Bioinformatics, 34(11), 2018, 1934–1936
doi: 10.1093/bioinformatics/bty034
Advance Access Publication Date: 19 January 2018
Applications Note

Genome analysis

Sequanix: a dynamic graphical interface for Snakemake workflows

Dimitri Desvillechabrol[1,*], Rachel Legendre[1,2], Claire Rioualen[3], Christiane Bouchier[1], Jacques van Helden[3], Sean Kennedy[1] and Thomas Cokelaer[1,2,*]

[1]Institut Pasteur—Biomics Pole—CITECH, F-75015, Paris, France, [2]Institut Pasteur—Bioinformatics and Biostatistics Hub—C3BI, USR 3756 IP CNRS, F-75015, Paris, France and [3]Aix Marseille Univ, INSERM, TAGC, UMR_S 1090, Marseille 13288, France

*To whom correspondence should be addressed.
Associate Editor: John Hancock

Received on July 20, 2017; revised on January 15, 2018; editorial decision on January 17, 2018; accepted on January 18, 2018

Abstract

Summary: We designed a PyQt graphical user interface—Sequanix—aimed at democratizing the use of Snakemake pipelines in the NGS space and beyond. By default, Sequanix includes Sequana NGS pipelines (Snakemake format) (http://sequana.readthedocs.io), and is also capable of loading any external Snakemake pipeline. New users can easily, visually, edit configuration files of expert-validated pipelines and can interactively execute these production-ready workflows. Sequanix will be useful to both Snakemake developers in exposing their pipelines and to a wide audience of users.
Availability and implementation: Source on http://github.com/sequana/sequana, bio-containers on http://bioconda.github.io and Singularity hub (http://singularity-hub.org).
Contact: dimitri.desvillechabrol@pasteur.fr or thomas.cokelaer@pasteur.fr
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Context and motivation

Bioinformatics software dealing with biological data (large volumes and variety of data structures) are routinely put together to design sophisticated workflows (or pipelines). Fortuitously, many workflows can be decomposed into *embarrassingly parallel* problems: a task can be applied in parallel on similar datasets (e.g. several DNA samples). Yet, the long-term utility of these pipelines is often tenuous, hampered of interwoven scripts and software libraries, numerous files or complex dependencies between tasks.

Scripting languages may be sufficient to design workflows. However, they usually lack the ability to handle dependencies between rules, re-entrancies (starting from an intermediate step rather than from scratch), or distributed computing features to handle time-intensive and multi-parametric tasks. To overcome these problems, various workflow managers have emerged as popular tools in the field of bioinformatics (Leipzig, 2017). They usually possess all relevant features to design workflows effectively. Consequently, developers have the luxury of choosing a framework amongst many based on personal choice. Influencing factors may include the programming language or the presence of a graphical user interface (GUI). Workflow managers like Galaxy (Goecks, 2010) provides GUI drag and drop capability allowing non-specialists to implement de novo pipelines. On the other end of the spectrum, command line interfaces (CLI, hereafter) are still widely employed. Indeed, most developers would prefer a light-weight framework that could be more flexible or accelerate the development of new pipelines.

Amongst the recent CLI-based workflow managers, Snakemake (Köster and Rahmann, 2012) has been adopted by a large community of developers. This is especially pronounced in the field of NGS (see http://snakemake.readthedocs.io), where there is an increasing demand for production-ready pipelines to handle massive amounts of data from different technologies. Although Snakemake provides a GUI interface, it is server-oriented, which is a limitation on some distributed clusters. Moreover, the GUI is essentially a wrapper of the command line itself. Besides, scientists willing to change the behaviour of the pipelines needs to edit the configuration file or

masterize the numerous Snakemake arguments. In order to expose our pipelines to a wider audience, we design a graphical interface—Sequanix—to offer the ability to edit the configuration file interactively, associate dedicated widgets to parameter types, or add tooltips dynamically. Although Sequanix was designed to expose Sequana pipelines (Cokelaer, 2017), it can also load any Snakemake pipelines as demonstrated in the Supplementary Material with third-party pipelines available in the SnakeChunks library (Rioualen, 2017).

2 The graphical interface: Sequanix

Snakemake is a text-based workflow that uses Python and a definition language to define rules and workflow properties. A Snakemake pipeline is defined within a file called *Snakefile* (examples in Supplementary Material). Although not strictly required, pipeline parameters may be externalized within a *configuration* file in YAML or JSON formats.

Sequana project provides a set of Snakemake pipelines dedicated to NGS analysis (e.g. RNA-seq, variant calling, ...). Hereafter, we distinguish the Snakemake pipelines provided in Sequana from other Snakemake pipelines. We refer to the former as *Sequana pipelines* and to the latter as *Generic pipelines*. The first difference being that every Sequana pipeline is made of a Snakefile *and* a configuration file whereas Generic pipelines may not have a configuration file. The second difference is that the Sequana configuration files are in YAML format only. A third difference is that the configuration file in Sequana must define specific fields which refer to the location or type of inputs files (e.g. FastQ files).

Sequanix interface is designed in PyQt, which is a Python binding of the cross-platform GUI toolkit Qt (https://www.qt.io/). The user interface consists of a *main* dialog (see Fig. 1), a *Snakemake* dialog and a *Preferences* dialog (See Supplementary Material). The Snakemake dialog is used to configure the Snakemake framework behavior (e.g. number of CPUs to be used) while selection and execution of a pipeline is performed in the main dialog as explained below.

At the top of the main dialog (Fig. 1A) one can switch between the *Sequana* or *Generic* mode. In Sequana mode, all released pipelines are shown in *pipeline selection* tab. Once a pipeline is selected, its configuration file is dynamically loaded (Fig. 1C) and users can modify parameters interactively. Then, users can select the relevant input data (Fig. 1A). Finally, a working directory must be set where the project (Snakemake pipeline and configuration file edited by the user) is saved. The *Generic* mode works in a similar fashion except that the *input data* tab (specific to Sequana) is now replaced by the *config file* tab. Here, users may provide a configuration file (if needed). The only restriction is that the Snakemake pipeline must be executable in the Sequanix's environment. In other words, third-party libraries and applications required by the pipeline must be configured properly before starting Sequanix.

We designed Sequanix to build upon the Snakemake framework and add important functionalities. We felt that users should not manually edit configuration files. Therefore, configuration files are loaded in a dedicated widget and their sections and parameters are interpreted and shown in the interface (Fig. 1C). Some parameters are associated with specific widgets. For instance a parameter name ending in *_file* or *_browser* becomes a file or directory browser instead of a simple editable line. We also propose to annotate configuration files (YAML format) with comments written as Python docstrings (see Supplementary Material). Such comments are then

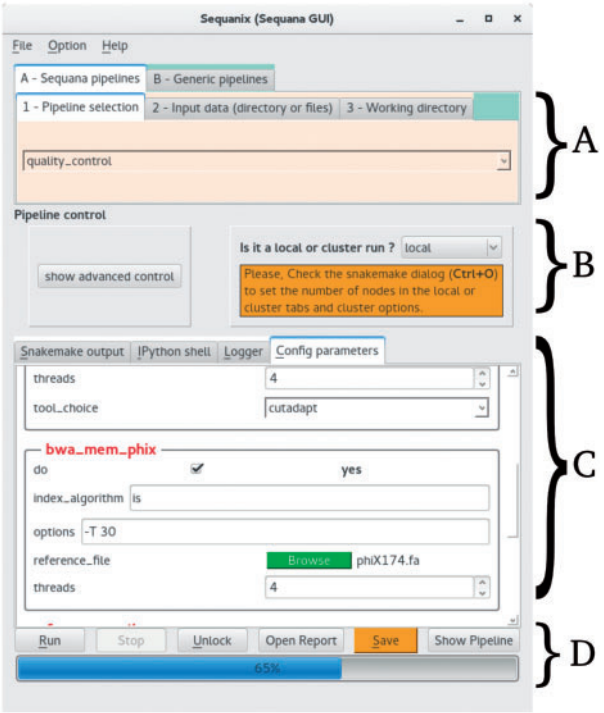

Fig. 1. Main Sequanix dialog. One can switch between Sequana and Generic modes (top panel). Then, one can select a Snakemake pipeline, or specify the working directory where analysis is performed and results stored (**A**). The analysis can be run locally or on a distributed computer (**B**). In the latter case, cluster options may be provided in the *Options* menu. Configuration file is editable (**C**). Finally, one can save, run or stop the pipeline execution (**D**)

interpreted and appear as tooltips in the GUI. We also implemented the ability for users to import a YAML schema file, which is used to further validate the configuration file (e.g. check that threading parameter are greater than zero).

The Snakemake framework scales without modification, from single and multi-core workstations to cluster engines. This ability is reflected in the *Sequanix* interface (Fig. 1B) and in the Snakemake dialog where one can switch between local and cluster mode, set the number of CPUs, or provide specific job scheduler arguments (e.g. memory requirements).

Once a pipeline (Sequana or Generic) and a working directory are set, the project can be saved (*Save* button) and the pipeline flow (a directed acyclic graph) visualized (*Pipeline* button). Finally, the pipeline can be executed (*Run* Button). Stopping the process (*Stop* button) behaves as a normal Snakemake interruption allowing re-entry. If an error occurs (e.g. missing file), one can quickly fix it and re-enter the execution.

3 Conclusion

Sequanix provides a GUI for Snakemake workflows. Its simple interface makes it easy for non-specialists to implement existing production-ready pipelines that have been created by Snakemake developers. Users can load a pipeline, edit the configuration file via various widgets or drop-down menus (reducing typographical errors), execute the pipeline locally or on a cluster, and track the progress of the analysis. Although Sequanix was primarily developed to expose Sequana pipelines, it should also benefit to the community of Snakemake developers willing to provide a graphical interface to their users. Currently, Sequanix is included in Sequana, that is available on Bioconda (Grüning, 2017) under the package named

sequana as well as on the singularity hub (Kurtzer, 2017) (see http://sequana.readthedocs.io for details). Finally, note that a Sequanix version, independent of Sequana, will be provided in the future.

Funding

This work has been supported by France Génomique consortium (ANR10-INBS-09-08 and ANR-10-INBS-09-10) and NIH grant GM0110597 & FOINS-CONACYT—Fronteras de la Ciencia 2015—ID 15.

Conflict of Interest: none declared.

References

Cokelaer,T. *et al*. (2017) Sequana: a set of snakemake NGS pipelines. *J. Open Source Softw*., **2**, 352.

Goecks,J. *et al*. (2010) Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol*., **11**, R86.

Grüning,B. *et al*. (2017). Bioconda: a sustainable and comprehensive software distribution for the life sciences. bioRxiv doi: 10.1101/207092

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

Kurtzer,G.M. *et al*. (2017) Singularity: scientific containers for mobility of compute. *PLoS ONE*, **12**, e0177459.

Leipzig,J. (2017) A review of bioinformatic pipeline frameworks. *Brief. Bioinf*., **18**, 530–536.

Rioualen,C. *et al*. (2017) SnakeChunks: modular blocks to build Snakemake workflows for reproducible NGS analyses. bioRxiv doi: 10.1101/165191