TECHNOLOGY REPORT
published: 28 June 2019
doi: 10.3389/fgene.2019.00614

Pipeliner: A Nextflow-Based Framework for the Definition of Sequencing Data Processing Pipelines

Anthony Federico[1,2]*, *Tanya Karagiannis*[1], *Kritika Karri*[1], *Dileep Kishore*[1], *Yusuke Koga*[2], *Joshua D. Campbell*[1,2] and *Stefano Monti*[1,2]*

[1] *Bioinformatics Program, Boston University, Boston, MA, United States,* [2] *Division of Computational Biomedicine, Boston University School of Medicine, Boston, MA, United States*

OPEN ACCESS

Edited by:
Vinicius Maracaja-Coutinho, Universidad de Chile, Chile

Reviewed by:
Pao-Yang Chen, Academia Sinica, Taiwan
Ernesto Picardi, University of Bari Aldo Moro, Italy

Correspondence:
Anthony Federico
anfed@bu.edu
Stefano Monti
smonti@BU.EDU

Specialty section:
This article was submitted to Bioinformatics and Computational Biology, a section of the journal Frontiers in Genetics

Received: *21 November 2018*
Accepted: *13 June 2019*
Published: *28 June 2019*

Citation:
Federico A, Karagiannis T, Karri K, Kishore D, Koga Y,Campbell JD and Monti S (2019) Pipeliner: A Nextflow-Based Framework for the Definition of Sequencing Data Processing Pipelines. Front. Genet. 10:614. doi: 10.3389/fgene.2019.00614

The advent of high-throughput sequencing technologies has led to the need for flexible and user-friendly data preprocessing platforms. The Pipeliner framework provides an out-of-the-box solution for processing various types of sequencing data. It combines the Nextflow scripting language and Anaconda package manager to generate modular computational workflows. We have used Pipeliner to create several pipelines for sequencing data processing including bulk RNA-sequencing (RNA-seq), single-cell RNA-seq, as well as digital gene expression data. This report highlights the design methodology behind Pipeliner that enables the development of highly flexible and reproducible pipelines that are easy to extend and maintain on multiple computing environments. We also provide a quick start user guide demonstrating how to setup and execute available pipelines with toy datasets.

Keywords: pipeline development, sequencing workflows, Nextflow, RNA-seq pipeline, scRNA-seq pipeline

INTRODUCTION

High-throughput sequencing (HTS) technologies are vital to the study of genomics and related fields. Breakthroughs in cost efficiency have made it common for studies to obtain millions of raw sequencing reads. However, processing these data requires a series of computationally intensive tools that can be unintuitive to use, difficult to combine into stable workflows that can handle large number of samples, and challenging to maintain over long periods of time in different environments. The effort to simplify this process has resulted in the development of sequencing pipelines such as RseqFlow (Wang et al., 2011), PRADA (Torres-García et al., 2014), and Galaxy (Goecks et al., 2010), among others. Some of these pipelines are open-source and either available for download or on publicly available servers. However, some drawbacks include difficulty when deploying on existing computational resources, limited selection of computational tools, and unintuitive or limited ability to make modifications. While other frameworks may be more flexible, they often require the user to install each needed tool separately, which may be challenging and reduce reproducibility.

Pipeliner is a framework for the definition of sequencing data processing pipelines that aims to solve these issues. Pipelines developed within the framework are platform independent and fully reproducible and inherit automated job parallelization and failure recovery. Their flexibility and modular architecture allows users to easily customize and modify processes based on their needs. Pipeliner also provides additional resources that allow developers to rapidly build and test their own pipelines in an efficient and scalable manner. Pipeliner is a complete and user-friendly solution to meet the demands of processing large amounts and various types of sequencing data.

MATERIALS AND METHODS

Design and Features

Pipeliner is a suite of tools and methods for defining sequencing pipelines. It uses Nextflow, a portable, scalable, and parallelizable domain-specific language, to define data workflows (Di Tommaso et al., 2017). Using Nextflow, each pipeline is modularized, consisting of a configuration file as well as a series of processes. These processes define the major steps in each pipeline and can be written in Linux-executable scripting languages such as Bash, Python, Ruby, etc. Nextflow processes are connected through channels—asynchronous first in, first out queues—which allow data to be passed between the different steps in each pipeline using a dataflow programming model. Using this architecture, pipelines developed within the Pipeliner framework inherit multiple features that contribute to their flexibility, reproducibility, and extensibility (**Figure 1**).

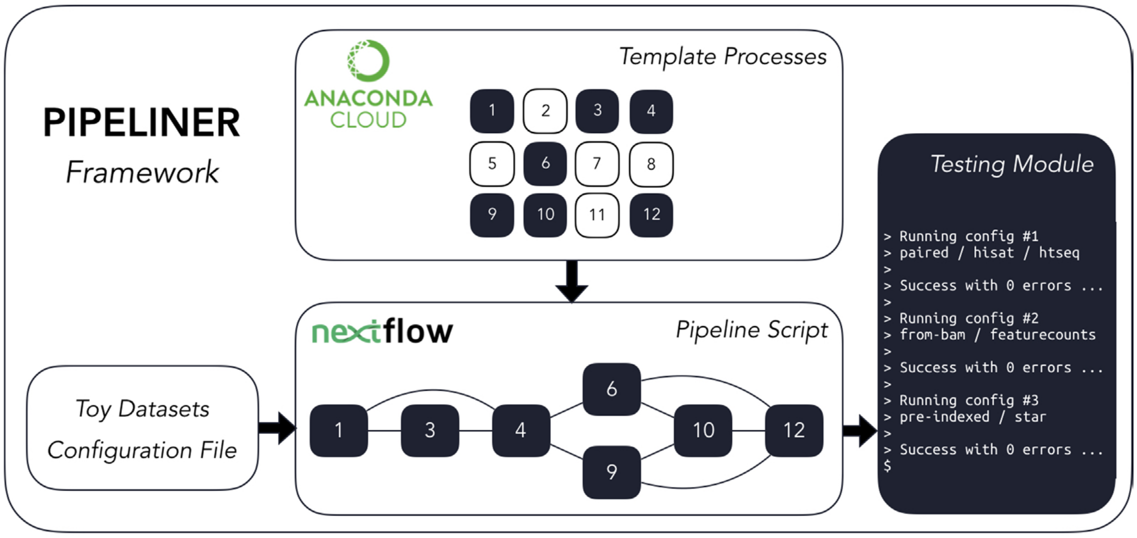
Pipeline Flexibility

Pipeliner enables flexible customization of pipeline options and parameters. Pipeliner currently offers three pipelines to demonstrate its applicability in processing different types of data, including bulk RNA-seq, single-cell RNA-seq (scRNA-seq), as well as digital gene expression (DGE) data (Soumillon et al., 2014). For the RNA-seq pipeline, sequencing reads are checked for quality with FastQC (Andrews, 2010), trimmed with TrimGalore (Krueger, 2016), mapped to a reference genome with either STAR (Dobin et al., 2012) or HISAT2 (Kim et al., 2015), and quantified with either StringTie (Pertea et al., 2015), HTSeq (Anders et al., 2015), or featureCounts (Liao et al., 2014). After alignment, mapping quality is checked with RSeQC (Wang et al., 2012), and a comprehensive summary report of all processes is generated with MultiQC (Ewels et al., 2016). The scRNA-seq and DGE pipelines adopt a similar methodology, and the development of additional pipelines for microRNA-seq (miRNA-seq) and RNA-seq Variant Calling is currently underway.

Parameter Configuration

All pipeline options and process parameters are set from a single configuration file (**Figure 2**). Users have the option to select and skip various steps as well as customize parameters and allocate computing resources for specific processes. This flexibility gives rise to many different use cases. For example, a user may opt to provide a pre-indexed reference genome or start the pipeline after the mapping step with saved alignment files or output an ExpressionSet data structure with count and phenotypic data. Thus, each pipeline is multipurpose and allows users to frequently tweak settings without adding complexity or sacrificing reproducibility.

The default configuration file defines variables for common parameters of third-party software tools used in each pipeline. These tools are wrapped into templates—one for



```
> Running config #1
> paired / hisat / htseq
>
> Success with 0 errors ...
>
> Running config #2
> from-bam / featurecounts
>
> Success with 0 errors ...
>
> Running config #3
> pre-indexed / star
>
> Success with 0 errors ...
$
```

FIGURE 1 | The Pipeliner framework employs reusable template processes strung together *via* Nextflow's scripting language to create workflows in addition to developer tools such as toy datasets and testing modules.

```
process {
  executor = 'sge'

  mapping.clusterOptions  = "-P project -pe omp 16 -l mem_total=94G"
  counting.clusterOptions = "-P project -pe omp 8  -l mem_total=16G"

  indir  = "/Users/pipeliner/pipelines/toy_data/rna-seq"
  outdir = "/Users/pipeliner/pipelines/rna-seq-results"
  fasta  = "${params.indir}/genome_reference.fa"
  gtf    = "${params.indir}/genome_annotation.gtf"

  // General pipeline parameters
  paired = true
  aligner = "hisat"
  quantifier = "featurecounts"

  index.use_existing = true
  index.path = "${params.indir}/alignment_indices/hisat_index/index/part"

  // Process-specific parameters
  feature_counts.cpus   = 8
  feature_counts.type   = "exon"
  feature_counts.id     = "gene_id"
  feature_counts.xargs  = ""
  feature_counts.ainj   = ""
}
```

FIGURE 2 | A shortened example of a configuration file, highlighting the key components. This configuration includes resource allocations for cluster executions, input and output paths to data, general pipeline parameters, as well as process-specific parameters.

```
featureCounts \\

# Common flags directly defined by the user
-T ${params.feature_counts.cpus} \\
-t ${params.feature_counts.type} \\
-g ${params.feature_counts.id} \\

# Flags handled by the pipeline
-a ${gtf} \\
-o "counts.raw.txt" \\

# Arguments indirectly defined by the user
${feature_counts_sargs} \\

# Extra arguments
${params.feature_counts.xargs} \\

# Input data
${bamfiles};

# After injection
${params.feature_counts.ainj}
```

FIGURE 3 | A code example of a template defined for the software tool featureCounts. The template wraps user-defined parameters, paths to data, as well as code injections into an executable bash script used in one of the pipeline steps.

each process—which are executed sequentially within the pipeline script. Because some software tools have hundreds of arguments, users have the option to insert code injections from the configuration file. These code injections can be used to pass uncommon keyword arguments or to append *ad hoc* processing steps (**Figure 3**). These features provide unrestricted control over each step in the execution of a pipeline. Furthermore, since all modifications are made within the configuration file—which is copied with each run—the pipeline script is left intact, preserving the reproducibility of each run regardless of any execution-specific changes the user may make.

Workflow Reproducibility

Pipeliner is designed to create reproducible workflows. An abstraction layer between Nextflow and Pipeliner logic enables platform independence and seamless compatibility with high-performance cloud computing executors such as Amazon Web Services. Pipeliner also uses Anaconda—a multi-platform package and environment manager—to manage all third-party software dependencies and handle pre-compilation of all required tools before a pipeline is executed (Continuum Analytics, 2016).

Pipeliner is bundled with a prepackaged environment hosted on Anaconda Cloud that contains all software packages necessary to run any of the three pipelines available. This virtual environment ensures consistent versioning of all software tools used during each pipeline execution. Additionally, all file paths, pipeline options, and process parameters are recorded, time stamped, and copied into a new configuration file with each run, ensuring pipelines are fully reproducible regardless of where and when they are executed.

Extensibility

Pipeliner makes the development of bioinformatics pipelines more efficient. The configuration file and processes that makeup each pipeline are inherited from shared blocks of code called template processes. For example, if a major update to an alignment tool requires modification to its template process, these changes propagate to all pipelines inheriting it (**Figure 4**). This property also minimizes the amount of code introduced as new pipelines are created, making them quicker to develop and easier to maintain. If a pipeline can inherit all of its processes with predefined templates, the user is only required to link these processes *via* Nextflow's scripting language and create a basic configuration file.

Rapid Development and Testing

Users can rapidly develop pipelines by using the toy datasets conveniently included with Pipeliner, enabling developers to test modifications made to their pipeline in minutes rather than hours. When testing, each execution covers only one configuration of parameters, meaning some processes may be skipped or partially executed depending on the configuration file. Therefore, to increase decision coverage, that is, the amount of tested reachable code, Pipeliner includes a custom testing module that automatically executes and logs a series of independent tests and configuration files (**Figure 5**). With these tools, users can efficiently build, test, and maintain multiple sequencing pipelines.

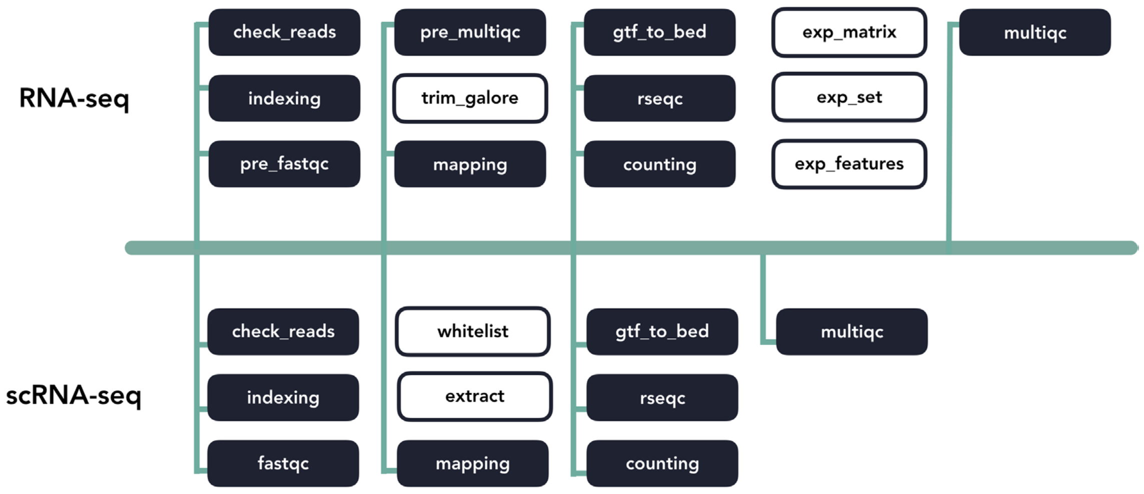


FIGURE 4 | A diagram of template code sharing between the RNA-seq and scRNA-seq pipelines. Each block represents an individual workflow step. Shaded blocks share template code, while unshaded blocks are unique.

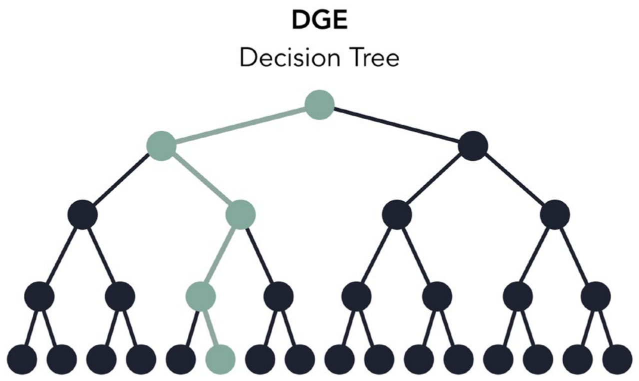


FIGURE 5 | A decision tree of the digital gene expression (DGE) pipeline based on available options in the configuration file. When testing, one execution of the pipeline will test one path in the decision tree (highlighted in green). The testing module automates the execution and logging of all possible paths.

Comparisons with Other Available Tools

Pipeliner has several characteristics that distinguish it from existing sequencing data workflows, such as RseqFlow (Wang et al., 2011), PRADA (Torres-García et al., 2014), or Galaxy (Goecks et al., 2010) (**Table 1**). These tools and their dependencies can be difficult to install and setup, lack sufficient documentation, and are rigid in their design, making customization challenging. Downloading and setting up Pipeliner is simple, and all dependencies are automatically installed through a virtual environment, ensuring data reproducibility and compatibility across various computing environments. Pipeliner is designed to be modular and flexible; therefore, workflow steps can be modified, skipped, removed, or extended. Pipeliner provides comprehensive documentation for general use as well as for developers who wish to extend the framework.

Usage Guide

In addition to comprehensive documentation of the framework and to demonstrate its ease of use, we provide a tutorial for processing the toy datasets available for each pipeline.

Processing Toy Datasets

The Pipeliner framework requires Nextflow and Anaconda. Nextflow requires Java 8 (or higher) to be installed and can be used on Linux and OS X machines. Third-party software tools will be installed and managed through an Anaconda virtual environment. Once the prerequisites are installed, the repository can be cloned from GitHub to any location through the following command:

```
$ git clone https://github.com/montilab/pipeliner
```

The next step is to clone and activate the virtual environment. The easiest method is to recreate the environment through the yml files provided in the repository. There is a single yml file for both Linux and OS X operating systems, containing all dependencies for all available pipelines.

```
$ conda env create -f pipeliner/envs/linux_env.yml
# Linux
$ conda env create -f pipeliner/envs/osx_env.yml #
OS X
$ source activate pipeliner
```

Pipeliner requires configuration of paths to input data such as fastq reads, bam alignments, references files, etc. When cloning Pipeliner to a new machine, all paths must be reconfigured. This process can be automated by running a script that will reconfigure any paths to the same directory of your clone.

```
$ python pipeliner/scripts/paths.py
```

The final step is to download a Nextflow executable package in the same directory as the available pipelines.

TABLE 1 | Comparison of Pipeliner with common sequencing data workflows.

	Pipeliner	RseqFlow	PRADA	Galaxy
Flexibility	Any computational tool or script can be incorporated into existing workflows. Workflow steps can be skipped, modified, or removed.	Computational tools are determined and difficult to remove or modify.	Computational tools are determined and difficult to remove or modify.	Users are limited to existing tools supported by the platform.
Extensibility	Extensible by design. Includes modular workflows, testing capabilities, and documentation specific to extending the framework.	Requires extensive Python experience to be extended by users. Limited documentation for making changes.	Requires extensive Python experience to be extended by users. Limited documentation for making changes.	The platform is not designed to be modified by its users.
Reproducibility	Dependencies are installable from Anaconda cloud or environment files for Linux or OS X. Workflow configuration files are recorded and reusable.	No virtualization methods used; correct dependency versions must be installed manually. Workflow steps are not logged.	No virtualization methods used; correct dependency versions must be installed manually. Workflow steps are not logged.	Workflows can be saved, shared, and reproduced on the platform.
Installation	A few simple steps to configure environment and install dependencies.	Must install dependencies and configure environment manually.	Must install dependencies and configure environment manually.	N/A
Ease of Use	Simple config file. Provides small example datasets for local machines. One command to run entire workflow. Extensive documentation.	Simple config file. Workflow steps must be run individually. Provides example datasets. Limited documentation.	Simple config file. Workflow steps must be run individually. Provides example datasets. Limited documentation	Click and drag interface. Extensive documentation.
Data Types	RNA-seq scRNA-seq digital gene expression	RNA-seq	RNA-seq	RNA-seq ChIP-seq Mass Spec 16S
Interface Type	Command Line	Command Line	Command Line	Web-based
Available for Download	GitHub repository	SourceForge Tarball	Google Code Tarball	GitHub repository
Publicly Available	Yes	Yes	Yes	Yes

```
$ cd pipeliner/pipelines
$ curl -s https://get.nextflow.io | bash
```

With the setup complete, any of the available pipelines can be executed with their respective toy datasets with the following commands.

```
$./nextflow rnaseq.nf -c rnaseq.config
$./nextflow scrnaseq.nf -c scrnaseq.config
$./nextflow dge.nf -c dge.config
```

Proof of Concept

To showcase the applicability of Pipeliner to real-world datasets, we reprocessed 48 RNA-seq-paired read files for the lymphoid neoplasm diffuse large B-cell lymphoma (DLBC) cohort from The Cancer Genome Atlas (TCGA). For each cohort, the TCGA uses a standardized pipeline where reads are mapped to a reference genome with STAR and quantified by HTSeq. While the TCGA provides open access to the count matrix, some researchers have opted to use alignment and quantification algorithms specific to their research interests (Rahman et al., 2015). For this reason, the TCGA also provides raw sequencing data; however, its large size requires parallelization on a high-performance computing platform. We argue that Pipeliner is a suitable choice for users looking for alternative reprocessing of TCGA datasets with minimal pipeline development.

Pipeliner makes alternative processing of TCGA and other publicly available data straightforward. In processing raw RNA-seq data for DLBC, paired fastq reads were downloaded from the Genomic Data Commons Data Portal. For each sample, Pipeliner requires an absolute file path to reads. After specifying this information, Pipeliner was able to successfully process all data with HISAT2, featureCounts, and the remaining settings left to default. Data processing methods can have subtle effects on downstream analysis of sequencing data. This is exemplified by an increase in assigned features and decrease in multi-mapping when using HISAT2/featureCounts instead of STAR/HTSeq (**Figure 6**). The ability for researchers to reprocess publicly available datasets to suit their specific interests is important, and Pipeliner is a useful software tool that meets those needs. The flexibility provided by Pipeliner is ideal for users experimenting with different tools and parameters. For example, because Pipeline is capable of taking aligned bam files as input and skipping preceding steps, we were able to rapidly try all three quantification options without rerunning unrelated processes. This level of control is critical for downstream analysis of the processed data. To help researchers extend this example to other datasets, we provide the scripts used to obtain and organize TCGA data from the Genomic Data Commons as well as the configuration file used by Pipeliner to process the data in the supplementary information.

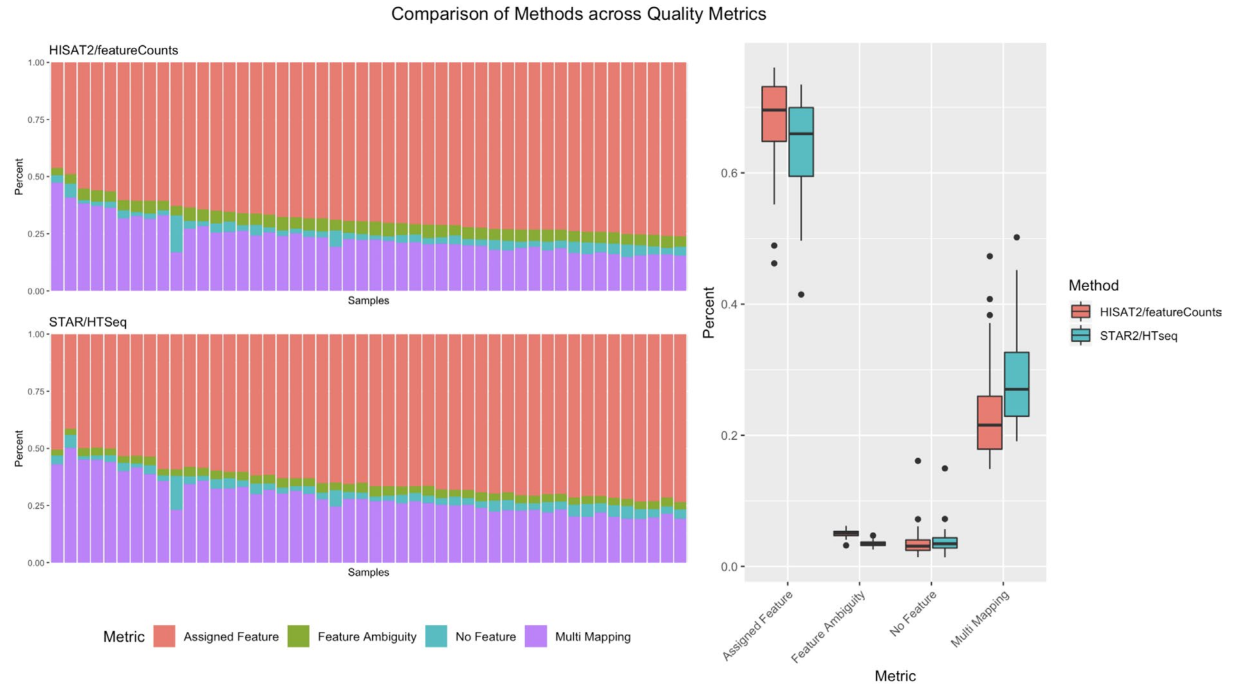


FIGURE 6 | A summary of counting and mapping quality metrics when processing with Pipeliner using HISAT2 and featureCounts compared with processed counts from The Cancer Genome Atlas (TCGA)-diffuse large B-cell lymphoma (DLBC) using STAR and HTSeq.

CONCLUSIONS

Together with Nextflow and Anaconda, Pipeliner enables users to process large and complex sequencing datasets with pipelines that are customizable, reproducible, and extensible. The framework provides a set of user-friendly tools for rapidly developing and testing new pipelines for various types of sequencing data that will inherit valuable design features of existing pipelines. We apply the RNA-seq pipeline to real-word data by processing raw sequencing reads from the DLBC cohort provided by the TCGA and provide supplementary files that can be used to repeat the analysis or serve as a template for applying Pipeliner to other publicly available datasets.

AVAILABILITY AND FUTURE DIRECTIONS

Pipeliner is implemented in Nextflow, Python, R, and Bash and released under a General Public License 3.0 license. It is publicly available at https://github.com/montilab/pipeliner and supports Linux and OS X operating systems. Comprehensive documentation is generated with Sphinx and hosted by Read the Docs at https://pipeliner.readthedocs.io/. We will continue to develop the Pipeliner framework as the Nextflow programming language matures, and we plan to provide additional pipelines for other types of sequencing data and analysis workflows in the future.

AUTHOR CONTRIBUTIONS

AF—Developed the current version of Pipeliner, wrote the manuscript, and generated the figures; TK—Initiated the project and developed early versions of Pipeliner; KK—Initiated the project and developed early versions of Pipeliner; DK—Initiated the project and developed early versions of Pipeliner; YK—Assisted in development of individual sequencing pipelines; JC—Initiated, oversaw, and guided the project as well as helped in writing the manuscript; SM—Initiated, oversaw, and guided the project as well as helped in writing the manuscript.

FUNDING

This work was supported by a Superfund Research Program grant P42ES007381 (SM) and the LUNGevity Career Development Award (JC).

ACKNOWLEDGMENTS

The authors would like to thank P. Di Tommaso for his assistance with Nextflow-related inquiries and A. Gower for his advice for improving and testing Pipeliner.

REFERENCES

Anders, S., Pyl, P. T., and Huber, W. (2015). HTSeq - a Python framework to work with high-throughput sequencing data. *Bioinformatics* 31 (2), 166–169. doi: 10.1093/bioinformatics/btu638

Andrews, S. (2010). FastQC: a quality control tool for high throughput sequence data. *Babraham Bioinforma.* doi: 10.1016/S1048-9843(02)00144-3

Continuum Analytics. (2016). Anaconda Software Distribution: Version 2-2.4.0. Available online at: https://continuum.io.

Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., and Notredame, C. (2017). Nextflow enables reproducible computational workflows. *Nat. Biotechnol.* 35 (4), 316–319. doi: 10.1038/nbt.3820

Dobin, A., Davis, C. A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., et al. (2012). STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* 29 (1), 15–21. doi: 10.1093/bioinformatics/bts635

Ewels, P., Magnusson, M., Lundin, S., and Käller, M. (2016). MultiQC: summarize analysis results for multiple tools and samples in a single report. *Bioinformatics* 32 (19), 3047–3048. doi: 10.1093/bioinformatics/btw354

Goecks, J., Nekrutenko, A., Taylor, J., Afgan, E., Ananda, G., Baker, D., et al. (2010). Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol.* 11. doi: 10.1186/gb-2010-11-8-r86

Kim, D., Langmead, B., and Salzberg, S. L. (2015). HISAT: a fast spliced aligner with low memory requirements. *Nat. Methods* 12 (4), 357–360. doi: 10.1038/nmeth.3317

Krueger, F. (2016). Trim Galore. *Babraham Bioinforma.*

Liao, Y., Smyth, G. K., and Shi, W. (2014). Featurecounts: an efficient general purpose program for assigning sequence reads to genomic features. *Bioinformatics* 30 (7), 923–930. doi: 10.1093/bioinformatics/btt656

Pertea, M., Pertea, G. M., Antonescu, C. M., Chang, T. C., Mendell, J. T., and Salzberg, S. L. (2015). StringTie enables improved reconstruction of a transcriptome from RNA-seq reads. *Nat. Biotechnol.* 33 (3), 290–295. doi: 10.1038/nbt.3122

Rahman, M., Jackson, L. K., Johnson, W. E., Li, D. Y., Bild, A. H., and Piccolo, S. R. (2015). Alternative preprocessing of RNA-Sequencing data in the Cancer Genome Atlas leads to improved analysis results. *Bioinformatics* 31 (22), 3666–3672. doi: 10.1093/bioinformatics/btv377

Soumillon, M., Cacchiarelli, D., Semrau, S., van Oudenaarden, A., and Mikkelsen, T. S. (2014). Characterization of directed differentiation by high-throughput single-cell RNA-seq. *BioRxiv.* doi: 10.1101/003236

Torres-García, W., Zheng, S., Sivachenko, A., Vegesna, R., Wang, Q., Yao, R., et al. (2014). PRADA: pipeline for RNA sequencing data analysis. *Bioinformatics* 30, 2224–2226. doi: 10.1093/bioinformatics/btu169

Wang, L., Wang, S., and Li, W. (2012). RSeQC: quality control of RNA-seq experiments. *Bioinformatics* 28 (16), 2184–2185. doi: 10.1093/bioinformatics/bts356

Wang, Y., Mehta, G., Mayani, R., Lu, J., Souaiaia, T., Chen, Y., et al. (2011). RseqFlow: workflows for RNA-Seq data analysis. *Bioinformatics* 27, 2598–2600. doi: 10.1093/bioinformatics/btr441

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2019 Federico, Karagiannis, Karri, Kishore, Koga, Campbell and Monti. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.