Streamlining data-intensive biology with workflow systems

This manuscript (permalink) was automatically generated from bluegenes/2020-gep@e511f0e on May 14, 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.

The emergence of workflow systems designed for bioinformatics has altered the landscape of ...

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 insight on workflow systems that have emerged to fill the gap for biologists....

Here, we provide a series of tips, tools, and “good enough” practices for biologists venturing into the realm of biological sequence analysis. The guidelines and tools presented below are designed to apply to novel or publicly-available sequencing data sets and across the range of computational resource options available to researchers.

The majority of this manuscript will covers understanding how to conduct computational analyses on sequencing data, Except for data acquisition, the tools and guidelines presented below apply to either novel or publicly-available data.

Author Summary

In this paper, we present our guide for biological sequence data analysis, developed through our own teaching, training and analysis. We recognize that this is currently biased towards our own use casea nd experiences, but we hope to engage in robust discussion with the open source community in order in include the best set of practices.

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. %Our main goal is to accelerate biologists/bioinformaticians into organized workflow practices that benefit their own research while also facilitating open, reproducible analyses

Introduction

Sequencing data are now widely available for species across the tree of life, and new sequencing data continues to be generated at a fantastic clip. %(cite sra growth?). The wealth of information present in sequencing data has the potential to revolutionize our understanding of the diversity and function of communities, building basic understanding from ecosystems to human health. However, sequence analysis remains both complex and computationally intensive, problems that are compounded during analysis of large datasets.

The magnitude of sequencing data requires a principled approach to management, analysis, and dissemination of results. As sequencing analysis has matured over the past decade, several papers have presented “best” or “good enough” practices for computational biological analyses . These recommendations have both helped build consensus and fueled additional tool and workflow development. Since the latest paper in 2017 , a number of important tools have greatly reduced the barrier to entry and opened the door to end-to-end reproducible analyses. % simple, shareable, etc Many of these changes owe their origin, at least in part, to the open science movement and the recognition of the importance of entry-level training, such as that provided by The Carpentries (open sci movement CITE).

The key advancements over the past few years have come in workflow scripting, software management, and tools that handle biological data at scale. % and sharing? Role of github/open code? The combination of workflow languages (e.g. snakemake, nextflow, common workflow language) and package installations (e.g. conda) have revolutionized bioinformatic analysis development. These tools enable researchers to build reproducible analyses that can be automatically executed in a directed fashion. With integrated installations, these workflows can work across different computational systems, and can even serve as a form of documentation for the analysis. Finally, when paired with new tools leveraging computational approximations, this suite of tools enables researchers to cope with the enormity of sequencing data. %have emerged a promising solution to coping with the enormity of sequencing data. %..provide researchers a framework/structure %Adopting workflow-based systems may be the single best step you can take to improve your analyses (here’s where to talk up workflows!) %bonus: these integrate with software installation! Also provide a bunch of other neat data-sciencey logging and benchmarking.

In this paper, we build on our experiences training researchers as part of The Carpentries and other courses and workshops. We present a roadmap for biological sequence analysis, beginning at data acquisition and providing specific recommendations for tools that ensure the integrity of your data along the way. We emphasize the importance of adopting a workflow-based approach to enhance documentation, automation and reproducibility of your science. Adopting these approaches will not only propel your own research, but will also facilitate sharing, discussion, peer review, etc. Here, we present our best advice on how to get the most out of your sequencing data and time.

Workflows and Software Management

Workflows are the workhorses of modern bioinformatics – most analyses require researchers to combine computational steps that involve multiple tools and often multiple programming languages. When multiple steps are combined together to execute a single analysis, this can generically be referred to as workflow. While a workflow can be manually executed step-by-step, this is both time-consuming and has the potential to introduce unintended errors. Automating a workflow using workflow languages can instead ensure that the entire data analysis is documented and repeatable from start to finish.

A number of tailored workflow systems have emerged that empower researchers to execute scalable workflows. Within these systems, the users specify steps using a system-specific syntax. The system then represents these steps as a directed graph in which each node is a step in the workflow, and edges connect sequential steps. This back-end organization, paired with additional scaffolding, makes workflows encoded in workflow systems automated, scalable, and transferable across systems. These traits are highly desirable in bioinformatic workflows where many steps commonly produce thousands of intermediate files.

Couldn't load plugin.

Figure 1: **Workflow Systems** Bioinformatic workflow systems have built-in functionality that simplifies 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, sigularity) can automate software installation for each step. **C. Branching, F. Ordering, G. Parallization** 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 reexecuting 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.

Why are workflows now useful? what has changed? (aka why were people not using them before)

The need for workflow management systems has increased with the plummeting cost of sequencing and availability of public data. While workflows are ubiquitous in bioinformatics and scientific computing in general, workflow systems designed by bioinformaticians for bioinformatic tasks are relatively new. Prior to the development of robust workflow systems, common tools for scripting a workflow included make, pydoit, or bash scripting. These systems required the user to implement substantial scaffolding around core commands on a per-workflow basis. Workflow systems have alleviated this overhead and simplified the process of scripting a workflow.

Advances in workflow systems have led to wide-spread community adoption in part attributable to the open science movement . A critical mass of workflow system-specific code has accumulated such that many routine tasks are already encoded and available for others to use . At the same time, consensus approaches for routine tasks have emerged, further encouraging reuse of existing code .

How will your life be changed with workflows?

Workflow systems have revolutionized computing in data intensive biology. Pipelines written in a workflow system are better documented, repeatable, transferable, and scalable. Because workflow systems generate directed graphs to execute a workflow, relationships between steps need to be precisely and completely specified for a workflow to execute properly. This leads to two beneficial side effects. First, workflows are more likely to be fully enclosed without undocumented steps that are executed by hand. Second, 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 6).

When a workflow is specified in this way, each step is executed in the proper order and will be rerun if a failure occurs. This frees the user from manually keeping track of execution and monitoring of each step. Fully-contained workflows (when paired with software management, SEE NEXT SECTION) scripted in a workflow system instill confidence that the code will still execute the same set of commands with little investment by the user in weeks, months, or years from the time of writing. Similarly, the standard syntax used to specify each step in the workflow lends itself to easy reuse in future project. This is in part enabled by cross-system compatibility of most workflow systems, which allows users to develop a workflow e.g. locally, and scale it on a cluster or a cloud computer.

Workflow systems contain powerful built-in features for workflow management that are available to users without the need to write additional code. The workflow system coordinates execution of the full workflow, including parallelization of non-independent steps. When a step fails, it optionally continues with independent steps. The workflow system keeps track of finished files and removes files if there was a problem in execution. In addition to coordinating runtime behavior, workflow systems can self-monitor and document resource usage, as well as compile reports that document the results of a workflow. The minimal overhead associated with implementing these features greatly empowers the user.

Getting the benefits without having to learn a scriptable 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 software.

Many research groups have used workflow software to build user-friendly pipelines that do not require learning or working with the underlying workflow software. These tools allow users to take advantage of the benefits of workflow software without needing to invest in curating and writing their own pipeline. These tools are specified in an underlying workflow language, but the user interacts with a command-line script and configuration file that coordinate and execute the workflow. Often times, workflow parameters are exposed to the user, allowing the user to control certain behaviors such as parallelization or dry-runs" when executing the command-line script. Some examples include the ATLAS metagenome assembly and binning pipeline (https://github.com/metagenome-atlas/atlas) , the Sunbeam metagenome analysis pipeline (https://github.com/sunbeam-labs/sunbeam) , the Elvers transcriptome and differential expression pipeline (https://github.com/dib-lab/elvers), the dammit eukaryotic transcriptome pipeline (https://github.com/dib-lab/dammit), and the nf-core RNA-seq pipeline (https://github.com/nf-core/rnaseq/).

Workflow systems are also available as graphical user interface systems. Websites like Galaxy, Cavatica, and EMBL-EBI MGnify offer online portals in which users build workflows around publicly-available or user-uploaded data .

How to learn to use workflows systems?

There are many scriptable workflow systems that offer similar benefits for data intensive biology. Given the plethora of choices and the steep learning curve associated with integrating a workflow system into daily workflow management, it can be difficult to decide which workflow system to adopt. While there are many workflow softwares to choose from, each software has it own strengths, meaning each software will meet an individuals computing goals differently (see). Our lab has adopted Snakemake given its integration with Python and its flexibility to execute code with different languages (e.g. bash and R) and software management tools (see section XXX) . Software like Nextflow and Common Workflow Language scale better to pipelines with hundreds of thousands of steps and support containerization more rigidly, making them ideal for production-level pipelines . There are also language-specific workflow managers, such as ROpenSci's Drake for R . Further, workflow systems are not necessarily exclusive entities; Snakemake can export pipelines in Common Workflow Language, allowing the same workflow to be fully compatible with two separate workflow systems.

Independent of computational needs, selecting a workflow system with a strong local or online community can facilitate the adoption process. These communities can provide support for new users, and have likely generated many open and accessible workflows that can be modified to analyze new data.

Alternatives to workflow systems

Workflow systems are not the only option for constructing and executing a workflow. Workflow automation can be conducted by scripting the ordered execution of each step in a language such as bash. While command line scripting is an effective solution to coordinate and execute a workflow, workflows automated in this way do not take advantage of the built-in infrastructure in workflow systems. In our experience, it is more difficult to identify partially-completed workflow steps that produced truncated files, to rerun specific steps in a workflow, and to add additional files to a workflow when using bash scripting. These shortcomings are magnified when executing workflows on large-scale sequencing datasets.

Wrangling Scientific Software

Most workflows rely on multiple software packages to generate final results. Bioinformatics research software is heterogeneous, where tools are written by different research groups, in different languages, and with varied target audiences. 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 .

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. Conda has a emerged as the leading software installation and management solution (). Conda is a package management and environment management system that allows users to install and organize software. By enabling installation of software from many languages and disciplines, and managing dependency needs and conflicts for those installations, conda brought about a revolution in package management. Although portions of conda may eventually be superceded by alternative solutions (e.g. https://github.com/QuantStack/mamba), the model of software installation and management established by conda will likely remain. Alternatively, container solutions like docker and singularity allow for for the entire computational environment to be captured and distributed, including the operating systems. This ensures that an environment is completely reproducible across different computer systems, and is common for production workflows.

While package managers and containers greatly increase reprodubility, there are a number of ways to test out software without 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 the user for language-specific tools.

Couldn't load plugin.

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) . \ **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 python3.5). 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.

Strategies to get the most out of your workflows

Developing steps that go into your workflow

Build your workflow using subsampled data

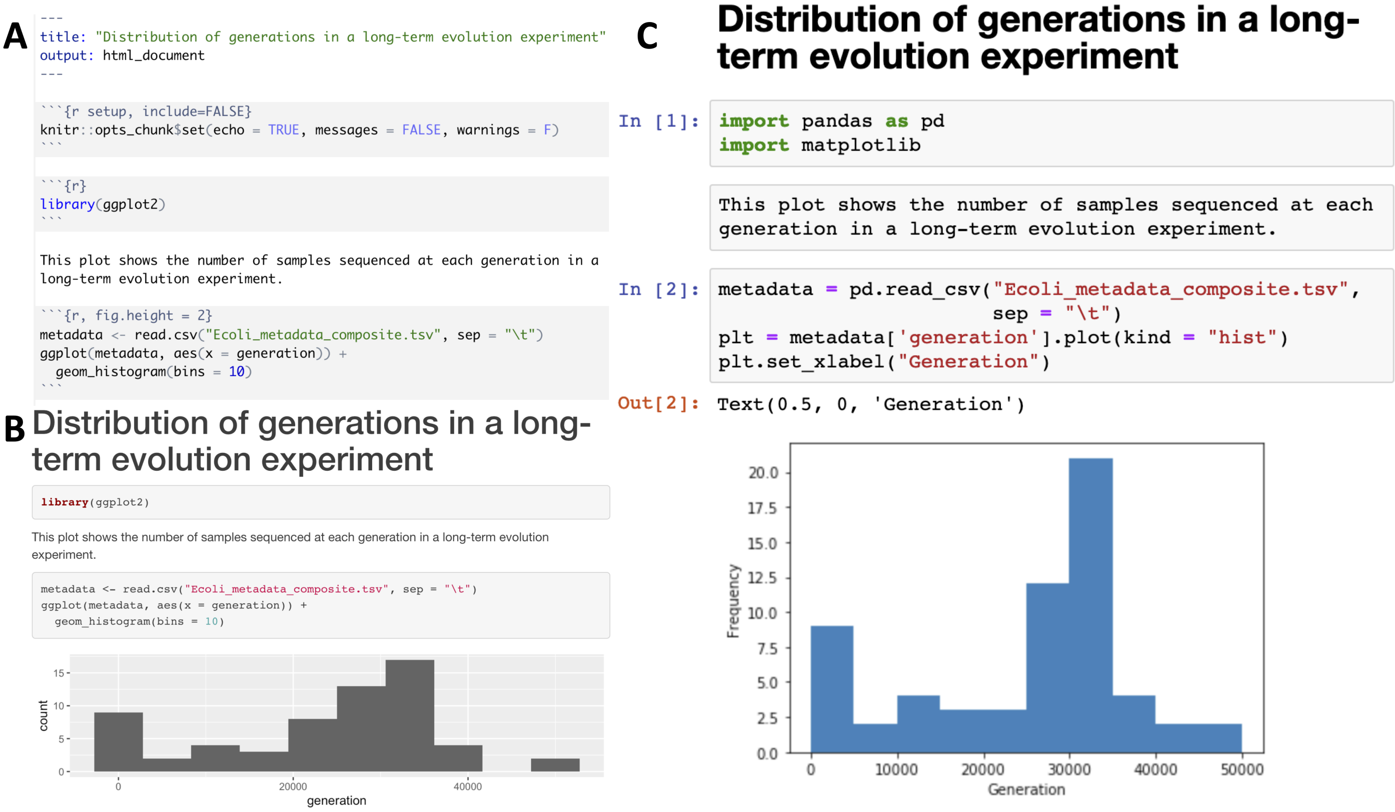
It is rare to find a workflow that will analyze your data from start to finish without testing, troubleshooting, and iteration. Testing each step with a small dataset prior to running the full analysis greatly facilitates workflow design and saves resources. After installing a program, if the program comes with test data, run it and check results against the expected results, to verify that it is working on your system. After that, subsample your own data and check you can run the program on this subsampled data. For example, if working with FASTQ data, you can subsample the first million lines of your data (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. Note, some programs will fail with too few reads or too few results, so be sure to examine that possibility if running into errors at this stage, either in the literature, program manual, or by running larger subsets of data. % add sentence here to reiterate concept of testing on public data!...and that workflows make this easy.

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. Jupyter notebooks and Rmarkdown are the two most popular notebook platforms (see Figure {fig:nb_figure}) . Notebooks 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.



~~~
---
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)
```
```

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)
```

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 3: Examples of computational notebooks. Computational notebooks allow the user to mix text, code, and results in one document. **A** shows a raw RMarkdown document viewed in the RStudio integrated development environment, while **B** shows a rendered HTML file produced by knitting the RMarkdown. **C** shows 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 . Computational notebooks facilitate sharing by packaging narrative, code, and visualizations together.

Version Control

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 that will inevitably undergo multiple changes in parameters, data, visualizations, and analysis, it is essential to keep track of which analysis produced which output files, so that the final analysis is reproducible. However, version control can also save time and effort at every step. 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. If multiple people are working on the same project, version control can both avoid conflict and ensure that no productivity is lost.

Version control systems such as Git or Mercurial can be used to properly keep track of all changes over time, even across multiple users, scripting languages, and including visualizations. In particular,
```

Git has emerged as the dominant version control system for biological code, particularly when combined with online repositories such as Github, GitLab, or Bitbucket, which store online version histories for all tracked files . In addition to acting as an additional backup location, the online services support drag-and-drop file addition and full control over the repository using the web interface, which greatly lowers the barrier to getting started with version control systems. While these systems do not work well with Google Docs or Microsoft Word, they can greatly simplify asynchronous collaborative manuscript writing when combined with services such as Overleaf and Manubot .

Git version control is primarily designed to handle small text files, but version control also exists for data sets. 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 paramters or versions. The Open Science Framework (OSF) , maintained and developed by the Center for Open Science, provides free storage of an unlimited number of files up to 5GB each in size, and allows the user to keep the data private until they are ready to share (make the project public). OSF provides built-in version control and is supported by a data preservation fund that will keep the data available for 50+ years. While OSF and other similar repositories (e.g. figshare) are suitable for use at any stage of a research project, repositories such as Zenodo and the Dryad Digital Repository (Dryad), are designed to make publication-ready data discoverable, citable, and reusable. Other services are compatible with git version control, e.g. Git Large File Storage (LFS) and Data Version Control (DVC).

These version control systems can 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 DOI for that release using Zenodo and make it available to reviewers and beyond (see “sharing” section, below).

Couldn't load plugin.

Figure 4: **Version Control** In this example, a typo in version 1 (in red) was corrected (green). Version control systems such as git and mercurial track line-by-line changes and store the information. These systems are particularly useful to handle accidental deletions or early code or text you’d like to return to after deletion.

## Documenting reproducible workflows for yourself and others

As with experimental biology, it is essential to write down everything you do - that is, record the origin of every file (e.g. download URL) and all metadata, record the version of software and each parameter you used, record any manual filtering or data preprocessing steps, and keep track of the order in which you executed each program. Without the ability to fully examine and reproduce your analysis, it will be impossible for you or your collaborators to assess whether the results are accurate, or even to understand how the heuristic decisions you took impact the conclusions made from an analysis.

Computational project management is a learned skill that will take time to implement. There are a myriad of ways to document your computational work, and you’ll need to experiment with the ways that work for you. For some portions of your project, you may want to document your work using a narrative approach, using written language to detail what steps you took and to communicate how the steps relate to one another. For other portions, it may be more useful to keep short, bulleted notes with your code or intersperse your commands with helpful diagrams. What is most important is to develop a clear documentation strategy and stick with it tenaciously. While the preferred tools discussed below will certainly change over time, these principles apply broadly and will help you design clear, well-documented, and reproducible analyses.

### Use consistent and descriptive names

Consistent, descriptive names keep your project organized and interpretable for yourself and collaborators. This applies to your files, your scripts, your variables, your workflows, your manuscripts, and even your projects, each of which should have a unique and descriptive identifier. Since the number of files in data-intensive biology can quickly get out of hand, consistent file naming is especially important. For example, you can implement a numbering scheme for your files, where the first file in your analysis starts with “00”, the next with “01”, etc. You can also append the tool name to output to make it clear where the file came from. Additionally, using a standardized yet flexible folder structure from the outset of your project facilitates file organization, even as a project becomes increasingly complex. Keeping independent portions of your analysis in descriptive folders can help keep your project workspace clean and organized. Within your files, using consistent and descriptive variable names will help build a readable codebase.

Couldn't load plugin.

Figure 5: filenaming caption goes here

### Store metadata about your workflow with your workflow

Biological analyses often span hundreds of steps and involve many small decisions: What parameters 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. 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 be (or contain) documentation - you can add comments with the reasoning behind parameter 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 (https://github.com/dib-lab/dib-MMETSP) .

### Add visual representations

Visual representations illustrate the connections in a workflow. 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. Whenever possible, adding visualizations can help improve the readability and reproducibility of your project. Figure 6 illustrates a workflow visualization modified from a graph produced by the workflow software Snakemake .

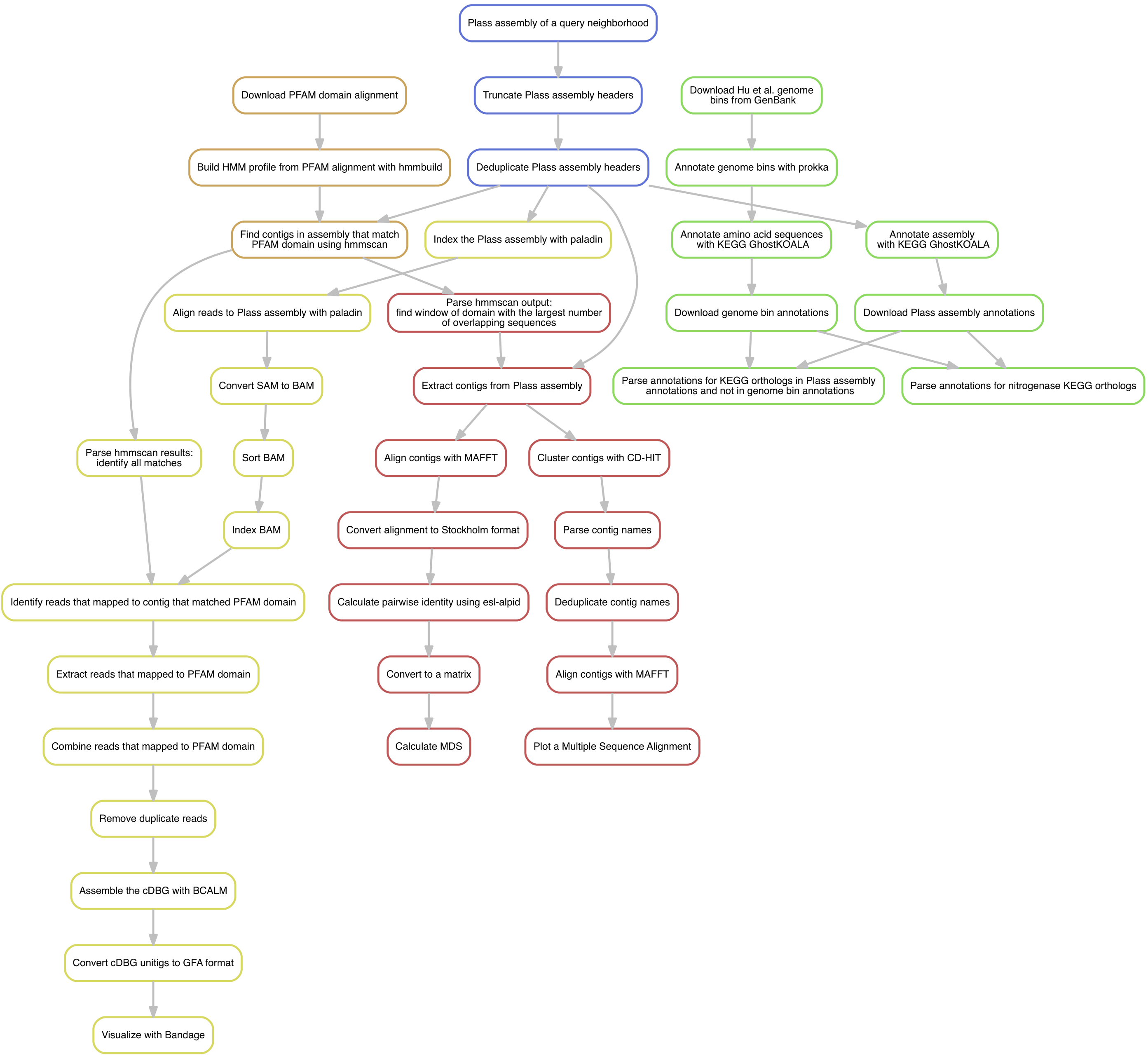

Plass assembly of a query neighborhood
Download PFAM domain alignment
Truncate Plass assembly headers
Download Hu et al. genome bins from GenBank
Build HMM profile from PFAM alignment with hmmbuild
Deduplicate Plass assembly headers
Annotate genome bins with prokka
Find contigs in assembly that match PFAM domain using hmmscan
Index the Plass assembly with paladin
Annotate amino acid sequences with KEGG GhostKOALA
Annotate assembly with KEGG GhostKOALA
Align reads to Plass assembly with paladin
Parse hmmscan output: find window of domain with the largest number of overlapping sequences
Download genome bin annotations
Download Plass assembly annotations
Convert SAM to BAM
Extract contigs from Plass assembly
Parse annotations for KEGG orthologs in Plass assembly annotations and not in genome bin annotations
Parse annotations for nitrogenase KEGG orthologs
Parse hmmscan results: identify all matches
Sort BAM
Align contigs with MAFFT
Cluster contigs with CD-HIT
Index BAM
Convert alignment to Stockholm format
Parse contig names
Identify reads that mapped to contig that matched PFAM domain
Calculate pairwise identity using esl-alpid
Deduplicate contig names
Extract reads that mapped to PFAM domain
Convert to a matrix
Align contigs with MAFFT
Combine reads that mapped to PFAM domain
Calculate MDS
Plot a Multiple Sequence Alignment
Remove duplicate reads
Assemble the cDBG with BCALM
Convert cDBG unitigs to GFA format
Visualize with Bandage

**Figure 6:** 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 . 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.

## Sharing Your Reproducible Analyses

Sharing your workflow is a useful way to communicate every step you took in a data analysis pipeline. Your collaborators, peer reviewers, and scientists seeking to use a similar method as your own will all benefit from open and accessible code. 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. However, sharing code in this way can still be burdensome for others to interact with given the need for software installation and differences in user operating systems. Tools like Binder, Whole Tale, and Shiny apps can reduce the time to reproduction by other scientists by constructing controlled environments identical to those in which the original computation was performed . 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. These tools are also great for teaching, as they provide consistent learner interfaces and environments.

### sharing greybox

**Binder (mybinder.org)** Binder is a tool that makes a GitHub repository executable in a specified environment . It uses package management by R, pip, or conda to build a docker container with the software required to run the analyses contained in a GitHub repository. This is especially useful for computational notebooks. The binder can then be shared with collaborators (or students in a classroom setting), where the analysis or visualizations can be reproduced using your provided code. Binder instances are not static and can be modified by collaborators during a binder session. The underlying code will not be changed unless changes are committed into the original GitHub repository, preserving the reproducibility of the original analysis.

**Shiny Apps** Shiny is a tool that allows you to build interactive web pages using R code. It allows you to package data that 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. For example, if a Shiny app contained RNA-seq differential expression data, it might allow the user to specify which gene counts it should plot. Shiny apps allow collaborators who may or may not know R to change R visualisations to fit their interests.

**Other tools** There are many other tools that aid in reproducibility or sharing of results. Some tools, such as Whole Tale, aim to package the entire research object by capturing data, code and software environments in one executable package . Other tools such as Vega-lite and plotly produce single interactive visualizations that can be shared with collaborators or integrated into websites .

## Scaling Workflows

### Assess required resources

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. While it can be difficult to estimate resources required for each tool, workflow systems provide built-in tools to monitor resource usage for each step. This reporting can be used while running a workflow on a single or a few samples to estimate required resources. These resources can then be specified to run the workflow on all samples.

### Scale workflows with tools that leverage computational approximations

Many bioinformatics workflows take a long time and significant computational resources to run, and interpretable results are often only produced by the last few steps. This means that time-to-insight from sequencing data is often very high.

Understanding the basic structure of data, the relationship between samples, and the approximate composition of each sample is 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** 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 the user 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 .

**Read quasi-mapping vs alignment** 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 . However, these tools generate information-rich output, specifying per-base alignments for each read. Quasi-mapping produces the minimum information necessary for read quantification, thereby reducing the time and resources needed to generate and store read count information .

discuss blast approximations?

## Practical considerations for workflow-enabled biology

While workflows enable data intensive biology, many steps like finding or backing up data are orchestrated outside of a workflow. Below we discuss practical considerations for working with large-scale data sets.

### Obtaining data

As with all biological analyses, a critical step of sequence analysis is obtaining high-quality data for your scientific question. 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.

### Accessing publicly-available data

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 . Additional curated databases focus on processed data instead, such as gene expression in the Gene Expression Omnibus (GEO) . Organism-specific databases such as **Wormbase** (*Caenorhabditis elegas*) specialize on curating and integrating sequencing and other data associated with a model organism . 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 . 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 ).

## 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 go about designing your experiment (see Table ??), 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.

```
|  Sequencing type | Resources |
| --- | --- |
|  RNA-sequencing | \cite{conesa2016, schurch2016, ching2014} |
|  Metagenomic sequencing | \cite{knight2012, quince2017, eisenhofer2019} |
|  Amplicon sequencing | \cite{mclaren2019, murray2015, sinha2017 } |
|  Microbial isolate sequencing | \cite{liao2015} |
|  Eurkaryotic genome sequencing |  |
|  Whole-genome resequencing | \cite{fuentes2017} |
|  Rad seq |  |
|  Chip seq |  |
|  ATAC seq |  |
|  single cell RNA-seq | \cite{bacher2016, haque2017} |
|  ? |  |
Table: References for experimental design and considerations for common
        sequencing chemistries.
{#tbl:seq_resources}
```

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. When working with metadata, using a standardized format that is easy for a computer to read can simplify downstream analysis. Standard guidelines for formatting data for scientific computing are given in .

Securing Computational Resources

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 . 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 ??). 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.

```
|  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 |  |
|  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 |

Table: **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.
{#tbl:computational\_resources}
```

Transferring Data (or not)

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 . 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). Checksums can be used to to check file integrity and ensure proper transfer (see Figure 7). File compression (gzip, bzip2, BAM/CRAM, etc.) can improve transfer speed and save space. Tools like Rsync and Rclone automate file transfer between computers or between computers and remote storage providers . These tools automatically use checksums to verify that files were transferred properly. Some GUI file transfer tools (e.g. cyberduck) also assess checksums when they are provided.

Couldn't load plugin.

Figure 7: **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. %hash algorithm. 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). For publicly-available files, a checksum value is often provided, so that you can check the integrity of the file after download.

Backup and storage

Workflow sysems allow users to generate an automatable, repeatable workflow. This removes the necessity of storing some intermediate files as these can be easily regenerated. However, storing or backing up raw data or computationally intensive results can be a burden in data intensive workflows. Many universities provide cloud storage space (e.g. through google drive, box, dropbox etc), and researchers can pay for individual storage on these services, or services attached to cloud computing (e.g. Amazon Web Services). Full computer backups can be conducted to these storage locations (e.g. with rclone), or there are also a number of paid services that will conduct backups at regular intervals (e.g. Backblaze, Dropbox Pro). Free online repositories mentioned in the version control section like OSF, Dryad, and Zenodo can also store or backup important files.

Regardless of which storage or backup strategy you choose, it is a good idea to have multiple independent backups of raw data and workflows; these cannot be easily regenerated if lost to a computer failure or other unforseeable event like a lab fire.

% Link out to data/project management info? e.g. Many of these considerations are addressed in a data carpentry lesson (https://datacarpentry.org/organization-genomics/).

Ensuring you have quality data

The adage, "garbage in, garbage out" describes most sequencing data analysis: the quality of the input data determines the quality of the output results. This is true whether your workflow analyzes six samples or 600 samples. Assessing data at every analysis step can reveal problems and errors early. You are the single most effective quality control tool that you have, so its important to interrogate your data to search for problems. While simply looking at your data is sometimes sufficient to catch issues, visualization and software tools make quality control easier.

Critically assess your data

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 (Table ??). 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.

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 8) 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 .

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 8).

```
|  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 |

Table: 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.
{#tbl:bash_commands}
```

Couldn't load plugin.

Figure 8: **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 . 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 . 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 78 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.

Handling 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. For example, PCR duplicates can cause problems in libraries that underwent an amplification step, and often need to be removed prior to downstream analysis . 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 . Libraries sequenced with high concentrations of free adapters or with low concentration samples may have increased barcode hopping, leading to contamination between samples . Stringent trimming of RNA-sequencing data may reduce isoform discovery . The exact biases in a specific data set or workflow will vary greatly between experiments. 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. Therefore, it is important to think critically about your expectations, and what patterns you expect to see given your data and your biological problem.

References