Highly efficient pipeline to perform integrated analysis of transposable elements and tandem repeats in complex genomes

Lucas Soares de Brito[1], *Jaire Alves Ferreira Filho*[2], *Marcelo Soares Souza*[3], *Eduardo Fernandes Formighieri*[4]

Graduados

Abstract

The study of repetitive regions is gaining attention with increase of sequenced genome, mainly the Transposable Elements (TEs, such as LTR) and Tandem Repeats (mainly as microsatellite candidates). This type of analysis demands deep Bioinformatics/IT knowledge and usually takes long processing time. In our study, we present an alternative way to facilitate the user's work by using a single Perl pipeline script, that automates full identification and annotation and improve the analysis performance, including automatization of intermediate steps and avoiding not so rare manual compilation errors. To evaluate the pipeline efficiency and robustness, we tested parameters and run it with different sequences, varying the number of processing cores of the machine. By automating the analysis, the pipeline reduces the execution time for *E. guineensis* from several weeks or months (comparing to previous manual work) to some days, by just using one CPU processor. With 48 processors, it takes only 2 days. This quick and efficient tool can help other works that aims to discover and annotate repetitive content (e.g., SSRs for breeding) and/or improve the repetitive content filtering to obtain better genome assemblies.

Introduction

In the last decades, the DNA sequencing technology passed through several advances, which allowed a huge leap for the genomic sequences analysis. The

[1] Engenheiro de Redes de Comunicações, mestre em Engenharia de Sistemas Eletrônicos e de Automação, Universidade de Brasília, lucas.brito@colaborador.embrapa.br
[2] Biólogo, doutorando em Genética e Biologia Molecular, Universidade Estadual de Campinas, jaire_hp@yahoo.com.br
[3] Informata, graduado em Informática pela Universidade Católica do Salvador, marcelo@juntadados.org
[4] Engenheiro-agrônomo, doutor em Biologia Funcional e Molecular, pesquisador da Embrapa Agroenergia, eduardo.formighieri@embrapa.br

New Generation Sequencing (NGS), represented by platforms such as Roche/454 and Illumina/Solexa technologies, and the third generation (e.g., PacBio), has contributed with the generation of several new genomic sequencing data. One of the consequences of this phenomenon is that the analysis size and complexity is growing constantly. The genome characterization plays an important role in these analyses, where the functionality of each gene can be described. However, to understand each gene functionality, we also have to characterize non-coding sequences like *repetitive regions*. These regions have several implications on genomic characterization; e.g., it may complicate the assembly process by generating gaps in the DNA sequence. The most important kinds of structures related with repetitive content are tandem repeats and Transposable Elements (TEs).

Tandem repeats are motifs repeated in tandem, and can be classified as microsatellites or minisatellites. Microsatellites are composed by motifs from 1 to 6 base pairs, while the second have larger motifs. Because this type of repetition makes difficult correct transcription, inducing mutation in DNA replication, these regions tend to have a high degree of polymorphism. Thus, considering the variation of the number of similar motifs in different genotypes, these structures are used as potential molecular markers for breeding studies.

There are two classes in *Transposable Elements* (*TEs*): class *I* and class *II*, subdivided in families, and these last ones into types. In class I we have *Retrotransposons*, which are the most complex, because they replicate to another genome position into the genome through intermediates (transcriptase and reverse transcriptase). Thus, the Retrotransposons replicates by a *copy/paste* mechanism, increasing genome size (GRANDBASTIEN, 1992). The Retrotransposons are also the most abundant TEs in plant genomes, and the main families are *Copia* and *Gipsy*. In the class II, the mechanism of replication is similar to Class I, but without the requirement of intermediates.

Transposable elements are related to the genome size variation and genetic variation. These elements can also be used as markers, with different applications, such as gene therapy, gene silencing, evolutionary studies and epigenetics.

A complete repetitive analysis from a genome, including identification and classification, is a laborious and complex job. Ferreira Filho (2015) performed this analysis from scratch in some plants, made a complete annotation of TEs and microsatellites of an Embrapa genotype of *E. oleifera* (manuscript being

submitted) and it cost a lot of work and time (it took literally months). It was a great effort, which could have been addressed to not automatable problems. This fact motivated us to develop a pipeline to really facilitate and accelerate these analyzes, consistently and reducing the risk of errors. The first version of this pipeline will be presented in this work. The work presented here was developed by Bioinformatics Research Group at the Bioinformatics and Bioenergy Laboratory – LBB. All uses software products are free to use and run in Linux OS.

Material and methods

Software

- **TRF –** *Tandem Repeats Finder* program (https://tandem.bu.edu/trf/trf.html), to identify repetitions with any pattern or size, perfect and imperfect. Output with tandem repeats (html format).
- **TRAP –** *Tandem Repeats Analysis Program* (www.coccidia.icb.usp.br/trap/), to compile and categorize results from TRF. Output with sorted tandem repeats (html format).
- **LTR Finder –** (https://code.google.com/archive/p/ltr-finder/), to predict locations and structure of full-length LTR retrotransposons accurately. Output all LTRs annotated (text format).
- **RepeatModeler –** (http://www.repeatmasker.org/RepeatModeler.html), to identify and model repeat families. Depends on RECON and RepeatScout (*de novo* repeats finding); TRF; Perl programming language, RepeatMasker (described below) and a search engine (RMBlast or ABBlast). Output repeat families (fasta format).
- **NCBI Blast –** *Blast Search Alignment Tool* (www.ncbi.nlm.nih.gov/guide/howto/run-blast-local), to compare sequences against TEs databases aiming annotation. Output matchs (‘-t 6’ tabular format).
- **RepeatMasker –** (http://www.repeatmasker.org/, Tarailo-Graovac and Chen (2009), to identify, annotate and mask repetitive elements. The masked version (Ns at TEs regions) permits better genome assemblies. Output: i) repeats (‘cat’ format, similar to fasta); ii) annotation (tabular format); iii) masked sequences (fasta format); and iv) report (text format).

Biological material (DNA sample)

We tested the pipeline into four plant genomes (data not shown), but used one of them to build performance metrics for this work: *Elaeis guineensis* (EG5) (from NCBI, January 2015). We used the .fna file (fasta format), which contains the draft of the 16 chromosomes and other 40,044 unplaced scaffolds. The total length of this fasta is ~1.5 Gbp (Giga base pairs), the maximum scaffold length is ~65 Mbp and the *N5O* is ~1.2 Mbp. The average GC content is 37.21%, and it presents 164,421 gaps, or 31.14% of the genome.

Methods

The goal of this pipeline is the simplified execution of chosen repeats analyzes efficiently, respecting the dependencies and processing intermediate files in order not to require any user interaction after it starts.

To achieve that, the strategy was: i) select the best tools for each step; ii) choose a suitable programming language; iii) setup hardware and software infrastructure (more infrastructure details on the work of Marcelo S. Souza, this same event); iv) test tools parameters; v) build auxiliary scripts; vi) develop and test basic pipeline; vii) improve pipeline; viii) validate pipeline; ix) write documentation and pipeline disclosing.

Chosen language was Perl (structured), and the Operational System (OS) was Linux (all used tools was developed just for this OS). Additional scripts, needed to connect some of the steps, provide auxiliary functionalities, such as well as data format conversion, concatenation and filtering. To make possible the full automation, we used a unique configuration file (text format), which includes parameters for all stages of the pipeline, such as original options for each of the bioinformatics software and choose of modules to run (one, some or all pipeline steps).

For the pipeline optimization, we used parallelism techniques (Perl *fork*) and normalized the multithreads capacity into a single pipeline parameter, maximizing the use of processors for all tools. In other words, with one parameter setup, you can get the best out of your machine processing for all analyses. More details about the pipeline can be found in the pipeline documentation, at the code itself, and, as soon as possible, in a publication of the final version.

Results and discussion

We develop the pipeline with Perl language. The final script is automated, modular and parallelized. Its structure is shown in Figure 1, where we can see the parallelism between tools (TRAP, LTR Finder and RepeatModeler) and the main steps and dependencies.

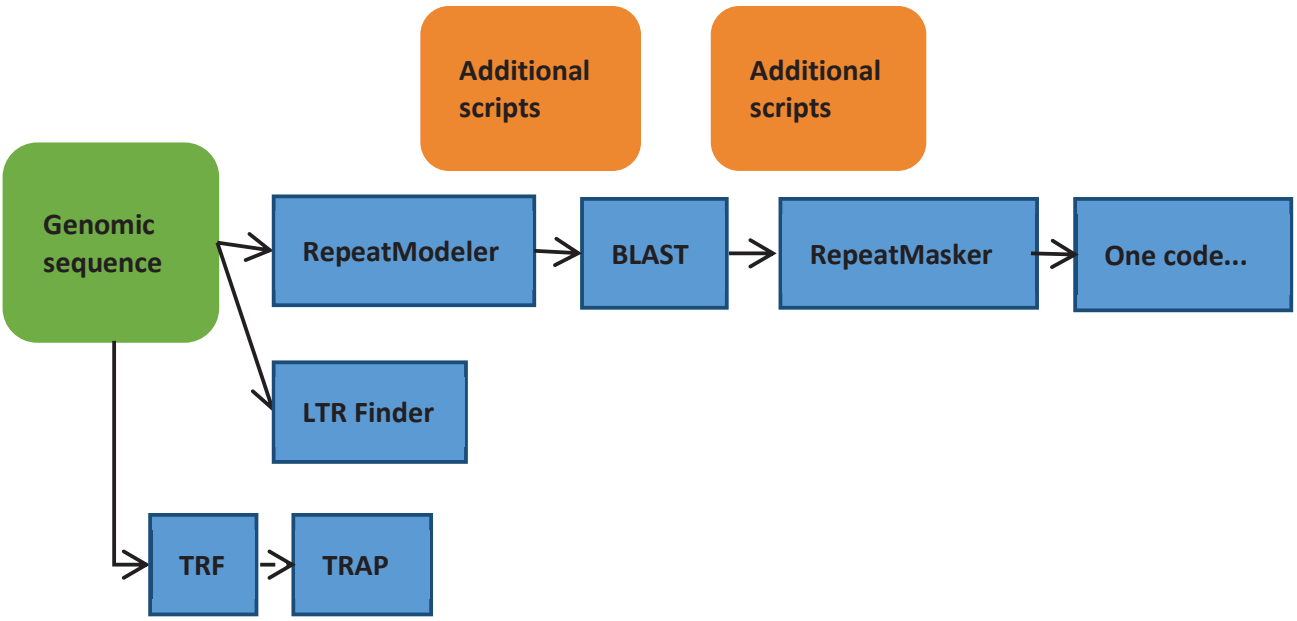

Figure 1. Pipeline structure.

To run the pipeline, up to main three steps: i) install Linux software packages (if is the very first time); ii) setup config file (the config file includes suggestions whenever possible); and iii) run a unique and simple command line.

Automation

To be done manually, one needs to choose tools, install each one (and its dependencies), learn and test, understand, convert and in some cases still need to process and filter output from one tool to the demanded for the next one. All of this depends on Bioinformatics/IT skills. We turned all these steps (and a few more things) into one command line.

How to run

Before running the pipeline command, there are a few requirements to follow, listed below:

- Define and/or create a folder for the results (Linux OS), and set the path into the config file.
- Put the completed config file and your fasta file into the results folder.
- Execute the main script (below) inside this results folder:

  $ rep_pipeline <name of your fasta file>

Multithread performance

We tested all multi thread tools and normalized all of them into one configuration parameter. Table 1 shows the user execution time, for each main pipeline software and for two cases: one processor and 48 processors. By looking at this table, we can see the huge gain of performance in execution time. The largest gain was in the *RepeatMasker* step, followed by *RepeatModeler*.

Table 1. Pipeline execution time according to the number of processors.

Time x Processors	1	48
TRF	0 day(s), 00:58:05	0 day(s), 00:58:03
TRAP	0 day(s), 00:14:22	0 day(s), 00:13:32
LTR Finder	0 day(s), 04:54:37	0 day(s), 04:47:08
RepeatModeler	2 day(s), 07:00:13	0 day(s), 21:46:09
Auxiliary	0 day(s), 00:00:03	0 day(s), 00:00:02
Blast	0 day(s), 00:00:52	0 day(s), 00:00:28
Post Blast	0 day(s), 00:00:10	0 day(s), 00:00:12
RepeatMasker	9 day(s), 02:40:57	0 day(s), 16:17:24
Post RepeatMasker	0 day(s), 04:14:00	0 day(s), 00:47:33

Format: “day, hh:mm:ss”.

Conclusions

The developed pipeline is able to execute in a simple and efficient way a complex and complete analysis of repetitive regions, including Tandem Repeats and Transposable Elements of all present classes, families and types.

A configuration file works very well in this context, since it can contain into just one file: parameters for all tools and steps, optimized parameters from literature, and also comments explaining each tool and each parameter.

In the near future, we aim to provide installation packages for used tools, to do some additional improvement and then publish the full work and the codes.

Financial support

This study was supported by the Brazilian Ministry of Science, Innovation and Technology (MCTI) through a grant (DENDEPALM) provided by Conselho Nacional de Desenvolvimento Científico (CNPq).

References

GRANDBASTIEN, M.A. Retroelements in higher plants. **Trends in Genetics**, Cambridge, v. 8, n. 3, p. 103-108, 1992.

TARAILO-GRAOVAC, M.; CHEN, N. Using RepeatMasker to Identify Repetitive Elements in Genomic Sequences. **Current Protocols in Bioinformatics**, New York, chapter 4, unit 4.10, 2009.

FERREIRA FILHO, J. A. Caracterização de sítios polimórficos e sequências repetitivas, e estabelecimento de coleção nuclear de caiaué [*Elaeis oleífera* (Kunth) Cortés]. 2015. 112 p. Dissertação (Mestrado) – Universidade de Lavras, Lavras, MG. Originalmente apresentada como dissertação de mestrado à EMBRAPA Agroenergia.