CWL + Research Object == Complete Provenance

Farah Zaib Khan\textsuperscript{1}, Stian-Soiland-Reyes\textsuperscript{2}, Andrew Lonie\textsuperscript{1}, Richard Sinnott\textsuperscript{1}

\textsuperscript{1}The University of Melbourne, Australia. Email: farah.khan@unimelb.edu.au
\textsuperscript{2}The University of Manchester, UK.

License: Apache Licence, Version 2.0, see http://www.apache.org/licenses/LICENSE-2.0

Abstract

The term Provenance is referred to as ‘The beginning of something’s existence; something’s origin’ or ‘A record of ownership of a work of art or an antique, used as a guide to authenticity or quality’. Provenance tracking is crucial in scientific studies where workflows have emerged as an exemplar approach to mechanize data-intensive analyses. Gil et al. analyze challenges of scientific workflows and concluded that formally specified workflow helps ‘accelerate the rate of scientific process’ and facilitates others to reproduce the given experiment provided that provenance of end-to-end process at every level is captured.

We have implemented exemplar \textit{GATK variant calling workflow} using three approaches to workflow definition namely \textit{Galaxy}, \textit{CWL} and \textit{Cpipe} to identify assumptions implicit in these approaches.

These \textit{assumptions} lead to limited or no understanding of reproducibility requirements due to lack of documentation and comprehensive provenance tracking and resulted in identification of provenance information crucial for genomic workflows as shown in Figure 1.

CWL provides a declarative approach to workflow declaration making minimal assumptions about precise software environment, base software dependencies, configuration settings, alteration of parameters and software versions. It aims to provide an open source extensible standard to build flexible and customized workflows including intricate details of every process. It facilitates capture of information by supporting declaration of requirements, ‘cwl:tool’ and checksums etc. Currently, there is no mechanism to gather the produced information as a result of a workflow run into one bundle for future use. We propose to demonstrate the implementation of a module for CWL as shown in Figure 2.