Julien Pernet, Matthieu Genais, Juliette Cooke

Project summary
Masters in Bioinformatics 2018/2019

Developing an RNA-seq differential analysis pipeline in order to compare two tools, featuring STAR as the grandfather and Kallisto as the newborn promising son

Key-words: RNA-seq, pipeline, benchmark(ing), differential analysis

Julien Pernet, Matthieu Genais, Juliette Cooke

Introduction

Our project sponsor is Vincent Rocher from the CBI laboratory. He is a PhD student in the Chromatin and DNA repair Team. He requested this project in order to compare the characteristics of two different RNA-seq analysis tools. In the future, the existing one will be replaced by the new one we have developed and implemented. Improving this RNA-seq analysis pipeline could be an important contribution in the advancement of our sponsor's PhD project.

Cellular DNA can be broken because of multiple factors: external ones such as radiation, cigarette smoke, or biological ones like during meiosis. One type of lesion is DSB (Double Strand Breaks), which is known to be the most deleterious since they can lead to various mutations and chromosomal rearrangements in the cell and during its development. The cell can identify and repair this damage using DNA Damage Response (DDR) processes.

The Chromatin and DNA repair team at the CBI has developed a cellular line called DIvA [Iacovoni et al, EMBO Journal 2010], which allows them to study these DSB-specific DDRs. Our project focused on improving the RNA-seq data (produced by the DIvA system) analysis pipeline. In the future, studying this data will allow further identification of the genes associated with DSB-specific DDR. If a cell fails to repair DSBs, it could lead to genomic instability and cancer.

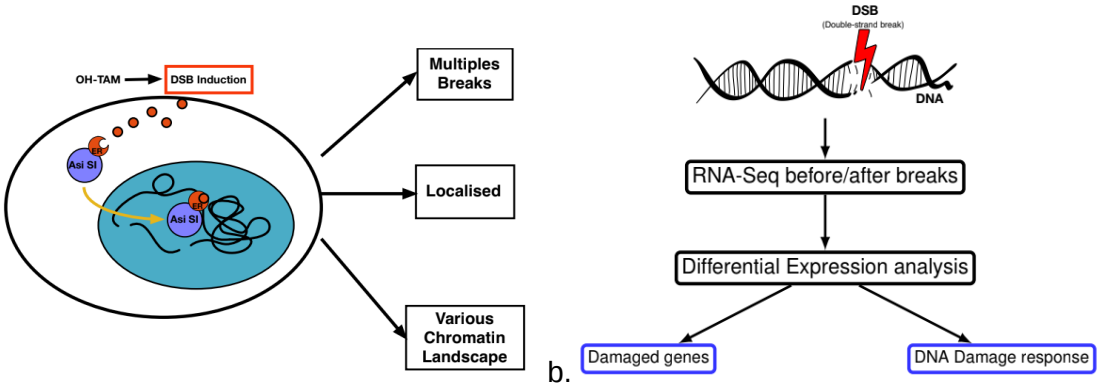

Figure 1 : (a) DIvA cellular line, (b) DSBs and differential analysis

Methodology

First of all, we had to integrate a piece of software called Kallisto [Bray et al, Nature 2016] into a RNA-seq data analysis pipeline, which we built from scratch using Snakemake [Köster et al, Bioinformatics 2012] (python programming framework used to develop and manage bioinformatics pipelines). A second pipeline was developed along the lines of an existing one, which used an older piece of software, STAR [Doblin et al, Bioinformatics 2013], along with htseq-count [Anders et al, Bioinformatics 2014]. Kallisto is considered to be better (higher performance) than the existing tools as it is not the same algorithm, STAR needs to align the reads (a **read** is an inferred sequence of base pairs from a DNA fragment) to the reference genome, while Kallisto can quickly identify the origin of each read using a specific programming structure based on k-mers (“words” of length = k) : a De Bruijn Graph.

After the integration of Kallisto, we had to benchmark (i.e : compare the performance) STAR and Kallisto. The goal of this was to give our project sponsor proof that it is in his best interest to use the Kallisto program in terms of execution time and for similar performances, or not, for future analyses.
In order to do this, we also used Snakemake, as it has a built-in benchmarking option.

Once the pipeline had been completed, we set it up on the Genotoul computing cluster. The task/job management was on the same cluster nodes, and used SLURM (also built-in within snakemake).

Finally, we had to compare the ability of the two pipelines to find differentially expressed genes, using edgeR [Robinson et al, Bioinformatics 2010] (edgeR is a statistical R package that can identify differentially expressed genes between different conditions using RNA-Seq data).

Results

We successfully implemented both Snakemake pipelines using STAR paired with htseq-count, and Kallisto. Using Snakemake's benchmarking option, we managed to benchmark the pipelines on their running time, memory usage and input-output read/write time.

We also created a config file containing a Python dictionary in JSON format (https://json.org/) with keys and values. This file allowed us to change the file names inside the config file, without having to go into the Snakefile to change the code directly.

The pipelines allow the user to start with the Snakefile script, and an input folder containing the reads, the reference genome and the GTF file (file format used to hold information about gene structure). The user only has to type `snakemake` in the console for it to run the entire pipeline. The output folders for each rule are generated automatically.

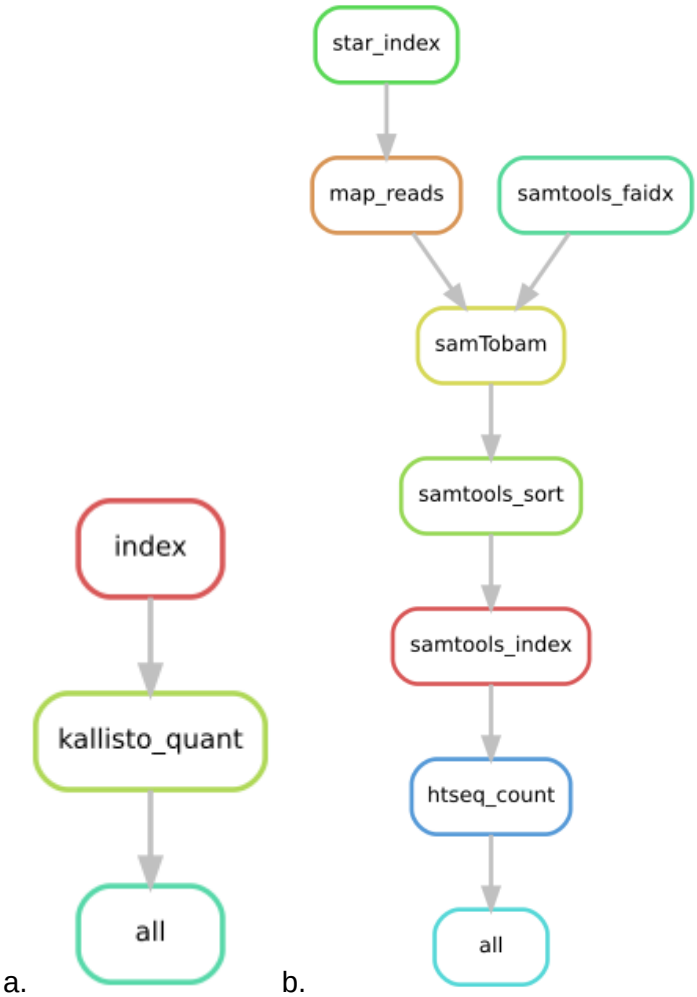

Figure 2 : DAG (Directed Acyclic Graph) of our pipelines, generated using snakemake --rulegraph | dot | display ***(a)*** *Kallisto DAG,* ***(b)*** *STAR - htseq-count DAG*

s	h:m:s	max_rss	max_vms	max_uss	max_pss	io_in	io_out	mean_load
71.3655	0:01:11	771.73	3507.36	765.70	766.44	49.74	532.49	0.00
74.0102	0:01:14	771.63	3507.36	765.68	766.46	0.52	532.49	0.00
87.7650	0:01:27	1873.46	3507.36	1867.50	1868.21	0.18	1768.43	0.00

Figure 3 : Example of a benchmarking result
s: *run-time in seconds,* ***h:m:s:*** *hours:minutes:seconds,* ***max_rss:*** *max resident set size (portion of RAM occupied),* ***max_vms:*** *virtual memory size,* ***max_uss****: max unique set size (unshared memory),* ***max_pss****: max proportional set size ((USS + RSS)/number of processes sharing RAM),* ***io_in****: input read in MB,* ***io_out****: output write in MB*

We then ran the benchmarking analysis on the previous results with R and obtained the following results. This barplot represents the average time for each method / tool of our two pipelines.

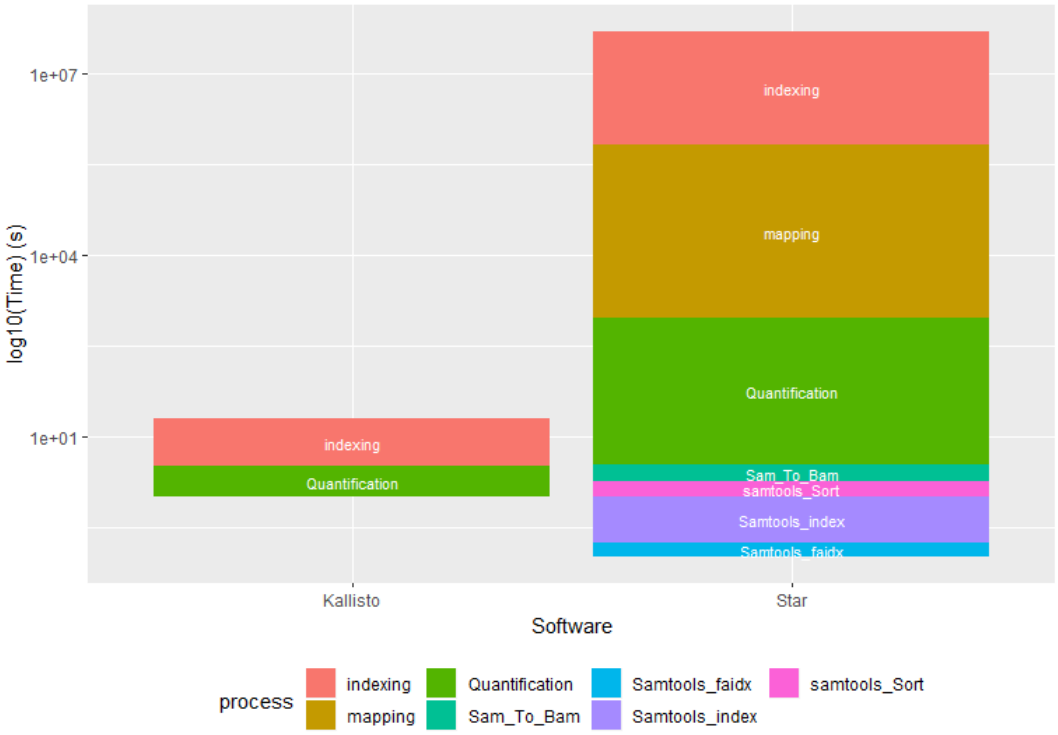

Figure 4 : Bar plot of log(run-time), Kallisto vs Star

As we can see, Kallisto, in addition to being easier to use, is much faster than the STAR + htseq-count pipeline.

Discussion

From our benchmarking results on test data, we found that Kallisto runs much quicker than STAR + htseq-count. This was expected, as Kallisto uses pseudo alignments to map the reads to the genome much faster than traditional alignments. As we can see, the mapping (=alignment) phase of STAR took the longest to run, whereas the Kallisto quant command was much faster.

Furthermore, kallisto quant not only pseudo-aligns (i.e looks if the read is compatible with the sequence instead of comparing its similarity), but also does the quantification phase, which is also what htseq-count does. As we can see, kallisto quant is faster than htseq-count.

While Kallisto is much faster, only differential analysis will allow us to determine whether Kallisto is as accurate in its detection of differentially expressed genes. However, the differential analysis using edgeR is a small part of our project, and it should only be done once the rest of the project is finished.

The next step is to set up the pipelines on the Genotoul server, allowing us to run benchmarks with large RNA-seq input files. However, the Genotoul cluster is organized via a queueing system, which means it could take days for our pipeline to be tested. This process is therefore long, which is why we did not have the results for the real dataset provided by our sponsor.
For now, we can only assess that our pipeline works and Kallisto is significantly better in term of execution time than STAR.