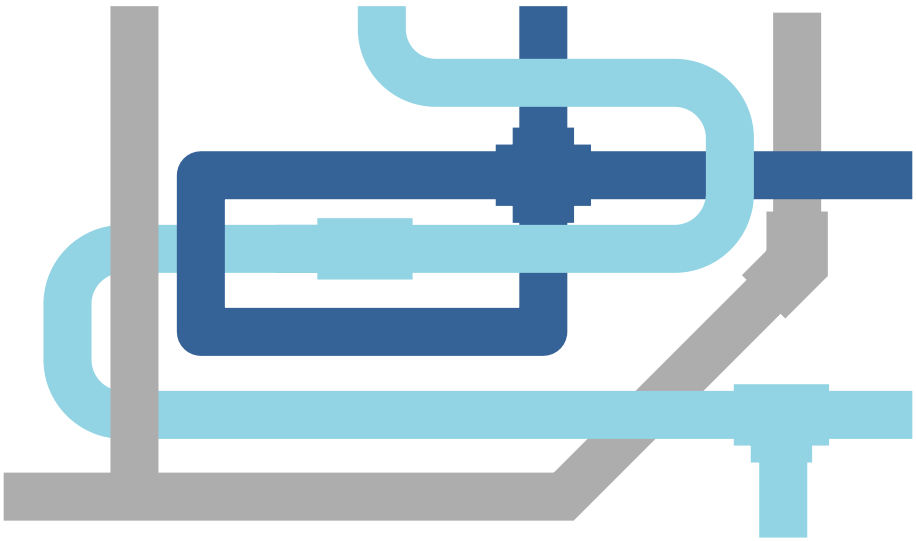
When computational pipelines go 'clank'

'Omics pipeline builders and users face options and tough decisions.

Vivien Marx

"I love the smell of fresh baked Snakefiles in the morning!" tweeted University of California, Davis, computational biologist C. Titus Brown earlier this year. Brown and his team, including postdoctoral fellow N. Tessa Pierce, are putting together a computational pipeline for decontaminating genome assemblies in metagenomics and are using the workflow manager Snakemake[1] to do so. It's one of over 100 workflow manager tools pipeline builders might use. Without data analysis pipelines, many 'big data' fields would move dramatically slower, says UCLA cancer researcher and computational biologist Paul Boutros. Pipelines and their infrastructure become joint workspaces for computational biologists and software engineers, he says. Not all pipelines are as standardized as, say, the Genome Analysis Tool Kit steps for mapping sequence to a reference. The choice of pipeline can lead to differing results on the same data, as a study in neuroimaging has revealed[2]. Using, building and maintaining pipelines presents researchers with a forked road. (You can read more on our blog.)



Not all data analysis pipelines are standardized. Labs face many decisions when using, building and maintaining pipelines.

17,000 is a lot

Pipeline building means tool choice. The bio.tools registry of life science software tools[3] includes over 17,000 entries from more than 1,000 contributors. The profusion of tools in bioinformatics is great but also an immense challenge for creating pipelines, says Boutros. "Each tool has its own idiosyncratic features around how it wants inputs and outputs." Tools might require specific operating systems or have software dependencies; versions will differ or have specific directory structures. "A huge fraction of the pipeline-builder's dilemma is to figure out how to handle all that variability," he says. Some of the variability in tool formats, such as for outputs, is reasonable, he says, given how varied tools are. "The number of tools is both a blessing and a curse," says Manfred Grabherr, a computational biologist at Uppsala University and chief technology officer at Methority, a company with algorithms and applications in machine learning and artificial intelligence. The choices slim down with standard tasks in genome assembly, gene expression or population genetics.

When researchers pursue new ground, hundreds of tools might seem suited, "some of which promise to do what you want — except they don't quite fit," he says. Or they don't run well. But resources such as bio.tools can help. Juan Antonio Vizcaino, proteomics team leader at the European Molecular Biology Laboratory's European Bioinformatics Institute (EBI), agrees on bio.tools, but users will always need "to talk to experts before doing anything serious," he says. An analysis[4] of the over 1,000 proteomics tools in bio.tools noted difficulty in finding even "basic information" about tools sometimes, which points to a pressing need for better standards of information for life science software. With Galaxy and other open source platforms, users can set up workflows from the platform's 'toolshed' of nearly 8,000 tools and run a workflow in the cloud. "We definitely do not serve old tools, but we do keep them for reproducibility," says Anton Nekrutenko, a Pennsylvania State University researcher and one of Galaxy's cofounders. Users will want to investigate tools they might choose, says Irene Papatheodorou, EBI team leader on gene expression, and scour the literature to see how results compare. "Review papers benchmarking different tools are extremely helpful, but in many cases the pipeline builder will need to run their own comparisons," she says.

It broke

"Broken pipelines have been a major issue from the beginning," says Grabherr. Dependencies might be missing, which are additional software bits a tool needs to run well. A solution, says Boutros, chosen by the Dialogue on Reverse-Engineering Assessment and Methods (DREAM) challenges for genomics tools, is to predefine the input and output and to require containerized

"I love the smell of fresh baked Snakefiles in the morning!" tweeted C. Titus Brown, University of California, Davis.

software solutions. It eases dependency issues when a pipeline workflow is packaged in a virtualized environment with executable tools. Containers might be Docker systems; there are also Singularity and rkt, among others. The issue of dependencies "is a problem everyone needs to solve before redistributing their software to the users," says Jinghui Zhang, chair of computational biology at St. Jude Children's Research Hospital. With containers, pipeline developers can configure an environment with all the required packages to ship it to users, says her St. Jude colleague Clay McLeod, a bioinformatician. To minimize issues with dependencies and pipeline breaking, says Papatheodorou, she and her team modularize pipelines into smaller, distinct micro-services with specific functions that can be updated separately as needed without breaking the surrounding ecosystem.

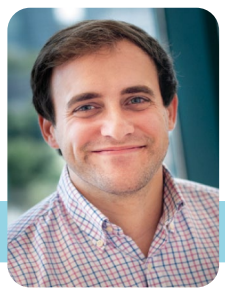
Small lab pipelines

It's not too challenging for a lab to build small pipelines for its own purposes with existing tools and draw on colleagues with computing experience, says Zhang. Deploying tools for public use "is a different story." A smaller lab, says McLeod, might want to first carefully craft a set of input examples and expected output and "let that truth set evolve over time as you check your pipeline's performance." Users need to know the context of a tool or pipeline. "There is no silver bullet tool or parameter set that works for all applications in bioinformatics," he says. Pipeline authors need to reveal the assumptions that went into development and the implications for results. "Too often, you see bioinformatics tools picked up off the shelf and applied in a context which doesn't make sense because of this communication breakdown." "Developers regularly obviate important aspects of development such as error-handling, scalability or documentation, mostly because they develop for their own benefit," says David Ochoa, platform coordinator for EBI's Open Targets project, in which therapeutic targets are identified and assessed with genome-scale analyses. Distributing the code and building a community of developers and users around a pipeline is very often underestimated. "Reading code is hard and not deeply rewarding; writing it is," says Boutros. Scientists thus prefer building to poring over others' code, and the number of cloud-based pipelines keeps growing. "Many labs want to create their infrastructure, which leads to a lot of poorly maintained pipeline code." Instead of building from the ground up each time, smaller teams might benefit from making their pipeline easier to embed into a larger one. The time saved can be used for optimizing it for their specific needs.

It costs

Pipelines advance 'big data' fields, which the life sciences are becoming. But to get this right, says Boutros, testing and benchmarking need to be almost daily activities. That's work, and it's costly. Standardization is common in routine clinical applications but less so in scientific discovery, where labs explore new study designs and research questions. "The more specific, detailed and complex a pipeline is, the less likely it is to be useful to anyone else," says Björn Nystedt. To build and maintain broadly useful computational pipelines for research, "the developer therefore has to find a sweet spot," he says, to address a task common enough to be relevant to many, yet complex enough to provide substantial added value for the user. Nystedt directs the National Bioinformatics Infrastructure Sweden for SciLifeLab, a research initiative started as joint effort between Karolinska Institute, KTH Royal Institute of Technology, Stockholm University and Uppsala University that now supports research activities all across Sweden in the life sciences and medicine. Rapid change makes pipeline maintenance ever more challenging, he says. In the life sciences over the last decade, data types, data volumes and analysis software have seen rapid development. Computational pipelines become outdated more quickly, and staying valid takes resources for updates and maintenance, he says. Several good initiatives related to pipelines have been launched with open-source code and contributions from a community of developers, says Nystedt, among them bcbio-nextgen, a community portal for automated high-throughput sequence analysis pipelines, and the Nf-core[5] framework, with community-based curated pipelines. Carole Goble of the University of Manchester and her team are putting final touches on a workflow registry that is in "pre-pre-beta," she says. Scientists can test pipelines on OpenEBench and Life Monitor.

It's not too hard for small labs to build pipelines for their own use, but deploying tools for public use "is a different story," says Jinghui Zhang.

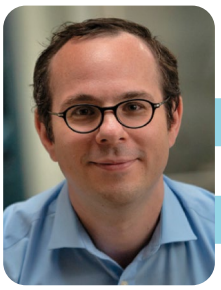


"As a pipeline matures and scales, continuous integration is needed," says Clay McLeod.

Credit: St. Jude Children's Research Hospital

Go with the workflow

To ultimately calculate differential expression, says Brown, might involve, for example, RNA-seq data, downloading reference genes, and applying software. Within a platform — whether in one's own lab, a Galaxy instance or other virtual environment — workflow managers help with assembling and executing a workflow. Tools can be hard to compose, says Brown, and much pipeline-building involves converting formats for passing information between tools. There's a big difference between production workflows that can run at scale and almost never fail without a useful error message and research workflows run on a dozen samples where much can happen. Workflow managers help users set up a computational analysis pipeline and monitor its activity as it runs. "I used to spend a lot of my time worrying about intermediate errors in my automation, and workflow managers take care of that," he says. With Snakemake, workflow steps phrased in a Python-like language are set up in a Snakefile. Snakemake can determine how the workflow can unfold, whether



"Many labs want to create their infrastructure, which leads to a lot of poorly maintained pipeline code," says Paul Boutros, UCLA.

Credit: UCLA Jonsson Comprehensive Cancer Center

over one or several cores, and can help with scaling a pipeline from a workstation with a single core to many-core systems. In Brown's view, bioinformatics teams mainly use four workflow managers: Workflow description language (WDL), Common Workflow Language (CWL), Snakemake and Nextflow.

Even when workflows just cobble together software in a simple way, every computing decision is a choice and involves implicit assumptions, says Brown. Unintended consequences of joining different programs make it harder to track and evaluate bugs. Especially for production-grade pipeline development and operation, workflow managers are becoming an increasingly important part of the equation, says McLeod. "Investing in learning these workflow languages and rewriting pipelines to conform to them has to be balanced with the benefit you will receive in return, like scalability and platform portability," he says. A smaller lab working on a single workstation is likely to be fine with simple bash commands or Python scripts and no workflow language, says McLeod. Snakemake and Nextflow appear popular in medium-sized or larger individual labs with multiple people working on a project because they are relatively easy to pick up and can be well-integrated with typical local computing setups. Based on what he has seen of community projects and cloud providers, for larger operations, such as at St. Jude, open workflow standards like CWL and WDL appear to be the leading workflow languages.

Make it sustainable

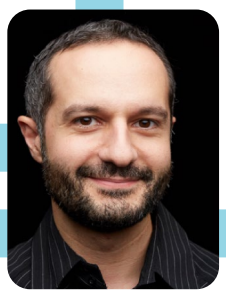
Brown works with others in a new Common Fund Data Ecosystem project geared toward increasing uptake and reuse of Common Fund data and tools and for training. The project also addresses gnarly issues of sustainability and preservation of data and tools in cases of defunding — for example, of the Human Microbiome Project. In general, the US National Institutes of Health wants scientists to head to the cloud for data analysis, says Brown, but not all labs are well-versed in doing so. With the cloud, the main problem is upload and download, says Grabherr. Opening a virtual machine or several of them on different clouds is relatively easy, but the major bottleneck is getting the data there and downloading results. For permanent storage of all the input data, researchers need their local resources, and a laptop won't hold terabytes of data. Unless data can be read through the network instead of from local disks, he says, people will continue to make copies of their large datasets. Scalability is another issue. "Especially for larger datasets, it might take many thousands of CPU hours before realizing that a tool or pipeline will never work for this data."

Validating, troubleshooting

Validation has to be continuous, says Zhang, and benefits from use of a benchmark dataset. User feedback helps with fixing errors. At a minimum, says Grabherr, pipelines should, but don't always, come with a small test data set for verifying that they generate correct results. "Next, there should be a clear versioning, not just a download date," he says, along with notes about changes. Larger outfits, such as EBI, the Broad Institute or University of California, Santa Cruz, "are generally very good at transparency and support, others less so." For validation, says McLeod, for starters it's okay to manually run end-to-end checks for errors and consistency of results at regular intervals. "As a pipeline matures and scales, continuous integration is needed," he says. Every incremental change to a pipeline kicks off the entire end-to-end pipeline check to find errors and assure consistent results. This way, pipeline developers can focus on development and let automation handle the rest, he says. Validation is challenging in research, not just with pipelines, says Nystedt. Assuring valid results is the responsibility of the individual research project and cannot be 'outsourced' to the pipeline maker. "And exactly how this evaluation should be done is almost always project-specific, and might in fact amount to a substantial part of the entire research project." Boutros agrees on the need to benchmark pipelines against standard datasets, such as Genome in a Bottle or DREAM. Pipeline changes should be accompanied by a test against that dataset to show how performance, both in terms of computing resources and results accuracy, has changed. Big improvements might hint at over-fitting, he says, so it's good to have many, diverse benchmarking datasets. With his recent move to UCLA, he began daily testing of all pipeline aspects against standard datasets. Pipelines are notoriously difficult to debug, says Grabherr. Processes can fail for any number of reasons. When a pipeline runs for several days or weeks, "then fiddling with parameters becomes a major wastebasket for both time and compute resources." He has seen, for example, that after many unsuccessful attempts, the one parameter that needed changing was hard-coded in the pipeline. Structural problems are easy to trace and debug, says Miguel Carmona, software developer with the Open Targets project. He and his colleagues use continuous integration and continuous development measures, along with tools such as the cloud-based application Travis, to leverage the compilation, packaging and execution that, for instance, Docker gives and to ensure that commands run without host dependencies. "Semantic problems are rather more intricate and difficult to spot as you need expert human intervention," says Carmona. One example is assigning annotation to a gene that is computationally correct but biological nonsense. He and

Several good open-source initiatives related to pipelines have been launched from a community of developers, says Björn Nystedt, SciLifeLab.

Bioinformatics is changing as fast as science, says John Ellithorpe, DNAnexus executive vice president and chief product officer.



"Compute is getting more ubiquitous and less expensive and the data are getting larger," says George Asimenos, DNAnexus chief technology officer.

his team developed checkomatic, a tool that checks expert-curated knowledge as a gold-standard ground truth. It helps to minimize the most common problems related to the lack of data or mis-typed data. Proteomics researcher Jürgen Cox from the Max Planck Institute of Biochemistry says his lab's approach is unlike mainstream pipeline building in bioinformatics and "ours is probably a minority view." The classic pipeline building approach in bioinformatics has many merits, he says. However, in developing MaxQuant, software for quantitative analysis of large mass spectrometry datasets, and its cousin Perseus for statistical workflows such as cross-omics analyses, his team took a different path. They built scientific software solutions from scratch with a local team of scientists and programmers, which is like industrial software development. Having thousands of independently developed pipeline components "available for algorithmic plumbing might seem like a paradise for some of the students who will be able to quickly solve a computational problem that they encounter in their data analysis," says Cox. But all too frequently this leads to difficult-to-maintain solutions with a short life cycle. Reuse of such pipelines can be hard because of usability issues, and it can be tough to scale the pipeline to other hardware platforms. Maintenance keeps an approach updated so it can, for example, reflect data from a given field's new technologies. "This is why we favor the complete system-building approach over the pipeline approach," says Cox. This does not exclude users from contributing to the software. For instance, Perseus has a plug-in architecture through which users can add their own workflow commands in any of a number of programming languages, such as R, Python and C#.

Sharing at scale

Selecting the right tools for a particular analysis requires thorough evaluation, especially for analysis of a new data types, says Zhang. She and her team have built the St. Jude Cloud to share large-scale pediatric cancer data and tools, many of which were developed at St. Jude. It's also a sandbox where tool developers and users can try analysis pipelines. Initially, the Zhang lab tools were on the lab's research cluster and shared via the lab page or Github. People could download tools for installation wherever they chose. In 2014, Zhang began using the cloud, and most of her lab's new methods are on St. Jude Cloud. In cancer research, scientists might look and find tools in many locations, such as the NCI Cloud or the International Cancer Genome Consortium portal. The St. Jude Cloud is specifically devoted to pediatric cancers. Users can browse the St. Jude Cloud data, but permission is required for access and analysis, says George Asimenos, chief technology officer at DNAnexus. His company built a layer that runs on top of the Microsoft Azure and Amazon Web Services clouds for data storage, analysis and computing nodes. Some organizations, including St. Jude, the Encyclopedia of DNA elements (ENCODE) consortium, the UK Biobank and the Vertebrate Genomes Project, involve DNAnexus to scale up pipelines. A regulatory environment can shape a pipeline's scope, but research-focused consortia, such as one in population genetics, might control pipelines too. They might need to process all samples uniformly so researchers can later "compare apples to apples," says Asimenos. During the ENCODE pilot project, a pipeline faced a versioning issue when, mid-analysis, the human reference genome changed. The Regeneron Genetics Center has processed exomes for the UK Biobank and had to tweak the pipeline version and communicate this widely. Especially in large endeavors, he says, scientists have "to assure whatever they do today is relevant years from today." Bioinformatics is changing as fast as science is, says John Ellithorpe, DNAnexus executive vice president and chief product officer. On the pipeline engineering side, people might want to lock things down before release whereas on the science side, teams want to get a pipeline to the community as quickly as possible. "Those two worlds don't mix particularly well," he says. DNAnexus mediates by building different pacing into a project. They might, for example, let users or developers revert to a pipeline four versions ago. Many pipelines are for readying data to allow association studies across datasets. This brings issues of architecture, organization and how to avoid copying data from here to there, he says. Over time, more and more groups will work on structural aspects instead of relearning a new pipeline environment.

Classic software practices such as those for maintaining large bodies of code and pipelines are making inroads in bioinformatics, says Ellithorpe. "All of these things are going to end up feeding major data science environments." Workers can use pipelines for simple association and correlation analyses with smaller sets of data or perform large-scale approaches with machine learning and artificial intelligence techniques. "Compute is getting more ubiquitous and less expensive, and the data are getting larger," says Asimenos. Data, such as that from the UK Biobank, makes it increasingly possible for labs around the world to ask new kinds of biomedical questions. The days are vanishing quickly, notes Asimenos, in which a researcher says: "I'm using one tool and it's running for a specific amount time, that's all I can afford, and I'm using it on my dinky dataset." ❐

Vivien Marx ✉
Nature Methods.
✉*e-mail: v.marx@us.nature.com*

Published online: 29 June 2020
https://doi.org/10.1038/s41592-020-0886-9

References

1. Köster, J. & Rahmann, S. *Bioinformatics* **28**, 2520–2522 (2012).
2. Bovotnik-Nezer, R. et al. *Nature* **582**, 84–88 (2020).
3. Ison, J. et al. *Nucleic Acids Res.* **44**(D1), D38–D47 (2016).
4. Tsiamis, V. et al. *J. Proteome Res.* **18**, 3580–3585 (2019).
5. Ewels, P. A. et al. *Nat. Biotechnol.* **38**, 276–278 (2020).