Streamlining data-intensive biology with workflow systems

This manuscript (permalink) was automatically generated from bluegenes/2020-gep@2d19e7e 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.

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 ??).

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.

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.

References