Title	CWL in Practice: Experiences, challenges, and results from adopting Common Workflow Language
Authors	*Dan Leehr*, Alejandro Barrera, Tim Reddy, Hilmar Lapp
Affiliation	Duke University, Center for Genomic and Computational Biology
Contact	dan.leehr@duke.edu
URL	https://github.com/Duke-GCB/GGR-cwl
License	MIT

Assembling individual bioinformatics software packages to run together as a pipeline is often done by scripting them together in an implementation language like Bash, Python, or Perl. While simple to build and understand, this often yields pipelines that are difficult to repeat across computing environments, are not generalizable across data sets, and require nontrivial effort for tracking and archiving the exact versions of the tools that make up pipeline components. As a result, reusing, adapting, or even only repeating a bioinformatics analysis pipeline later in an environment other than its original one has become notorious for the difficulties involved. To address this problem, the Common Workflow Language (CWL) endeavors to provide a blueprint for architecting workflows declaratively. With open source implementations and a comprehensive specification, CWL provides a basis for building reusable components and reproducible, modular bioinformatics pipelines that facilitate swapping different analytic methods or algorithm implementations in and out. Because CWL is agnostic of workflow engine implementation, pipeline definitions do not need to be tied to a particular execution environment. CWL also encourages the use of Docker images for pipeline components, which not only isolate tools and their myriad of dependencies from each other, but also allows entire compute environments to be faithfully archived, shared, and reused.

Here we report on using CWL in support of the Genomics of Gene Regulation (GGR) project, a multi-institutional collaboration which aims to comprehensively characterize and better understand the first 12 hours of the glucocorticoid response (GCR). The lab of one of the project PIs at Duke, Tim Reddy, uses gene editing techniques to turn on and off genes, with the goal of elucidating the network of gene regulation involved in the GCR. This involves comparing the results of numerous genomic experiments in each condition, which requires developing data processing pipelines that can be confidently repeated, reused and customized, both by collaborators within the project, and by scientists at large wishing to reproduce and build on the team's findings.

We will highlight our experiences, challenges, and results from transforming the team's existing scripted workflows to workflows defined in the CWL standard. In comparison, GGR workflows migrated to CWL became more flexible, easier to modify, and amenable to replacing entire parts with alternative methods, which improved the project's analytic capabilities. Another benefit was the ability to reuse pipeline components shared between the analysis of ChIP-seq, RNA-seq, and DNase-seq experiments. Finally, publishing alongside a paper the CWL definitions of the underlying computational pipelines and the data sets allows others to reproduce the results, even though our specific high-performance computing environment, like most others, is not a public resource.

During the development of these CWL pipelines we have also faced two major challenges, and we will detail our approaches to them. One of them was translating the control flow and imperative scheduling logic found in the existing workflows expressed in a Turing-complete scripting language, to CWL's declarative language that is evaluated at compile time. The second challenge is the practical aspects of adopting CWL workflows in a shared academic HPC cluster environment. Our research center operates a computational genomics workload-tailored HPC cluster running Slurm as the schedular, and as a shared HPC environment users do not have the elevated privileges needed to run Docker containers or virtual machines, as CWL would favor. To address this, we have started to create pipeline component alternatives as loadable Environment Modules using helmod. In addition, we are extending toil, a CWL-supporting workflow engine that interfaces with job schedulers, to integrate with Slurm, which will allow us to better scale up CWL workflows to the available HPC resources.