NeatSeq-Flow: A Lightweight Software for Efficient Execution of High Throughput Sequencing Workflows

Menachem Sklarz[1,*], Michal Gordon[1] and Vered Chalifa-Caspi[1,*]

[1]Bioinformatics Core Facility, National Institute for Biotechnology in the Negev, Ben-Gurion University of the Negev, 84105, Beer-Sheva, Israel,

*To whom correspondence should be addressed.

Abstract

Summary: Bioinformatics workflows (WFs) in general, and those involving High Throughput Sequencing data in particular, typically involve executing a sequence of programs on raw sequence files from as many as thousands of samples. Management of these WFs is laborious and error-prone. We have developed NeatSeq-Flow, a python package that manages WF creation for execution on computer clusters. NeatSeq-Flow creates shell scripts as well as a directory structure for storing analysis results, error messages, and execution logs. The user maintains full control over the execution of the WF, while the computer cluster enforces sequential execution and parallelization. NeatSeq-Flow also supplies tools for version tracking, documentation and execution logging.

Availability: https://github.com/bioinfo-core-BGU/neatseq-flow

Contact: sklarz@bgu.ac.il

Introduction

Modern biological experiments involving High Throughput Sequencing (HTS) produce large amounts of data, which scientists must analyze in order to reach the kernel of information of interest. Usually, analysis of the data is composed of several steps, each of which consists of calling a program with inputs,

receiving the outputs and passing them on to the next step. Often, the analysis is readily parallelized and is executed on different processing units (CPUs) or cluster nodes, thus saving execution time. The bioinformatician will typically write short shell scripts that execute the different steps and send them to a computer cluster job scheduler for execution on distributed nodes.

Creating and executing these script-based workflows (WFs) is time consuming and error prone, especially when considering projects with hundreds or thousands of samples, or when the same analyses has to be repeated with different combinations of parameters. Additionally, the user has to ensure the WF is executed sequentially, sending latter steps for execution only after completion of former steps.

To address these issues, many commendable efforts have been made to create platforms for automating execution of such WFs, *e.g.* (Hatakeyama *et al.*, 2016; Köster and Rahmann, 2012; Linke *et al.*, 2011; Sadedin *et al.*, 2012; Stocker *et al.*, 2004). Recently, a review of these efforts has been published (Leipzig, 2016).

As has been pointed out previously (Hatakeyama *et al.*, 2016), most of the available WF platforms do not address the methodology described above, *i.e.* generation of shell scripts designed for execution by a cluster job scheduler. The user will also find the existing WF management tools difficult to use without knowledge of special programming languages, such as make, or learning a new language entirely designed for creating WFs, such as Bpipe.

Using the manual experience we have gained in our group over the years, we have developed NeatSeq-Flow, a lightweight python software package that manages WF creation for execution on computer clusters. NeatSeq-Flow has proved indispensable in routine analysis of various types of HTS data.

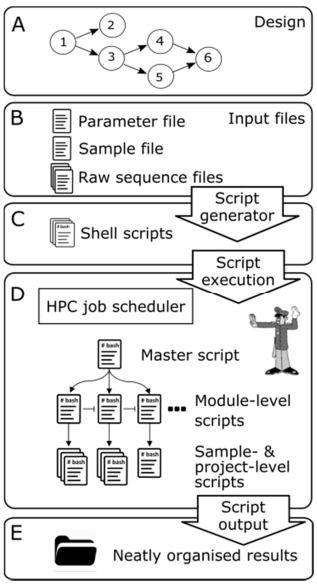

Figure 1. Outline of workflow (WF) execution with NeatSeq-Flow. A conceptual design of the WF (A) takes on the form of a directed acyclic graph where vertices represent steps, edges represent interdependencies between steps and convergence (*e.g.* step 5 in A) represents a step which is dependent on several previous steps. Based on the WF design, the user creates sample- and parameter-defintion files (B). NeatSeq-Flow is executed, creating a set of shell scripts (C). These are executed on a computer cluster using the cluster job manager, which manages step dependencies (D). Script outputs and WF log files are neatly organized in a directory structure (E).

Main advantages

NeatSeq-Flow separates workflow creation into distinct elements, which match the practicing bioinformatician's frame of mind. Each workflow consists of a set of samples and their files; and a set of modular operations to be performed on the samples and the order of operation. NeatSeq-Flow generates a set of concise and self-explanatory shell scripts which can be executed in parallel on a cluster at various levels: the whole WF at once, step-by-step or sample-by-sample. The job scheduler enforces dependencies and manages parallel execution.

NeatSeq-Flow is written in pure python, and is therefore platform independent. New steps can be programmed by defining a set of functions based on a template.

Implementation

Basic usage

The user first formulates a conceptual design of the WF (Fig. 1A), which can take the form of a directed acyclic graph, as described below. Based on the design, the user creates a parameter file (Fig. S1), defining the planned WF and the parameters to be applied at each step as well as a sample-file (Fig. S2) defining the samples to be analyzed (Fig. 1B).

NeatSeq-Flow is then executed to create a directory structure containing different elements of the WF: The shell scripts and additional scripts that control submission to the job scheduler and contain dependency information (Fig. 1C); a directory for WF results (*i.e.* script outputs); and additional directories that store information regarding the WF construction and execution. The user may then decide whether to execute the entire WF automatically, in a step-by-step manner, or even sample-by-sample (Fig. 1D). In all instances, irrespective of the execution manner, the job scheduler enforces step dependencies, *i.e.* it executes steps only after previous steps, on which they depend, have finished running. Thus, WFs are executed stepwise, while enabling parallelization of sample-level analyses as well as steps on independent branches of the WF.

The script outputs are neatly organized in the results directory by step and sample, making it easy to locate required information (Fig. 1E). Additionally, execution start and end time as well as maximum memory requirements are written to a log file.

In NeatSeq-Flow, steps are coded by modules. Each module creates scripts for a particular step. Often, the user will want to try out different parameters for specific steps. Rather than create a new WF for each set of parameters, NeatSeq-Flow permits defining different instances of the same module. For each

module instance, the user defines a base instance, thus determining the instance`s input files and creating a dependency of the instance on its base. Consequently, the analysis takes on the form of a directed acyclic graph representing different approaches to the analysis (Fig. 1A and Fig. S3).

Example sample- and parameter-files for a basic WF are provided in supplementary figures S2 and S1. This example performs quality testing and trimming on a set of fastq sequence files, aligns the sequences to a reference genome and creates bigwig files for display in the UCSC genome browser.

Modules

The modules currently included in the package are listed in supplementary table S4. Users may add modules for open source, commercial or custom programs not included in the basic package using basic python code (see template in Fig. S5), as described in the online documentation. It is our hope that the community of users will contribute additional modules to the public.

Conclusion and Future developments

NeatSeq-Flow allows the bioinformatician to pursue his routine HTS analysis work methodology on computer clusters, while avoiding the tedious task of composing error free shell scripts. Execution of the actual WF is controlled by the cluster job scheduler, while the user has control over which steps and which samples to execute. A WF in NeatSeq-Flow is defined by sample and parameter files, which ensure clear documentation and reproducibility. NeatSeq-Flow is written in plain python, such that adding modules to the software is a straightforward process. Accordingly, NeatSeq-Flow may easily be extended to include new protocols and software packages. NeatSeq-Flow is in constant use by our group for diverse analyses, and has proven to be priceless in time saving and error reduction. NeatSeq-Flow is under continuous agile development and improvement. NeatSeq-Flow can be generalized to work on many types of biological data other than HTS data.

Acknowledgements

We would like to thank Drs. Esti Yeger-Lotem and Barak Marcus for critically reading the manuscript. This research used the High Performance Computing Facility at Ben-Gurion University

References

Hatakeyama,M., Opitz,L., *et al.* (2016) SUSHI: an exquisite recipe for fully documented, reproducible and reusable NGS data analysis. *BMC Bioinformatics*, 17, 228.

Kent,W.J., Sugnet,C.W., *et al.* (2002) The Human Genome Browser at UCSC. *Genome Res.*, 12, 996-1006.

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

Leipzig,J. (2016) A review of bioinformatic pipeline frameworks. *Briefings in Bioinformatics*, bbw020.

Linke,B., Giegerich,R., *et al.* (2011) Conveyor: a workflow engine for bioinformatic analyses. *Bioinformatics*, 27, 903-911.

Sadedin,S.P., Pope,B., *et al.* (2012) Bpipe: a tool for running and managing bioinformatics pipelines. *Bioinformatics*, 28, 1525-1526.

Stocker,G., Rieder,D., *et al.* (2004) ClusterControl: a web interface for distributing and monitoring bioinformatics applications on a Linux cluster. *Bioinformatics*, 20, 805-807.