JAST

DELESTRE Clément

June 29, 2014

Contents

1 About JAST

JAST stands for **J**ava **A**ssembly and **S**caffolding **T**ool. This program performs assembling and scaffolding from paired-end Illumina library and use a reference genome.

1.1 Dependencies

JAST uses the following softwares : (see 1)

- Flexbar for filtering reads.
- A5 for assembling reads.
- Bowtie to map the filtered reads against the reference genome (a SAM file is requiered for Colombus).
- Colombus thanks to VelvetOptimizer for scaffolding using a reference genome.
- And SSPACE to improve obtained scaffolds.

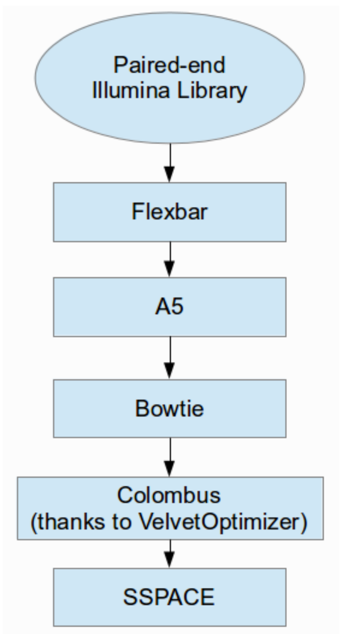

Figure 1: Simple pipeline of JAST.

These programs have to be well installed on your computer before you can use JAST. As for every Java programs, you also need a JRE to be able to use it. Please note the one used to test JAST was 1.7.

2 How it works

Here we describe how to use `JAST`. Please note this software doesn't have `GUI`[1] so we assume you are able to deal with command lines. All options will be not discussed here, for know all of theme use -h or - -help.

2.1 Input file(s)

`Illumina` library

`JAST` was designed to work with paired-end `Illumina` library. Your files name *MUST* follow this format : *myReads_1.fastq* and *myReads_2.fastq*. The first one must be use with the -r or - -reads option and the second one with -p or - -paired.

Config files

If you want to specify some options to sofwares used by `JAST` you can do this with a config file. Each config files must contain each options separated by a break line. `VelvetOptimizer` is the only sofware requires a config file. Examples of config files can be found at : `https://github.com/AgResearch/JAST/tree/master/ExampleConfigFiles`

SSPACE library

`JAST` requires a `SSPACE` library file ; equivalent to file you specify with '-l' option when you use `SSPACE`. Please note that the first line must contains only the 3 last columns : the other information (name and fastq files) will be writen in a file nammed with _JAST extension and the library will be nammed JASTlib. This file must be specified with -l or - -sspacelib option.

2.2 Stdout and stderr

Messages displayed to stdout by `JAST` during its process, start with `[JAST]`. When an error appears, messages displayed to stderr start with `[JAST_Error]`. Messages do not start with these symbol belonging to softwares used.

2.3 About the pipeline

The pipeline is more detailed at the figure 2. On this figure we assume that "?" could be "1" or "2", "genome" is the reference genome (- - ref option) and "output" is the output name specified by -o or - - output option.

[1]**G**raphical **U**ser **I**nterface

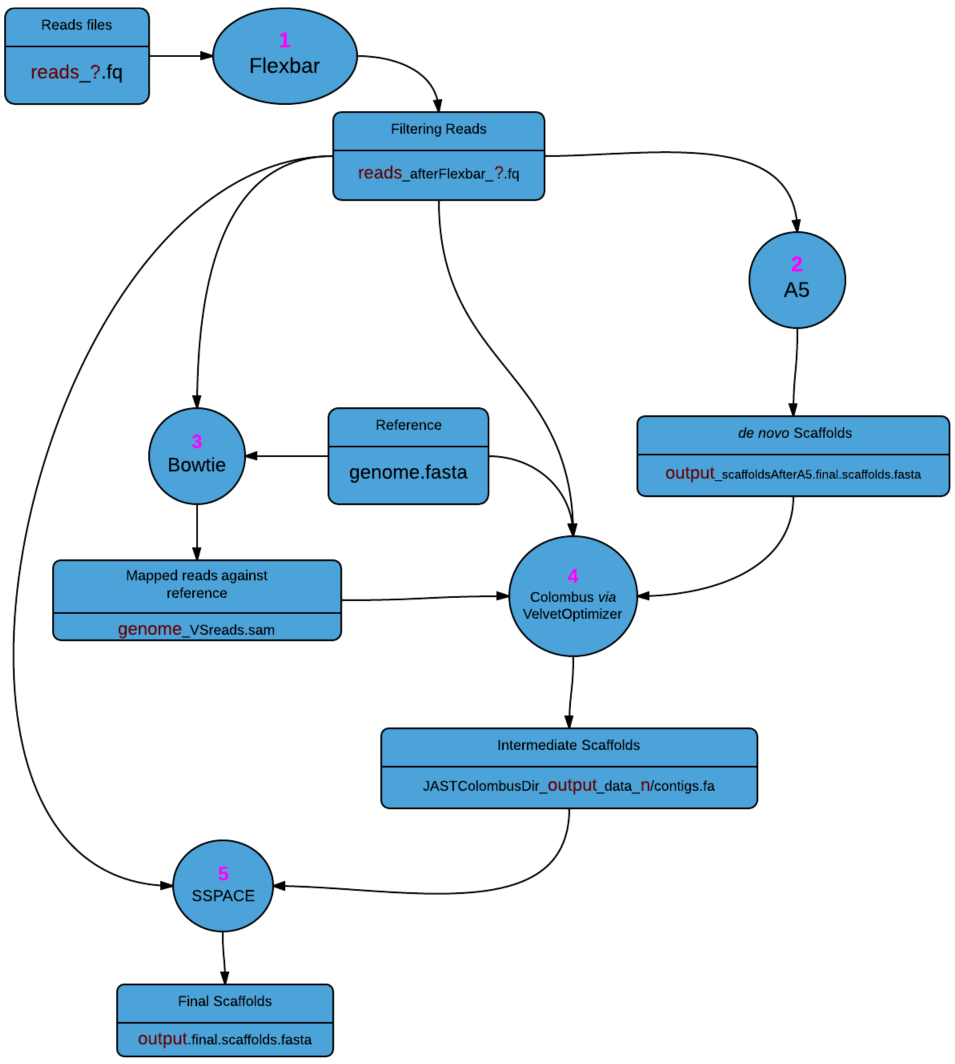

Figure 2: Complete pipeline of `JAST`.

3 For programmers

Dependencies

`JAST` use the `JSAP` library v2.1 downloaded in June 2014, under LGPL licence : `http://www.martiansoftware.com/jsap/` . You need to download it if you want to generate you own JAR file from source code or modify source code.

Software architecture

In `JAST`, all commands inherit from the `Command` abstract class (see 3), has the following attributs & methods :

- Attributs :
  - command (String) : The executable command i.e. for flexbar it's "flexbar"

  - forbiddenOptions (String[]) : Options already used by `JAST` the users not allowed to use.
  - totalCommand (List<String>) : All the command with executable and options.
  - config (Path) : The config file, can be `null`.
  - outputFile (Path) : The outputFile.
- Methods :
  - exec : Execute the command.
  - getOutPutFile : get the output file.
  - getCommand : get the executable command.

The source code can be found here : `https://github.com/AgResearch/JAST/tree/master/src` and the Javadoc can be found here : `https://github.com/AgResearch/JAST/tree/master/Javadoc` .

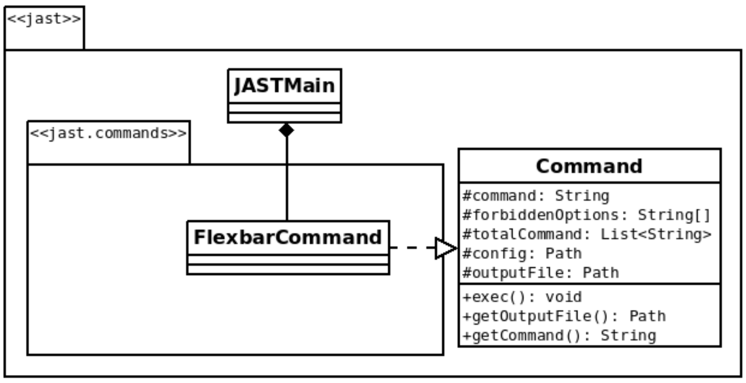

Figure 3: Class diagram example for one command (`Flebar`).

Considering that bioinformatics software evolve rapidly in the field of sequencing, it could be usefull to add a new software to the pipeline. Here is how do so :

```java
package jast.commands;
import java.nio.file.Path;
import java.nio.file.Paths;
import jast.Command;
public class newCommand extends Command {
    public newCommand(Path config,String [] arrayOfForbbidenOptions) {
        super(config,arrayOfForbbidenOptions);
        command="foo";
        // no need to add command to totalCommand
        totalCommand.add("-bar");
        totalCommand.add("myArgumentForBarOption");
    }
}
```