Bioinformatics, 36(22-23), 2020, 5556–5557
doi: 10.1093/bioinformatics/btaa1033
Advance Access Publication Date: 16 December 2020
Applications Note

Data and text mining

aCLImatise: automated generation of tool definitions for bioinformatics workflows

Michael Milton [1,2,*] **and Natalie Thorne**[1,2,3,4]

[1]Melbourne Genomics Health Alliance, Parkville, VIC 3052, Australia, [2]Walter and Eliza Hall Institute of Medical Research, Parkville, VIC 3052, Australia, [3]Murdoch Children's Research Institute, Royal Children's Hospital, Parkville, VIC 3052, Australia and [4]Department of Medical Biology, The University of Melbourne, Parkville, VIC 3010, Australia

*To whom correspondence should be addressed.

Associate Editor: Jonathan Wren

Received on October 2, 2020; revised on November 25, 2020; editorial decision on November 30, 2020; accepted on December 3, 2020

Abstract

Summary: aCLImatise is a utility for automatically generating tool definitions compatible with bioinformatics workflow languages, by parsing command-line help output. aCLImatise also has an associated database called the aCLImatise Base Camp, which provides thousands of pre-computed tool definitions.

Availability and implementation: The latest aCLImatise source code is available within a GitHub organisation, under the GPL-3.0 license: https://github.com/aCLImatise. In particular, documentation for the aCLImatise Python package is available at https://aclimatise.github.io/CliHelpParser/, and the aCLImatise Base Camp is available at https://aclimatise.github.io/BaseCamp/.

Contact: michael.milton@melbournegenomics.org.au

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Bioinformatics workflow languages are domain-specific languages which aim to simplify the process of writing workflows for bioinformatics analysis (Larsonneur *et al.*, 2018). Four popular workflow languages in bioinformatics are Nextflow, Snakemake, Workflow Definition Language (WDL) and Common Workflow Language (CWL) (Bedő, 2019). In each of these languages the author defines 'tool definitions' (variously referred to as 'command line tool descriptions', 'task definitions', 'process definitions' or 'wrappers'), which are then composed together using a separate 'workflow' definition (Chapman *et al.*, 2016; Di Tommaso *et al.*, 2017; Köster and Rahmann, 2012).

Tool definitions describe the interface to a piece of software, generally a command-line interface, including all of its inputs, outputs and execution requirements. While workflow definitions must be customized according to the use-case, tool definitions simply describe a piece of software, and are therefore not coupled to a single workflow or context (Chapman *et al.*, 2016). For this reason, it is common to collect tool definitions in online tool repositories that can be used by workflow designers, reducing the work involved in constructing a workflow. Such repositories exist for WDL (https://github.com/biowdl/tasks), Snakemake (https://snakemake-wrappers.readthedocs.io/en/stable/), Nextflow (https://github.com/nf-core/modules) and CWL (https://github.com/common-workflow-library/bio-cwl-tools), while registries such as Dockstore cater to multiple workflow languages simultaneously (O'Connor *et al.*, 2017). Despite these initiatives, most tool repositories are incomplete or out-of-date. Maintaining up-to-date tool definitions would require frequent updates to describe new software and accommodate updates to existing software, which is not feasible to perform manually. However, some automated techniques have been developed for generating these tool definitions, most notably argparse2tool (https://github.com/hexylena/argparse2tool). This approach has shown further promise when enhanced with metadata from the bio.tools registry, but as argparse2tool is only compatible with software written in the Python language, it does not provide a general solution to this problem (Hillion *et al.*, 2017).

Fortunately, all command-line software provides documentation in the form of the help output. This is the output that is generally printed by an application when invoked using the – help flag, as encouraged by Stallman (2015). Furthermore, this help is generally kept up-to-date, as it is the first point of reference for most users of the software, and in many cases is generated automatically by the argument parsing library (e.g. the argparse library for Python; https://docs.python.org/3/library/argparse.html). In addition, help output often follows a semi-formalised series of conventions, most notably the POSIX Utility Convention (IEEE, 2018) and more rigorously the docopt language (http://docopt.org/), making it a viable target for automated parsing.

aCLImatise is a new contribution to the bioinformatics workflow ecosystem designed to streamline the creation of new portable workflows by providing automatically generated tool definitions for any tool with a conventional command-line interface. aCLImatise is itself a command-line application written in the Python programming language. To produce a tool definition, aCLImatise first executes the command of interest by trying a variety of help flags and storing the standard output from each. The resulting help text is

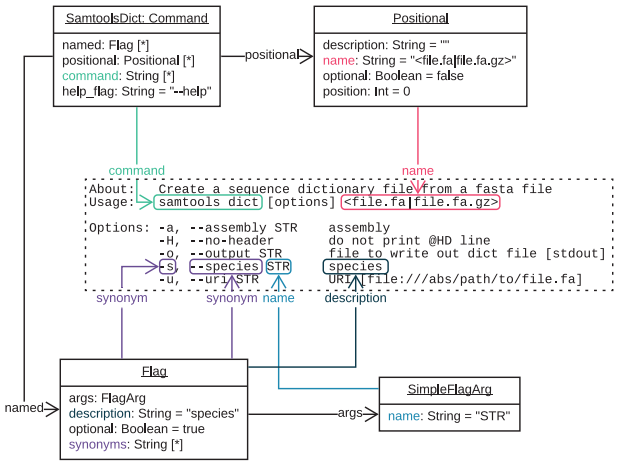

Fig. 1. The help text produced by the command-line tool SAMtools dict (surrounded by a dotted line), annotated with a subset of the aCLImatise object model (the four surrounding boxes), illustrating how the source text is mapped into Python classes via the parser (coloured arrows). SAMtools dict is a subcommand of the popular SAMtools suite of bioinformatics utilities (Li *et al.*, 2009). The object model is an adapted version of a Unified Modelling Language (UML) object diagram (Rumbaugh *et al.*, 2004), where each box represents the instance of an internal class, and each arrow represents an association between objects that is navigable in the direction of the arrow. The coloured arrows indicate an association with a String or array of Strings originally sourced from the help text. The Command class represents an entire command-line tool or subcommand, which has many inputs: Positionals (aka arguments) and Flags (aka options), each of which has an argument specification, such as SimpleFlagArg

then parsed with a Parsing Expression Grammar (PEG) defined using the powerful PyParsing library (McGuire, 2007). This parsing process, and the internal data format are briefly illustrated in Figure 1. Finally, the best intermediate data model is output as a YAML data structure, or translated into workflow formats such as CWL or WDL. Workflow designers are able to download the Python package from the PyPI, and then run aCLImatise on the software they intend to use in their workflow. To evaluate this approach, a detailed comparison between an automated tool definition generated by aCLImatise, and manually authored tool definition is available in Supplementary Appendix SA.

To simplify the writing of workflows even further, a large database of approximately 20 000 tool definitions called the aCLImatise Base Camp has been generated by running aCLImatise on the Bioconda database of bioinformatics software (Grüning *et al.*, 2018), facilitated by BioContainers Docker images (da Veiga Leprevost *et al.*, 2017). This database will be periodically and automatically regenerated from the latest version of Bioconda, limiting the need for manual curation and reducing the risk of outdated tool definitions being used in workflows. It is intended that workflow authors first refer to the Base Camp for up-to-date tool definitions, and only resort to installing aCLImatise when using a tool that is not available in Bioconda.

There are numerous potential directions for aCLImatise in the future. Firstly, we hope to expand the parser to support more unusual help formats that depart further from help text conventions. Secondly, there is the potential to add manual curation to the Base Camp database, allowing authors to refine the generated tool definitions and provide simple test suites. Finally, there has already been some effort made to expand the number of supported workflow languages from the initial two. We envisage that Galaxy (Afgan *et al.*, 2018), Nextflow and Snakemake could be supported in the future.

Funding

This work was supported by the State Government of Victoria and the 10 member organisations of the Melbourne Genomics Health Alliance.

Conflict of Interest: none declared.

Acknowledgements

The authors thank Sarah Payton and Edmund Lau for editorial assistance, and the Melbourne Genomics Bioinformatics Working Group for guidance and review.

Data Availability

The tool definitions produced by aCLImatise are available in Zenodo, at https://doi.org/10.5281/zenodo.4312329.

References

Afgan,E. *et al.* (2018) The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Res.*, **46**, W537–W544.

Bedő,J. (2019) BioShake: a Haskell EDSL for bioinformatics workflows. *PeerJ*, **7**, e7223.

Chapman,B. *et al.* (2016) Common Workflow Language, v1.0. *figshare*. United States.

da Veiga Leprevost,F. *et al.* (2017) BioContainers: an open-source and community-driven framework for software standardization. *Bioinformatics*, **33**, 2580–2582.

Di Tommaso,P. *et al.* (2017) Nextflow enables reproducible computational workflows. *Nat. Biotechnol.*, **35**, 316–319.

Grüning,B., The Bioconda Team. *et al.* (2018) Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nat. Methods*, **15**, 475–476.

Hillion,K.-H. *et al.* (2017) Using bio.tools to generate and annotate workbench tool descriptions. *F1000Research*, **6**, 2074.

IEEE. (2018) IEEE Standard for Information Technology–Portable Operating System Interface (POSIX(TM)) Base Specifications, Issue 7. *IEEE Std 1003.1-2017*, 1–3951.

Köster,J. and Rahmann,S. (2012) Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, **28**, 2520–2522.

Larsonneur,E. *et al.* (2018) Evaluating Workflow Management Systems: A Bioinformatics Use Case. *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2773–2775.

Li,H. *et al.*; 1000 Genome Project Data Processing Subgroup. (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics (Oxford, England)*, **25**, 2078–2079.

McGuire,P. (2007) *Getting Started with Pyparsing*. United States: O'Reilly Media, Inc.

O'Connor,B.D. *et al.* (2017) The Dockstore: enabling modular, community-focused sharing of Docker-based genomics tools and workflows. *F1000Research*, **6**, 52.

Rumbaugh,J. *et al.* (2004) *Unified Modeling Language Reference Manual*, 2nd edn. United Kingdom: Pearson Higher Education.

Stallman,R. (2015) *GNU Coding Standards*. Hong Kong: Samurai Media Limited.