Integrated Workflow for Automated Transcription Start Site Prediction in Prokaryotes

Friederike Hanssen, Julian Späth, Jonas Ditz

Eberhard Karls Universität Tübingen

Abstract

The exact position at base level where the DNA is transcribed into mRNA, is called a Transcription Start Side (TSS). Several steps have to be performed in order to preprocess raw sequencing data into a format, which can be used for TSS prediction. We developed a workflow that performs these steps fully automated and uses TSSPredator [1] for prediction. The whole workflow is highly modulized with Nextflow [2] and Docker[1], which makes the whole process easy to maintain and exchange single modules.

I. Evaluation of workflow languages

A suitable workflow language was chosen between the three most prominent open-source newcomers: *Snakemake* [3], *Nextflow* [2] and *WDL*[2]. As portrayed in Table 1, each of them has their own benefits and disadvantages. Their behavior regarding reproducibility was one of the main aspects affecting the decision.

In order to achieve a high reproducibility, the language should provide straightforward installability. *Snakemake* and *Nextflow* can be set up via *Bioconda*[3], *GitHub*[4], and several other sources. In order to run *WDL*, `wdltools.jar` has to be downloaded and executed using a working *Java* environment. Another important aspect of reproducibility is a software container support. It is provided by *Nextflow* for two prominent softwares: *Docker* and *Singularity*. *WDL* needs an additional execution engine (*Cromwell*) to utilize *Docker* images. Snakemake does not provide any *Docker* or *Singularity* support.

The workflow should be able to be transferred too a cloud environment. The *Executers* in *Nextflow* provide an easy definition of the workflow execution location. Further changes to the workflow script are not necessary.

Another interesting point would have been which big companies use the individual workflow languages. While *Nextflow* publishes some featured pipelines in a *GitHub* repository[5], it was not possible to find companies working with *WDL* or *Snakemake*. One big company using *Nextflow* is the *International Agency for Research on Cancer* [6], which is is the specialized cancer agency of the World Health Organization.

Out of this pre-evaluation in the context of reproducibility, we chose *Nextflow* for the TSS workflow. This choice was based on the *Docker* integration, an extensive manual, and the prospect of easy-to-implement future cloud support.

[1]https://www.docker.com/
[2]https://software.broadinstitute.org/wdl/
[3]https://bioconda.github.io/
[4]https://github.com/
[5]https://github.com/nextflow-io/awesome-nextflow
[6]http://www.iarc.fr/

Table 1: *Comparison of the workflow languages Snakemake, Nextflow and WDL [2]*

Information	Snakemake	Nextflow	WDL
Licensing	MIT	GPL3	BSD 3-clause
Source code availability	✓	✓	✓
Online community	✓	✓	✓
Installability	Bioconda/GitHub/ PIP	Bioconda/GitHub/ Package Manager/Homebrew	GitHub
Platform	Python	Groovy/JVM	domain specific
Docker support	×	✓	✓
Singularity support	×	✓	×
Cloud integration	×	✓	×
Job scheduler support	✓	✓	✓

II. Evaluation of TSS Workflow

I. Workflow Structure

The general structure of our TSS workflow can be found in Figure 1. An important prerequesite is the storage of all needed input files in a common folder as well as the adjustment of nextflow's configuration file template provided by the GitHub directory. After all prerequisites are achieved, *BWA* [4] maps all provided reads to a reference genome. The resulting alignment will be converted into a binary file using *Samtools* [5]. Furthermore, *Samtools* is used to seperate forward and reverse strand. These binary files are converted into graph files using *TSSTools* and, finally, TSS prediction in calculated using TSSPredator.

II. Docker support

All submodules used by the workflow are implemented inside of Docker containers that are uploaded to *DockerHub*[7]. The usage of containers leads to a vast flexibility by providing new features with a simple exchange of Docker images. The Docker builds on *DockerHub* are automatically updated when the Dockerfile in the corresponding *GitHub* repository is changed and additionally allows an easy versioning by using tags.

III. Analysis

The workflow was evaluated on one dataset 'SRR1951997/SRR1951998' and 'SRR1951999/SRR1952000' using Ubuntu in a virtual machine. The host computer had a Intel i5-7500 quad-core CPU with 3.4 GHz Processor rate and 6 Gb memory. Ubuntu 16.04 in the 64-bit version was used as the operating system.

. The data originates from wildtype *H.pylori* samples. The times were measured for an sequential and a parallel run and can be found in Table 2. The times for the parallel workflow are larger than for the sequential one for both samples.

[7]https://hub.docker.com/

Table 2: *Runtimes on the two normal datasets SRR1951997 and SRR1951999 and the two TEX enriched ones SRR1951998 and SRR1952000. Each origin from the wildtype of H.pylori*

Dataset	Length (Bp)	Enriched	Iterative(min)	Parallel(min)
SRR1951997	458.2M Basen	×	8.17	9.08
SRR1951998	399.4M Basen	✓		
SRR1951999	814M Basen	×	8.11	8.86
SRR1952000	698.9M Basen	✓		

III. Discussion

During this Project we were able to implement a workflow automating TSS prediction starting with raw read data. Since all Docker images used during the workflow can be found in a DockerHub repository, we achive a high reproducibility due to the possibility to fetch specific versions of images using a tag system. Furthermore, our workflow is highly modularized. Each prozess executes exactly one tool and each tool is contained in a separate docker container. This allows for vast flexibility in the tools applied. Each of them can simply be replaced by calling a different docker image, e.g. for mapping use BowTie [6] instead of BWA.

Although Nextflow allows a docker integration, it was not used during this project. In order to run the docker containers, scripts were invoked. Furthermore, in a post-project evaluation, we found that the Nextflow script was difficult to implement. A manual exists, but the contained examples often only cover basic approaches. More advanced issues are hard to resolve. This was particularly apparent when the task was to parallelize the workflow. Although this workflow language aims at executing as many process as possible in parallel, we were unable to execute all needed TSSTools runs parallely. Processes have to be ordered by opening channels between two of them. The previous SamTools process is only executed twice. Therefore, the TSSTools process recieves a channel of size two and can only run twice as well. However, it would need to be executed four times as can be seen in Figure 1. Presently, this was implemented with invoking a parallelized shell script. Additionally, the online community is still rather small likely due to its relative novelty.

The unexpected runtime behavior during sequential and parallel execution of our workflow, namely the longer runtime for parallel runs, could occur due to the use of a virtual machine as a host system. We have not tested the same datasets on a different computer, but a smaller dataset tested on a Linux host computer (Linux Mint 18 Cinnamon, 64-bit, Intel Core i5-6200U CPU @ 2.30 GHz x 2, 8 Gb memory) resulted in a faster runtime for parallel execution of our workflow. Therefore, further tests are needed to determine whether the faster sequential runtime solely occurs due to issues with the virtual machine host.

In the future more containers could be added to the workflow. This would allow the user to produce results based on different tools, mainly different mappers. Furthermore, multiple configuration files for TSSPredator could be used to execute multiple settings in parallel without having to recompute the preprocessing steps. This would make an analysis of different parameters time-efficient and simple.

References

[1] G. Dugar, A. Herbig, K. U. Förstner, N. Heidrich, R. Reinhardt, K. Nieselt, and C. M. Sharma, "High-resolution transcriptome maps reveal strain-specific regulatory features of multiple campylobacter jejuni isolates," *PLoS genetics*, vol. 9, no. 5, p. e1003495, 2013.

[2] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, "Nextflow enables reproducible computational workflows," *Nature Biotechnology*, vol. 35, no. 4, pp. 316–319, 2017.

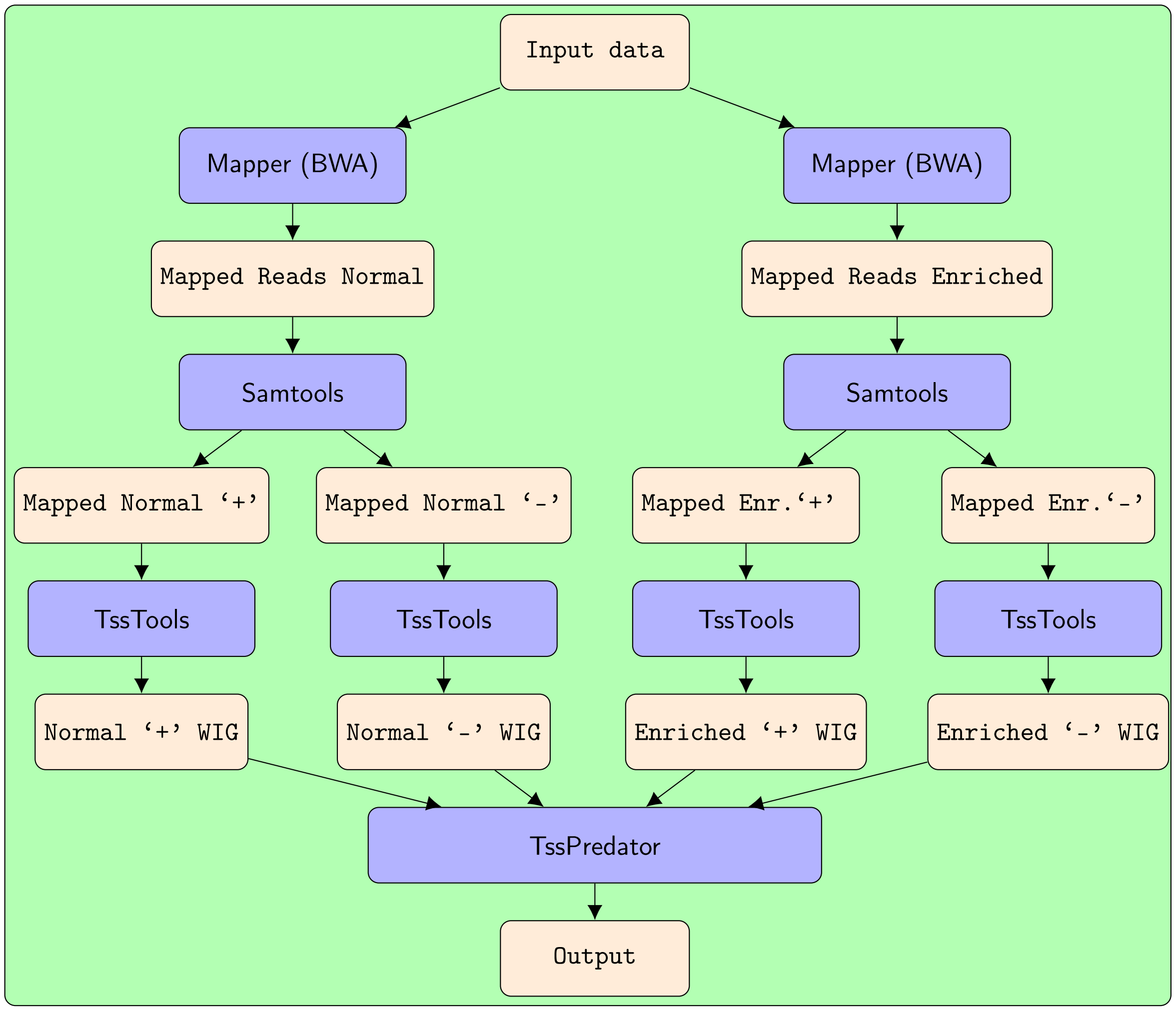
[3] J. Koester and S. Rahmann, "Snakemake - a scalable bioinformatics workflow engine," *Bioinformatics*, vol. 28, no. 19, p. 2520, 2012.

[4] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with bwa-mem," *arXiv preprint arXiv:1303.3997*, 2013.

[5] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, and R. Durbin, "The sequence alignment/map format and samtools," *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, 2009.

[6] B. Langmead and S. L. Salzberg, "Fast gapped-read alignment with bowtie 2," *Nature methods*, vol. 9, no. 4, pp. 357–359, 2012.

Figure 1: *Flowchart of our TSS workflow*


Input data
Mapper (BWA)
Mapper (BWA)
Mapped Reads Normal
Mapped Reads Enriched
Samtools
Samtools
Mapped Normal '+'
Mapped Normal '-'
Mapped Enr.'+'
Mapped Enr.'-'
TssTools
TssTools
TssTools
TssTools
Normal '+' WIG
Normal '-' WIG
Enriched '+' WIG
Enriched '-' WIG
TssPredator
Output