Streamlining data-intensive biology with workflow systems

This manuscript (permalink) was automatically generated from bluegenes/2020-gep@9d31aa8 on May 18, 2020.

Authors

- **Taylor Reiter**
  0000-0002-7388-421X · taylorreiter · ReiterTaylor
  Department of Population Health and Reproduction, University of California, Davis · Funded by Grant XXXXXXXX

- **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

Abstract

As both sequencing technologies and data have proliferated, the bottleneck of biological sequence analysis has shifted from data generation to analysis. Fortunately, analysis tools and techniques have evolved to cope with this ever-increasing flood of data. Reliable and user-friendly workflow systems and software management have emerged to facilitate interrogation of many thousands of samples. For fundamental steps such as quality control, standardized protocols are now available meaning researchers can spend less time rewriting common analyses and more time examining the biological intricacies of their data. In cases where the data are too large for even high-performance computing environments, a series of tools have emerged that are capable of using small, representative subsets of massive datasets to produce comparable results. While adoption of these tools can both facilitate and expedite reproducible data analysis, knowledge of and training in these techniques is still lacking. Here, we provide a series of tips, tools, and “good enough” practices for leveraging workflow systems to streamline biological analysis.

Author Summary

In this paper, we present our guide for workflow-enabled biological sequence data analysis, developed through our own teaching, training and analysis. We recognize that this is currently biased towards our own use cases and experiences, but we hope to engage in robust discussion with the open source community to build a comprehensive resource. Our main goal is to accelerate scientists conducting sequence analyses into organized workflow practices that benefit their own research while also facilitating open and reproducible science.

Introduction

(draft)

Sequencing data are now widely available for species across the tree of life, and new sequencing data continues to be generated at an incredible rate [1]. The wealth of information present in high-throughput sequencing data has already revolutionized our understanding of the diversity and function of organisms and communities, building basic understanding from ecosystems to human health.

As sequencing analysis has matured over the past decade, several papers have presented "best" or "good enough" practices for computational biological analyses [2,3,4]. These recommendations have both helped build consensus and fueled additional tool and workflow development. Since the latest paper in 2017 [4], key advancements in workflow scripting, software management, and tools that handle biological data at scale have vastly increased the power of analysis management. In particular, the emergence of bioinformatics-focused workflow systems has empowered biologists to analyze biological data at scale.

Bioinformatics-focused workflow systems and software management tools contain powerful infrastructure for task and tool management and can self-monitor progress, resource usage, and dependencies. When paired with proper software management, fully-contained workflows are scaleable, robust to software updates, and executable across platforms. These workflow systems have quickly become the workhorses of modern bioinformatics, enabling researchers to build workflows that execute dozens of analysis tools in a systematic manner across all experimental samples, producing hundreds to thousands of intermediate files for a single analysis. In concert, sharing of data analysis lessons and workflows has created a critical mass of analysis code now openly available for research and training, including that done by nonprofit organizations such as the Carpentries [5].

In our experiences with both research and training biologists in computational techniques, workflow systems have greatly reduced the barrier to entry for data analysis at scale and opened the door to end-to-end reproducible analyses. Here, we present some tools, strategies, and "good enough" practices for leveraging workflow systems to streamline data-intensive biology and to enhance the documentation, automation and reproducibility of your science.

Workflows facilitate data-intensive biology

Sequence analyses require researchers to build workflows that synthesize multiple analytic tools and execute them in a systematic manner across all experimental samples. These workflows commonly produce hundreds to thousands of intermediate files and require incremental changes as experimental insights demand tool and parameter modifications. Managing these steps can be both time-consuming and error-prone, even when automated using scripting languages (e.g. bash).

The emergence and maturation of workflow systems designed with bioinformatic challenges in mind has revolutionized computing in data intensive biology [6]. Workflow systems contain powerful infrastructure for workflow management that can coordinate runtime behavior, self-monitor progress and resource usage, and compile reports documenting the results of a workflow (**Figure 1**). These features ensure that the steps for data analysis are documented and repeatable from start to finish. When paired with proper software management, fully-contained workflows are scaleable, robust to software updates, and executable across platforms, meaning they will likely still execute the same set of commands with little investment by the user in weeks, months, or years from the time of writing.

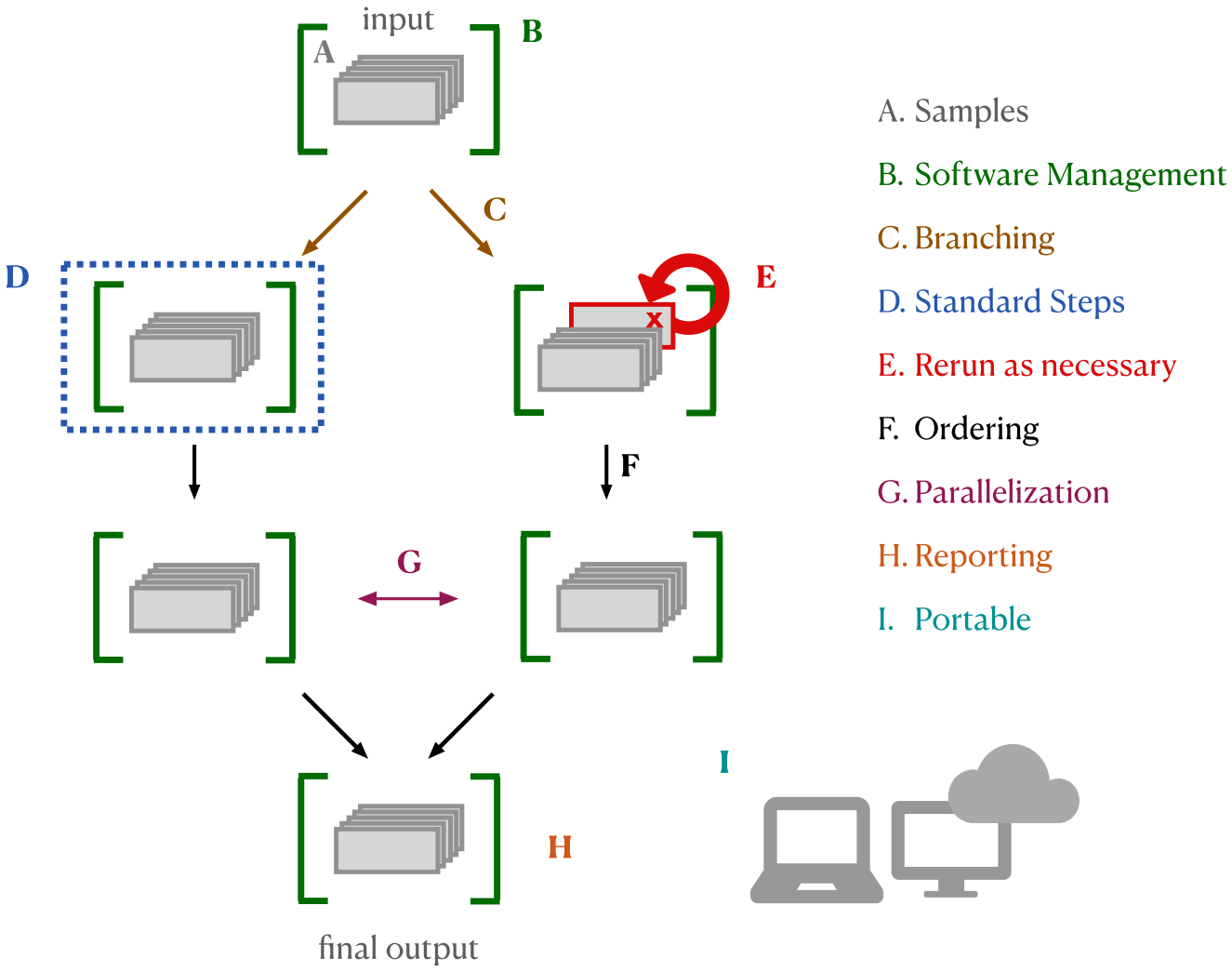

Figure 1: Workflow Systems Bioinformatic workflow systems have built-in functionality that facilitate/streamline/simplify running analysis pipelines. **A. Samples** Workflow systems enable you to use the same code to run each step on each sample. Samples can be easily added if the analysis expands. **B. Software Management** Integration with software management tools (e.g. conda, singularity) can automate software installation for each step. **C. Branching, F. Ordering, G. Parallelization** Workflow systems ensure tasks are executed in the correct order for each sample file, and can automatically execute independent steps in parallel. **D.Standard Steps** Many steps are now considered "standard" (e.g. quality control). Workflow languages keep all information for a step together and can be written to enable you to remix and reuse each individual step across pipelines. **E. Rerun as necessary** Workflow systems keep track of which steps executed properly and on which samples, and allow you to rerun failed steps (or additional steps) rather than re-executing the entire workflow. **H. Reporting** Workflow languages enable comprehensive reporting on workflow execution and resource utilization by each tool. **I. Portability** Analyses written in workflow languages (with integrated software management) can be run across computing systems without changes to code.

In order to properly direct a workflow, workflow systems need to encode information about the relationships between workflow steps. In practice, this means that each analysis step must specify the input (or types of inputs) needed for that step, and the output (or types of outputs) being produced. This structure results in several added benefits, beyond those mentioned above. First, workflows become self-documented; the directed graph produced by workflow systems can be exported and

visualized, producing a graphical representation of the relationships between all steps in a pipeline (see **Figure 5**). Next, workflows are more likely to be fully enclosed without undocumented steps that are executed by hand, meaning analyses are more likely to be reproducible. Finally, each step becomes a self-contained unit that can be used and re-used across multiple analysis workflows, so scientists can spend less time implementing standard steps, and more time on their specific research questions. In sum, the internal scaffolding provided by workflow systems helps build analyses that are generally better documented, repeatable, transferable, and scalable.

Choosing a workflow system

While the benefits of encoding a workflow in a workflow system are immense, the learning curve associated with implementing complete workflows in a new syntax can be daunting. It is possible to obtain the benefits of workflow systems without learning a workflow system. Websites like Galaxy, Cavatica, and EMBL-EBI MGnify offer online portals in which users build workflows around publicly-available or user-uploaded data [7,8,9]. On the command line, many research groups have used workflow systems to build user-friendly pipelines that do not require learning or working with the underlying workflow software. These tools are specified in an underlying workflow language, but are packaged in a more user-friendly manner command-line script that coordinates and executes the workflow. Rather than writing each workflow step, the user can specify data and parameters in a configuration file to customize the run. Some examples include the nf-core RNA-seq pipeline [10,11], the ATLAS metagenome assembly and binning pipeline [12,13], the Sunbeam metagenome analysis pipeline [14,15], and two from our own lab, the Elvers *de novo* transcriptome and differential expression pipeline [16], and dammit eukaryotic transcriptome annotation pipeline [17]. These tools allow users to take advantage of the benefits of workflow software without needing to invest in curating and writing their own pipeline. They are designed to execute a series of standard steps, and provide varying degrees of customizability.

While these systems can be incredibly powerful and versatile, bioinformatic analyses are highly dependent on experimental design, and researchers often need to build custom workflows or steps for their analyses. At this time, there are several scriptable workflow systems that offer similar benefits for data intensive biology. Each has it own strengths, meaning each software will meet an individuals computing goals differently (see **Table 1**). Our lab has adopted Snakemake, in part due to its similarity and integration with Python, its flexibility for building and testing new analyses in different languages, and its intuitive integration with software management tools (SEE SECTION XXX) [18]. Software like Nextflow and Common Workflow Language require slightly more infrastructure, but in return, scale much better to pipelines with hundreds of thousands of steps and support containerization more rigidly, making them ideal for production-level pipelines [19,20]. Language-specific workflow systems, such as ROpenSci's Drake [21], are limited in the scope of tasks they can execute, but are powerful within their language and easier to integrate if comfortable in that language.

Table 1: Popular bioinformatics workflow systems, documentation, example workflows, and tutorials. While we have linked to general tutorials, there may be more relevant tutorials online for your field. Not all workflow systems are not necessarily exclusive entities: some workflow systems can translate workflows between languages or run tools or tasks written in other languages.

Workflow System	Documentation	Example Workflow	Tutorial
Snakemake	https://snakemake.readthedocs.io/	https://github.com/snakemake-workflows/chipseq	https://snakemake.readthedocs.io/en/stable/tutorial/tutorial.html
Nextflow	https://www.nextflow.io/	https://github.com/nf-core/sarek	https://www.nextflow.io/docs/latest/getstarted.html

Workflow System	Documentation	Example Workflow	Tutorial
Common workflow language	https://www.commonwl.org/	https://github.com/EBI-Metagenomics/pipeline-v5	https://www.commonwl.org/user_guide/02-1st-example/index.html
Workflow description language	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

The best workflow system to adopt may be the one with a strong and accessible local or online community in your field, somewhat independent of your computational needs. The availability of field-specific data analysis code for reuse and modification, as well as community support for new users can facilitate the adoption process. Fortunately, the standardized syntax required by workflow systems, combined with widespread adoption in the open science community, has resulted in a proliferation of open access workflow-system code for routine analysis steps [22,23]. At the same time, consensus approaches for data analysis are emerging, further encouraging reuse of existing code [24,25,26,27,28].

Getting started with workflows

The first place to start with any workflow language is the tutorials and available code. We have linked some examples in **Table 1**, but you may be able to find tutorials that are specific to your analysis or field. After working through at least one tutorial, you can move to running example workflows that somewhat closely resemble the tool or task you need to accomplish. Read through the script with the system documentation on hand in order to make sense of the structure. If available, run the test data to see what the output will look like. The “Getting started developing workflows” section will contain strategies for developing and modifying workflows for your own analyses.

Wrangling Scientific Software

Analysis workflows rely on multiple software packages to generate final results. These tools are heterogeneous in nature: written by researchers working in different coding languages, with varying types software design and optimization, and often for specific analysis goals. Each program has a number of other programs it depends upon to function (“dependencies”), and as software changes over time to meet research needs, the results may change, even when run with identical parameters. As a result, it is critical to take an organized approach to installing, managing, and keeping track of software and software versions. To meet this need, most workflow managers integrate with software management systems like conda, singularity, and docker [**???**,29,30].

Software management systems perform some combination of software installation, management, and packaging that alleviate problems that arise from dependencies and that facilitate documentation of software versions. On many systems, system-wide software management is overseen by system administrators, who ensure commonly-used and requested software is installed into a “module” system available to all users. Unfortunately, this system does not lend itself well for exploring new workflows and software, as researchers do not have permission to install software themselves. The Conda package manager has emerged as a leading software installation and management solution, largely because it 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 (**Figure 2**). Conda enables lightweight software installation and can be used with the same commands across platforms. Alternatively, container solutions like docker and singularity allow for for the entire computational environment to be captured and

distributed, including the operating system. This ensures that an environment is completely reproducible, and is common for production workflows.

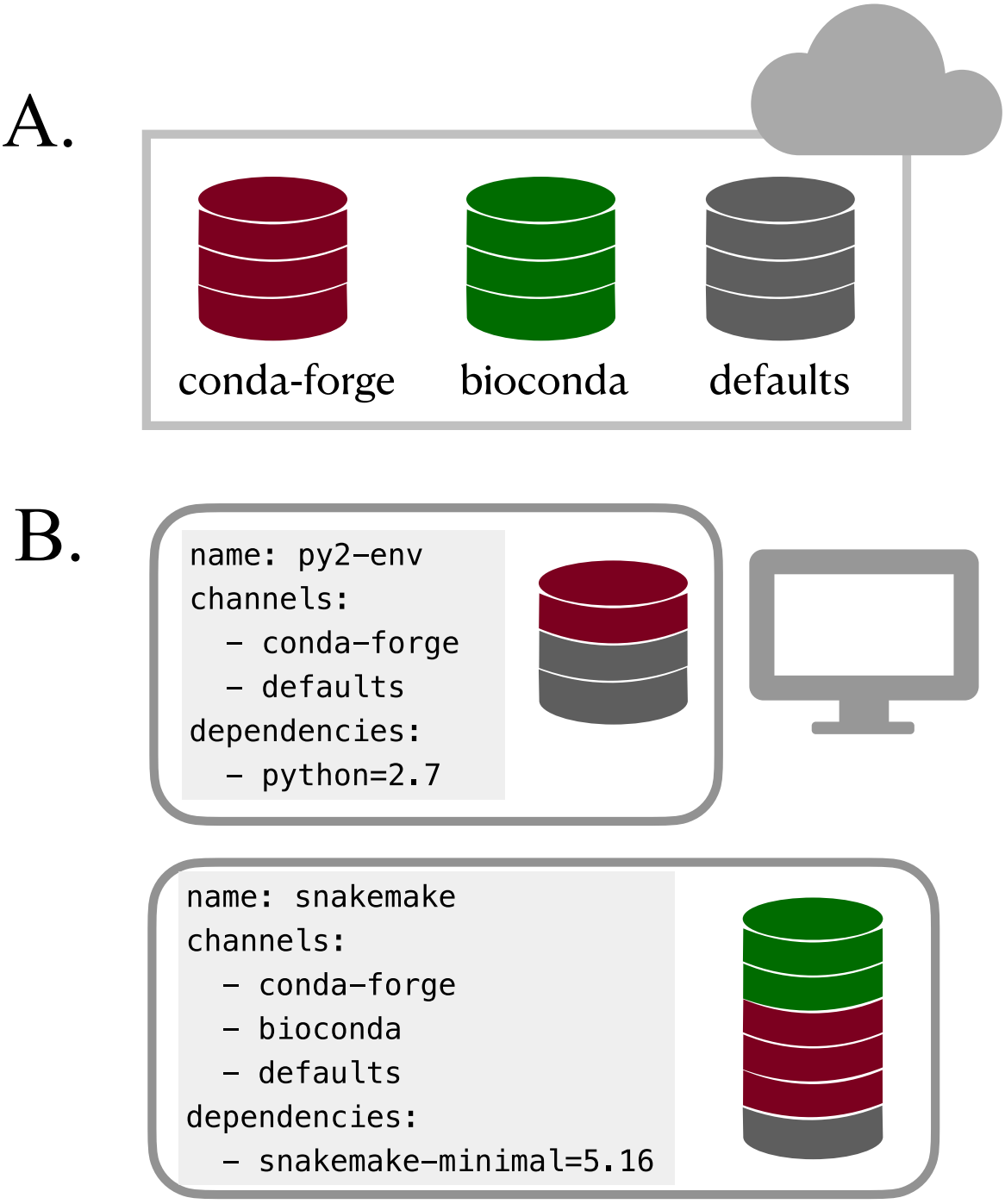

Figure 2: The conda package and environment manager simplifies software installation and management. A. Conda Recipe Repositories Each program distributed via Conda has a "recipe" describing all software dependencies needed for Conda installation (each of which must also be installable via conda). These are stored and managed in separate "channels", some of which specialize (e.g. "bioconda" specializes in bioinformatic software, "r" specializes in R language packages) [6]. **B. Use Conda Environments to Avoid Installation Conflicts** Conda does not require root privileges for software installation, thus enabling use by researchers working on shared cluster systems. However, even user-based software installation can encounter dependency conflicts. For example, you might need to use python2 to install and run a program (e.g. older scripts written by members of your lab), while also using snakemake to execute your workflows (requires python>=3.5). By installing each program into an isolated "environment" that contains only the software required to run that program, you can ensure all programs will run without issue. Using small, separate environments for your software and building many simple environments to accommodate different steps in your

workflow also reduces the amount of time it takes conda to resolve dependency conflicts between different software tools ("solve" an environment). Conda virtual environments can be created and installed either on the command line, or via an environment YAML file, as shown. In this case, the environment file also specifies which Conda channels to search and download programs from. When specified in a YAML file, conda environments are easily transferable between computers and operating systems. Further, because the version of each package installed in an environment is recorded, workflow reproducibility is enhanced. Although portions of conda may be superceded by alternative solutions [31], this model of software installation and management will likely remain.

Getting started with software management

Using these software management tools within workflow systems is relatively straightforward: install the manager if needed, then add a line to each workflow step specifying information for software installation. (see conda environment example in **Figure 2, B**). When the workflow is executed, the specified environment or container will be downloaded and installed (or system module will be loaded), used to run that analysis step, and stored for future analysis runs. The specific syntax for using package managers with your chosen workflow system will be available in the documentation.

While package managers and containers greatly increase reproducibility, there are a number of ways to test software before (or without ever) needing to worry about installation. Some software packages are available as web-based tools and through a series of data upload and parameter specifications, allow the user to interact with a tool that is running on a back-end server. This approach is ideal for testing a tool prior to installation to determine whether it produces an appropriate or useful output on your data. Integrated development environments like PyCharm and RStudio can also manage software installation for language-specific tools. In our experience, the complete solution for using scientific software involves a combination of conda with these tools for developing new analyses, as well as workflow-integrated software management via conda, singularity, or docker for executing full workflows on many samples.

Workflow-Based Project Management

Computational project management is a learned skill that takes time to implement. Biological analyses often span hundreds of steps and involve a myriad of decisions ranging from small-scale tool and parameter choices to larger-scale decisions around computational experimental design and statistical analyses. It is rare (if not impossible) to build or find a workflow that analyzes your data from start to finish without testing, troubleshooting, and iteration. Fortunately, with a little direction, workflow systems both simplify and improve computational project management.

Systematically document your workflows

Workflow systems link each analysis step, so that downstream files are regenerated if upstream files and parameters change, as long as you re-run the workflow manager each time (steps will only be rerun if necessary). In this way, the coded portions of workflows can be self-documenting, with each analysis step (and parameters) completely specified. The simplest way to document your workflow, then, is to include as much of it as possible within the automated workflow framework. However, this is limited in a few ways. First, there may be portions of the workflow that you have not been able to automate, such as downloads of data from protected servers (e.g. containing protected identity information), or manual data and metadata cleaning steps. Second, some steps may take a long time to rerun, so you may end up breaking your workflow into several independently-executed steps. Third, even if the code and parameters are documented, it is critical to record the reasoning behind each particular analysis decision, so that you can assess the influence of each choice on the downstream results.

Use consistent, self-documenting names

For analysis workflows, it is useful to be able to look at a file and understand exactly which sample it belongs to, and which processing or analysis steps were used to generate it. Using consistent and descriptive identifiers for your files, scripts, variables, workflows, projects, and even manuscripts helps keep your projects organized and interpretable for yourself and collaborators. For workflow systems, this strategy can be implemented by tagging output files with a descriptive identifier for each analysis step, either in the filename or by placing output files within a descriptive output folder. For example, the file shown in **Figure** 3 has been preprocessed with a quality control trimming step. For large workflows, placing results from each step of your analysis in isolated, descriptive folders can be essential for keeping your project workspace clean and organized.

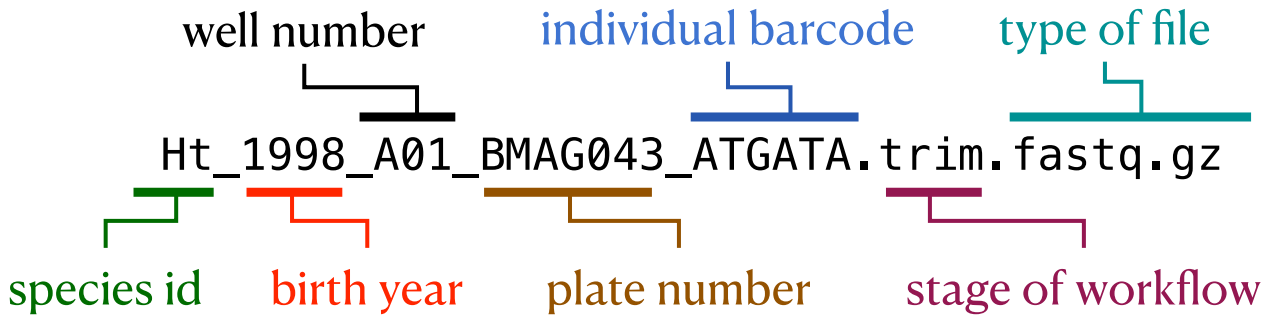

Figure 3: Since the number of files in data-intensive biology can quickly get out of hand, consistent file naming is especially important. It is useful to keep unique sample identification in the filename, often with a metadata file explaining the meaning of each unique descriptor. For analysis scripts, it can help to implement a numbering scheme, where the name of first file in the analysis begins with "00", the next with "01", etc. For output files, it can help to add a short, unique identifier to output files processed with each analysis step. This particular file is a RADsequencing fastq file of a fish species that has been preprocessed with a fastq quality trimming tool (courtesy of Shannon Joslin).

Store workflow metadata with the workflow

Developing biological analysis workflows can involve hundreds of small decisions: What parameters work best for each step? Why did you use a certain reference file for annotation as compared with other available files? How did you finally manage to get around the program or installation error? All of these pieces of information contextualize your results and may be helpful when writing your manuscript. Keeping information about these decisions in an intuitive and easily accessible place helps you find it when you need it. To capitalize on the utility of version control systems described below, it is most useful to store this information in plain text files. Each main directory should include notes on the data or scripts contained within, so that a collaborator could look into the directory and understand what to find there (especially since that "collaborator" is likely to be you, a few months from now!). Code itself can contain documentation - you can include comments with the reasoning behind algorithm choice or include a link to the seqanswers post that helped you decide how to shape your differential expression analysis. Larger pieces of information can be kept in "README" or notes documents kept alongside your code and other documents. For example, a GitHub repository documenting the reanalysis of the Marine Microbial Eukaryote Transcriptome Sequencing Project uses a README alongside the code to document the workflow and digital object identifiers for data products [32,33]. While this particular strategy cannot be automated, it is critical for interpreting the final results of your workflow.

Document data and analysis exploration using computational notebooks

Computational notebooks allow users to combine narrative, code, and code output (e.g. visualizations) in a single location, enabling the user to conduct analysis and visually assess the results in a single file (see **Figure** 4). These notebooks allow for fully documented iterative analysis

development, and are particularly useful for data exploration and developing visualizations prior to integration into a workflow or as a report generated by a workflow that can be shared with collaborators.

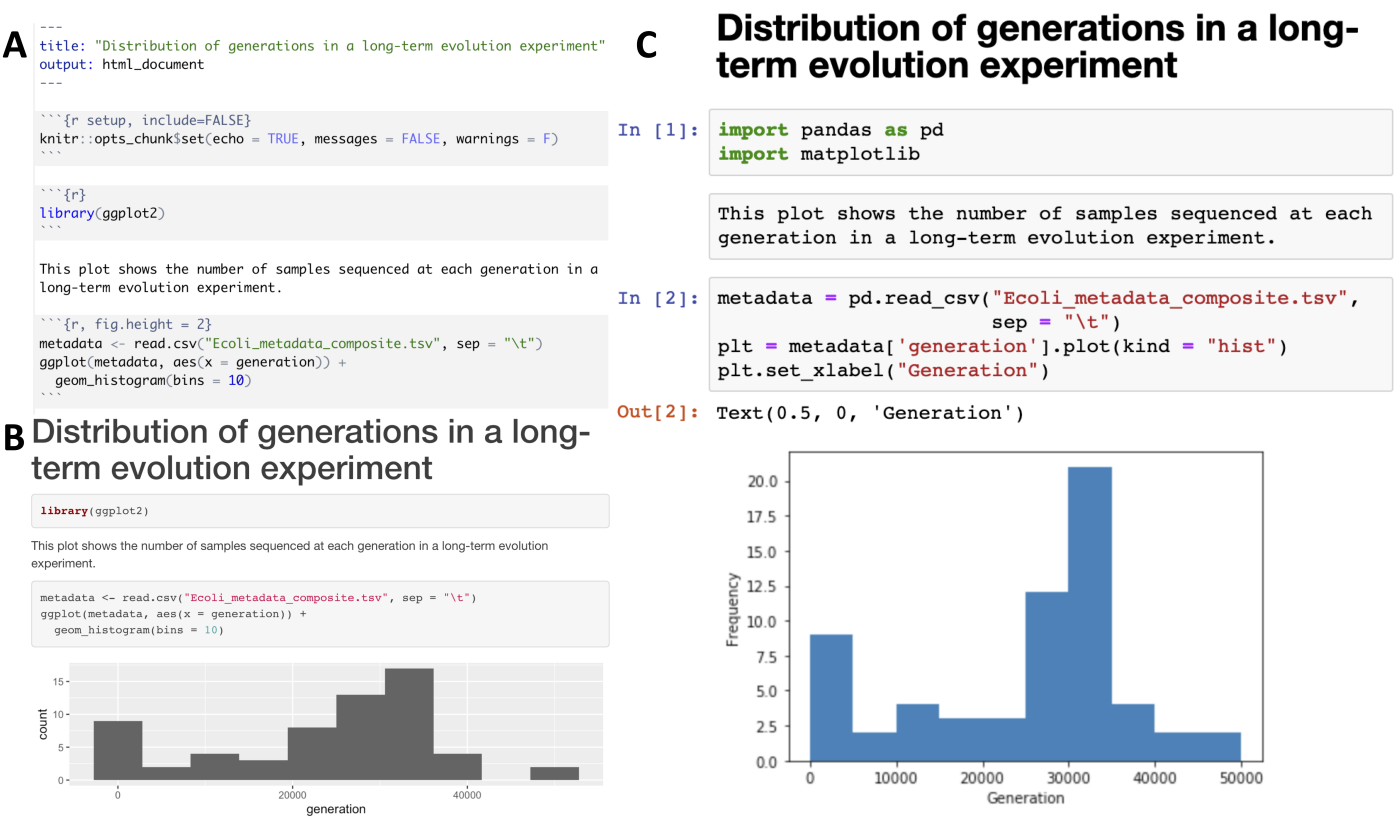

Figure 4: Examples of computational notebooks. Computational notebooks allow the user to mix text, code, and results in one document. **A** A shows an RMarkdown document viewed in the RStudio integrated development environment, while **B** shows a rendered HTML file produced by knitting the RMarkdown document [34]. **C** A Jupyter Notebook, where code, text, and results are rendered inline as each code chunk is executed [**???**]. The second grey chunk is a raw markdown chunk with text that will be rendered inline when executed. Both notebooks generate a histogram of a metadata feature, number of generations, from a long-term evolution experiment with *Escherichia coli* [35]. Computational notebooks facilitate sharing by packaging narrative, code, and visualizations together. Computational notebooks can be packaged with tools like Binder [36]. Binder makes a GitHub repository executable, using package management systems and docker to build reproducible and executable software environments specified in the repository. Binders can be shared with collaborators (or students in a classroom setting), and analysis and visualization can be ephemerally reproduced or altered from the code provided in computational notebooks.

Visualize your workflow

Visual representations can help illustrate the connections in a workflow and improve the readability and reproducibility of your project. At the highest level, flowcharts that detail relationships between steps of a workflow can help provide big-picture clarification, especially when the pipeline is complicated. For individual steps, a graphical representation of the output can show the status of the project or provide insight on additional analyses that should be added. For example, **Figure 5** illustrates a workflow visualization modified from a graph produced by the workflow software Snakemake [37].

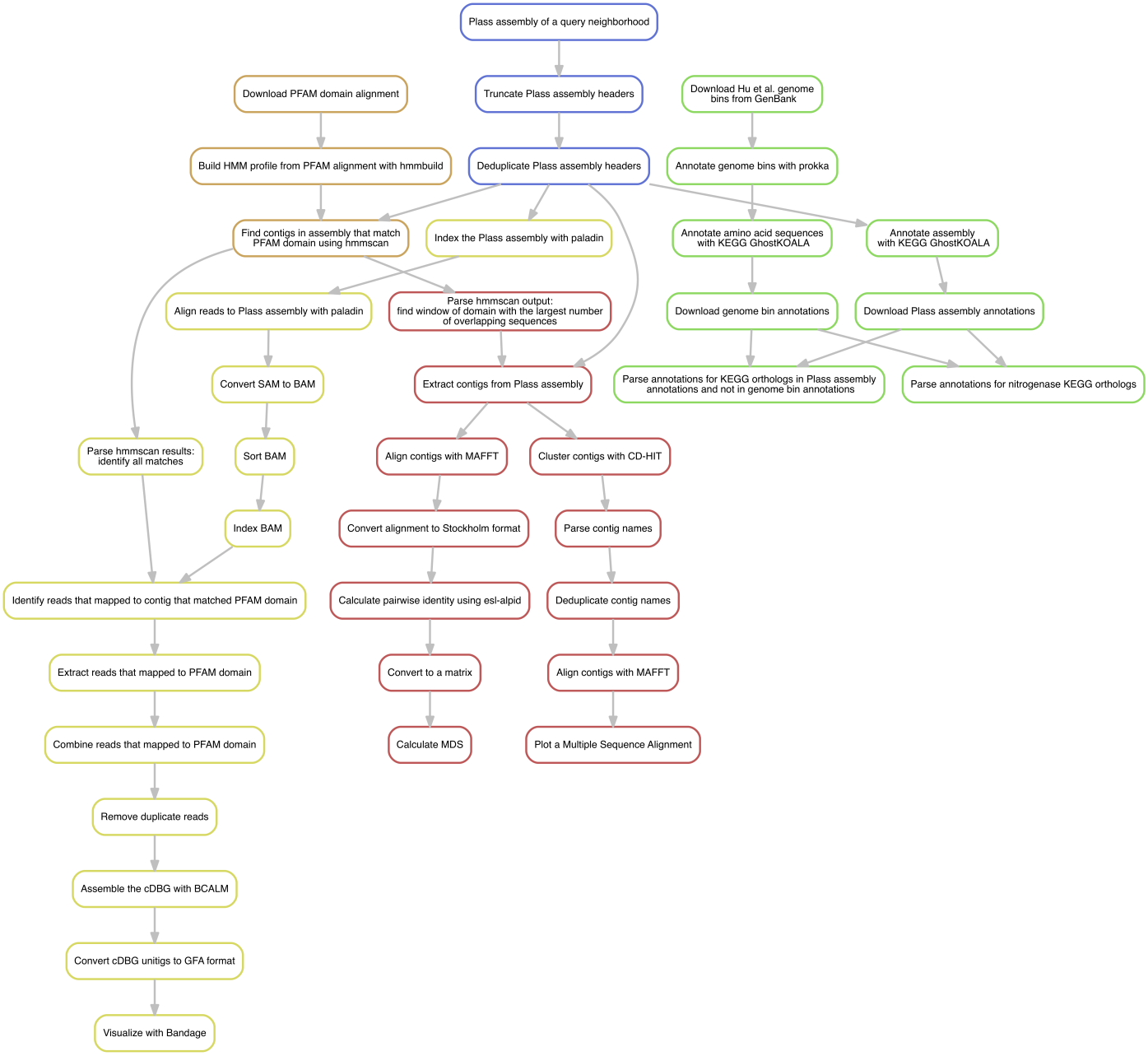

Figure 5: A directed acyclic graph (DAG) that illustrates connections between all steps of a sequencing data analysis workflow. Each box represents a step in the workflow, while lines connect sequential steps. The DAG shown in this figure illustrates a real bioinformatics workflow and was generated by modifying the default Snakemake workflow DAG [37]. The colors represent arms of the workflow that achieve a final result, such as a multiple sequence alignment of a protein of interest. While the workflow is complex, it is coordinated by a workflow system, alleviating the need for a user to manage file interdependencies.

Version control your project

As your project develops, version control allows you to keep track of changes over time. You may already do this in some ways, perhaps with frequent hard drive backups or by manually saving different versions of the same file - e.g. by appending the date to a script name or appending “version_1” or “version_FINAL” to a manuscript draft. For computational workflows, using version control systems such as Git or Mercurial can be used to properly keep track of all changes over time, even across multiple systems, scripting languages, and project contributors (see **Figure 6**). If a key piece of a workflow inexplicably stops working, good version control can allow you to rewind in time and identify differences from when the pipeline worked to when it stopped working.

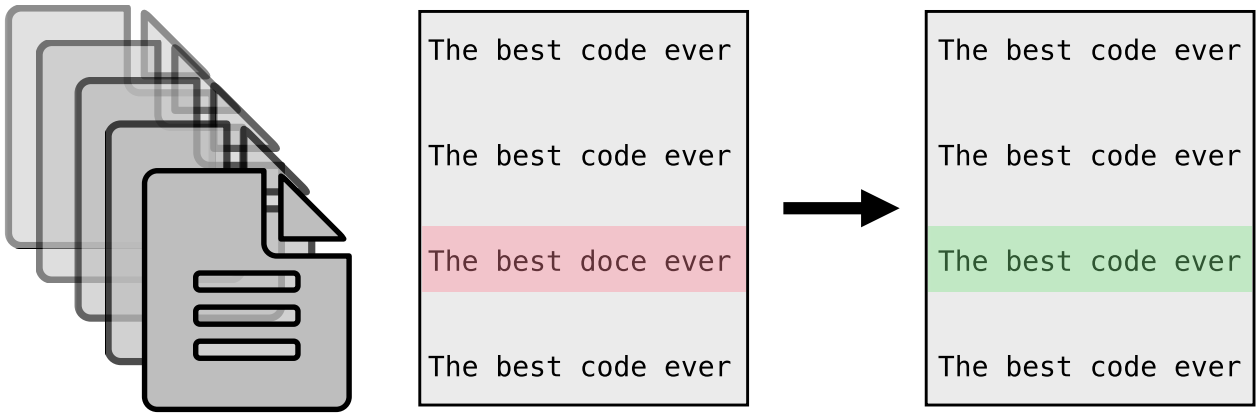

Figure 6: Version Control Version control systems (e.g. Git, Mercurial) work by storing incremental differences in files from one saved version ("commit") to the next. To visualize the differences between each version, text editors such as Atom and online services such as GitHub, GitLab and Bitbucket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

Version control systems also facilitate code and data availability and reproducibility for publication. For example, to ensure the correct version of the code is preserved, you can create a "release", a snapshot of the current code and files in a GitHub repository. You can then generate a digital object identifier (DOI) for that release using Zenodo and make it available to reviewers and beyond (see "sharing" section, below).

Share your workflow and analysis code

Sharing your workflow code with collaborators, peer reviewers, and scientists seeking to use a similar method can foster discussion and review of your analysis. Sticking to a clear documentation strategy, using a version control system, and packaging your code in notebooks or as a workflow prepare them to be easily shared with others. To go one step further, you can package your code with a tool like Binder, Whole Tale, or Shiny apps, which run the code on cloud computers in environments identical to those in which the original computation was performed (**Figure 4**, **Figure 7**) [36,38]. These tools substantially reduce overhead associated with interacting with someones code base and data, and in doing so, make it fast and easy to rerun portions of the analysis, check accuracy, or even tweak the analysis to produce new results. If you choose to share your code and workflows publicly, you will also help contribute to the growing resource of biological analyses available for your chosen workflow system.

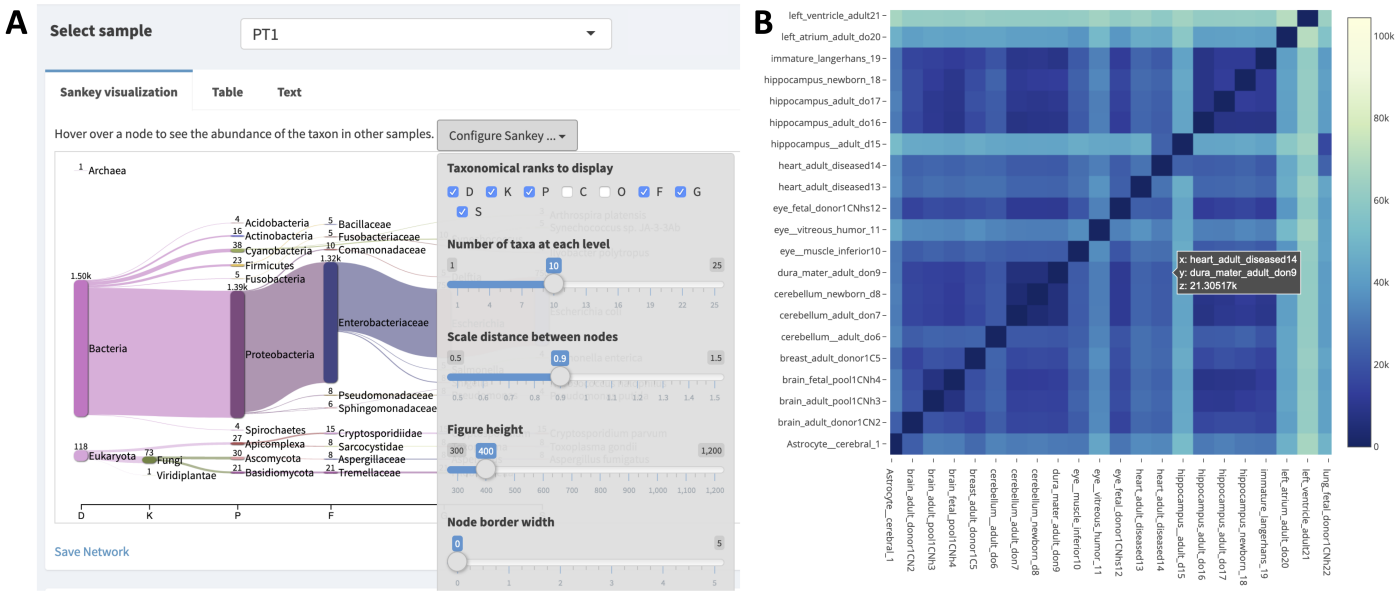

Figure 7: Interactive vizualizations facilitate sharing and repeatability. **A** Interactive vizualization dashboard in the Pavian Shiny app for metagenomic analysis [39,40]. Shiny allows you to build interactive web pages using R code. Data is manipulated by R code in real-time in a web page, producing analysis and visualizations of a data set. Shiny apps can contain user-specifiable parameters, allowing a user to control visualizations or analyses. As seen above, sample "PT1" is selected, and taxonomic ranks class and order are excluded. Shiny apps allow collaborators who may or may not know R to change R visualisations to fit their interests.
B Plotly heatmap of transcriptional profiling in human brain samples [41]. Hovering over a cell in the heatmap displays the sample names from the x and y axis, as well as the intensity value. Plotting tools like plotly and vega-lite produce single interactive plots that can be shared with collaborators or integrated into websites [42,43]. Interactive visualizations are also helpful in exploratory data analysis.

Getting started developing workflows

In our experience, the best way to have your workflow system work *for* you is to include as much of your analysis as possible within the automated workflow framework, use self-documenting names, include analysis visualizations, and keep rigorous documentation alongside your workflow that enables you to understand each decision and entirely reproduce any manual steps. Some of the tools discussed above will inevitably change over time, but these principles apply broadly and will help you design clear, well-documented, and reproducible analyses. Ultimately, you'll need to experiment with the ways that work for you – what is most important is to develop a clear set of strategies and implement them tenaciously. Here are a few practical tips to get started.

Start with working code The best way to develop your own workflow analysis is to start from working code: either from the workflow tutorials or code sharing websites like GitHub, GitLab, and Bitbucket. If a workflow is available through Binder, you can test and experiment with workflow modification on Binder's cloud system, without needing to install a workflow manager or software management tool. When you're building a workflow for the first time, choose code in official repositories or reasonably well-used that provides sample data that can be run to verify the analysis works. The "Workflows and Software Management" Section links to a number of official repositories containing tutorials and example biological analysis workflows.

Test with subsampled data After verifying your chosen example workflow is functional, try running it with your own data or some public data related to your species or condition of interest (see **Table 2**) To save yourself time, energy, and computational resources, first create a small test subset of the data. For example, if working with FASTQ data, you can subsample the first million lines of a file (first 250k reads) by running:

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

While there are many more sophisticated ways to subsample reads, this technique should be sufficient for testing each step of a workflow prior to running your full dataset.

Document your process Write documentation in plain text, so your notes files can be saved under version control with your workflow. We recommend using Markdown language for your notes so they can include helpful headings, code formatting, and embedded images. We recommend using software with visual text previewing such as Hackmd or Atom [44].

Develop your workflow Now that you have the workflow running on new data, you can begin to modify a workflow step to meet your needs. Run each modified step on your subsampled data, checking that the output files look correct. Keep the tool documentation and tutorials on hand; as with any language, remembering workflow-specific syntax takes time and practice.

Back up early and often As you write new code, back up your changes in an online repository such as GitHub, GitLab, or Bitbucket. These services support drag-and-drop interaction: after creating and naming a project repository for your code, you can upload files by dragging and dropping on the web browser. Data backup will be discussed in the next section, "Practical considerations for workflow-enabled biology".

Scale up your workflow Bioinformatic tools vary in the resources they require: some analysis steps are compute-intensive, other steps are memory intensive, and still others will have large intermediate storage needs. If using high-performance computing system, you will need to request resources for running your pipeline, often provided as a simultaneous execution limit or purchased by your research group on a cost-per-compute basis. Workflow systems provide built-in tools to monitor resource usage for each step. Turn on this reporting according to workflow documentation and record the resources required for running each tool on a full-size single sample. Use these estimates to set appropriate resource limits for each step when executing the workflow on your remaining samples.

Find a community and ask for help when you need it If there are local user groups for your workflow language, try to get involved with the community. When you are first learning, help from more advanced users can save you hours of frustration. After you've progressed, providing that same help to new users can help you cement the syntax in your mind and tackle more advanced uses. If no local community is available, look online for forums related to your workflow language. These workflow systems have been enthusiastically adopted by the open science community, and as a consequence, there is a critical mass of tutorials and open access code, code discussion on forums and via social media, particularly twitter. Be respectful of people's time and energy: post in the relevant workflow forums only when you have hit a stopping point you are unable to work through.

Data and resource management for workflow-enabled biology

Advancements in sequencing technologies have greatly increased the volume of data available for biological query [1]. Workflow systems, by virtue of automating many of the time-intensive project management steps traditionally required for data-intensive biology, can increase our capacity for data analysis. However, conducting biological analyses at this scale requires a coordinated approach to data and computational resource management. Below, we provide recommendations for data acquisition, management, and quality control that have become especially important as the scale of data has increased. In the "Getting Started" section, we offer a selection of tools and strategies useful for critically assessing sequencing data, quickly gaining insights from large datasets, and conducting computational resource management.

Managing large-scale datasets

Two important aspects of workflow-enabled data management cannot yet be automated: experimental design and quality control. There is no substitute for taking the time to properly design your analysis, identify appropriate data, and conduct sanity checks on your files. Many tools can aid in this process by providing quality control metrics, but each type of data will have different characteristics that are important to assess.

maybe link: Many of these considerations are addressed in a data carpentry lesson (https://datacarpentry.org/organization-genomics/).

Look for appropriate publicly-available data

With vast amounts of sequencing data already available in public repositories, it is often possible to begin investigating your research question by seeking out publicly available data. In some cases, these data will be sufficient to conduct your entire analysis. In others cases, particularly for biologists conducting novel experiments, these data can inform decisions about sequencing type, depth, and replication, and can help uncover potential pitfalls before they cost valuable time and resources.

Most journals now require data for all manuscripts to be made accessible, either at publication or after a short moratorium. You can find relevant sequencing data either by starting from the "data accessibility" sections of papers relevant to your research or by directly searching for your organism, environment, or treatment of choice in public data portals and repositories. The International Nucleotide Sequence Database Collaboration (INSDC), which includes the Sequence Read Archive (SRA), European Nucleotide Archive (ENA), and DataBank of Japan (DDBJ) is the largest repository for raw sequencing data [46]. Additional curated databases focus on processed data instead, such as gene expression in the Gene Expression Omnibus (GEO) [47]. Organism-specific databases such as **Wormbase** (*Caenorhabditis elegas*) specialize on curating and integrating sequencing and other data associated with a model organism [48]. The SRA, ENA, and DDBJ no longer accept raw sequencing data from large consortia projects, so data from these efforts are often hosted in consortia-specific databases such as those hosted by the Tara Ocean Foundation [49]. Unlike the SRA and associated databases which are centralized and searchable, databases overseen by consortia often require domain-specific knowledge and have unique download and authentication protocols. Finally, rather than focusing on certain data types or organisms, some repositories are designed to hold any data and metadata associated with a specific project or manuscript (e.g. Open Science Framework, Dryad, Zenodo [50]).

Consider analysis when generating your own data

If generating your own data, proper experimental design and planning are essential. For cost-intensive sequencing data, there are a range of decisions about experimental design and sequencing (including sequencing type, sequencing depth per sample, and biological replication) that impact your ability to properly address your research question. These considerations will be different for different types of sequence analysis. While we have curated a series of domain-specific references that may be useful as you go about designing your experiment (see **Table 2**), conducting discussions with experienced bioinformaticians and statisticians, **prior to beginning your experiments** if possible, is the best way to ensure you will have sufficient statistical power to detect effects. Given the resources invested in collecting samples for sequencing, it's important to build in a buffer to preserve your experimental design in the face of unexpected laboratory or technical issues. Once generated, it is always a good idea to have multiple independent backups of raw sequencing data, as it typically cannot be easily regenerated if lost to computer failure or other unforeseeable events.

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

Sequencing type	Resources

Sequencing type	Resources
RNA-sequencing	[24,51,52]
Metagenomic sequencing	[25,53,54]
Amplicon sequencing	[55,56,57]
Microbial isolate sequencing	[58]
Eurkaryotic genome sequencing	
Whole-genome resequencing	[59]
Rad seq	
Chip seq	
ATAC seq	
single cell RNA-seq	[60,61]
?	

As your experiment progresses, keep track of as much information as possible: dates and times of sample collection, storage, and extraction, sample names, aberrations that occurred during collection, kit lot used for extraction, and any other sample measurements you might be able to obtain (temperature, location, metabolite concentration, name of collector, etc). This metadata allows you to keep track of your samples, to control for batch effects that may arise from unintended batching during sampling or experimental procedures and makes the data you collect reusable for future applications and analysis by yourself and others. Wherever possible, follow the standard guidelines for formatting data for scientific computing to limit downstream processing and simplify analyses requiring these metadata (see: [4]).

Protect valuable data

When building your workflow, keep a write-protected copy of your raw data available (as well as backed up) in order to facilitate running the analysis in a fully reproducible manner. When used this way, workflow systems remove the imperative of backing up or storing intermediate files that can be easily regenerated. Given the size of data involved, backing up and storing files associated with data-intensive biology can be burdensome. However, depending on your analysis, you may wish to back up these intermediate files anyway, particularly when they require significant computational time to regenerate. Data version control can be used to store a read-only copy of raw sequencing files and accompanying metadata, or to keep track of difference in intermediate files that change with tool parameters or versions. The version control tools discussed in the “Project Management” section are primarily designed to handle small files, but version control and backup repositories also exist for larger files and datasets.

The Open Science Framework (OSF; [50]) is a free service that provides powerful collaboration and sharing tools, provides built-in version control, integrates with other storage and version control repositories, guarantees data preservation, and enables you to keep projects private until they are ready to share. Like other services geared towards data sharing, OSF also enables generation of a digital object identifier (doi) for each project. While other services such as Git Large File Storage (LFS), Figshare [62], Zenodo [63], and the Dryad Digital Repository [64] each provide important services for sharing and version control, OSF provides the most comprehensive set of free tools for managing data storage and backup. As free tools often limit the size of files that can be stored, a number of cloud backup and storage services are also available for purchase or via university contract, including Google Drive, Box, Dropbox, Amazon Web Services, and Backblaze. Full computer backups can be conducted to these storage locations with tools like rclone [65].

Ensure data integrity during transfers

If you're working with publicly-available data, you may be able to work on a compute system where the data are already available, circumventing time and effort required for downloading and moving the data. Databases such as the Sequence Read Archive (SRA) are now available on commercial cloud computing systems, and open source projects such as Galaxy enable import and work on SRA sequence files directly from a web browser [7,66]. Ongoing projects such as the NIH Common Fund Data Ecosystem aim to develop a data portal to make NIH Common Fund data, including biomedical sequencing data, more findable, accessible, interoperable, and reusable (FAIR).

In most cases, you'll still need to transfer some data - either downloading raw data or transferring important intermediate and results files for backup and sharing (or both). Transferring compressed files (gzip, bzip2, BAM/CRAM, etc.) can improve transfer speed and save space, and checksums can be used to to ensure file integrity after transfer (see **Figure 8**). For publicly-available files, a checksum value is often provided with each data file, so that you can check the integrity of the file after download.

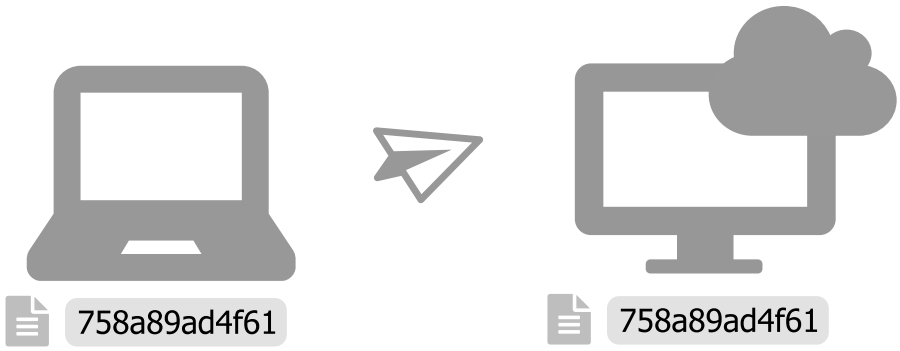

Figure 8: Use Checksums to ensure file integrity Checksum programs (e.g. md5, sha256) encode file size in a single value known as a "checksum". For any file, this value will be identical across platforms when calculated using the same checksum program. When transferring files, calculate the value of the checksum prior to transfer, and then again after transfer. If the value is not identical, there was an error introduced during transfer (e.g. file truncation, etc). Tools like Rsync and Rclone that automate file transfers use checksums internally to verify that files were transferred properly [65]. Some GUI file transfer tools (e.g. cyberduck) also assess checksums when they are provided [67].

Getting started: strategies for quality control

The quality of your input data has a major impact on the quality of the output results, no matter whether your workflow analyzes six samples or 600 samples. Assessing data at every analysis step can reveal problems and errors early, before they waste valuable time and resources. Using quality control tools that provide metrics and visualizations can help you assess your datasets, particularly as the size of your input data scales up. However, data from different species or sequencing types can produce different quality control results. You are ultimately the single most effective quality control tool that you have, so it is important to critically assess each metric to determine those that are relevant for your particular data.

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

Table 3: Some bash commands are useful to quickly explore the contents of a file. By using these commands, the user can 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

Visualize your data Visualization is another powerful way to pick out unusual or unexpected patterns. Although large abnormalities may be clear from looking at files, others may be small and difficult to find. Visualizing raw sequencing data with FastQC (**Figure 9A**) and processed sequencing data with tools like the Integrative Genome Viewer and plotting tabular results files using python or R can make aberrant or inconsistent results easier to track down [68,69].

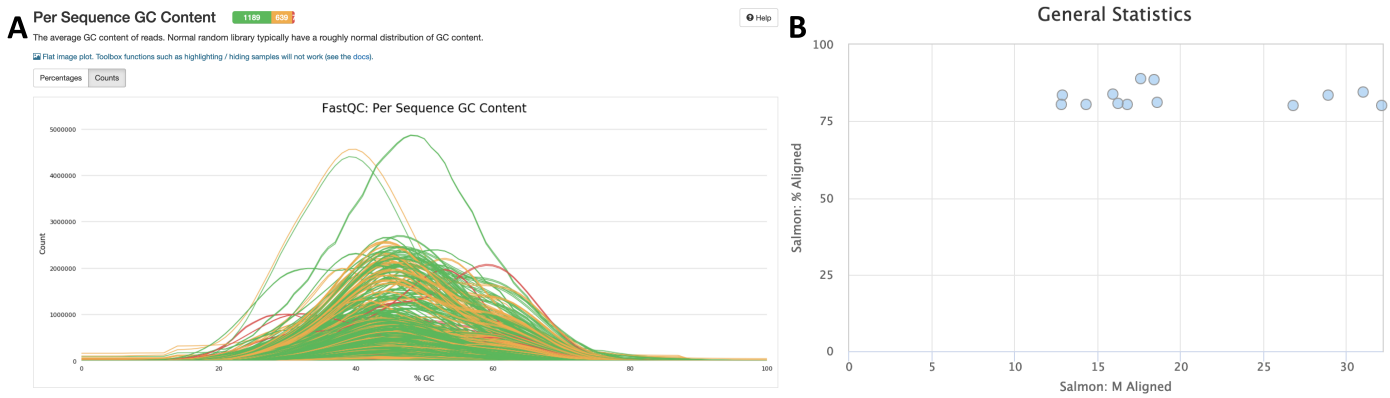

Figure 9: Visualizations produced by MultiQC. A MultiQC summary of FastQC Per Sequence GC Content for 1905 metagenome samples. FastQC provides quality control measurements and visualizations for raw sequencing data, and is a near-universal first step in sequencing data analysis because of the insights it provides [68,69]. FastQC measures and summarizes 10 quality metrics and provides recommendations for whether the sample is within an acceptable quality range. Not all metrics readily apply to all sequencing data types. For example, while multiple GC peaks might be concerning in whole genome sequencing of a bacterial isolate, we would expect a non-normal distrubtion for some metagenome samples that contain organisms with diverse GC content. Samples like this can be seen in red in this figure. **B** MultiQC summary of Salmon *quant* reads mapped per sample for RNA-seq samples [70]. MultiQC finds and automatically parses log files from other tools and generates a combined report and parsed data tables that include all samples. MultiQC currently supports 88 tools. In this figure, we see that MultiQC summarizes the number of reads mapped and percent of reads mapped, two values that are reported in the Salmon log files.

Pay attention to warnings and log files Many tools generate log files or messages while running. These files contain information about the quantity, quality, and results from the run, or error messages about why a run failed. Inspecting these files can be helpful to make sure tools ran properly

and consistently, or to debug failed runs. Parsing and visualizing log files with a tool like MultiQC can improve interpretability of program-specific log files (**Figure 9** [71]).

Look for common biases in sequencing data Biases in sequencing data originate from experimental design, methodology, sequencing chemistry, or workflows, and are helpful to target specifically with quality control measures. The exact biases in a specific data set or workflow will vary greatly between experiments. For example, PCR duplicates can cause problems in libraries that underwent an amplification step, and often need to be removed prior to downstream analysis [72,73,74,75,76].

Check for contamination Contamination can arise during sample collection, nucleotide extraction, library prepartion, or through sequencing spike-ins like PhiX, and could change data interpretation if not removed [77,78,79]. Libraries sequenced with high concentrations of free adapters or with low concentration samples may have increased barcode hopping, leading to contamination between samples [80].

Consider the costs and benefits of stringent quality control for your data Good quality data is essential for good downstream analysis. However, stringent quality control can sometimes do more harm than good. For example, depending on sequencing depth, stringent quality trimming of RNA-sequencing data may reduce isoform discovery [81]. To determine what issues are most likely to plague your specific data set, it can be helpful to find recent publications using a similar experimental design, or to speak with experts at a sequencing core.

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

Securing and managing appropriate computational resources

Independent of workflows, sequence analysis requires access to computing systems with adequate storage and analysis power for your data. For some smaller-scale datasets, local desktop or even laptop systems can be sufficient, especially if using tools that implement data-reduction strategies such as minhashing [82]. However, larger projects require additional computing power, or may be restricted to certain operating systems (e.g. linux). For these projects, solutions range from research-focused high performance computing systems to research-integrated commercial analysis platforms. Both research-only and and commercial clusters provide avenues for research and educational proposals to enable access to their computing resources (see **Table 4**). In preparing for data analysis, be sure to allocate sufficient computational resources and funding for storage and analysis, including large intermediate files and resources required for personnel training.

Table 4: Research cloud resources Cloud provider indicates the name of the cloud, standard model indicates the most common route toward using the cloud, and limitations indicates limitations in access or services provided by the cloud.

Cloud Provider	Standard Model	Limits
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	

Cloud Provider	Standard Model	Limits
Google Colab	Free	computational notebooks, no resource guaruntees
Microsoft Azure	Paid	
NSF XSEDE	Research allocation	USA researchers or collaborators
Open Science Data Cloud	Research allocation	
Wasabi	Paid	data storage solution only

Getting started with resource management

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

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

Use workflow reporting Workflow systems can facilitate use of your available resources. Workflow systems provide built-in reporting of resource usage by each tool that can be used to limit requested computing resources. However, reports or benchmarking often need to be enabled with workflow-specific syntax. Enable reporting for all of your workflow steps so that you can see exactly how many resources each step uses for each of your files. You can then use workflow resource specifications to set the resources requested per task to match your recorded values (include a small excess buffer to account for differences in samples and file sizes).

Develop on a local computer when possible Since workflows transfer easily across systems, it can be useful to develop individual analysis steps on a local laptop. If the analysis tool will run on your local system, test the step with subsampled data, such as that created in the “Getting started developing workflows” section. Once working, the new workflow component can be run at scale on a larger computing system. For researchers without access to free or granted computing resources, this strategy can save significant cost.

Gain quick insights using sketching algorithms Understanding the basic structure of data, the relationship between samples, and the approximate composition of each sample can very helpful at the beginning of data analysis, and can often drive analysis decisions in different directions than those originally intended. Although most bioinformatics workflows generate these types of insights, there are a few tools that do so rapidly, allowing the user to generate quick hypotheses that can be further tested by more extensive, fine-grained analyses. Sketching algorithms work with compressed approximate representations of sequencing data and thereby reduce runtimes and computational resources. These approximate representations retain enough information about the original sequence to recapitulate the main findings from many exact but computationally intensive workflows. Most sketching algorithms estimate sequence similarity in some way, allowing you to gain insights from these comparisons. For example, sketching algorithms can be used to estimate all-by-all sample similarity which can be visualized as a Principle Component Analysis or a multidimensional scaling plot, or can be used to build a phylogenetic tree with accurate topology. Sketching algorithms also dramatically reduce the runtime for comparisons against databases (e.g. all of GenBank), allowing users to quickly compare their data against large public databases. Sketching algorithms have been reviewed in-depth by Rowe [82].

Use the right tools for your question RNA-seq analysis approaches like differential expression or transcript clustering rely on transcript or gene counts. Many tools can be used to generate these counts by quantifying the number of reads that overlap with each transcript or gene. For example, tools like STAR and HISAT2 produce alignments that can be post-processed to generate per-transcript read counts [83,84]. However, these tools generate information-rich output, specifying per-base alignments for each read. If you are only interested in counts per read, quasi-mapping tools produce the minimum information necessary for read quantification, thereby reducing the time and resources needed to generate and store read count information [85].

Troubleshooting: how to help yourself and when to get help

If you have tried the strategies above and are having trouble with your workflow, it's time to ask for help. The first point of attack is always to Google the error, including any identifying error message or code, the program name, and if necessary, the type of data you're running. There are a vast array of online resources for bioinformatic help ranging from question sites such as Stack Overflow and BioStars, to personal or academic blogs or even tutorials and lessons written by experts in the field [86]. In most cases, the error you've encountered has been encountered many times before, and often the solution is readily available. If you can't find any solutions in the relevant search results, it's time to escalate. If the error is with a specific program, it's best to post on that program's help or issue location (e.g. GitHub Issues), or google group mailing list. Often the authors of your software will post their preferred location for answering questions and solving errors related to their program. Be sure to include the relevant details of your error, including terminal output and the version of the software you are using. If your error or question is more general, such as asking about program choice or workflows, Stack Overflow is a good choice. First, search through related topics to ensure your question has not already been answered. If it hasn't, make a post to a relevant section, and be sure to include all relevant information in your post - type of data you have, approaches you've tried already, relevant error message, etc.

While there is lots of help available online, there's no substitute for local communities where you can get help working with your data and learning to troubleshoot. Many people around you may be experiencing similar issues and finding it difficult to find appropriate help. Developing a local bioinformatics community, either via seminar series or meetup sessions for data analysis, can also help in both improving and expanding your work in this area. Once you establish a local community, it may also be useful to set up a local online sites (e.g. discourse) for group troubleshooting. While this may seem like just a local version of Stack Overflow, the local, member-only nature can help create a safe and collaborative online space for troubleshooting problems often encountered by your local bioinformatics community. The benefit to beginners is clear: learning the best way to post questions and the important parts of errors, while getting their questions answered so they can move forward in their research. However, intermediate users may find these communities most useful, as they can also accelerate their own troubleshooting skills by helping others solve issues that they have already struggled through. While it can be helpful to have some experts available to help answer questions or to know when to escalate to Stack Overflow or other communities, a collaborative community of practice with members at all experience levels can help all its members move their science forward faster.

Conclusion

Bioinformatics-focused workflow systems have revolutionized data-intensive biology, and enabled biologists to cope with the massive scale of data now becoming available. Workflow-integrated software management tools remove tool installation as a barrier to data exploration and analysis, and free biologists to less time rewriting common analysis steps, and spend more time on interesting biological questions. With a few directed strategies for project, data, and resource management, these

workflow systems can quickly streamline bioinformatic analyses and reduce the time to insight. We hope the tips contained here will enable more biologists take advantage of workflow-enabled data-intensive biology.

Acknowledgements

thanks!

Competing Interests

Author	Competing Interests	Last Reviewed

Author Contributions

Author	Contributions

References

1. **Documentation** https://www.ncbi.nlm.nih.gov/sra/docs/sragrowth/

2. **Best Practices for Scientific Computing**
   Greg Wilson, D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, … Paul Wilson
   PLoS Biology (2014-01-07) https://doi.org/qtt
   DOI: 10.1371/journal.pbio.1001745 · PMID: 24415924 · PMCID: PMC3886731

3. **Computing Workflows for Biologists: A Roadmap**
   Ashley Shade, Tracy K. Teal
   PLOS Biology (2015-11-24) https://doi.org/f72r9m
   DOI: 10.1371/journal.pbio.1002303 · PMID: 26600012 · PMCID: PMC4658184

4. **Good enough practices in scientific computing**
   Greg Wilson, Jennifer Bryan, Karen Cranston, Justin Kitzes, Lex Nederbragt, Tracy K. Teal
   PLOS Computational Biology (2017-06-22) https://doi.org/gbkbwp
   DOI: 10.1371/journal.pcbi.1005510 · PMID: 28640806 · PMCID: PMC5480810

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

6. **Bioconda: sustainable and comprehensive software distribution for the life sciences**
   Björn Grüning, Ryan Dale, Andreas Sjödin, Brad A. Chapman, Jillian Rowe, Christopher H. Tomkins-Tinch, Renan Valieris, Johannes Köster, The Bioconda Team
   Nature Methods (2018-07-02) https://doi.org/gd2xzp
   DOI: 10.1038/s41592-018-0046-7 · PMID: 29967506

7. **The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update**
   Enis Afgan, Dannon Baker, Bérénice Batut, Marius van den Beek, Dave Bouvier, Martin Čech, John Chilton, Dave Clements, Nate Coraor, Björn A Grüning, … Daniel Blankenberg
   Nucleic Acids Research (2018-07-02) https://doi.org/gdwxpr
   DOI: 10.1093/nar/gky379 · PMID: 29790989 · PMCID: PMC6030816

8. **Data Commons to Support Pediatric Cancer Research**
   Samuel L. Volchenboum, Suzanne M. Cox, Allison Heath, Adam Resnick, Susan L. Cohn, Robert Grossman
   American Society of Clinical Oncology Educational Book (2017) https://doi.org/ggv5zs
   DOI: 10.14694/edbk_175029 · PMID: 28561664

9. **MGnify: the microbiome analysis resource in 2020**
   Alex L Mitchell, Alexandre Almeida, Martin Beracochea, Miguel Boland, Josephine Burgin, Guy Cochrane, Michael R Crusoe, Varsha Kale, Simon C Potter, Lorna J Richardson, … Robert D Finn
   Nucleic Acids Research (2019-11-07) https://doi.org/ggctnm
   DOI: 10.1093/nar/gkz1035 · PMID: 31696235 · PMCID: PMC7145632

10. **nf-core/rnaseq**
nf-core
(2020-05-17) https://github.com/nf-core/rnaseq

11. **The nf-core framework for community-curated bioinformatics pipelines**
Philip A. Ewels, Alexander Peltzer, Sven Fillinger, Harshil Patel, Johannes Alneberg, Andreas Wilm, Maxime Ulysse Garcia, Paolo Di Tommaso, Sven Nahnsen
Nature Biotechnology (2020-02-13) https://doi.org/ggk3qh
DOI: 10.1038/s41587-020-0439-x · PMID: 32055031

12. **metagenome-atlas/atlas**
metagenome-atlas
(2020-05-17) https://github.com/metagenome-atlas/atlas

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

14. **sunbeam-labs/sunbeam**
Sunbeam Labs
(2020-05-05) https://github.com/sunbeam-labs/sunbeam

15. **Sunbeam: an extensible pipeline for analyzing metagenomic sequencing experiments**
Erik L. Clarke, Louis J. Taylor, Chunyu Zhao, Andrew Connell, Jung-Jin Lee, Bryton Fett, Frederic D. Bushman, Kyle Bittinger
Microbiome (2019-03-22) https://doi.org/gf9sd6
DOI: 10.1186/s40168-019-0658-x · PMID: 30902113 · PMCID: PMC6429786

16. **dib-lab/elvers**
The Lab for Data Intensive Biology
(2020-05-08) https://github.com/dib-lab/elvers

17. **dib-lab/dammit**
The Lab for Data Intensive Biology
(2020-05-12) https://github.com/dib-lab/dammit

18. **Snakemake–a scalable bioinformatics workflow engine**
J. Koster, S. Rahmann
Bioinformatics (2012-08-20) https://doi.org/gd2xzq
DOI: 10.1093/bioinformatics/bts480 · PMID: 22908215

19. **Nextflow enables reproducible computational workflows**
Paolo Di Tommaso, Maria Chatzou, Evan W Floden, Pablo Prieto Barja, Emilio Palumbo, Cedric Notredame
Nature Biotechnology (2017-04-11) https://doi.org/gfj52z
DOI: 10.1038/nbt.3820 · PMID: 28398311

20. **Common Workflow Language, v1.0**
Peter Amstutz, Michael R. Crusoe, Nebojša Tijanić, Brad Chapman, John Chilton, Michael Heuer, Andrey Kartashov, Dan Leehr, Hervé Ménager, Maya Nedeljkovich, … Luka Stojanovic

Figshare (2016) https://doi.org/gf6ppg
DOI: 10.6084/m9.figshare.3115156.v2

21. **The drake R package: a pipeline toolkit for reproducibility and high-performance computing**
William Michael Landau
The Journal of Open Source Software (2018-01-26) https://doi.org/gd876m
DOI: 10.21105/joss.00550

22. **Scalable Workflows and Reproducible Data Analysis for Genomics**
Francesco Strozzi, Roel Janssen, Ricardo Wurmus, Michael R. Crusoe, George Githinji, Paolo Di Tommaso, Dominique Belhachemi, Steffen Möller, Geert Smant, Joep de Ligt, Pjotr Prins
Methods in Molecular Biology (2019) https://doi.org/ggv5zh
DOI: 10.1007/978-1-4939-9074-0_24 · PMID: 31278683

23. **"Sequana": a Set of Snakemake NGS pipelines**
Thomas Cokelaer, Dimitri Desvillechabrol, Rachel Legendre, Mélissa Cardon
The Journal of Open Source Software (2017-08-30) https://doi.org/ggv5zt
DOI: 10.21105/joss.00352

24. **A survey of best practices for RNA-seq data analysis**
Ana Conesa, Pedro Madrigal, Sonia Tarazona, David Gomez-Cabrero, Alejandra Cervera, Andrew McPherson, Michał Wojciech Szcześniak, Daniel J. Gaffney, Laura L. Elo, Xuegong Zhang, Ali Mortazavi
Genome Biology (2016-01-26) https://doi.org/f3vhpj
DOI: 10.1186/s13059-016-0881-8 · PMID: 26813401 · PMCID: PMC4728800

25. **Shotgun metagenomics, from sampling to analysis**
Christopher Quince, Alan W Walker, Jared T Simpson, Nicholas J Loman, Nicola Segata
Nature Biotechnology (2017-09-12) https://doi.org/gbv6nf
DOI: 10.1038/nbt.3935 · PMID: 28898207

26. **Current best practices in single-cell RNA-seq analysis: a tutorial**
Malte D Luecken, Fabian J Theis
Molecular Systems Biology (2019-06-19) https://doi.org/gf4f4d
DOI: 10.15252/msb.20188746 · PMID: 31217225 · PMCID: PMC6582955

27. **Next-generation biology: Sequencing and data analysis approaches for non-model organisms**
Rute R. da Fonseca, Anders Albrechtsen, Gonçalo Espregueira Themudo, Jazmín Ramos-Madrigal, Jonas Andreas Sibbesen, Lasse Maretty, M. Lisandra Zepeda-Mendoza, Paula F. Campos, Rasmus Heller, Ricardo J. Pereira
Marine Genomics (2016-12) https://doi.org/f9h2s2
DOI: 10.1016/j.margen.2016.04.012 · PMID: 27184710

28. **Best practices for analysing microbiomes**
Rob Knight, Alison Vrbanac, Bryn C. Taylor, Alexander Aksenov, Chris Callewaert, Justine Debelius, Antonio Gonzalez, Tomasz Kosciolek, Laura-Isobel McCall, Daniel McDonald, … Pieter C. Dorrestein
Nature Reviews Microbiology (2018-05-23) https://doi.org/gdj32p
DOI: 10.1038/s41579-018-0029-9 · PMID: 29795328

29. **Conda: a cross-platform, python-agnostic binary package manager.**
Lex Hider

PyVideo.org https://pyvideo.org/pycon-au-2015/conda-a-cross-platform-python-agnostic-binary-p.html

30. **Singularity: Scientific containers for mobility of compute**
Gregory M. Kurtzer, Vanessa Sochat, Michael W. Bauer
PLOS ONE (2017-05-11) https://doi.org/f969fz
DOI: 10.1371/journal.pone.0177459 · PMID: 28494014 · PMCID: PMC5426675

31. **QuantStack/mamba**
QuantStack
(2020-05-18) https://github.com/QuantStack/mamba

32. **dib-lab/dib-MMETSP**
The Lab for Data Intensive Biology
(2019-10-13) https://github.com/dib-lab/dib-MMETSP

33. **Re-assembly, quality evaluation, and annotation of 678 microbial eukaryotic reference transcriptomes**
Lisa K Johnson, Harriet Alexander, C Titus Brown
GigaScience (2019-04) https://doi.org/ggrd6x
DOI: 10.1093/gigascience/giy158 · PMID: 30544207 · PMCID: PMC6481552

34. **R Markdown** https://rmarkdown.rstudio.com/

35. **Tempo and mode of genome evolution in a 50,000-generation experiment**
Olivier Tenaillon, Jeffrey E. Barrick, Noah Ribeck, Daniel E. Deatherage, Jeffrey L. Blanchard, Aurko Dasgupta, Gabriel C. Wu, Sébastien Wielgoss, Stéphane Cruveiller, Claudine Médigue, … Richard E. Lenski
Nature (2016-08-01) https://doi.org/f8283v
DOI: 10.1038/nature18959 · PMID: 27479321 · PMCID: PMC4988878

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

37. **Exploring neighborhoods in large metagenome assembly graphs reveals hidden sequence diversity**
C. Titus Brown, Dominik Moritz, Michael P. O'Brien, Felix Reidl, Taylor Reiter, Blair D. Sullivan
bioRxiv (2020-01-22) https://doi.org/cwrw
DOI: 10.1101/462788

38. **Computing environments for reproducibility: Capturing the "Whole Tale"**
Adam Brinckman, Kyle Chard, Niall Gaffney, Mihael Hategan, Matthew B. Jones, Kacper Kowalik, Sivakumar Kulasekaran, Bertram Ludäscher, Bryce D. Mecum, Jarek Nabrzyski, … Kandace Turner
Future Generation Computer Systems (2019-05) https://doi.org/ggv5zj
DOI: 10.1016/j.future.2017.12.029

39. **Pavian** https://fbreitwieser.shinyapps.io/pavian/

40. **Pavian: interactive analysis of metagenomics data for microbiome studies and pathogen identification**

Florian P Breitwieser, Steven L Salzberg
Bioinformatics (2019-09-25) https://doi.org/ggnb3n
DOI: 10.1093/bioinformatics/btz715 · PMID: 31553437

41. **Visualizing Biological Data**https://plotly.com/python/v3/ipython-notebooks/bioinformatics/

42. **Plotly: The front-end for ML and data science models**/

43. **Vega-Lite: A Grammar of Interactive Graphics**
Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, Jeffrey Heer
IEEE Transactions on Visualization and Computer Graphics (2017-01) https://doi.org/f92f32
DOI: 10.1109/tvcg.2016.2599030 · PMID: 27875150

44. **HackMD - Collaborative Markdown Knowledge Base**
HackMD
https://hackmd.io

45. **A hackable text editor for the 21st Century**
Atom
https://atom.io/

46. **The International Nucleotide Sequence Database Collaboration**
Guy Cochrane, Ilene Karsch-Mizrachi, Toshihisa Takagi, International Nucleotide Sequence Database Collaboration
Nucleic Acids Research (2016-01-04) https://doi.org/gf28sk
DOI: 10.1093/nar/gkv1323 · PMID: 26657633 · PMCID: PMC4702924

47. **Gene Expression Omnibus: NCBI gene expression and hybridization array data repository**
R. Edgar
Nucleic Acids Research (2002-01-01) https://doi.org/fttpkn
DOI: 10.1093/nar/30.1.207 · PMID: 11752295 · PMCID: PMC99122

48. **WormBase: a modern Model Organism Information Resource**
Todd W Harris, Valerio Arnaboldi, Scott Cain, Juancarlos Chan, Wen J Chen, Jaehyoung Cho, Paul Davis, Sibyl Gao, Christian A Grove, Ranjana Kishore, … Paul W Sternberg
Nucleic Acids Research (2019-10-23) https://doi.org/ggv5zk
DOI: 10.1093/nar/gkz920 · PMID: 31642470 · PMCID: PMC7145598

49. **Open science resources for the discovery and analysis of Tara Oceans data**
Stéphane Pesant, Fabrice Not, Marc Picheral, Stefanie Kandels-Lewis, Noan Le Bescot, Gabriel Gorsky, Daniele Iudicone, Eric Karsenti, Sabrina Speich, Romain Troublé, … Sarah Searson
Scientific Data (2015-05-26) https://doi.org/gd32xw
DOI: 10.1038/sdata.2015.23 · PMID: 26029378 · PMCID: PMC4443879

50. **Open Science Framework (OSF)**
Erin D. Foster, MSLS, Ariel Deardorff, MLIS
Journal of the Medical Library Association (2017-04-04) https://doi.org/gfxvhq
DOI: 10.5195/jmla.2017.88 · PMCID: PMC5370619

51. **Erratum: How many biological replicates are needed in an RNA-seq experiment and which differential expression tool should you use?**
Nicholas J. Schurch, Pietà Schofield, Marek Gierliński, Christian Cole, Alexander Sherstnev, Vijender Singh, Nicola Wrobel, Karim Gharbi, Gordon G. Simpson, Tom Owen-Hughes, … Geoffrey J. Barton

RNA (2016-09-16) https://doi.org/ggv5zr
DOI: 10.1261/rna.058339.116 · PMID: 27638913 · PMCID: PMC5029462

52. **Power analysis and sample size estimation for RNA-Seq differential expression**
Travers Ching, Sijia Huang, Lana X. Garmire
RNA (2014-11) https://doi.org/f6nstp
DOI: 10.1261/rna.046011.114 · PMID: 25246651 · PMCID: PMC4201821

53. **Unlocking the potential of metagenomics through replicated experimental design**
Rob Knight, Janet Jansson, Dawn Field, Noah Fierer, Narayan Desai, Jed A Fuhrman, Phil Hugenholtz, Daniel van der Lelie, Folker Meyer, Rick Stevens, … Jack A Gilbert
Nature Biotechnology (2012-06-07) https://doi.org/f32nds
DOI: 10.1038/nbt.2235 · PMID: 22678395 · PMCID: PMC4902277

54. **Contamination in Low Microbial Biomass Microbiome Studies: Issues and Recommendations**
Raphael Eisenhofer, Jeremiah J. Minich, Clarisse Marotz, Alan Cooper, Rob Knight, Laura S. Weyrich
Trends in Microbiology (2019-02) https://doi.org/gfpfkt
DOI: 10.1016/j.tim.2018.11.003 · PMID: 30497919

55. **Consistent and correctable bias in metagenomic sequencing experiments**
Michael R McLaren, Amy D Willis, Benjamin J Callahan
eLife (2019-09-10) https://doi.org/gf7wbr
DOI: 10.7554/elife.46923 · PMID: 31502536 · PMCID: PMC6739870

56. **From Benchtop to Desktop: Important Considerations when Designing Amplicon Sequencing Workflows**
Dáithí C. Murray, Megan L. Coghlan, Michael Bunce
PLOS ONE (2015-04-22) https://doi.org/f7hjz7
DOI: 10.1371/journal.pone.0124671 · PMID: 25902146 · PMCID: PMC4406758

57. **Assessment of variation in microbial community amplicon sequencing by the Microbiome Quality Control (MBQC) project consortium**
Rashmi Sinha, Galeb Abu-Ali, Emily Vogtmann, Anthony A Fodor, Boyu Ren, Amnon Amir, Emma Schwager, Jonathan Crabtree, Siyuan Ma, Christian C Abnet, … The Microbiome Quality Control Project Consortium
Nature Biotechnology (2017-10-02) https://doi.org/gbzrdx
DOI: 10.1038/nbt.3981 · PMID: 28967885 · PMCID: PMC5839636

58. **Completing bacterial genome assemblies: strategy and performance comparisons**
Yu-Chieh Liao, Shu-Hung Lin, Hsin-Hung Lin
Scientific Reports (2015-03-04) https://doi.org/f686w8
DOI: 10.1038/srep08747 · PMID: 25735824 · PMCID: PMC4348652

59. **Whole-genome sequencing approaches for conservation biology: Advantages, limitations and practical recommendations**
Angela P. Fuentes-Pardo, Daniel E. Ruzzante
Molecular Ecology (2017-10) https://doi.org/gfzn9r
DOI: 10.1111/mec.14264 · PMID: 28746784

60. **Design and computational analysis of single-cell RNA-sequencing experiments**
Rhonda Bacher, Christina Kendziorski

Genome Biology (2016-04-07) https://doi.org/gc958g
DOI: 10.1186/s13059-016-0927-y · PMID: 27052890 · PMCID: PMC4823857

61. **A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications**
Ashraful Haque, Jessica Engel, Sarah A. Teichmann, Tapio Lönnberg
Genome Medicine (2017-08-18) https://doi.org/gfgppz
DOI: 10.1186/s13073-017-0467-4 · PMID: 28821273 · PMCID: PMC5561556

62. **figshare - credit for all your research** https://figshare.com/

63. **Zenodo - Research. Shared.** https://zenodo.org/

64. **Dryad Home - Publish and Preserve your Data** https://datadryad.org/stash

65. **RClone: a package to identify MultiLocus Clonal Lineages and handle clonal data sets in .**
Diane Bailleul, Solenn Stoeckel, Sophie Arnaud-Haond
Methods in Ecology and Evolution (2016-08) https://doi.org/f82t65
DOI: 10.1111/2041-210x.12550

66. **SRA in the Cloud** https://www.ncbi.nlm.nih.gov/sra/docs/sra-cloud/

67. **Cyberduck | Libre server and cloud storage browser for Mac and Windows with support for FTP, SFTP, WebDAV, Amazon S3, OpenStack Swift, Backblaze B2, Microsoft Azure & OneDrive, Google Drive and Dropbox** https://cyberduck.io/

68. **Babraham Bioinformatics - FastQC A Quality Control tool for High Throughput Sequence Data** https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

69. **Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration**
H. Thorvaldsdottir, J. T. Robinson, J. P. Mesirov
Briefings in Bioinformatics (2012-04-19) https://doi.org/f4sc43
DOI: 10.1093/bib/bbs017 · PMID: 22517427 · PMCID: PMC3603213

70. **Salmon provides fast and bias-aware quantification of transcript expression**
Rob Patro, Geet Duggal, Michael I Love, Rafael A Irizarry, Carl Kingsford
Nature Methods (2017-03-06) https://doi.org/gcw9f5
DOI: 10.1038/nmeth.4197 · PMID: 28263959 · PMCID: PMC5600148

71. **MultiQC: summarize analysis results for multiple tools and samples in a single report**
Philip Ewels, Måns Magnusson, Sverker Lundin, Max Käller
Bioinformatics (2016-10-01) https://doi.org/f3s996
DOI: 10.1093/bioinformatics/btw354 · PMID: 27312411 · PMCID: PMC5039924

72. **Identifying and mitigating bias in next-generation sequencing methods for chromatin biology**
Clifford A. Meyer, X. Shirley Liu
Nature Reviews Genetics (2014-09-16) https://doi.org/ggd9m9
DOI: 10.1038/nrg3788 · PMID: 25223782 · PMCID: PMC4473780

73. **The impact of amplification on differential expression analyses by RNA-seq**
Swati Parekh, Christoph Ziegenhain, Beate Vieth, Wolfgang Enard, Ines Hellmann

Scientific Reports (2016-05-09) https://doi.org/f8kzcp
DOI: 10.1038/srep25533 · PMID: 27156886 · PMCID: PMC4860583

74. **Detection and Removal of PCR Duplicates in Population Genomic ddRAD Studies by Addition of a Degenerate Base Region (DBR) in Sequencing Adapters**
Hannah Schweyen, Andrey Rozenberg, Florian Leese
The Biological Bulletin (2014-10) https://doi.org/f6q76c
DOI: 10.1086/bblv227n2p146 · PMID: 25411373

75. **Elimination of PCR duplicates in RNA-seq and small RNA-seq using unique molecular identifiers**
Yu Fu, Pei-Hsuan Wu, Timothy Beane, Phillip D. Zamore, Zhiping Weng
BMC Genomics (2018-07-13) https://doi.org/ggv5zp
DOI: 10.1186/s12864-018-4933-1 · PMID: 30001700 · PMCID: PMC6044086

76. **Biased estimates of clonal evolution and subclonal heterogeneity can arise from PCR duplicates in deep sequencing experiments**
Erin N Smith, Kristen Jepsen, Mahdieh Khosroheidari, Laura Z Rassenti, Matteo D'Antonio, Emanuela M Ghia, Dennis A Carson, Catriona HM Jamieson, Thomas J Kipps, Kelly A Frazer
Genome Biology (2014-08-07) https://doi.org/ggfhv7
DOI: 10.1186/s13059-014-0420-4 · PMID: 25103687 · PMCID: PMC4165357

77. **Evidence for extensive horizontal gene transfer from the draft genome of a tardigrade**
Thomas C. Boothby, Jennifer R. Tenlen, Frank W. Smith, Jeremy R. Wang, Kiera A. Patanella, Erin Osborne Nishimura, Sophia C. Tintori, Qing Li, Corbin D. Jones, Mark Yandell, … Bob Goldstein
Proceedings of the National Academy of Sciences (2015-12-29) https://doi.org/9gs
DOI: 10.1073/pnas.1510461112 · PMID: 26598659 · PMCID: PMC4702960

78. **No evidence for extensive horizontal gene transfer in the genome of the tardigrade *Hypsibius dujardini***
Georgios Koutsovoulos, Sujai Kumar, Dominik R. Laetsch, Lewis Stevens, Jennifer Daub, Claire Conlon, Habib Maroon, Fran Thomas, Aziz A. Aboobaker, Mark Blaxter
Proceedings of the National Academy of Sciences (2016-05-03) https://doi.org/f8nrtb
DOI: 10.1073/pnas.1600338113 · PMID: 27035985 · PMCID: PMC4983863

79. **Large-scale contamination of microbial isolate genomes by Illumina PhiX control**
Supratim Mukherjee, Marcel Huntemann, Natalia Ivanova, Nikos C Kyrpides, Amrita Pati
Standards in Genomic Sciences (2015-03-30) https://doi.org/ggv5zn
DOI: 10.1186/1944-3277-10-18 · PMID: 26203331 · PMCID: PMC4511556

80. **Index hopping on the Illumina HiseqX platform and its consequences for ancient DNA studies**
Tom van der Valk, Francesco Vezzi, Mattias Ormestad, Love Dalén, Katerina Guschanski
Molecular Ecology Resources (2019-05-05) https://doi.org/gfwtvw
DOI: 10.1111/1755-0998.13009 · PMID: 30848092

81. **On the optimal trimming of high-throughput mRNA sequence data**
Matthew D. MacManes
Frontiers in Genetics (2014) https://doi.org/gfn5g7
DOI: 10.3389/fgene.2014.00013 · PMID: 24567737 · PMCID: PMC3908319

82. **Streaming histogram sketching for rapid microbiome analytics**
Will PM Rowe, Anna Paola Carrieri, Cristina Alcon-Giner, Shabhonam Caim, Alex Shaw, Kathleen

Sim, J. Simon Kroll, Lindsay J. Hall, Edward O. Pyzer-Knapp, Martyn D. Winn
Microbiome (2019-03-16) https://doi.org/ggv5zq
DOI: 10.1186/s40168-019-0653-2 · PMID: 30878035 · PMCID: PMC6420756

83. **STAR: ultrafast universal RNA-seq aligner**
Alexander Dobin, Carrie A. Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, Thomas R. Gingeras
Bioinformatics (2013-01) https://doi.org/f4h523
DOI: 10.1093/bioinformatics/bts635 · PMID: 23104886 · PMCID: PMC3530905

84. **Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype**
Daehwan Kim, Joseph M. Paggi, Chanhee Park, Christopher Bennett, Steven L. Salzberg
Nature Biotechnology (2019-08-02) https://doi.org/gf5395
DOI: 10.1038/s41587-019-0201-4 · PMID: 31375807

85. **RapMap: a rapid, sensitive and accurate tool for mapping RNA-seq reads to transcriptomes**
Avi Srivastava, Hirak Sarkar, Nitish Gupta, Rob Patro
Bioinformatics (2016-06-15) https://doi.org/f8vm2j
DOI: 10.1093/bioinformatics/btw277 · PMID: 27307617 · PMCID: PMC4908361

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