Kronos: a workflow assembler for genome analytics and informatics

M Jafar Taghiyar[1,2], Jamie Rosner[1], Diljot Grewal[1,2], Bruno Grande[3], Radhouane Aniba[1,2], Jasleen Grewal[3], Paul C Boutros[4,5], Ryan D Morin[3], Ali Bashashati[1,2], and Sohrab P Shah[1,2]

[1]Department of Molecular Oncology, British Columbia Cancer Agency. Email: jtaghiyar@bccrc.ca
[2]Department of Pathology and Laboratory Medicine, University of British Columbia.
[3]Department of Molecular Biology and Biochemistry, Simon Fraser University.
[4]Ontario Institute for Cancer Research.
[5]Department of Medical Biophysics, University of Toronto.

Project URL: https://github.com/jtaghiyar/kronos
https://github.com/MO-BCCRC

Background
The field of next generation sequencing informatics has matured to a point where algorithmic advances in sequence alignment and individual feature detection methods have stabilized. However, practical and robust implementation of complex analytical workflows still requires significant programming investment and expertise. For example, analysis of cancer genomes may invoke sequence alignment followed by mutation, copy number, and rearrangement detection, followed by annotation of predicted functional effects of these alterations. In both research and clinical settings it is vital to encode specific analytical protocols for transparency of derivative results and clinical audits.

Results
We present Kronos, a software platform for automating the development and execution of reproducible, auditable and distributable bioinformatics workflows. Kronos obviates the need for explicit coding of workflows by transforming a text configuration file into an executable Python script. Given a set of existing components (analysis modules), a new workflow can be created in three simple steps:

Step 1. Create a configuration file template by running *kronos make_config* command.
Step 2. In the template, specify the order by which the components in the workflow should be run.
Step 3. Create the workflow by running *kronos init* command with the template as input.

The result is an executable Python script that uses the built-in run manager of Kronos. The run manager provides scalability by enabling users to run the workflow on a single machine or a cluster of computing nodes. Moreover, each component in the workflow can individually be run either locally, or on a cluster. Workflows created by Kronos are highly modular and configurable through configuration file editing. The workflows are fully encoded for ease of distribution and can be instantiated on external systems, promoting and facilitating reproducible research and comparative analyses. Kronos also has a framework for making components. Each component is an analysis module wrapped according to the framework. A component template can be made using *kronos make_component* command. Developers can use the framework to implement custom tools, reuse existing tools, and contribute to the community at large. Kronos is shipped with Docker and Amazon AWS machine images. It is free, open source and available through Python Package Index.

Conclusion
This work provides a framework towards rapid integration of new (and optimal) genomic analysis advances in high-throughput studies. The flexibility, customization, and modularity of the resulting pipelines make it an attractive system to use in any high-throughput genomics analysis endeavour. We expect Kronos will be a small but foundational step towards standardization and distribution of NGS workflows in both clinical and research applications.