additional curated databases focus on processed data instead, such as gene
496 expression in the Gene Expression Omnibus (GEO) [[68](#)]. Organism-specific databases such as
497 **Wormbase** (*Caenorhabditis elegans*) specialize on curating and integrating sequencing and other data
498 associated with a model organism [[69](#)]. Finally, rather than focusing on certain data types or organisms,
499 some repositories are designed to hold any data and metadata associated with a specific project or
500 manuscript (e.g. Open Science Framework, Dryad, Zenodo [[70](#)]).

501 **Consider analysis when generating your own data**

502 If generating your own data, proper experimental design and planning are essential. For cost-intensive
503 sequencing data, there are a range of decisions about experimental design and sequencing (including
504 sequencing type, sequencing depth per sample, and biological replication) that impact your ability to
505 properly address your research question. Conducting discussions with experienced bioinformaticians and

506 statisticians, *prior to beginning your experiments* if possible, is the best way to ensure you will have
 507 sufficient statistical power to detect effects. These considerations will be different for different types of
 508 sequence analysis. To aid in early project planning, we have curated a series of domain-specific
 509 references that may be useful as you go about designing your experiment (see **Table 2**). Given the
 510 resources invested in collecting samples for sequencing, it's important to build in a buffer to preserve
 511 your experimental design in the face of unexpected laboratory or technical issues. Once generated, it is
 512 always a good idea to have multiple independent backups of raw sequencing data, as it typically cannot be
 513 easily regenerated if lost to computer failure or other unforeseeable events.

514 *Table 2: References for experimental design and considerations for common sequencing chemistries.*

Sequencing type	Resources
RNA-sequencing	[32 , 71 , 72]
Metagenomic sequencing	[33 , 73 , 74]
Amplicon sequencing	[75 , 76 , 77]
Microbial isolate sequencing	[78]
Eukaryotic genome sequencing	[79 , 80 , 81 , 82]
Whole-genome resequencing	[83]
RAD-sequencing	[84 , 84 , 85 , 86 , 87 , 88]
single cell RNA-seq	[89 , 90]

515 As your experiment progresses, keep track of as much information as possible: dates and times of sample
 516 collection, storage, and extraction, sample names, aberrations that occurred during collection, kit lot used
 517 for extraction, and any other sample and sequencing measurements you might be able to obtain
 518 (temperature, location, metabolite concentration, name of collector, well number, plate number, machine
 519 your data was sequenced, on etc). This metadata allows you to keep track of your samples, to control for
 520 batch effects that may arise from unintended batching during sampling or experimental procedures and
 521 makes the data you collect reusable for future applications and analysis by yourself and others. Wherever
 522 possible, follow the standard guidelines for formatting metadata for scientific computing to limit

523 downstream processing and simplify analyses requiring these metadata (see: [\[10\]](#)). We have focused here
524 on sequencing data; for data management over long-term ecological studies, we recommend [\[91\]](#).

525 **Getting started with sequencing data**

526 **Protect valuable data**

527 Aside from the code itself, raw data are the most important files associated with a workflow, as they
528 cannot be regenerated if accidentally altered or deleted. Keeping a read-only copy of raw data alongside a
529 workflow as well multiple backups protects your data from accidents and computer failure. This also
530 removes the imperative of storing intermediate files as these can be easily regenerated by the workflow.

531 When sharing or storing files and results, data version control can keep track of differences in files such
532 as changes from tool parameters or versions. The version control tools discussed in the [Workflow-based](#)
533 [project management](#) section are primarily designed to handle small files, but GitHub provides support for
534 Git Large File Storage (LFS), and repositories such as the Open Science Framework (OSF), Figshare,
535 Zenodo, and Dryad can be used for storing larger files and datasets [\[49,70,92,93,94\]](#).

536 In addition to providing version control for projects and datasets, these tools also facilitate sharing and
537 attribution by enabling generation of digital object identifiers (doi) for datasets, figures, presentations,
538 code, and preprints. As free tools often limit the size of files that can be stored, a number of cloud backup
539 and storage services are also available for purchase or via university contract, including Google Drive,
540 Box, Dropbox, Amazon Web Services, and Backblaze. Full computer backups can be conducted to these
541 storage locations with tools like rclone [\[95\]](#).

542 **Ensure data integrity during transfers**

543 If you're working with publicly-available data, you may be able to work on a compute system where the
544 data are already available, circumventing time and effort required for downloading and moving the data.

545 Databases such as the Sequence Read Archive (SRA) are now available on commercial cloud computing
546 systems, and open source projects such as Galaxy enable working with SRA sequence files directly from
547 a web browser [12,96]. Ongoing projects such as the NIH Common Fund Data Ecosystem aim to develop
548 a data portal to make NIH Common Fund data, including biomedical sequencing data, more findable,
549 accessible, interoperable, and reusable (FAIR).

550 In most cases, you'll still need to transfer some data - either downloading raw data or transferring
551 important intermediate and results files for backup and sharing (or both). Transferring compressed files
552 (gzip, bzip2, BAM/CRAM, etc.) can improve transfer speed and save space, and checksums can be used
553 to to ensure file integrity after transfer (see **Figure 8**).

554

555 *Figure 8: Use Checksums to ensure file integrity* Checksum programs (e.g. md5, sha256) encode file size
556 and content in a single value known as a “checksum”. For any given file, this value will be identical
557 across platforms when calculated using the same checksum program. When transferring files, calculate
558 the value of the checksum prior to transfer, and then again after transfer. If the value is not identical,
559 there was an error introduced during transfer (e.g. file truncation, etc). Checksums are often provided
560 alongside publicly available files, so that you can verify proper download. Tools like rsync and rclone
561 that automate file transfers use checksums internally to verify that files were transferred properly, and
562 some GUI file transfer tools (e.g. Cyberduck) can assess checksums when they are provided [95]. If you
563 generated your own data and received sequencing files from a sequencing center, be certain you also
564 receive a checksum for each of your files to ensure they download properly.

565 **Perform quality control at every step**

566 The quality of your input data has a major impact on the quality of the output results, no matter whether
567 your workflow analyzes six samples or six hundred. Assessing data at every analysis step can reveal

568 problems and errors early, before they waste valuable time and resources. Using quality control tools that
 569 provide metrics and visualizations can help you assess your datasets, particularly as the size of your input
 570 data scales up. However, data from different species or sequencing types can produce anomalous quality
 571 control results. You are ultimately the single most effective quality control tool that you have, so it is
 572 important to critically assess each metric to determine those that are relevant for your particular data.

573 **Look at your files** Quality control can be as simple as looking at the first few and last few lines of input
 574 and output data files, or checking the size of those files (see **Table 3**). To develop an intuition for what
 575 proper inputs and outputs look like for a given tool, it is often helpful to first run the test example or data
 576 that is packaged with the software. Comparing these input and output file formats to your own data can
 577 help identify and address inconsistencies.

578 *Table 3: Some commands to quickly explore the contents of a file. These commands can be used on Unix*
 579 *and Linux operating systems to detect common formatting problems or other abnormalities.*

command	function	example
ls -lh	list files with information in a human-readable format	ls -lh *fastq.gz
head	print the first 6 lines of a file to standard out	head samples.csv
tail	print the last 6 lines of a file to standard out	tail samples.csv
less	show the contents of a file in a scrollable screen	less samples.csv
zless	show the contents of a gzipped file in a scrollable screen	zless sample1.fastq.gz
wc -l	count the number of lines in a file	wc -l ecoli.fasta
cat	print a file to standard out	cat samples.csv
grep	find matching text and print the line to standard out	grep ">" ecoli.fasta
cut	cut columns from a table	cut -d"," -f1 samples.csv

580 **Visualize your data** Visualization is another powerful way to pick out unusual or unexpected patterns.
 581 Although large abnormalities may be clear from looking at files, others may be small and difficult to find.
 582 Visualizing raw sequencing data with FastQC (**Figure 9A**) and processed sequencing data with tools like
 583 the Integrative Genome Viewer and plotting tabular results files using python or R can make aberrant or
 584 inconsistent results easier to track down [[98,99](#)].

585

586 *Figure 9: Visualizations produced by MultiQC. MultiQC finds and automatically parses log files from*
587 *other tools and generates a combined report and parsed data tables that include all samples. MultiQC*
588 *currently supports 88 tools. A. MultiQC summary of FastQC Per Sequence GC Content for 1905*
589 *metagenome samples. FastQC provides quality control measurements and visualizations for raw*
590 *sequencing data from a single sample, and is a near-universal first step in sequencing data analysis*
591 *because of the insights it provides [98,99]. FastQC measures and summarizes 10 quality metrics and*
592 *provides recommendations for whether an individual sample is within an acceptable quality range.*
593 *Not all metrics readily apply to all sequencing data types. For example, while multiple GC peaks might*
594 *be concerning in whole genome sequencing of a bacterial isolate, we would expect a non-normal*
595 *distribution for some metagenome samples that contain organisms with diverse GC content. Samples like*
596 *this can be seen in red in this figure. B. MultiQC summary of Salmon quant reads mapped per sample for*
597 *RNA-seq samples [100]. In this figure, we see that MultiQC summarizes the number of reads mapped and*
598 *percent of reads mapped, two values that are reported in the Salmon log files.*

599 **Pay attention to warnings and log files** Many tools generate log files or messages while running. These
600 files contain information about the quantity, quality, and results from the run, or error messages about
601 why a run failed. Inspecting these files can be helpful to make sure tools ran properly and consistently, or
602 to debug failed runs. Parsing and visualizing log files with a tool like MultiQC can improve
603 interpretability of program-specific log files (**Figure 9** [101]).

604 **Look for common biases in sequencing data** Biases in sequencing data originate from experimental
605 design, methodology, sequencing chemistry, or workflows, and are helpful to target specifically with
606 quality control measures. The exact biases in a specific data set or workflow will vary greatly between
607 experiments so it is important to understand the sequencing method you have chosen and incorporate
608 appropriate filtration steps into your workflow. For example, PCR duplicates can cause problems in

609 libraries that underwent an amplification step, and often need to be removed prior to downstream analysis
610 [\[102,103,104,105,106\]](#).

611 **Check for contamination** Contamination can arise during sample collection, nucleotide extraction,
612 library preparation, or through sequencing spike-ins like PhiX, and could change data interpretation if not
613 removed [\[107,108,109\]](#). Libraries sequenced with high concentrations of free adapters or with low
614 concentration samples may have increased barcode hopping, leading to contamination between samples
615 [\[110\]](#).

616 **Consider the costs and benefits of stringent quality control for your data** Good quality data is
617 essential for good downstream analysis. However, stringent quality control can sometimes do more harm
618 than good. For example, depending on sequencing depth, stringent quality trimming of RNA-sequencing
619 data may reduce isoform discovery [\[111\]](#). To determine what issues are most likely to plague your
620 specific data set, it can be helpful to find recent publications using a similar experimental design, or to
621 speak with experts at a sequencing core.

622 Because sequencing data and applications are so diverse, there is no one-size-fits-all solution for quality
623 control. It is important to think critically about the patterns you expect to see given your data and your
624 biological problem, and consult with technical experts whenever possible.

625 **Securing and managing appropriate computational resources**

626 Sequence analysis requires access to computing systems with adequate storage and analysis power for
627 your data. For some smaller-scale datasets, local desktop or even laptop systems can be sufficient,
628 especially if using tools that implement data-reduction strategies such as minhashing [\[112\]](#). However,
629 larger projects require additional computing power, or may be restricted to certain operating systems
630 (e.g. linux). For these projects, solutions range from research-focused high performance computing
631 systems to research-integrated commercial analysis platforms. Both research-only and commercial

632 clusters provide avenues for research and educational proposals to enable access to their computing
 633 resources (see **Table 4**). In preparing for data analysis, be sure to allocate sufficient computational
 634 resources and funding for storage and analysis, including large intermediate files and resources required
 635 for personnel training. Note that workflow systems can greatly facilitate faithful execution of your
 636 analysis across the range of computational resources available to you, including distribution across cloud
 637 computing systems.

638 *Table 4: **Computing Resources** Bioinformatic projects often require additional computing resources. If a*
 639 *local or university-run high-performance computing cluster is not available, computing resources are*
 640 *available via a number of grant-based or commercial providers.*

Provider	Access Model	Restrictions
Amazon Web Services	Paid	
Bionimbus Protected Data Cloud	Research allocation	users with eRA commons account
Cyverse Atmosphere	Free with limits	storage and compute hours
EGI federated cloud	Access by contact	European partner countries
Galaxy	Free with storage limits	data storage limits
Google Cloud Platform	Paid	
Google Colab	Free	computational notebooks, no resource guarantees
Microsoft Azure	Paid	
NSF XSEDE	Research allocation	USA researchers or collaborators
Open Science Data Cloud	Research allocation	
Wasabi	Paid	data storage solution only

641

642 **Getting started with resource management**

643 As the scale of data increases, the resources required for analysis can balloon. Bioinformatic workflows
 644 can be long-running, require high-memory systems, or involve intensive file manipulation. Some of the
 645 strategies below may help you manage computational resources for your project.

646 **Apply for research units if eligible** There are a number of cloud computing services that offer grants
647 providing computing resources to data-intensive researchers (**Table 4**). In some cases, the resources
648 provided may be sufficient to cover your entire analysis.

649 **Develop on a local computer when possible** Since workflows transfer easily across systems, it can be
650 useful to develop individual analysis steps on a local laptop. If the analysis tool will run on your local
651 system, test the step with subsampled data, such as that created in the [Getting started developing](#)
652 [workflows](#) section. Once working, the new workflow component can be run at scale on a larger
653 computing system. Workflow system tool resource usage reporting can help determine the increased
654 resources needed to execute the workflow on larger systems. For researchers without access to free or
655 granted computing resources, this strategy can save significant cost.

656 **Gain quick insights using sketching algorithms** Understanding the basic structure of data, the
657 relationship between samples, and the approximate composition of each sample can very helpful at the
658 beginning of data analysis, and can often drive analysis decisions in different directions than those
659 originally intended. Although most bioinformatics workflows generate these types of insights, there are a
660 few tools that do so rapidly, allowing the user to generate quick hypotheses that can be further tested by
661 more extensive, fine-grained analyses. Sketching algorithms work with compressed approximate
662 representations of sequencing data and thereby reduce runtimes and computational resources. These
663 approximate representations retain enough information about the original sequence to recapitulate the
664 main findings from many exact but computationally intensive workflows. Most sketching algorithms
665 estimate sequence similarity in some way, allowing you to gain insights from these comparisons. For
666 example, sketching algorithms can be used to estimate all-by-all sample similarity which can be
667 visualized as a Principal Component Analysis or a multidimensional scaling plot, or can be used to build
668 a phylogenetic tree with accurate topology. Sketching algorithms also dramatically reduce the runtime for
669 comparisons against databases (e.g. all of GenBank), allowing users to quickly compare their data against
670 large public databases.

671 Rowe 2019 [\[113\]](#) reviewed programs and genomic use cases for sketching algorithms, and provided a
672 series of tutorial workbooks (e.g. Sample QC notebook: [\[114\]](#)).

673 **Use the right tools for your question** RNA-seq analysis approaches like differential expression or
674 transcript clustering rely on transcript or gene counts. Many tools can be used to generate these counts by
675 quantifying the number of reads that overlap with each transcript or gene. For example, tools like STAR
676 and HISAT2 produce alignments that can be post-processed to generate per-transcript read counts
677 [\[115,116\]](#). However, these tools generate information-rich output, specifying per-base alignments for
678 each read. If you are only interested in read quantification, quasi-mapping tools provide the desired
679 results while reducing the time and resources needed to generate and store read count information
680 [\[117,118\]](#).

681 **Seek help when you need it** In some cases, you may find that your accessible computing system is ill-
682 equipped to handle the type or scope of your analysis. Depending on the system, staff members may be
683 able to help direct you to properly scale your workflow to available resources, or guide you in tailoring
684 computational unit allocations or purchases to match your needs.

685 **Strategies for troubleshooting**

686 Workflows, and research software in general, invariably require troubleshooting and iteration. When first
687 starting with a workflow system, it can be difficult to interpret code and usage errors from unfamiliar
688 tools or languages [\[2\]](#). Further, the iterative development process of research software means
689 functionality may change, new features may be added, or documentation may be out of date [\[119\]](#). The
690 challenges of learning and interacting with research software require time and patience [\[4\]](#).

691 One of the largest barriers to surmounting these challenges is learning how, when, and where to ask for
692 help. Below we outline a strategy for troubleshooting that can help build your own knowledge while
693 respecting both your own time and that of research software developers and the larger bioinformatic

694 community. In the “where to seek help” section, we also recommend locations for asking general
695 questions around data-intensive analysis, including discussion of tool choice, parameter selection, and
696 other analysis strategies. Beyond these tips, workshops and materials from training organizations such as
697 the Carpentries, R-Ladies, RStudio can arm you with the tools you need to start troubleshooting and
698 jump-start software and data literacy in your community [[120](#)]. Getting involved with these workshops
699 and communities not only provides educational benefits but also networking and career-building
700 opportunities.

701 **How to help yourself: Try to pinpoint your issue or error**

702 Software errors can be the result of syntax errors, dependency issues, operating system conflicts, bugs in
703 the software, problems with the input data, and many other issues. Running the software on the provided
704 test data can help narrow the scope of error sources: if the test data successfully runs, the command is
705 likely free of syntax errors, the source code is functioning, and the tool is likely interacting appropriately
706 with dependencies and the operating system. If the test data runs but the tool still produces an error when
707 run with your data and parameters, the error message can be helpful in discovering the cause of the error.
708 In many cases, the error you’ve encountered has been encountered many times before, and searching for
709 the error online can turn up a working solution. If there is a software issue tracker for the software (e.g. on
710 the GitHub, GitLab, or Bitbucket repository), or a Gitter, Slack, or Google Groups page, performing a
711 targeted search with the error message may provide additional context or a solution for the error. If
712 targeted searches do not return a results, Googling the error message with the program name is a good
713 next step. Searching with several variants and iteratively adding information such as the type of input
714 data, the name of the coding language or computational platform, or other relevant information, can
715 improve the likelihood that a there will be a match. There are a vast array of online resources for
716 bioinformatic help ranging from question sites such as Stack Overflow and BioStars, to personal or

717 academic blogs and even tutorials and lessons written by experts in the field [[121](#)]. This increases the
718 discoverability of error messages and their solutions.

719 Sometimes, programs fail without outputting an error message. In cases like these, the software's help
720 (usually accessible on the command line via `tool-name --help`) and official documentation may provide
721 clues or additional example use cases that may be helpful in resolving an error. Syntax errors are
722 extremely common, and typos as small as a single, misplaced character or amount of whitespace can
723 affect the code. If a command matches the documentation and appears syntactically correct, the software
724 version (often accessible at the command line `tool-name --version`) may be causing the error.
725 Best practices for software development follow “semantic versioning” principles, which aim to keep the
726 arguments and functionality the same for all minor releases of the program (e.g. 1.1 to 1.2) and only
727 change functions with major releases (e.g. 1.x to 2.0).

728 **How to seek help: include the right details with your question**

729 When searching for the error message and reading the documentation do not resolve an error, it is usually
730 appropriate to for seek help either from the software developers or from a bioinformatics community.
731 When asking for help, it's essential to provide the right details so that other users and developers can
732 understand the exact conditions that produced the error. At minimum, include the name and version of the
733 program, the method used to install it, whether or not the test data ran, the exact code that produced the
734 error, the error message, and the full output text from the run (if any is produced). The type and version of
735 the operating system you are using is also helpful to include. Sometimes, this is enough information for
736 others to spot the error. However, if it appears that there may bug in the underlying code, specifying or
737 providing the minimum amount of data required to reproduce the error (e.g. reproducible example
738 [[122,123](#)]) enables other to reproduce and potentially solve the error at hand. Putting the effort into
739 gathering this information both increases your own understanding of the problem and makes it easier and

740 faster for others to help solve your issue. Furthermore, it signals respect for the time that these developers
741 and community members dedicate to helping troubleshoot and solve user issues.

742 **Where to seek help: online and local communities of practice**

743 Online communities and forums are a rich source of archived bioinformatics errors with many helpful
744 community members. For errors with specific programs, often the best place to post is the developers'
745 preferred location for answering questions and solving errors related to their program. For open source
746 programs on GitHub, GitLab, or Bitbucket, this is often the "Issues" tab within the software repository,
747 but it could alternatively be a Google groups list, gitter page, or other specified forum. Usually, the
748 documentation indicates the best location questions. If question is more general, such as asking about
749 program choice or workflows, forums relevant to your field such as Stack Overflow, BioStars, or
750 SEQanswers are good choices, as posts here are often seen by a large community of researchers. Before
751 posting, search through related topics to double check the question has not already been answered. As
752 more research software development and troubleshooting is happening openly in online repositories, it is
753 becoming more important than ever to follow a code of conduct that promotes open and harassment-free
754 discussion environment [[124](#)]. Look for codes of conduct in the online forums you participate in, and
755 make sure you do your part to help ensure a welcoming community for participants of all backgrounds
756 and computational competencies.

757 While there is lots of help available online, there is no substitute for local communities. Local
758 communities may come in the form of a tech meetup, a users group, a hacky hour, or an informal meetup
759 of researchers using similar tools. While this may seem like just a local version of Stack Overflow, the
760 local, member-only nature can help create a safe and collaborative online space for troubleshooting
761 problems often encountered by your local bioinformatics community. The benefit to beginners is clear:
762 learning the best way to post questions and the important parts of errors, while getting questions answered
763 so they can move forward in their research. Intermediate users may actually find these communities most

764 useful, as they can also accelerate their own troubleshooting skills by helping others solve issues that they
765 have already struggled through. While it can be helpful to have some experts available to help answer
766 questions or to know when to escalate to Stack Overflow or other communities, a collaborative
767 community of practice with members at all experience levels can help all its members move their science
768 forward faster.

769 If such a community does not yet exist in your area, building this sort of community (discussed in detail
770 in [\[125\]](#)), can be as simple as hosting a seminar series or starting meetup sessions for data analysis co-
771 working. In our experience, it can also be useful to set up a local online forum (e.g. discourse) for group
772 troubleshooting.

773 **Conclusion**

774 Bioinformatics-focused workflow systems have reshaped data-intensive biology, reducing execution
775 hurdles and empowering biologists to conduct reproducible analyses at the massive scale of data now
776 available. Shared, interoperable research code is enabling biologists to spend less time rewriting common
777 analysis steps, and more time on interesting biological questions. We believe these workflow systems will
778 become increasingly important as dataset size and complexity continue to grow. This manuscript provides
779 a directed set of project, data, and resource management strategies for adopting workflow systems to
780 facilitate and expedite reproducible biological research. While the included data management strategies
781 are tailored to our own experiences in high-throughput sequencing analysis, we hope that these principles
782 enable biologists both within and beyond our field to reap the benefits of workflow-enabled data-intensive
783 biology.

784 **Acknowledgements**

785 Thank you to all the members and affiliates of the Lab for Data-Intensive Biology at UC Davis
786 for providing valuable feedback on earlier versions of this manuscript and growing these
787 practices alongside us. We also thank the Carpentries Community for fundamentally shaping

788 many of the ideas and practices we cover in this manuscript. A collaborative “living document”
789 version of the manuscript written with [manubot](#) is available in a GitHub repository [126].

790

791 Author Contributions

Author	Contributions
TER	Conceptualization; Methodology; Writing - Original Draft; Writing - Review and Editing; Visualization
PTB**	Methodology; Writing - Review and Editing
LCI**	Methodology; Writing - Review and Editing
SEJ**	Methodology; Visualization; Writing - Review and Editing
CMR**	Methodology; Writing - Review and Editing
CSW**	Methodology; Writing - Review and Editing
CTB	Methodology; Writing - Review and Editing; Supervision; Funding Acquisition
NTP	Conceptualization; Methodology; Writing - Original Draft; Writing - Review and Editing; Visualization; Supervision; Funding Acquisition

792 **co-equal contributions

793 Competing Interests

794 The authors declare no competing interests.

795 References

796 1. The nf-core framework for community-curated bioinformatics pipelines

797 Philip A. Ewels, Alexander Peltzer, Sven Fillinger, Harshil Patel, Johannes Alneberg, Andreas Wilm,

798 Maxime Ulysse Garcia, Paolo Di Tommaso, Sven Nahnsen

799 *Nature Biotechnology* (2020-02-13) <https://doi.org/ggk3qh>

800 DOI: [10.1038/s41587-020-0439-x](https://doi.org/10.1038/s41587-020-0439-x) · PMID: [32055031](https://pubmed.ncbi.nlm.nih.gov/32055031/)

801 2. Unmet needs for analyzing biological big data: A survey of 704 NSF principal investigators

802 Lindsay Barone, Jason Williams, David Micklos

803 *PLOS Computational Biology* (2017-10-19) <https://doi.org/gcgdgc>

804 DOI: [10.1371/journal.pcbi.1005755](https://doi.org/10.1371/journal.pcbi.1005755) · PMID: [29049281](https://pubmed.ncbi.nlm.nih.gov/29049281/) · PMCID: [PMC5654259](https://pubmed.ncbi.nlm.nih.gov/PMC5654259/)

805 3. Practical Computational Reproducibility in the Life Sciences

806 Björn Grüning, John Chilton, Johannes Köster, Ryan Dale, Nicola Soranzo, Marius van den Beek, Jeremy

807 Goecks, Rolf Backofen, Anton Nekrutenko, James Taylor

808 *Cell Systems* (2018-06) <https://doi.org/ggdv3z>

809 DOI: [10.1016/j.cels.2018.03.014](https://doi.org/10.1016/j.cels.2018.03.014) · PMID: [29953862](https://pubmed.ncbi.nlm.nih.gov/29953862/) · PMCID: [PMC6263957](https://pubmed.ncbi.nlm.nih.gov/PMC6263957/)

- 810 **4. A graduate student perspective on overcoming barriers to interacting with open-source software**
811 Oihane Cereceda, Danielle E. A. Quinn
812 *FACETS* (2020-01-01) <https://doi.org/ggw7f3>
813 DOI: [10.1139/facets-2019-0020](https://doi.org/10.1139/facets-2019-0020)
- 814 **5. Scientific workflows: Past, present and future**
815 Malcolm Atkinson, Sandra Gesing, Johan Montagnat, Ian Taylor
816 *Future Generation Computer Systems* (2017-10) <https://doi.org/ggs77s>
817 DOI: [10.1016/j.future.2017.05.041](https://doi.org/10.1016/j.future.2017.05.041)
- 818 **6. Rule-based workflow management for bioinformatics**
819 John S. Conery, Julian M. Catchen, Michael Lynch
820 *The VLDB Journal* (2005-09-09) <https://doi.org/fhzqvp>
821 DOI: [10.1007/s00778-005-0153-9](https://doi.org/10.1007/s00778-005-0153-9)
- 822 **7. Robust Cross-Platform Workflows: How Technical and Scientific Communities Collaborate to**
823 **Develop, Test and Share Best Practices for Data Analysis**
824 Steffen Möller, Stuart W. Prescott, Lars Wirzenius, Petter Reinholdtsen, Brad Chapman, Pjotr Prins, Stian
825 Soiland-Reyes, Fabian Klötzl, Andrea Bagnacani, Matúš Kalaš, ... Michael R. Crusoe
826 *Data Science and Engineering* (2017-11-16) <https://doi.org/ggwrrk>
827 DOI: [10.1007/s41019-017-0050-4](https://doi.org/10.1007/s41019-017-0050-4)
- 828 **8. Best Practices for Scientific Computing**
829 Greg Wilson, D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H.
830 D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, ... Paul Wilson
831 *PLoS Biology* (2014-01-07) <https://doi.org/qtt>
832 DOI: [10.1371/journal.pbio.1001745](https://doi.org/10.1371/journal.pbio.1001745) · PMID: [24415924](https://pubmed.ncbi.nlm.nih.gov/24415924/) · PMCID: [PMC3886731](https://pubmed.ncbi.nlm.nih.gov/PMC3886731/)
- 833 **9. Computing Workflows for Biologists: A Roadmap**
834 Ashley Shade, Tracy K. Teal
835 *PLOS Biology* (2015-11-24) <https://doi.org/f72r9m>
836 DOI: [10.1371/journal.pbio.1002303](https://doi.org/10.1371/journal.pbio.1002303) · PMID: [26600012](https://pubmed.ncbi.nlm.nih.gov/26600012/) · PMCID: [PMC4658184](https://pubmed.ncbi.nlm.nih.gov/PMC4658184/)
- 837 **10. Good enough practices in scientific computing**
838 Greg Wilson, Jennifer Bryan, Karen Cranston, Justin Kitzes, Lex Nederbragt, Tracy K. Teal
839 *PLOS Computational Biology* (2017-06-22) <https://doi.org/gbkbwp>
840 DOI: [10.1371/journal.pcbi.1005510](https://doi.org/10.1371/journal.pcbi.1005510) · PMID: [28640806](https://pubmed.ncbi.nlm.nih.gov/28640806/) · PMCID: [PMC5480810](https://pubmed.ncbi.nlm.nih.gov/PMC5480810/)
- 841 **11. Bioconda: sustainable and comprehensive software distribution for the life sciences**
842 Björn Grüning, Ryan Dale, Andreas Sjödin, Brad A. Chapman, Jillian Rowe, Christopher H. Tomkins-
843 Tinch, Renan Valieris, Johannes Köster, The Bioconda Team
844 *Nature Methods* (2018-07-02) <https://doi.org/gd2xzp>
845 DOI: [10.1038/s41592-018-0046-7](https://doi.org/10.1038/s41592-018-0046-7) · PMID: [29967506](https://pubmed.ncbi.nlm.nih.gov/29967506/)
- 846 **12. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018**
847 **update**
848 Enis Afgan, Dannon Baker, Bérénice Batut, Marius van den Beek, Dave Bouvier, Martin Čech, John

849 Chilton, Dave Clements, Nate Coraor, Björn A Grüning, ... Daniel Blankenberg
850 *Nucleic Acids Research* (2018-07-02) <https://doi.org/gdwxpr>
851 DOI: [10.1093/nar/gky379](https://doi.org/10.1093/nar/gky379) · PMID: [29790989](https://pubmed.ncbi.nlm.nih.gov/29790989/) · PMCID: [PMC6030816](https://pubmed.ncbi.nlm.nih.gov/PMC6030816/)

852 **13. Data Commons to Support Pediatric Cancer Research**
853 Samuel L. Volchenboun, Suzanne M. Cox, Allison Heath, Adam Resnick, Susan L. Cohn, Robert
854 Grossman
855 *American Society of Clinical Oncology Educational Book* (2017) <https://doi.org/ggv5zs>
856 DOI: [10.14694/edbk_175029](https://doi.org/10.14694/edbk_175029) · PMID: [28561664](https://pubmed.ncbi.nlm.nih.gov/28561664/)

857 **14. MGnify: the microbiome analysis resource in 2020**
858 Alex L Mitchell, Alexandre Almeida, Martin Beracochea, Miguel Boland, Josephine Burgin, Guy
859 Cochrane, Michael R Crusoe, Varsha Kale, Simon C Potter, Lorna J Richardson, ... Robert D Finn
860 *Nucleic Acids Research* (2019-11-07) <https://doi.org/ggctnm>
861 DOI: [10.1093/nar/gkz1035](https://doi.org/10.1093/nar/gkz1035) · PMID: [31696235](https://pubmed.ncbi.nlm.nih.gov/31696235/) · PMCID: [PMC7145632](https://pubmed.ncbi.nlm.nih.gov/PMC7145632/)

862 **15. nf-core/rnaseq**
863 GitHub
864 <https://github.com/nf-core/rnaseq>

865 **16. PiGx: reproducible genomics analysis pipelines with GNU Guix**
866 Ricardo Wurmus, Bora Uyar, Brendan Osberg, Vedran Franke, Alexander Godschan, Katarzyna
867 Wreczycka, Jonathan Ronen, Altuna Akalin
868 *GigaScience* (2018-12) <https://doi.org/ghdx9w>
869 DOI: [10.1093/gigascience/giy123](https://doi.org/10.1093/gigascience/giy123) · PMID: [30277498](https://pubmed.ncbi.nlm.nih.gov/30277498/) · PMCID: [PMC6275446](https://pubmed.ncbi.nlm.nih.gov/PMC6275446/)

870 **17. metagenome-atlas/atlas**
871 GitHub
872 <https://github.com/metagenome-atlas/atlas>

873 **18. ATLAS: a Snakemake workflow for assembly, annotation, and genomic binning of metagenome**
874 **sequence data**
875 Silas Kieser, Joseph Brown, Evgeny M. Zdobnov, Mirko Trajkovski, Lee Ann McCue
876 *bioRxiv* (2019-08-20) <https://doi.org/ggv5zm>
877 DOI: [10.1101/737528](https://doi.org/10.1101/737528)

878 **19. sunbeam-labs/sunbeam**
879 GitHub
880 <https://github.com/sunbeam-labs/sunbeam>

881 **20. Sunbeam: an extensible pipeline for analyzing metagenomic sequencing experiments**
882 Erik L. Clarke, Louis J. Taylor, Chunyu Zhao, Andrew Connell, Jung-Jin Lee, Bryton Fett, Frederic D.
883 Bushman, Kyle Bittinger
884 *Microbiome* (2019-03-22) <https://doi.org/gf9sd6>
885 DOI: [10.1186/s40168-019-0658-x](https://doi.org/10.1186/s40168-019-0658-x) · PMID: [30902113](https://pubmed.ncbi.nlm.nih.gov/30902113/) · PMCID: [PMC6429786](https://pubmed.ncbi.nlm.nih.gov/PMC6429786/)

- 886 21. **dib-lab/dammit**
887 GitHub
888 <https://github.com/dib-lab/dammit>
- 889 22. **dib-lab/elvers**
890 GitHub
891 <https://github.com/dib-lab/elvers>
- 892 23. **Snakemake—a scalable bioinformatics workflow engine**
893 J. Koster, S. Rahmann
894 *Bioinformatics* (2012-08-20) <https://doi.org/gd2xzq>
895 DOI: [10.1093/bioinformatics/bts480](https://doi.org/10.1093/bioinformatics/bts480) · PMID: [22908215](https://pubmed.ncbi.nlm.nih.gov/22908215/)
- 896 24. **Sustainable data analysis with Snakemake**
897 Felix Mölder, Kim Philipp Jablonski, Brice Letcher, Michael B. Hall, Christopher H. Tomkins-Tinch,
898 Vanessa Sochat, Jan Forster, Soohyun Lee, Sven O. Twardziok, Alexander Kanitz, ... Johannes Köster
899 *Zenodo* (2020-10-02) <https://doi.org/ghdx9x>
900 DOI: [10.5281/zenodo.4067137](https://doi.org/10.5281/zenodo.4067137)
- 901 25. **Nextflow enables reproducible computational workflows**
902 Paolo Di Tommaso, Maria Chatzou, Evan W Floden, Pablo Prieto Barja, Emilio Palumbo, Cedric
903 Notredame
904 *Nature Biotechnology* (2017-04-11) <https://doi.org/gfj52z>
905 DOI: [10.1038/nbt.3820](https://doi.org/10.1038/nbt.3820) · PMID: [28398311](https://pubmed.ncbi.nlm.nih.gov/28398311/)
- 906 26. **Common Workflow Language, v1.0**
907 Peter Amstutz, Michael R. Crusoe, Nebojša Tijanić, Brad Chapman, John Chilton, Michael Heuer,
908 Andrey Kartashov, Dan Leehr, Hervé Ménager, Maya Nedeljkovich, ... Luka Stojanovic
909 *figshare* (2016) <https://doi.org/gf6ppg>
910 DOI: [10.6084/m9.figshare.3115156.v2](https://doi.org/10.6084/m9.figshare.3115156.v2)
- 911 27. **Terra.Bio**
912 Terra.Bio
913 <https://terra.bio>
- 914 28. **The Seven Bridges Platform**
915 Seven Bridges
916 <https://www.sevenbridges.com/platform/>
- 917 29. **The drake R package: a pipeline toolkit for reproducibility and high-performance computing**
918 William Michael Landau
919 *The Journal of Open Source Software* (2018-01-26) <https://doi.org/gd876m>
920 DOI: [10.21105/joss.00550](https://doi.org/10.21105/joss.00550)
- 921 30. **Scalable Workflows and Reproducible Data Analysis for Genomics**
922 Francesco Strozzi, Roel Janssen, Ricardo Wurmus, Michael R. Crusoe, George Githinji, Paolo Di
923 Tommaso, Dominique Belhachemi, Steffen Möller, Geert Smant, Joep de Ligt, Pjotr Prins

- 924 *Methods in Molecular Biology* (2019) <https://doi.org/ggv5zh>
925 DOI: [10.1007/978-1-4939-9074-0_24](https://doi.org/10.1007/978-1-4939-9074-0_24) · PMID: [31278683](https://pubmed.ncbi.nlm.nih.gov/31278683/)
- 926 **31. “Sequana”: a Set of Snakemake NGS pipelines**
927 Thomas Cokelaer, Dimitri Desvillechabrol, Rachel Legendre, Mélissa Cardon
928 *The Journal of Open Source Software* (2017-08-30) <https://doi.org/ggv5zt>
929 DOI: [10.21105/joss.00352](https://doi.org/10.21105/joss.00352)
- 930 **32. A survey of best practices for RNA-seq data analysis**
931 Ana Conesa, Pedro Madrigal, Sonia Tarazona, David Gomez-Cabrero, Alejandra Cervera, Andrew
932 McPherson, Michał Wojciech Szcześniak, Daniel J. Gaffney, Laura L. Elo, Xuegong Zhang, Ali
933 Mortazavi
934 *Genome Biology* (2016-01-26) <https://doi.org/f3vhpj>
935 DOI: [10.1186/s13059-016-0881-8](https://doi.org/10.1186/s13059-016-0881-8) · PMID: [26813401](https://pubmed.ncbi.nlm.nih.gov/26813401/) · PMCID: [PMC4728800](https://pubmed.ncbi.nlm.nih.gov/PMC4728800/)
- 936 **33. Shotgun metagenomics, from sampling to analysis**
937 Christopher Quince, Alan W Walker, Jared T Simpson, Nicholas J Loman, Nicola Segata
938 *Nature Biotechnology* (2017-09-12) <https://doi.org/gbv6nf>
939 DOI: [10.1038/nbt.3935](https://doi.org/10.1038/nbt.3935) · PMID: [28898207](https://pubmed.ncbi.nlm.nih.gov/28898207/)
- 940 **34. Current best practices in single-cell RNA-seq analysis: a tutorial**
941 Malte D Luecken, Fabian J Theis
942 *Molecular Systems Biology* (2019-06-19) <https://doi.org/gf4f4d>
943 DOI: [10.15252/msb.20188746](https://doi.org/10.15252/msb.20188746) · PMID: [31217225](https://pubmed.ncbi.nlm.nih.gov/31217225/) · PMCID: [PMC6582955](https://pubmed.ncbi.nlm.nih.gov/PMC6582955/)
- 944 **35. Next-generation biology: Sequencing and data analysis approaches for non-model organisms**
945 Rute R. da Fonseca, Anders Albrechtsen, Gonçalo Espregueira Themudo, Jazmín Ramos-Madrigal, Jonas
946 Andreas Sibbesen, Lasse Maretty, M. Lisandra Zepeda-Mendoza, Paula F. Campos, Rasmus Heller,
947 Ricardo J. Pereira
948 *Marine Genomics* (2016-12) <https://doi.org/f9h2s2>
949 DOI: [10.1016/j.margen.2016.04.012](https://doi.org/10.1016/j.margen.2016.04.012) · PMID: [27184710](https://pubmed.ncbi.nlm.nih.gov/27184710/)
- 950 **36. Best practices for analysing microbiomes**
951 Rob Knight, Alison Vrbanc, Bryn C. Taylor, Alexander Aksenov, Chris Callewaert, Justine Debelius,
952 Antonio Gonzalez, Tomasz Kosciolk, Laura-Isobel McCall, Daniel McDonald, ... Pieter C. Dorrestein
953 *Nature Reviews Microbiology* (2018-05-23) <https://doi.org/gdj32p>
954 DOI: [10.1038/s41579-018-0029-9](https://doi.org/10.1038/s41579-018-0029-9) · PMID: [29795328](https://pubmed.ncbi.nlm.nih.gov/29795328/)
- 955 **37. Singularity: Scientific containers for mobility of compute**
956 Gregory M. Kurtzer, Vanessa Sochat, Michael W. Bauer
957 *PLOS ONE* (2017-05-11) <https://doi.org/f969fz>
958 DOI: [10.1371/journal.pone.0177459](https://doi.org/10.1371/journal.pone.0177459) · PMID: [28494014](https://pubmed.ncbi.nlm.nih.gov/28494014/) · PMCID: [PMC5426675](https://pubmed.ncbi.nlm.nih.gov/PMC5426675/)
- 959 **38. Docker: lightweight Linux containers for consistent development and deployment**
960 Dirk Merkel
961 *Linux Journal* (2014-03-01)

962 39. **mamba-org/mamba**
963 GitHub
964 <https://github.com/mamba-org/mamba>

965 40. **BioContainers Registry: searching for bioinformatics tools, packages and containers**
966 Jingwen Bai, Chakradhar Bandla, Jiaxin Guo, Roberto Vera Alvarez, Juan Antonio Vizcaíno, Mingze
967 Bai, Pablo Moreno, Björn A. Grüning, Olivier Sallou, Yasset Perez-Riverol
968 *Cold Spring Harbor Laboratory* (2020-07-22) <https://doi.org/ghhfgn>
969 DOI: <https://doi.org/10.1101/2020.07.21.187609>

970 41. **dib-lab/dib-MMETSP**
971 GitHub
972 <https://github.com/dib-lab/dib-MMETSP>

973 42. **Re-assembly, quality evaluation, and annotation of 678 microbial eukaryotic reference**
974 **transcriptomes**
975 Lisa K Johnson, Harriet Alexander, C Titus Brown
976 *GigaScience* (2019-04) <https://doi.org/ggrd6x>
977 DOI: [10.1093/gigascience/giy158](https://doi.org/10.1093/gigascience/giy158) · PMID: [30544207](https://pubmed.ncbi.nlm.nih.gov/30544207/) · PMCID: [PMC6481552](https://pubmed.ncbi.nlm.nih.gov/PMC6481552/)

978 43. **R Markdown** <https://rmarkdown.rstudio.com/>

979 44. **IOS Press Ebooks - Jupyter Notebooks – a publishing format for reproducible computational**
980 **workflows** <http://ebooks.iospress.nl/publication/42900>

981 45. **Tempo and mode of genome evolution in a 50,000-generation experiment**
982 Olivier Tenaillon, Jeffrey E. Barrick, Noah Ribeck, Daniel E. Deatherage, Jeffrey L. Blanchard, Aurko
983 Dasgupta, Gabriel C. Wu, Sébastien Wielgoss, Stéphane Cruveiller, Claudine Médigue, ... Richard E.
984 Lenski
985 *Nature* (2016-08-01) <https://doi.org/f8283v>
986 DOI: [10.1038/nature18959](https://doi.org/10.1038/nature18959) · PMID: [27479321](https://pubmed.ncbi.nlm.nih.gov/27479321/) · PMCID: [PMC4988878](https://pubmed.ncbi.nlm.nih.gov/PMC4988878/)

987 46. **Binder 2.0 - Reproducible, interactive, sharable environments for science at scale**
988 Project Jupyter, Matthias Bussonnier, Jessica Forde, Jeremy Freeman, Brian Granger, Tim Head, Chris
989 Holdgraf, Kyle Kelley, Gladys Nalvarte, Andrew Osheroﬀ, ... Carol Willing
990 *SciPy* (2018) <https://doi.org/gfwcm6>
991 DOI: [10.25080/majora-4af1f417-011](https://doi.org/10.25080/majora-4af1f417-011)

992 47. **ngs-docs/2020-ggg-201b-rnaseq**
993 GitHub
994 <https://github.com/ngs-docs/2020-ggg-201b-rnaseq>

995 48. **Exploring neighborhoods in large metagenome assembly graphs using spacegraphcats reveals**
996 **hidden sequence diversity**
997 C. Titus Brown, Dominik Moritz, Michael P. O'Brien, Felix Reidl, Taylor Reiter, Blair D. Sullivan
998 *Genome Biology* (2020-07-06) <https://doi.org/d4bb>
999 DOI: [10.1186/s13059-020-02066-4](https://doi.org/10.1186/s13059-020-02066-4) · PMID: [32631445](https://pubmed.ncbi.nlm.nih.gov/32631445/) · PMCID: [PMC7336657](https://pubmed.ncbi.nlm.nih.gov/PMC7336657/)

- 1000 49. **Zenodo - Research. Shared.** <https://zenodo.org/>
- 1001 50. **Computing environments for reproducibility: Capturing the “Whole Tale”**
1002 Adam Brinckman, Kyle Chard, Niall Gaffney, Mihael Hategan, Matthew B. Jones, Kacper Kowalik,
1003 Sivakumar Kulasekaran, Bertram Ludäscher, Bryce D. Mecum, Jarek Nabrzyski, ... Kandace Turner
1004 *Future Generation Computer Systems* (2019-05) <https://doi.org/ggv5zj>
1005 DOI: [10.1016/j.future.2017.12.029](https://doi.org/10.1016/j.future.2017.12.029)
- 1006 51. **ReproZip**
1007 Fernando Chirigati, Rémi Rampin, Dennis Shasha, Juliana Freire
1008 *Association for Computing Machinery (ACM)* (2016) <https://doi.org/ggxs3d>
1009 DOI: [10.1145/2882903.2899401](https://doi.org/10.1145/2882903.2899401)
- 1010 52. **Pavian** <https://fbreitwieser.shinyapps.io/pavian/>
- 1011 53. **Pavian: interactive analysis of metagenomics data for microbiome studies and pathogen**
1012 **identification**
1013 Florian P Breitwieser, Steven L Salzberg
1014 *Bioinformatics* (2019-09-25) <https://doi.org/ggnb3n>
1015 DOI: [10.1093/bioinformatics/btz715](https://doi.org/10.1093/bioinformatics/btz715) · PMID: [31553437](https://pubmed.ncbi.nlm.nih.gov/31553437/)
- 1016 54. **Visualizing Biological Data** <https://plotly.com/python/v3/ipython-notebooks/bioinformatics/>
- 1017 55. **Plotly: The front end for ML and data science models** /
- 1018 56. **Vega-Lite: A Grammar of Interactive Graphics**
1019 Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, Jeffrey Heer
1020 *IEEE Transactions on Visualization and Computer Graphics* (2017-01) <https://doi.org/f92f32>
1021 DOI: [10.1109/tvcg.2016.2599030](https://doi.org/10.1109/tvcg.2016.2599030) · PMID: [27875150](https://pubmed.ncbi.nlm.nih.gov/27875150/)
- 1022 57. **HackMD - Collaborative Markdown Knowledge Base**
1023 HackMD
1024 <https://hackmd.io>
- 1025 58. **Documentation** <https://www.ncbi.nlm.nih.gov/sra/docs/sragrowth/>
- 1026 59. **The FAIR Guiding Principles for scientific data management and stewardship**
1027 Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton,
1028 Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, ...
1029 Barend Mons
1030 *Scientific Data* (2016-03-15) <https://doi.org/bdd4>
1031 DOI: [10.1038/sdata.2016.18](https://doi.org/10.1038/sdata.2016.18) · PMID: [26978244](https://pubmed.ncbi.nlm.nih.gov/26978244/) · PMCID: [PMC4792175](https://pubmed.ncbi.nlm.nih.gov/PMC4792175/)
- 1032 60. **The international nucleotide sequence database collaboration**
1033 Ilene Karsch-Mizrachi, Toshihisa Takagi, Guy Cochrane, on behalf of the International Nucleotide
1034 Sequence Database Collaboration
1035 *Nucleic Acids Research* (2018-01-04) <https://doi.org/ggwp7h>
1036 DOI: [10.1093/nar/gkx1097](https://doi.org/10.1093/nar/gkx1097) · PMID: [29190397](https://pubmed.ncbi.nlm.nih.gov/29190397/) · PMCID: [PMC5753279](https://pubmed.ncbi.nlm.nih.gov/PMC5753279/)

- 1037 61. **Public Microbial Resource Centers: Key Hubs for Findable, Accessible, Interoperable, and**
1038 **Reusable (FAIR) Microorganisms and Genetic Materials**
1039 P. Becker, M. Bosschaerts, P. Chaerle, H.-M. Daniel, A. Hellemans, A. Olbrechts, L. Rigouts, A.
1040 Wilmotte, M. Hendrickx
1041 *Applied and Environmental Microbiology* (2019-10-16) <https://doi.org/ggbpc9>
1042 DOI: [10.1128/aem.01444-19](https://doi.org/10.1128/aem.01444-19) · PMID: [31471301](https://pubmed.ncbi.nlm.nih.gov/31471301/) · PMCID: [PMC6803313](https://pubmed.ncbi.nlm.nih.gov/PMC6803313/)
- 1043 62. **Linking the International Wheat Genome Sequencing Consortium bread wheat reference**
1044 **genome sequence to wheat genetic and phenomic data**
1045 Michael Alaux, Jane Rogers, Thomas Letellier, Raphaël Flores, Françoise Alfama, Cyril Pommier, Nacer
1046 Mohellibi, Sophie Durand, Erik Kimmel, Célia Michotey, ... International Wheat Genome Sequencing
1047 Consortium
1048 *Genome Biology* (2018-08-17) <https://doi.org/gf23xv>
1049 DOI: [10.1186/s13059-018-1491-4](https://doi.org/10.1186/s13059-018-1491-4) · PMID: [30115101](https://pubmed.ncbi.nlm.nih.gov/30115101/) · PMCID: [PMC6097284](https://pubmed.ncbi.nlm.nih.gov/PMC6097284/)
- 1050 63. **FAIR: A Call to Make Published Data More Findable, Accessible, Interoperable, and Reusable**
1051 Leonore Reiser, Lisa Harper, Michael Freeling, Bin Han, Sheng Luan
1052 *Molecular Plant* (2018-09) <https://doi.org/ggwp69>
1053 DOI: [10.1016/j.molp.2018.07.005](https://doi.org/10.1016/j.molp.2018.07.005) · PMID: [30076986](https://pubmed.ncbi.nlm.nih.gov/30076986/)
- 1054 64. **The Integrative Human Microbiome Project**
1055 The Integrative HMP (iHMP) Research Network Consortium
1056 *Nature* (2019-05-29) <https://doi.org/gf3wp9>
1057 DOI: [10.1038/s41586-019-1238-8](https://doi.org/10.1038/s41586-019-1238-8) · PMID: [31142853](https://pubmed.ncbi.nlm.nih.gov/31142853/) · PMCID: [PMC6784865](https://pubmed.ncbi.nlm.nih.gov/PMC6784865/)
- 1058 65. **The Pfam protein families database in 2019**
1059 Sara El-Gebali, Jaina Mistry, Alex Bateman, Sean R Eddy, Aurélien Luciani, Simon C Potter, Matloob
1060 Qureshi, Lorna J Richardson, Gustavo A Salazar, Alfredo Smart, ... Robert D Finn
1061 *Nucleic Acids Research* (2019-01-08) <https://doi.org/gfhx7r>
1062 DOI: [10.1093/nar/gky995](https://doi.org/10.1093/nar/gky995) · PMID: [30357350](https://pubmed.ncbi.nlm.nih.gov/30357350/) · PMCID: [PMC6324024](https://pubmed.ncbi.nlm.nih.gov/PMC6324024/)
- 1063 66. **The International Nucleotide Sequence Database Collaboration**
1064 Guy Cochrane, Ilene Karsch-Mizrachi, Toshihisa Takagi, International Nucleotide
1065 Sequence Database Collaboration
1066 *Nucleic Acids Research* (2016-01-04) <https://doi.org/gf28sk>
1067 DOI: [10.1093/nar/gkv1323](https://doi.org/10.1093/nar/gkv1323) · PMID: [26657633](https://pubmed.ncbi.nlm.nih.gov/26657633/) · PMCID: [PMC4702924](https://pubmed.ncbi.nlm.nih.gov/PMC4702924/)
- 1068 67. **Open science resources for the discovery and analysis of Tara Oceans data**
1069 Stéphane Pesant, Fabrice Not, Marc Picheral, Stefanie Kandels-Lewis, Noan Le Bescot, Gabriel Gorsky,
1070 Daniele Iudicone, Eric Karsenti, Sabrina Speich, Romain Troublé, ... Sarah Searson
1071 *Scientific Data* (2015-05-26) <https://doi.org/gd32xw>
1072 DOI: [10.1038/sdata.2015.23](https://doi.org/10.1038/sdata.2015.23) · PMID: [26029378](https://pubmed.ncbi.nlm.nih.gov/26029378/) · PMCID: [PMC4443879](https://pubmed.ncbi.nlm.nih.gov/PMC4443879/)
- 1073 68. **Gene Expression Omnibus: NCBI gene expression and hybridization array data repository**
1074 R. Edgar
1075 *Nucleic Acids Research* (2002-01-01) <https://doi.org/fttpkn>
1076 DOI: [10.1093/nar/30.1.207](https://doi.org/10.1093/nar/30.1.207) · PMID: [11752295](https://pubmed.ncbi.nlm.nih.gov/11752295/) · PMCID: [PMC99122](https://pubmed.ncbi.nlm.nih.gov/PMC99122/)

- 1077 **69. WormBase: a modern Model Organism Information Resource**
1078 Todd W Harris, Valerio Arnaboldi, Scott Cain, Juancarlos Chan, Wen J Chen, Jaehyoung Cho, Paul
1079 Davis, Sibyl Gao, Christian A Grove, Ranjana Kishore, ... Paul W Sternberg
1080 *Nucleic Acids Research* (2019-10-23) <https://doi.org/ggv5zk>
1081 DOI: [10.1093/nar/gkz920](https://doi.org/10.1093/nar/gkz920) · PMID: [31642470](https://pubmed.ncbi.nlm.nih.gov/31642470/) · PMCID: [PMC7145598](https://pubmed.ncbi.nlm.nih.gov/PMC7145598/)
- 1082 **70. Open Science Framework (OSF)**
1083 Erin D. Foster, MSLS, Ariel Deardorff, MLIS
1084 *Journal of the Medical Library Association* (2017-04-04) <https://doi.org/gfxvhq>
1085 DOI: [10.5195/jmla.2017.88](https://doi.org/10.5195/jmla.2017.88) · PMCID: [PMC5370619](https://pubmed.ncbi.nlm.nih.gov/PMC5370619/)
- 1086 **71. Erratum: How many biological replicates are needed in an RNA-seq experiment and which**
1087 **differential expression tool should you use?**
1088 Nicholas J. Schurch, Pietà Schofield, Marek Gierliński, Christian Cole, Alexander Sherstnev, Vijender
1089 Singh, Nicola Wrobel, Karim Gharbi, Gordon G. Simpson, Tom Owen-Hughes, ... Geoffrey J. Barton
1090 *RNA* (2016-09-16) <https://doi.org/ggv5zr>
1091 DOI: [10.1261/rna.058339.116](https://doi.org/10.1261/rna.058339.116) · PMID: [27638913](https://pubmed.ncbi.nlm.nih.gov/27638913/) · PMCID: [PMC5029462](https://pubmed.ncbi.nlm.nih.gov/PMC5029462/)
- 1092 **72. Power analysis and sample size estimation for RNA-Seq differential expression**
1093 Travers Ching, Sijia Huang, Lana X. Garmire
1094 *RNA* (2014-11) <https://doi.org/f6nstp>
1095 DOI: [10.1261/rna.046011.114](https://doi.org/10.1261/rna.046011.114) · PMID: [25246651](https://pubmed.ncbi.nlm.nih.gov/25246651/) · PMCID: [PMC4201821](https://pubmed.ncbi.nlm.nih.gov/PMC4201821/)
- 1096 **73. Unlocking the potential of metagenomics through replicated experimental design**
1097 Rob Knight, Janet Jansson, Dawn Field, Noah Fierer, Narayan Desai, Jed A Fuhrman, Phil Hugenholtz,
1098 Daniel van der Lelie, Folker Meyer, Rick Stevens, ... Jack A Gilbert
1099 *Nature Biotechnology* (2012-06-07) <https://doi.org/f32nds>
1100 DOI: [10.1038/nbt.2235](https://doi.org/10.1038/nbt.2235) · PMID: [22678395](https://pubmed.ncbi.nlm.nih.gov/22678395/) · PMCID: [PMC4902277](https://pubmed.ncbi.nlm.nih.gov/PMC4902277/)
- 1101 **74. Contamination in Low Microbial Biomass Microbiome Studies: Issues and Recommendations**
1102 Raphael Eisenhofer, Jeremiah J. Minich, Clarisse Marotz, Alan Cooper, Rob Knight, Laura S. Weyrich
1103 *Trends in Microbiology* (2019-02) <https://doi.org/gfpfkt>
1104 DOI: [10.1016/j.tim.2018.11.003](https://doi.org/10.1016/j.tim.2018.11.003) · PMID: [30497919](https://pubmed.ncbi.nlm.nih.gov/30497919/)
- 1105 **75. Consistent and correctable bias in metagenomic sequencing experiments**
1106 Michael R McLaren, Amy D Willis, Benjamin J Callahan
1107 *eLife* (2019-09-10) <https://doi.org/gf7wbr>
1108 DOI: [10.7554/elife.46923](https://doi.org/10.7554/elife.46923) · PMID: [31502536](https://pubmed.ncbi.nlm.nih.gov/31502536/) · PMCID: [PMC6739870](https://pubmed.ncbi.nlm.nih.gov/PMC6739870/)
- 1109 **76. From Benchtop to Desktop: Important Considerations when Designing Amplicon Sequencing**
1110 **Workflows**
1111 Dáithí C. Murray, Megan L. Coghlan, Michael Bunce
1112 *PLOS ONE* (2015-04-22) <https://doi.org/f7hjz7>
1113 DOI: [10.1371/journal.pone.0124671](https://doi.org/10.1371/journal.pone.0124671) · PMID: [25902146](https://pubmed.ncbi.nlm.nih.gov/25902146/) · PMCID: [PMC4406758](https://pubmed.ncbi.nlm.nih.gov/PMC4406758/)
- 1114 **77. Assessment of variation in microbial community amplicon sequencing by the Microbiome**
1115 **Quality Control (MBQC) project consortium**

- 1116 Rashmi Sinha, Galeb Abu-Ali, Emily Vogtmann, Anthony A Fodor, Boyu Ren, Amnon Amir, Emma
1117 Schwager, Jonathan Crabtree, Siyuan Ma, Christian C Abnet, ... The Microbiome Quality Control Project
1118 Consortium
1119 *Nature Biotechnology* (2017-10-02) <https://doi.org/gbzrdx>
1120 DOI: [10.1038/nbt.3981](https://doi.org/10.1038/nbt.3981) · PMID: [28967885](https://pubmed.ncbi.nlm.nih.gov/28967885/) · PMCID: [PMC5839636](https://pubmed.ncbi.nlm.nih.gov/PMC5839636/)
- 1121 **78. Completing bacterial genome assemblies: strategy and performance comparisons**
1122 Yu-Chieh Liao, Shu-Hung Lin, Hsin-Hung Lin
1123 *Scientific Reports* (2015-03-04) <https://doi.org/f686w8>
1124 DOI: [10.1038/srep08747](https://doi.org/10.1038/srep08747) · PMID: [25735824](https://pubmed.ncbi.nlm.nih.gov/25735824/) · PMCID: [PMC4348652](https://pubmed.ncbi.nlm.nih.gov/PMC4348652/)
- 1125 **79. Earth BioGenome Project: Sequencing life for the future of life**
1126 Harris A. Lewin, Gene E. Robinson, W. John Kress, William J. Baker, Jonathan Coddington, Keith A.
1127 Crandall, Richard Durbin, Scott V. Edwards, Félix Forest, M. Thomas P. Gilbert, ... Guojie Zhang
1128 *Proceedings of the National Academy of Sciences* (2018-04-24) <https://doi.org/gdh5vz>
1129 DOI: [10.1073/pnas.1720115115](https://doi.org/10.1073/pnas.1720115115) · PMID: [29686065](https://pubmed.ncbi.nlm.nih.gov/29686065/) · PMCID: [PMC5924910](https://pubmed.ncbi.nlm.nih.gov/PMC5924910/)
- 1130 **80. Opportunities and challenges in long-read sequencing data analysis**
1131 Shanika L. Amarasinghe, Shian Su, Xueyi Dong, Luke Zappia, Matthew E. Ritchie, Quentin Gouil
1132 *Genome Biology* (2020-02-07) <https://doi.org/ggkpv7>
1133 DOI: [10.1186/s13059-020-1935-5](https://doi.org/10.1186/s13059-020-1935-5) · PMID: [32033565](https://pubmed.ncbi.nlm.nih.gov/32033565/) · PMCID: [PMC7006217](https://pubmed.ncbi.nlm.nih.gov/PMC7006217/)
- 1134 **81. Whole-genome sequencing of eukaryotes: From sequencing of DNA fragments to a genome**
1135 **assembly**
1136 K. S. Zadesenets, N. I. Ershov, N. B. Rubtsov
1137 *Russian Journal of Genetics* (2017-07-12) <https://doi.org/gg26n5>
1138 DOI: [10.1134/s102279541705012x](https://doi.org/10.1134/s102279541705012x)
- 1139 **82. Ten steps to get started in Genome Assembly and Annotation**
1140 Victoria Dominguez Del Angel, Erik Hjerde, Lieven Sterck, Salvadors Capella-Gutierrez, Cederic
1141 Notredame, Olga Vinnere Pettersson, Joelle Amselem, Laurent Bouri, Stephanie Bocs, Christophe Klopp,
1142 ... Henrik Lantz
1143 *F1000Research* (2018-02-05) <https://doi.org/gdq9zj>
1144 DOI: [10.12688/f1000research.13598.1](https://doi.org/10.12688/f1000research.13598.1) · PMID: [29568489](https://pubmed.ncbi.nlm.nih.gov/29568489/) · PMCID: [PMC5850084](https://pubmed.ncbi.nlm.nih.gov/PMC5850084/)
- 1145 **83. Whole-genome sequencing approaches for conservation biology: Advantages, limitations and**
1146 **practical recommendations**
1147 Angela P. Fuentes-Pardo, Daniel E. Ruzzante
1148 *Molecular Ecology* (2017-10) <https://doi.org/gfzn9r>
1149 DOI: [10.1111/mec.14264](https://doi.org/10.1111/mec.14264) · PMID: [28746784](https://pubmed.ncbi.nlm.nih.gov/28746784/)
- 1150 **84. Bioinformatic processing of RAD-seq data dramatically impacts downstream population genetic**
1151 **inference**
1152 Aaron B. A. Shafer, Claire R. Peart, Sergio Tusso, Inbar Maayan, Alan Brelsford, Christopher W. Wheat,
1153 Jochen B. W. Wolf
1154 *Methods in Ecology and Evolution* (2017-08) <https://doi.org/gbrdv5>
1155 DOI: [10.1111/2041-210x.12700](https://doi.org/10.1111/2041-210x.12700)

- 1156 85. **Selecting RAD-Seq Data Analysis Parameters for Population Genetics: The More the Better?**
1157 Natalia Díaz-Arce, Naiara Rodríguez-Ezpeleta
1158 *Frontiers in Genetics* (2019-05-29) <https://doi.org/gg26n7>
1159 DOI: [10.3389/fgene.2019.00533](https://doi.org/10.3389/fgene.2019.00533) · PMID: [31191624](https://pubmed.ncbi.nlm.nih.gov/31191624/) · PMCID: [PMC6549478](https://pubmed.ncbi.nlm.nih.gov/PMC6549478/)
- 1160 86. **Harnessing the power of RADseq for ecological and evolutionary genomics**
1161 Kimberly R. Andrews, Jeffrey M. Good, Michael R. Miller, Gordon Luikart, Paul A. Hohenlohe
1162 *Nature Reviews Genetics* (2016-01-05) <https://doi.org/gd85wf>
1163 DOI: [10.1038/nrg.2015.28](https://doi.org/10.1038/nrg.2015.28) · PMID: [26729255](https://pubmed.ncbi.nlm.nih.gov/26729255/) · PMCID: [PMC4823021](https://pubmed.ncbi.nlm.nih.gov/PMC4823021/)
- 1164 87. **Unbroken: RADseq remains a powerful tool for understanding the genetics of adaptation in**
1165 **natural populations**
1166 Julian M. Catchen, Paul A. Hohenlohe, Louis Bernatchez, W. Chris Funk, Kimberly R. Andrews, Fred W.
1167 Allendorf
1168 *Molecular Ecology Resources* (2017-05) <https://doi.org/f92pff>
1169 DOI: [10.1111/1755-0998.12669](https://doi.org/10.1111/1755-0998.12669) · PMID: [28319339](https://pubmed.ncbi.nlm.nih.gov/28319339/)
- 1170 88. **Responsible RAD: Striving for best practices in population genomic studies of adaptation**
1171 David B. Lowry, Sean Hoban, Joanna L. Kelley, Katie E. Lotterhos, Laura K. Reed, Michael F. Antolin,
1172 Andrew Storfer
1173 *Molecular Ecology Resources* (2017-05) <https://doi.org/gfzn6h>
1174 DOI: [10.1111/1755-0998.12677](https://doi.org/10.1111/1755-0998.12677) · PMID: [28382730](https://pubmed.ncbi.nlm.nih.gov/28382730/)
- 1175 89. **Design and computational analysis of single-cell RNA-sequencing experiments**
1176 Rhonda Bacher, Christina Kendzierski
1177 *Genome Biology* (2016-04-07) <https://doi.org/gc958g>
1178 DOI: [10.1186/s13059-016-0927-y](https://doi.org/10.1186/s13059-016-0927-y) · PMID: [27052890](https://pubmed.ncbi.nlm.nih.gov/27052890/) · PMCID: [PMC4823857](https://pubmed.ncbi.nlm.nih.gov/PMC4823857/)
- 1179 90. **A practical guide to single-cell RNA-sequencing for biomedical research and clinical**
1180 **applications**
1181 Ashrafal Haque, Jessica Engel, Sarah A. Teichmann, Tapio Lönnberg
1182 *Genome Medicine* (2017-08-18) <https://doi.org/gfgppz>
1183 DOI: [10.1186/s13073-017-0467-4](https://doi.org/10.1186/s13073-017-0467-4) · PMID: [28821273](https://pubmed.ncbi.nlm.nih.gov/28821273/) · PMCID: [PMC5561556](https://pubmed.ncbi.nlm.nih.gov/PMC5561556/)
- 1184 91. **Developing a modern data workflow for evolving data**
1185 Glenda M. Yenni, Erica M. Christensen, Ellen K. Bledsoe, Sarah R. Supp, Renata M. Diaz, Ethan P.
1186 White, S. K. Morgan Ernest
1187 *Cold Spring Harbor Laboratory* (2018-07-24) <https://doi.org/gdqbnz>
1188 DOI: [10.1101/344804](https://doi.org/10.1101/344804)
- 1189 92. **Git Large File Storage**
1190 Git Large File Storage
1191 <https://git-lfs.github.com/>
- 1192 93. **figshare - credit for all your research** <https://figshare.com/>
- 1193 94. **Dryad Home - Publish and Preserve your Data** <https://datadryad.org/stash>

- 1194 95. **RClone: a package to identify MultiLocus Clonal Lineages and handle clonal data sets in .**
1195 Diane Bailleul, Solenn Stoeckel, Sophie Arnaud-Haond
1196 *Methods in Ecology and Evolution* (2016-08) <https://doi.org/f82t65>
1197 DOI: [10.1111/2041-210x.12550](https://doi.org/10.1111/2041-210x.12550)
- 1198 96. **SRA in the Cloud** <https://www.ncbi.nlm.nih.gov/sra/docs/sra-cloud/>
- 1199 97. **Cyberduck | Libre server and cloud storage browser for Mac and Windows with support for**
1200 **FTP, SFTP, WebDAV, Amazon S3, OpenStack Swift, Backblaze B2, Microsoft Azure & OneDrive,**
1201 **Google Drive and Dropbox** <https://cyberduck.io/>
- 1202 98. **Babraham Bioinformatics - FastQC A Quality Control tool for High Throughput Sequence**
1203 **Data** <https://www.bioinformatics.babraham.ac.uk/projects/fastqc/>
- 1204 99. **Integrative Genomics Viewer (IGV): high-performance genomics data visualization and**
1205 **exploration**
1206 H. Thorvaldsdottir, J. T. Robinson, J. P. Mesirov
1207 *Briefings in Bioinformatics* (2012-04-19) <https://doi.org/f4sc43>
1208 DOI: [10.1093/bib/bbs017](https://doi.org/10.1093/bib/bbs017) · PMID: [22517427](https://pubmed.ncbi.nlm.nih.gov/22517427/) · PMCID: [PMC3603213](https://pubmed.ncbi.nlm.nih.gov/PMC3603213/)
- 1209 100. **Salmon provides fast and bias-aware quantification of transcript expression**
1210 Rob Patro, Geet Duggal, Michael I Love, Rafael A Irizarry, Carl Kingsford
1211 *Nature Methods* (2017-03-06) <https://doi.org/gcw9f5>
1212 DOI: [10.1038/nmeth.4197](https://doi.org/10.1038/nmeth.4197) · PMID: [28263959](https://pubmed.ncbi.nlm.nih.gov/28263959/) · PMCID: [PMC5600148](https://pubmed.ncbi.nlm.nih.gov/PMC5600148/)
- 1213 101. **MultiQC: summarize analysis results for multiple tools and samples in a single report**
1214 Philip Ewels, Måns Magnusson, Sverker Lundin, Max Källér
1215 *Bioinformatics* (2016-10-01) <https://doi.org/f3s996>
1216 DOI: [10.1093/bioinformatics/btw354](https://doi.org/10.1093/bioinformatics/btw354) · PMID: [27312411](https://pubmed.ncbi.nlm.nih.gov/27312411/) · PMCID: [PMC5039924](https://pubmed.ncbi.nlm.nih.gov/PMC5039924/)
- 1217 102. **Identifying and mitigating bias in next-generation sequencing methods for chromatin biology**
1218 Clifford A. Meyer, X. Shirley Liu
1219 *Nature Reviews Genetics* (2014-09-16) <https://doi.org/ggd9m9>
1220 DOI: [10.1038/nrg3788](https://doi.org/10.1038/nrg3788) · PMID: [25223782](https://pubmed.ncbi.nlm.nih.gov/25223782/) · PMCID: [PMC4473780](https://pubmed.ncbi.nlm.nih.gov/PMC4473780/)
- 1221 103. **The impact of amplification on differential expression analyses by RNA-seq**
1222 Swati Parekh, Christoph Ziegenhain, Beate Vieth, Wolfgang Enard, Ines Hellmann
1223 *Scientific Reports* (2016-05-09) <https://doi.org/f8kzcp>
1224 DOI: [10.1038/srep25533](https://doi.org/10.1038/srep25533) · PMID: [27156886](https://pubmed.ncbi.nlm.nih.gov/27156886/) · PMCID: [PMC4860583](https://pubmed.ncbi.nlm.nih.gov/PMC4860583/)
- 1225 104. **Detection and Removal of PCR Duplicates in Population Genomic ddRAD Studies by Addition**
1226 **of a Degenerate Base Region (DBR) in Sequencing Adapters**
1227 Hannah Schweyen, Andrey Rozenberg, Florian Leese
1228 *The Biological Bulletin* (2014-10) <https://doi.org/f6q76c>
1229 DOI: [10.1086/bblv227n2p146](https://doi.org/10.1086/bblv227n2p146) · PMID: [25411373](https://pubmed.ncbi.nlm.nih.gov/25411373/)
- 1230 105. **Elimination of PCR duplicates in RNA-seq and small RNA-seq using unique molecular**
1231 **identifiers**

- 1232 Yu Fu, Pei-Hsuan Wu, Timothy Beane, Phillip D. Zamore, Zhiping Weng
1233 *BMC Genomics* (2018-07-13) <https://doi.org/ggv5zp>
1234 DOI: [10.1186/s12864-018-4933-1](https://doi.org/10.1186/s12864-018-4933-1) · PMID: [30001700](https://pubmed.ncbi.nlm.nih.gov/30001700/) · PMCID: [PMC6044086](https://pubmed.ncbi.nlm.nih.gov/PMC6044086/)
- 1235 **106. Biased estimates of clonal evolution and subclonal heterogeneity can arise from PCR**
1236 **duplicates in deep sequencing experiments**
1237 Erin N Smith, Kristen Jepsen, Mahdieh Khosroheidari, Laura Z Rassenti, Matteo D'Antonio, Emanuela
1238 M Ghia, Dennis A Carson, Catriona HM Jamieson, Thomas J Kipps, Kelly A Frazer
1239 *Genome Biology* (2014-08-07) <https://doi.org/ggfhv7>
1240 DOI: [10.1186/s13059-014-0420-4](https://doi.org/10.1186/s13059-014-0420-4) · PMID: [25103687](https://pubmed.ncbi.nlm.nih.gov/25103687/) · PMCID: [PMC4165357](https://pubmed.ncbi.nlm.nih.gov/PMC4165357/)
- 1241 **107. Evidence for extensive horizontal gene transfer from the draft genome of a tardigrade**
1242 Thomas C. Boothby, Jennifer R. Tenlen, Frank W. Smith, Jeremy R. Wang, Kiera A. Patanella, Erin
1243 Osborne Nishimura, Sophia C. Tintori, Qing Li, Corbin D. Jones, Mark Yandell, ... Bob Goldstein
1244 *Proceedings of the National Academy of Sciences* (2015-12-29) <https://doi.org/9gs>
1245 DOI: [10.1073/pnas.1510461112](https://doi.org/10.1073/pnas.1510461112) · PMID: [26598659](https://pubmed.ncbi.nlm.nih.gov/26598659/) · PMCID: [PMC4702960](https://pubmed.ncbi.nlm.nih.gov/PMC4702960/)
- 1246 **108. No evidence for extensive horizontal gene transfer in the genome of the tardigrade *Hypsibius***
1247 ***dujardini***
1248 Georgios Koutsovoulos, Sujai Kumar, Dominik R. Laetsch, Lewis Stevens, Jennifer Daub, Claire Conlon,
1249 Habib Maroon, Fran Thomas, Aziz A. Aboobaker, Mark Blaxter
1250 *Proceedings of the National Academy of Sciences* (2016-05-03) <https://doi.org/f8nrtd>
1251 DOI: [10.1073/pnas.1600338113](https://doi.org/10.1073/pnas.1600338113) · PMID: [27035985](https://pubmed.ncbi.nlm.nih.gov/27035985/) · PMCID: [PMC4983863](https://pubmed.ncbi.nlm.nih.gov/PMC4983863/)
- 1252 **109. Large-scale contamination of microbial isolate genomes by Illumina PhiX control**
1253 Supratim Mukherjee, Marcel Huntemann, Natalia Ivanova, Nikos C Kyrpides, Amrita Pati
1254 *Standards in Genomic Sciences* (2015-03-30) <https://doi.org/ggv5zn>
1255 DOI: [10.1186/1944-3277-10-18](https://doi.org/10.1186/1944-3277-10-18) · PMID: [26203331](https://pubmed.ncbi.nlm.nih.gov/26203331/) · PMCID: [PMC4511556](https://pubmed.ncbi.nlm.nih.gov/PMC4511556/)
- 1256 **110. Index hopping on the Illumina HiSeqX platform and its consequences for ancient DNA studies**
1257 Tom van der Valk, Francesco Vezzi, Mattias Ormestad, Love Dalén, Katerina Guschanski
1258 *Molecular Ecology Resources* (2019-05-05) <https://doi.org/gfwtvw>
1259 DOI: [10.1111/1755-0998.13009](https://doi.org/10.1111/1755-0998.13009) · PMID: [30848092](https://pubmed.ncbi.nlm.nih.gov/30848092/)
- 1260 **111. On the optimal trimming of high-throughput mRNA sequence data**
1261 Matthew D. MacManes
1262 *Frontiers in Genetics* (2014) <https://doi.org/gfn5g7>
1263 DOI: [10.3389/fgene.2014.00013](https://doi.org/10.3389/fgene.2014.00013) · PMID: [24567737](https://pubmed.ncbi.nlm.nih.gov/24567737/) · PMCID: [PMC3908319](https://pubmed.ncbi.nlm.nih.gov/PMC3908319/)
- 1264 **112. Streaming histogram sketching for rapid microbiome analytics**
1265 Will PM Rowe, Anna Paola Carrieri, Cristina Alcon-Giner, Shabonam Caim, Alex Shaw, Kathleen Sim,
1266 J. Simon Kroll, Lindsay J. Hall, Edward O. Pyzer-Knapp, Martyn D. Winn
1267 *Microbiome* (2019-03-16) <https://doi.org/ggv5zq>
1268 DOI: [10.1186/s40168-019-0653-2](https://doi.org/10.1186/s40168-019-0653-2) · PMID: [30878035](https://pubmed.ncbi.nlm.nih.gov/30878035/) · PMCID: [PMC6420756](https://pubmed.ncbi.nlm.nih.gov/PMC6420756/)
- 1269 **113. When the levee breaks: a practical guide to sketching algorithms for processing the flood of**
1270 **genomic data**

1271 Will P. M. Rowe
1272 *Genome Biology* (2019-09-13) <https://doi.org/gf8bfj>
1273 DOI: [10.1186/s13059-019-1809-x](https://doi.org/10.1186/s13059-019-1809-x) · PMID: [31519212](https://pubmed.ncbi.nlm.nih.gov/31519212/) · PMCID: [PMC6744645](https://pubmed.ncbi.nlm.nih.gov/PMC6744645/)

1274 114. **will-rowe/genome-sketching**
1275 GitHub
1276 <https://github.com/will-rowe/genome-sketching>

1277 115. **STAR: ultrafast universal RNA-seq aligner**
1278 Alexander Dobin, Carrie A. Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe
1279 Batut, Mark Chaisson, Thomas R. Gingeras
1280 *Bioinformatics* (2013-01) <https://doi.org/f4h523>
1281 DOI: [10.1093/bioinformatics/bts635](https://doi.org/10.1093/bioinformatics/bts635) · PMID: [23104886](https://pubmed.ncbi.nlm.nih.gov/23104886/) · PMCID: [PMC3530905](https://pubmed.ncbi.nlm.nih.gov/PMC3530905/)

1282 116. **Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype**
1283 Daehwan Kim, Joseph M. Paggi, Chanhee Park, Christopher Bennett, Steven L. Salzberg
1284 *Nature Biotechnology* (2019-08-02) <https://doi.org/gf5395>
1285 DOI: [10.1038/s41587-019-0201-4](https://doi.org/10.1038/s41587-019-0201-4) · PMID: [31375807](https://pubmed.ncbi.nlm.nih.gov/31375807/) · PMCID: [PMC7605509](https://pubmed.ncbi.nlm.nih.gov/PMC7605509/)

1286 117. **Near-optimal probabilistic RNA-seq quantification**
1287 Nicolas L Bray, Harold Pimentel, Páll Melsted, Lior Pachter
1288 *Nature Biotechnology* (2016-04-04) <https://doi.org/f8nvsp>
1289 DOI: [10.1038/nbt.3519](https://doi.org/10.1038/nbt.3519) · PMID: [27043002](https://pubmed.ncbi.nlm.nih.gov/27043002/)

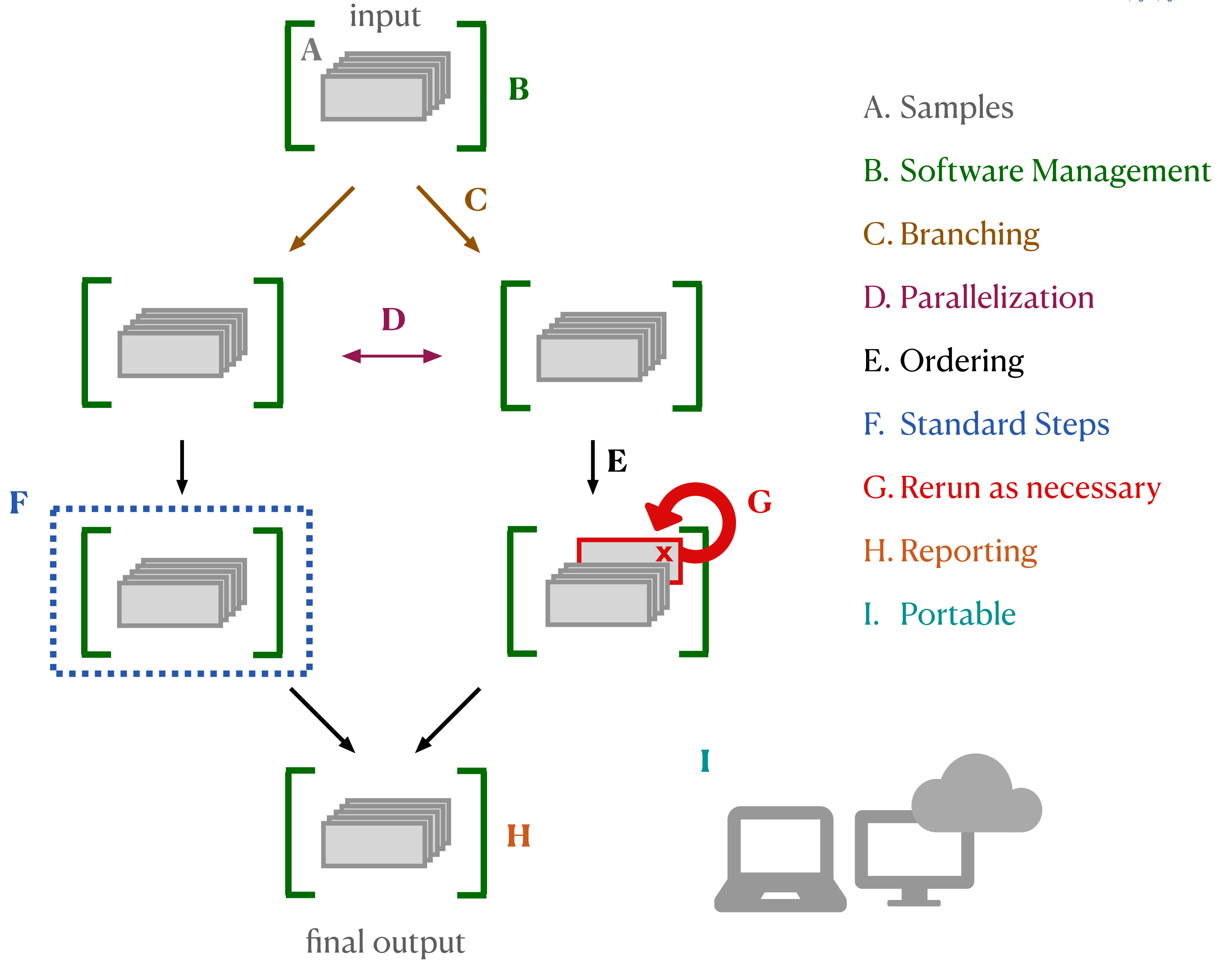
1290 118. **RapMap: a rapid, sensitive and accurate tool for mapping RNA-seq reads to transcriptomes**
1291 Avi Srivastava, Hirak Sarkar, Nitish Gupta, Rob Patro
1292 *Bioinformatics* (2016-06-15) <https://doi.org/f8vm2j>
1293 DOI: [10.1093/bioinformatics/btw277](https://doi.org/10.1093/bioinformatics/btw277) · PMID: [27307617](https://pubmed.ncbi.nlm.nih.gov/27307617/) · PMCID: [PMC4908361](https://pubmed.ncbi.nlm.nih.gov/PMC4908361/)

1294 119. **The Types, Roles, and Practices of Documentation in Data Analytics Open Source Software**
1295 **Libraries**
1296 R. Stuart Geiger, Nelle Varoquaux, Charlotte Mazel-Cabasse, Chris Holdgraf
1297 *Computer Supported Cooperative Work (CSCW)* (2018-05-29) <https://doi.org/gd4rhr>
1298 DOI: [10.1007/s10606-018-9333-1](https://doi.org/10.1007/s10606-018-9333-1)

1299 120. **Data Carpentry: Workshops to Increase Data Literacy for Researchers**
1300 Tracy K. Teal, Karen A. Cranston, Hilmar Lapp, Ethan White, Greg Wilson, Karthik Ram, Aleksandra
1301 Pawlik
1302 *International Journal of Digital Curation* (2015-03-18) <https://doi.org/gf5hjw>
1303 DOI: [10.2218/ijdc.v10i1.351](https://doi.org/10.2218/ijdc.v10i1.351)

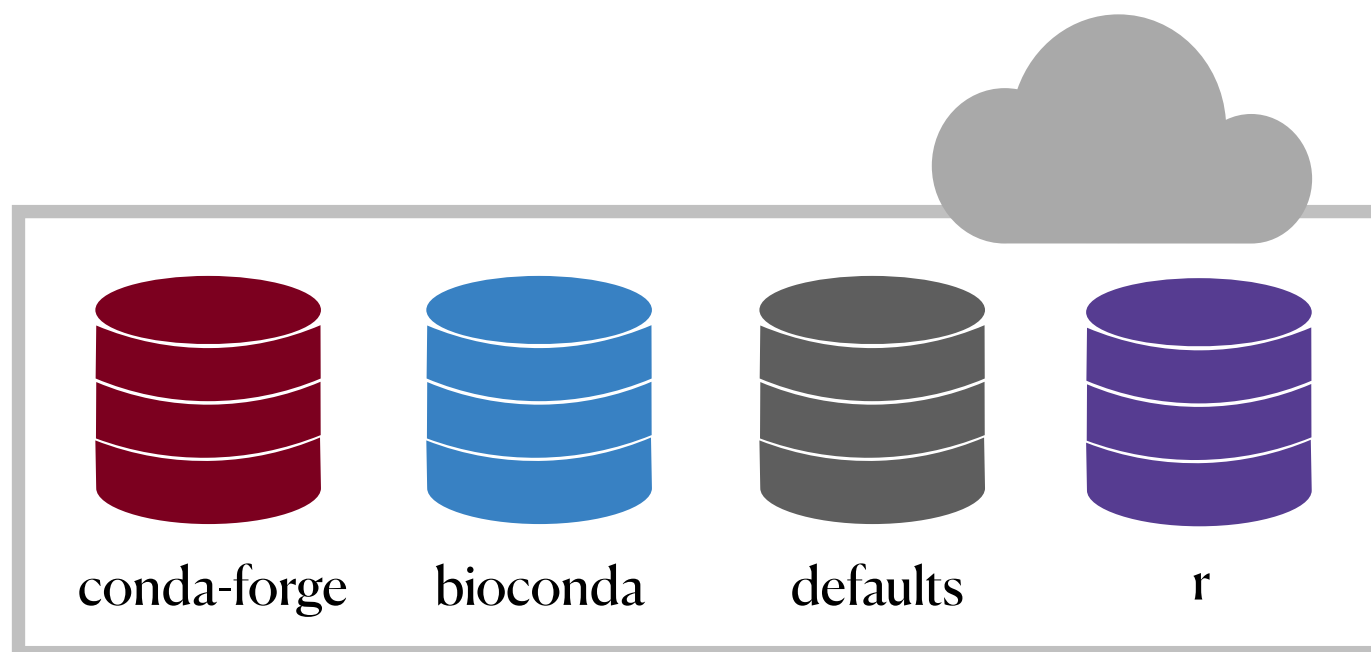
1304 121. **BioStar: An Online Question & Answer Resource for the Bioinformatics Community**
1305 Laurence D. Parnell, Pierre Lindenbaum, Khader Shameer, Giovanni Marco Dall'Olio, Daniel C. Swan,
1306 Lars Juhl Jensen, Simon J. Cockell, Brent S. Pedersen, Mary E. Mangan, Christopher A. Miller, Istvan
1307 Albert
1308 *PLoS Computational Biology* (2011-10-27) <https://doi.org/dhsz4k>
1309 DOI: [10.1371/journal.pcbi.1002216](https://doi.org/10.1371/journal.pcbi.1002216) · PMID: [22046109](https://pubmed.ncbi.nlm.nih.gov/22046109/) · PMCID: [PMC3203049](https://pubmed.ncbi.nlm.nih.gov/PMC3203049/)

- 1310 122. **How to create a Minimal, Reproducible Example - Help Center**
1311 Stack Overflow
1312 <https://stackoverflow.com/help/minimal-reproducible-example>
- 1313 123. **FAQ: How to do a minimal reproducible example (reпреx) for beginners**
1314 RStudio Community
1315 (2020-03-25) [https://community.rstudio.com/t/faq-how-to-do-a-minimal-reproducible-example-reprex-](https://community.rstudio.com/t/faq-how-to-do-a-minimal-reproducible-example-reprex-for-beginners/23061)
1316 [for-beginners/23061](https://community.rstudio.com/t/faq-how-to-do-a-minimal-reproducible-example-reprex-for-beginners/23061)
- 1317 124. **Code of conduct in open source projects**
1318 Parastou Tourani, Bram Adams, Alexander Serebrenik
1319 *Institute of Electrical and Electronics Engineers (IEEE)* (2017-02) <https://doi.org/gfzbs6>
1320 DOI: [10.1109/saner.2017.7884606](https://doi.org/10.1109/saner.2017.7884606)
- 1321 125. **Building a local community of practice in scientific programming for life scientists**
1322 Sarah L. R. Stevens, Mateusz Kuzak, Carlos Martinez, Aurelia Moser, Petra Bleeker, Marc Galland
1323 *PLoS Biology* (2018-11-28) <https://doi.org/gfpwkb>
1324 DOI: [10.1371/journal.pbio.2005561](https://doi.org/10.1371/journal.pbio.2005561) · PMID: [30485260](https://pubmed.ncbi.nlm.nih.gov/30485260/) · PMCID: [PMC6287879](https://pubmed.ncbi.nlm.nih.gov/PMC6287879/)
- 1325 126. Streamlining Data-Intensive Biology With Workflow Systems: Manubot repository
1326 <https://github.com/dib-lab/2020-workflows-paper/tree/09910b4e77280c979f332d6b11a8547fbb305d84>

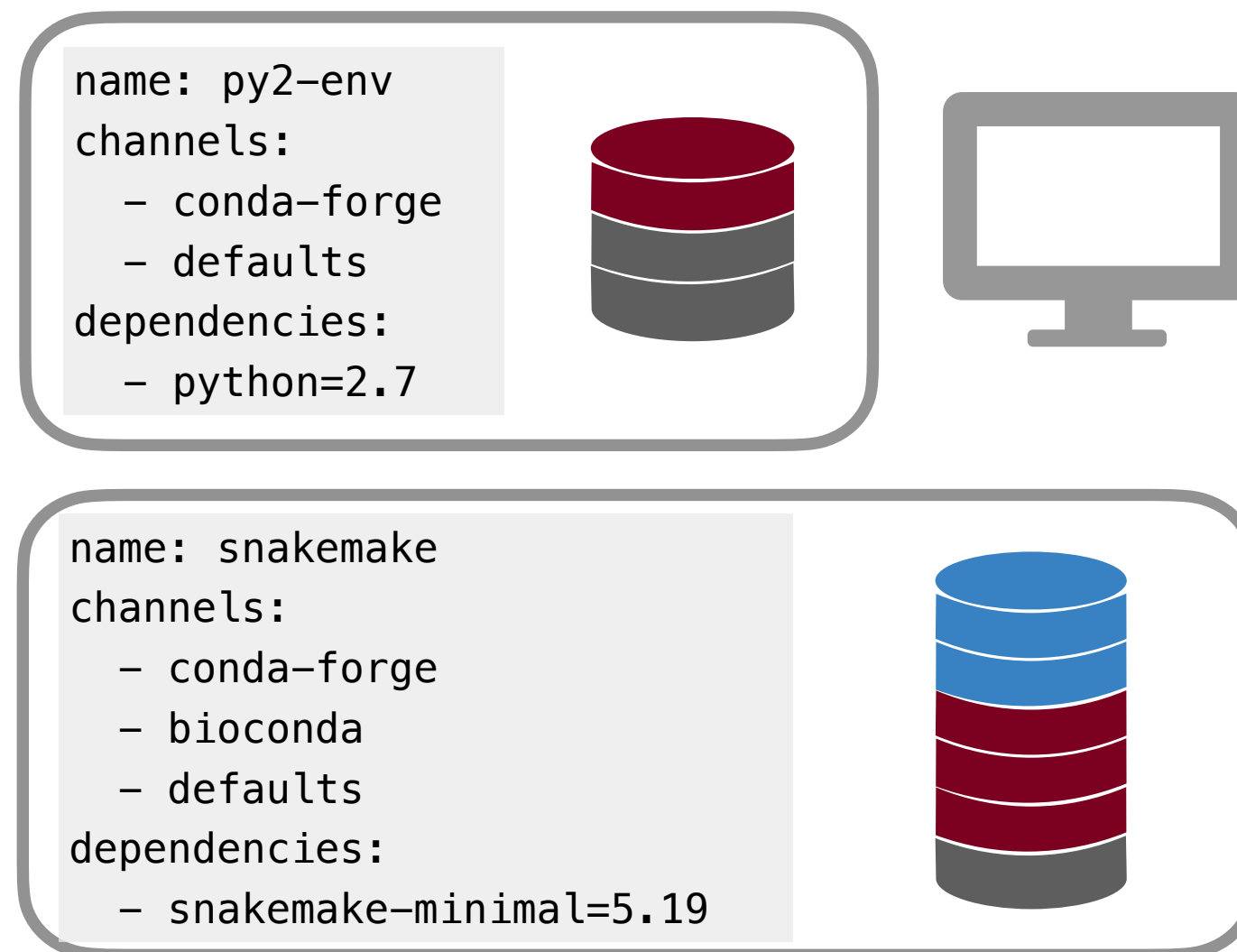


- A. Samples
- B. Software Management
- C. Branching
- D. Parallelization
- E. Ordering
- F. Standard Steps
- G. Rerun as necessary
- H. Reporting
- I. Portable

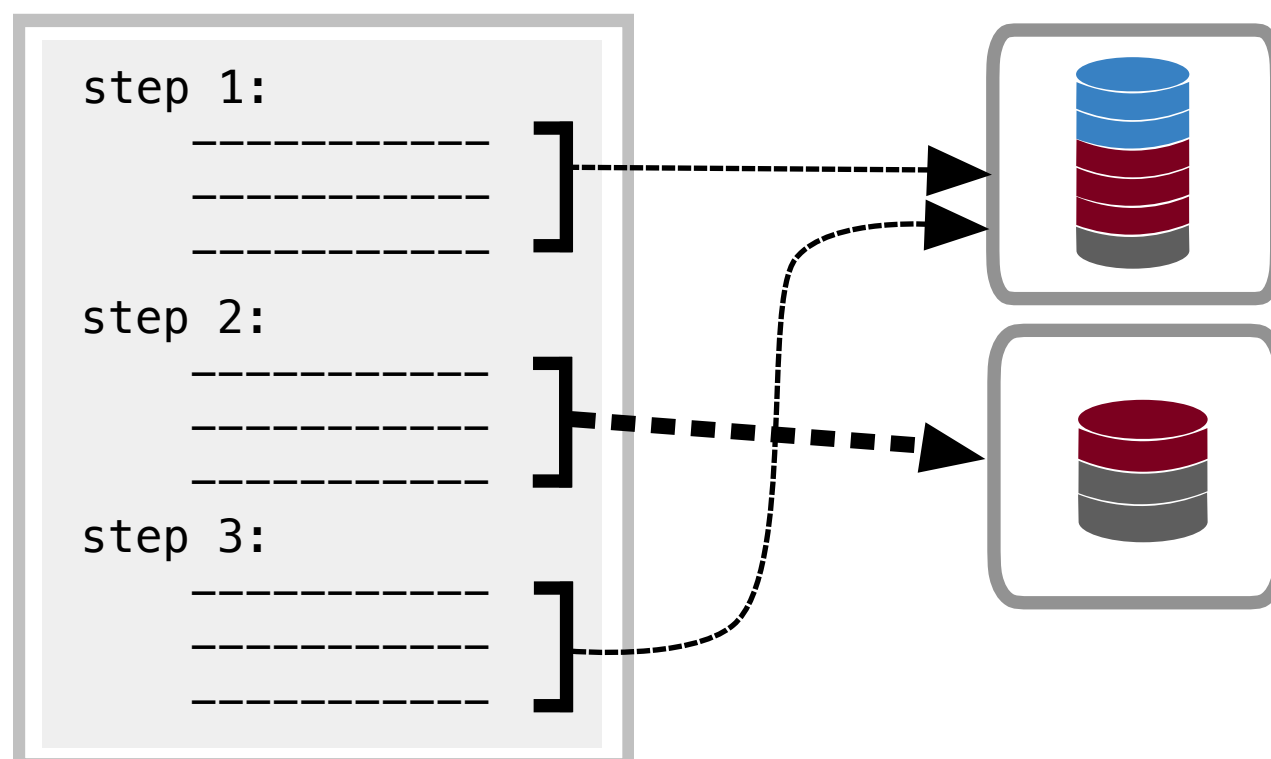
A.

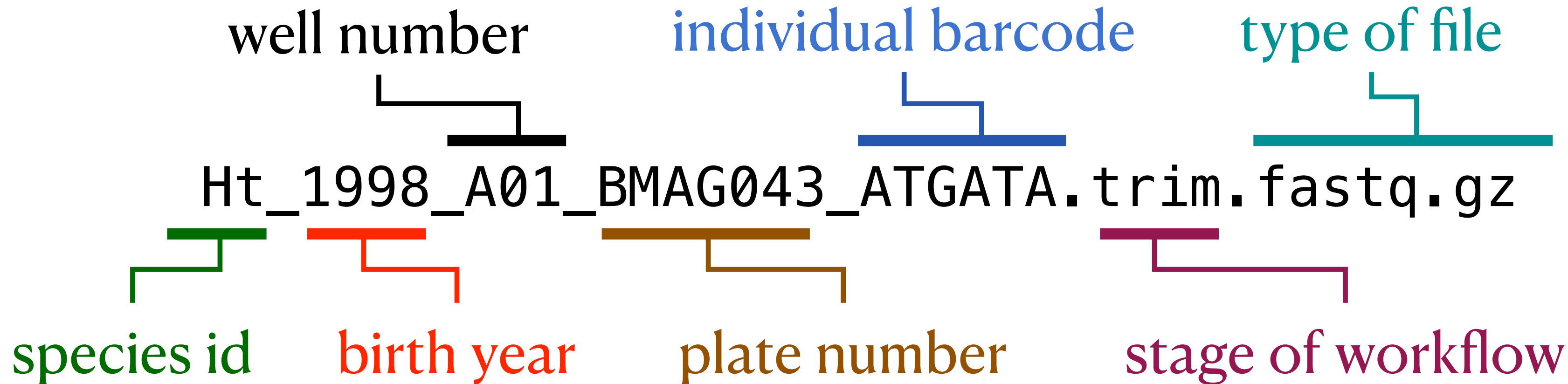


B.



C.





A title: "Distribution of generations in a long-term evolution experiment"

output: html_document

```
---  
{r setup, include=FALSE}  
knitr::opts_chunk$set(echo = TRUE, messages = FALSE, warnings = F)  
---
```

```
{r}  
library(ggplot2)  
---
```

This plot shows the number of samples sequenced at each generation in a long-term evolution experiment.

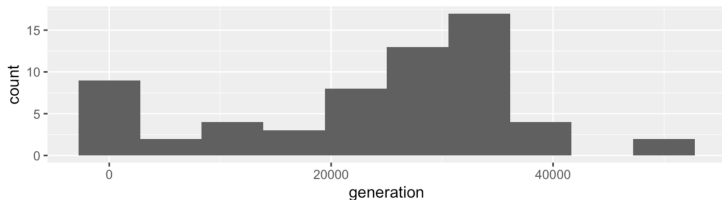
```
{r, fig.height = 2}  
metadata <- read.csv("Ecoli_metadata_composite.tsv", sep = "\t")  
ggplot(metadata, aes(x = generation)) +  
  geom_histogram(bins = 10)  
---
```

B Distribution of generations in a long-term evolution experiment

```
library(ggplot2)
```

This plot shows the number of samples sequenced at each generation in a long-term evolution experiment.

```
metadata <- read.csv("Ecoli_metadata_composite.tsv", sep = "\t")  
ggplot(metadata, aes(x = generation)) +  
  geom_histogram(bins = 10)
```



C Distribution of generations in a long-term evolution experiment

```
In [1]: import pandas as pd  
import matplotlib
```

This plot shows the number of samples sequenced at each generation in a long-term evolution experiment.

```
In [2]: metadata = pd.read_csv("Ecoli_metadata_composite.tsv",  
                               sep = "\t")  
plt = metadata['generation'].plot(kind = "hist")  
plt.set_xlabel("Generation")
```

```
Out[2]: Text(0.5, 0, 'Generation')
```

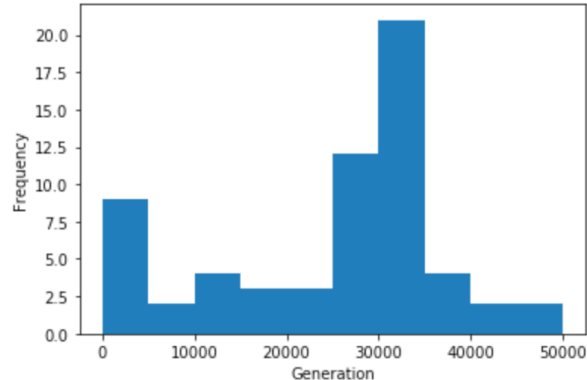
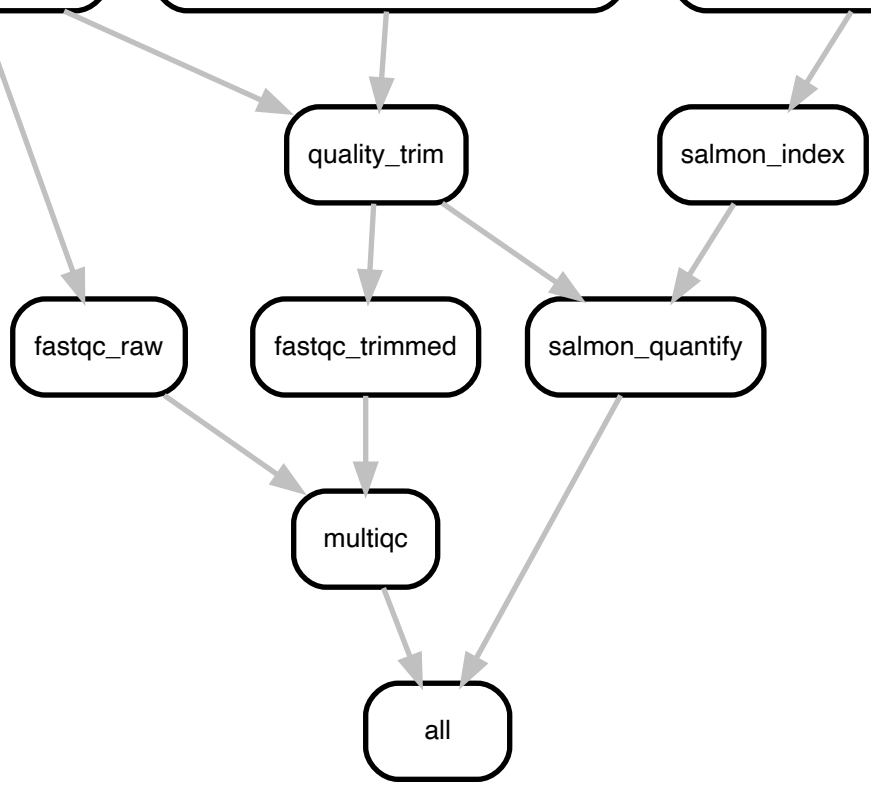


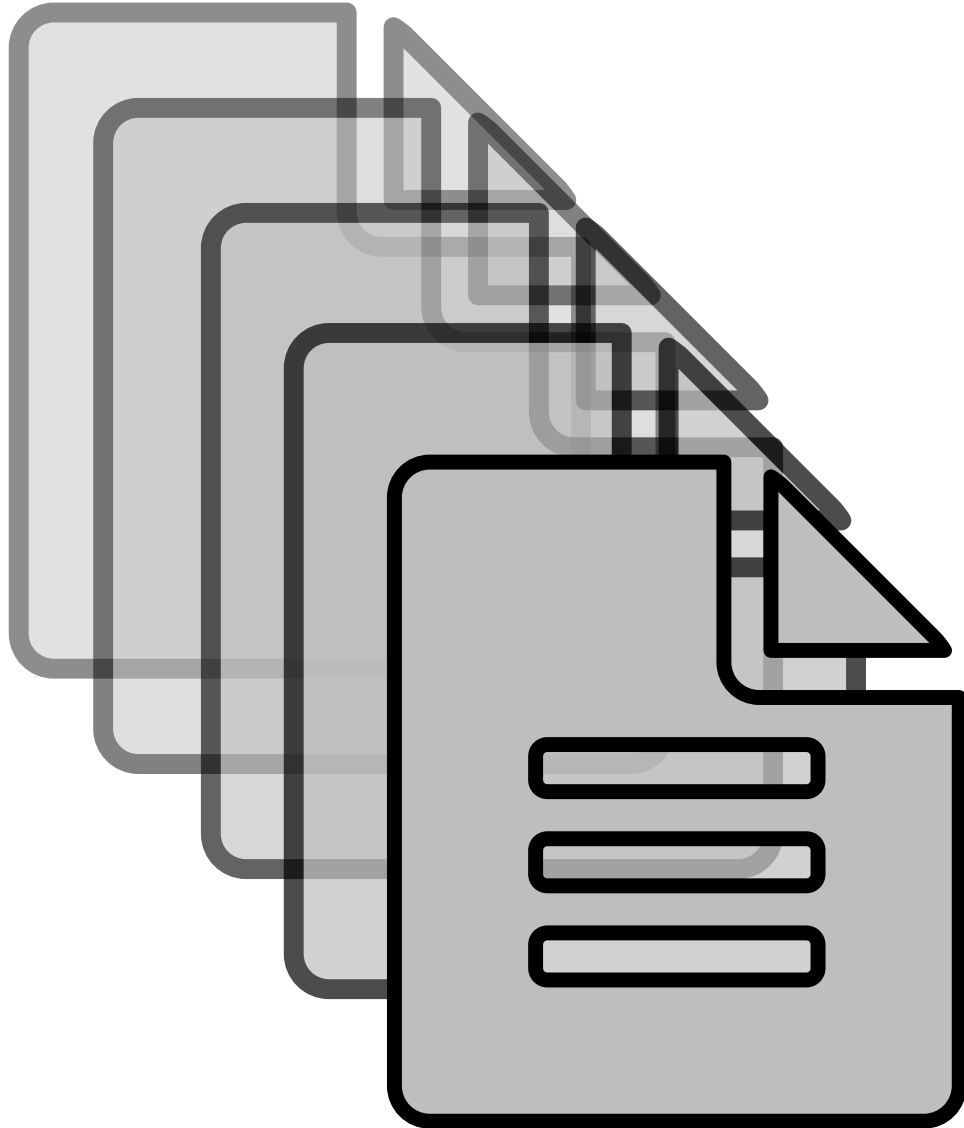
Figure 5: rnaseq dag
download_reads
sample: ERR458493

download_trimmomatic_adapter_file

download_yeast_transcriptome

[Click here to access/download;Figure;Figure5-rnaseq-dag.pdf](#)





```
Hello world  
This code is great  
The best doce ever  
Here is my code
```

commit 4c38c2c



```
Hello world  
This code is great  
The best code ever  
Here is my code
```

commit 5c6c28d

A

Select sample

PT1

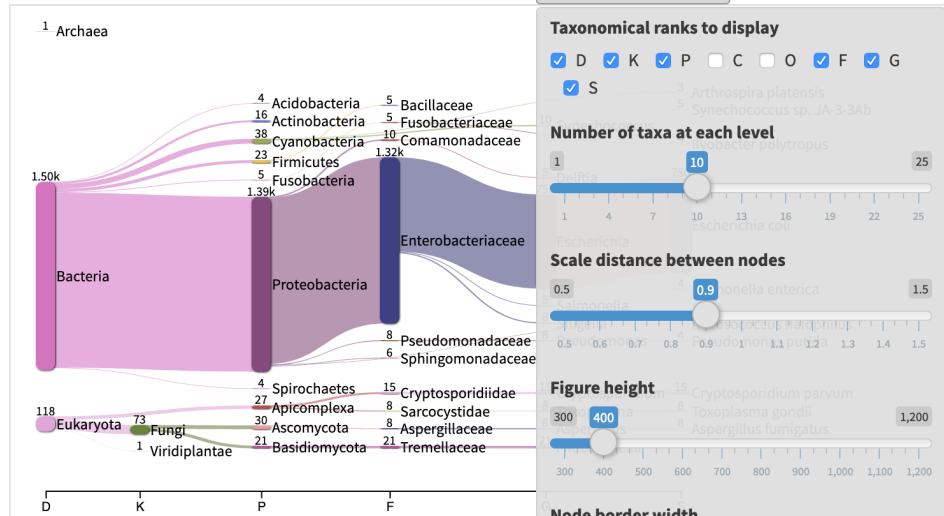
Sankey visualization

Table

Text

Hover over a node to see the abundance of the taxon in other samples.

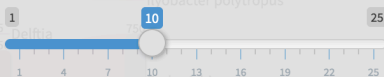
Configure Sankey ...



Taxonomical ranks to display

- D
- K
- P
- C
- O
- F
- G
- S

Number of taxa at each level



Scale distance between nodes

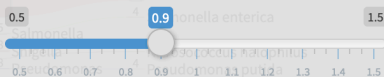
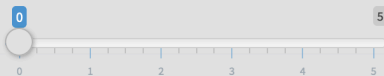


Figure height

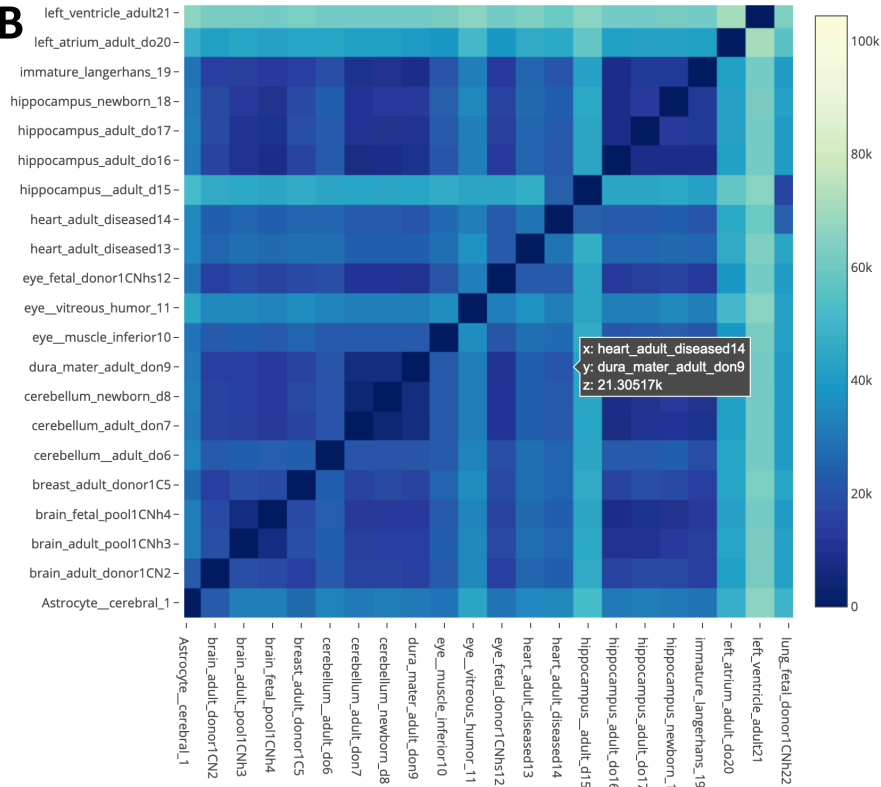


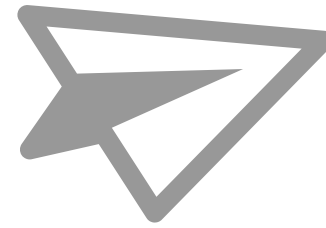
Node border width





Save Network


B






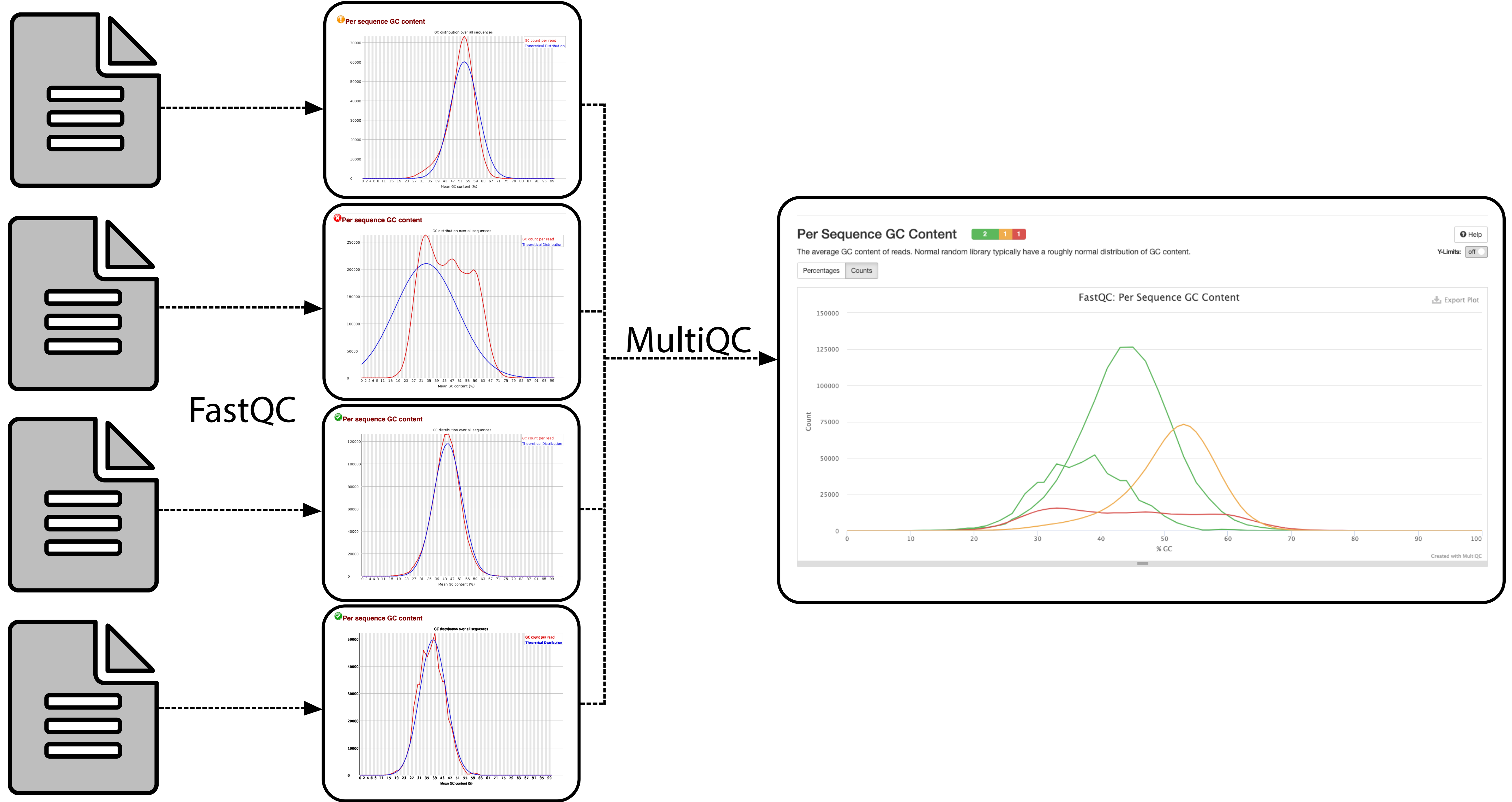
 = 758a89ad4f61

 = 98h1v48wnw3

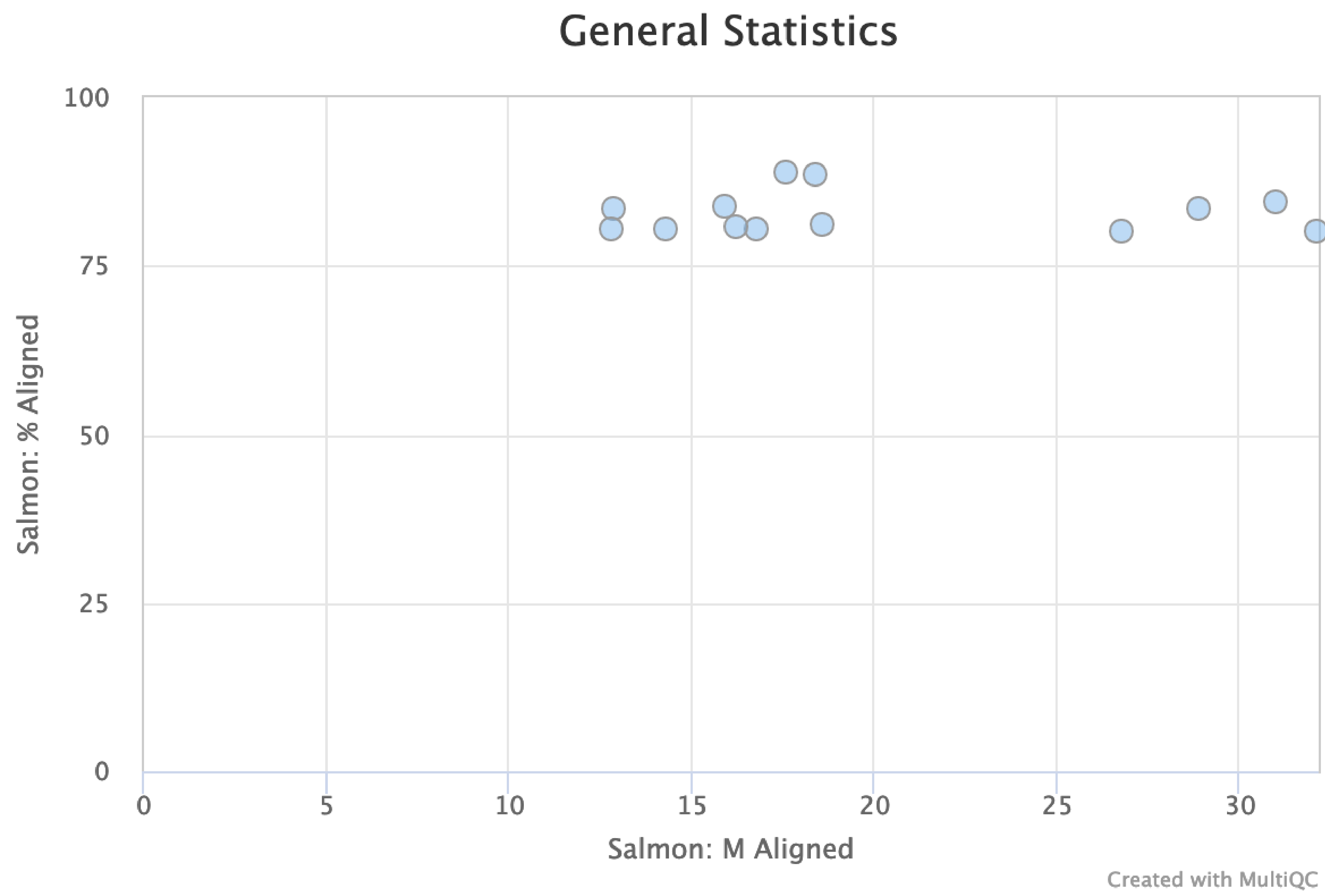
 = 758a89ad4f61 ✓

 = pq98wy4hvps ✗

A.



B.



Dear Editors,

Thank you for the opportunity to submit a revised draft of our manuscript “Streamlining Data-Intensive Biology with Workflow Systems” by Reiter et al. We appreciate the effort you and the reviewers have dedicated to providing valuable feedback for improving the manuscript. In particular, we thank the reviewers for their strong points on managing reproducible scientific software installations. We hope that the edited manuscript better describes the important compromises we see between usability, installability, and reproducibility.

Below, please find a point-by-point response to each review. The manuscript changes are also available on github, perhaps most easily viewed here: <https://github.com/dib-lab/2020-workflows-paper/pull/58/files>.

Reviewer #1:

This manuscript is a review that provides insights into creating workflows and executing them using workflow engines. The authors also provide extensive practical recommendations on performing data-intensive biology. Overall it is easy to understand, and the figures are nice. Below are some changes that I suggest.

1. The Abstract states, “These workflows commonly produce hundreds to thousands of intermediate files.” Can you provide an example of a workflow that would require such a large number of intermediate files? That would help to make this need more concrete.

We have described large workflows and linked to a larger directed acyclic graph (DAG) within the Figure 5 caption.

2. Line 76: “These features ensure that the steps for data analysis are minimally documented...” Consider rephrasing. To me it reads that minimal documentation is desirable. But I think you mean that things should be documented at least to a minimum requirement.

Yes - we have rephrased this sentence to better reflect the point.

3. The paper briefly emphasizes software containers. It puts more emphasize on software-management systems like conda. However, in my opinion, containers are a critical tool and should be emphasized more. Software-management systems sometimes cannot install all of the required dependencies, and not all tools are included in these systems. However, containers provide more flexibility for these types of situations and are supported by all or nearly all workflow management systems.

We have reworked the software management section to better describe the range of software management solutions, including both limitations of conda and benefits and limitations of container-based software installation.

4. Type-o on line 138: “developing”

fixed - thanks!

5. Line 205: “Using software without learning management systems.” I’m not sure that I understand the wording on this. This term has a very specific meaning in education administration (https://en.wikipedia.org/wiki/Learning_management_system), but I think you mean something different. Also, that who section feels a bit disjointed. It was difficult to wrap my mind around exactly what it is trying to say.

We have reworded this section for clarity. We hope it now reflects the scope of software usage and installation options and provides some useful context for how to choose between them.

6. Line 268: Readers may be unclear what “seqanswers” is. Please provide more context.

We removed this unnecessary reference. SeqAnswers is described instead in a later portion of the manuscript.

7. Lines 410-411: This sentence seems unnecessary. Same with lines 420-421.

Removed.

8. Line 474: consider using italics rather than bold text to emphasize this point.

Agreed - changed.

9. Line 502: You reference these repositories in the next paragraph, but it would be better to do so the first time you mention them.

We moved the references here, thanks.

10. Line 509: This statement is somewhat subjective. Consider removing it or providing a more detailed justification.

We have pared down this section.

11. Table 3: Some users may be unfamiliar with what bash is.

Removed reference to bash.

12. Line 638: Should be “Principal Component Analysis”

fixed!

13. The manuscript starts with a focus on using workflow systems. Later it provides a wide range of general recommendations for doing data-intensive biology. Maybe it's just me, but I felt like it got too long and a bit unfocused in the second half. It's up to you, but you might consider splitting it into two manuscripts: one on workflow systems and one on data-intensive biology. Or maybe consider removing/simplifying some of the sections so that it is not so long and is a bit more focused.

We recognize that the manuscript is long, but feel that it provides a comprehensive set of recommendations for researchers to get started using data-intensive workflows for working with sequencing data. In particular, we think that the quality control and data management recommendations we discuss in the latter half of the manuscript are critical for building high-quality, reproducible sequence analysis workflows.

Reviewer #2:

Reiter et al. provides a review on the current state of the data-centric workflows for data analysis tasks in biology. The authors touch all aspects of data analysis tasks, not only the workflow systems but also the whole ecosystem that contains everything from workflow management systems to data integrity and managing computational resources. I generally found the recommendations and the review very useful for the community except one point detailed below.

I understand this manuscript is based on personal experiences as authors acknowledged in their summary. However, in my opinion, this review misses important developments in reproducibility that are compatible with the main recommendations of the review.

The authors mention software wrangling as a crucial part in scientific reproducibility assuming that the initial data is intact and available. They mention Conda, Singularity, and Docker as methods to manage software for reproducibility. However, we see reproducibility as a spectrum. A fully reproducible workflow would have the following ingredients assuming the data is available: 1) code and usage transparency, 2) installability and 3) reproduction of runtime environment. The authors give reasonable recommendations for the code and usage transparency which is mainly making sure that the code is documented to the highest quality, and available publicly. Conda, Singularity and Docker remedies the installation problem. They make dependencies easily installable in most cases. However, authors do not mention short-comings of these solutions. The main short-coming of those tools is that they don't fully satisfy reproduction of the runtime environment if they do so in the case of docker/singularity they do this opaquely and are not transparent. It

is hard to verify exactly what are the contents of a container. Docker recommends use of docker files but they do not necessarily have the version of the software that is in the container. It is mostly a collection of commands installing software from package managers. Which brings us to the same problems I describe for conda below. These commands often include invocations of package managers like Apt (in the case of Debian-derived foundations). A package installed via apt today will *not* be identical to the same package installed a year ago. Containers are also harder to maintain if you do not analyze data and develop code on dockerized environments exclusively.

Conda packages, on the other hand, do not provide reproducibility at all. You can get different binaries for the same name+version query at different times and there is no way to track which source files of dependencies produced that binary. The system environment where the software is built is not isolated. During the build, processes have access to other libraries that are not in the package recipe and also conda assumes certain low-level packages to be available in all environments.

In essence, the authors don't mention that *all* the tools they recommend (conda, docker and singularity) are completely time-dependent. Some Conda channels provide archives of pre-built binaries, yes, but since not all dependencies (beyond the kernel) are taken into account at build time you will *not* be able to run these binaries without changes on a different system.

There are package managers like GNU Guix remedies most of these problems and provide the state-of-the-art reproducibility exemplified by the recent "Ten Years Reproducibility Challenge" (<https://www.nature.com/articles/d41586-020-02462-7>). This package management system is also incorporated in snakeMake-based pipelines providing gold-standard reproducibility for multiple NGS analysis pipelines (<https://academic.oup.com/gigascience/article/7/12/giy123/5114263>).

Altuna Akalin & Ricardo Wurmus

We strongly agree that reproducibility is a spectrum, and thank the reviewers for their thoughtful discussion and references. In response, we've extensively revised the software management portion of the manuscript, detailing the reproducibility limitations of each approach and adding information on functional package managers like GNU guix and Nix. Additionally, we have added some information on choosing between these systems that will hopefully convey the merits of each.

In our experience, software management systems currently present significant trade-offs between usability, installability, and reproducibility. In providing recommendations to the field, we feel it is important to prioritize usability, which is a critical determinant for widespread adoption of computational techniques.

Systems that provide 100% reproducibility but are not straightforward to implement are not a viable solution for researchers with limited computational training or experience.

While conda is not currently the best system for producing long-term (10-year) reproducibility, we feel that it provides a reasonable balance between usability and reproducibility, particularly when used according to the provided recommendations (small, isolated, version-pinned environments are critical for both usability and reproducibility). Furthermore, broad community adoption has resulted in proliferation of conda-installable tools and helps ensure that the ecosystem around these tools will be maintained and improved over the coming years.

The expanded text on software management systems is reproduced here:

“On the lightweight end, the conda package manager has emerged as a leading software management solution for research workflows (Figure 2). Conda handles both cluster permission and version conflict issues with a user-based software environment system, and features a straightforward “recipe” system which simplifies the process of making new software installable (including simple management of versions and updates). These features have led to widespread adoption within the bioinformatics community: packages for new software become quickly available, and can be installed easily across platforms. However, conda does not completely isolate software installations and aims neither for bitwise reproducibility nor long-term archiving of install packages, meaning installations will not be completely reproducible over time. Heavyweight software management systems package not only the software of interest, but also the runtime environment information, with the goal of ensuring perfect reproducibility in software installation over time. Tools such as singularity and docker [3,11,37,38] wrap software environments in “containers” that capture and reproduce the runtime environment information. Container-based management is particularly useful for systems where some dependencies may not be installable by lightweight managers. However, software installation within these containers can be limited by similar reproducibility issues, including changes in dependency installations over time. “Functional package managers” such as GNU Guix and Nix strictly require all dependency and configuration details be encoded within each software package, providing the most comprehensively reproducible installations. These have begun to be integrated into some bioinformatic tools [16], but have a steeper learning curve for independent use. In addition, standard installation of these managers requires system-wide installation permissions, requiring assistance from system administrators on most high-performance computing systems.”

All changes can be most easily seen here: <https://github.com/dib-lab/2020->

[workflows-paper/pull/58/files](#).

Overall, we hope the revised manuscript reflects the important points the reviewers have raised while still emphasizing accessible techniques that will move researchers towards fully automated, reproducible analyses.

Sincerely,

N. Tessa Pierce, corresponding author