Refatorando um *Pipeline* de Bioinformática: Um Estudo de Caso para Análise de Amplicons

Aline Mara Rudsit Bini[1], **Liliane Santana Oliveira** [2], **Heloisa Poliselo** [2], **Francismar Correa Marcelino Guimaraes**[2], **André Yoshiaki Kashiwabara**[1]

[1]Departamento de Computação – Universidade Tecnológica Federal do Paraná (UTFPR) – Cornélio Procópio – PR – Brazil

[2]Embrapa Soja – Londrina – PR – Brazil

alinebini@alunos.utfpr.edu.br

Abstract. *Bioinformatic pipelines are essential to make it possible to process the enormous amount of available biological data. An approach in analyzing biological data is to implement scripts using a programming language such as Perl, Python, R, or Bash. However, these scripts can be hard to maintain and to understand by other developers. In this work, we refactored a pipeline for amplicon analysis from NGS (next-generation sequencig) data implemented in Perl to make it scalable, portable, with simplified code and parallelization of existing processes. For this, we used a workflow manager called Nextflow.*

Resumo. *Os pipelines de bioinformática são essenciais para possibilitar o processamento da enorme quantidade de dados biológicos disponíveis. Uma abordagem na análise de dados biológicos consiste em implementar scripts usando uma linguagem de programação como Perl, Python, R ou Bash. No entanto, esses scripts podem ser difíceis de serem mantidos e compreendidos por outros desenvolvedores. Neste trabalho, reimplementamos um pipeline para análise de amplicons, implementado em Perl, para torná-lo escalável, portável, com código simplificado e para a paralelização de seus processos. Para isso, utilizamos um gerenciador de workflow denominado Nextflow.*

1. Introdução

Um *workflow* é uma forma de organizar diferentes tarefas a serem executadas, de forma a gerar um resultado final. É possível criar diferentes estruturas com estas tarefas, sendo uma delas o *pipeline*, caracterizado por ser uma arquitetura de *software* amplamente utilizada quando o objetivo da análise consiste no processamento de dados, visto que ele possui vantagens como: facilidade de compreensão do comportamento geral do sistema; suporte a reutilização; capacidade de aprimoramento; análise especializada; e execução paralela de seus componentes [Garlan and Shaw 1993]. Estas características provêm de sua topologia, a qual é baseada em *pipes* e filtros. Os *pipes* formam um canal unidirecional de comunicação entre os filtros, o qual recebe uma entrada e a direciona para uma saída após realizar uma tarefa [Richards and Ford 2020].

Com isto, muitos experimentos têm sido compartilhados em forma de *pipelines* no intuito de disponibilizá-los para a comunidade científica. Entretanto, eles são,

em sua maioria, projetados para uso local, o que aumenta as chances de ocorrer problemas como irreprodutibilidade computacional, má utilização de *software* e de banco de dados e variações entre plataformas, que pode resultar em instabilidade numérica e códigos não modulares, com difícil compreensão e adaptação. Estes fatores são prejudiciais principalmente quando trabalhamos com aplicações em ambientes computacionais de alto desempenho (HPC), os quais são frequentemente utilizados em análises ômicas [Di Tommaso et al. 2017].

Para ajudar a resolver alguns desses problemas, existem vários sistemas de gerenciamento de *workflows* para bioinformática: Galaxy [Goecks et al. 2010], Snakemake [Köster and Rahmann 2012], Bpipe [Sadedin et al. 2012], EGene [Durham et al. 2005] e o Nextflow [Di Tommaso et al. 2017].

O presente trabalho iniciou-se com um projeto da EMBRAPA-Soja relacionado a Seleção Assistida de plantas de soja para a resistência à Ferrugem Asiática através da tecnologia de sequenciamento *paired-end* de fragmentos amplificados em *PCR (amplicons)* em diferentes *loci*. Em sua fase final, foi aplicado um *pipeline* [Fresnedo-Ramírez et al. 2019] para realizar as análises dos polimorfismos presentes nestes *amplicons*. Contudo, devido à dificuldade de interpretação do código, programas não utilizados na análise, e de outros fatores que não garantem a reprodutibilidade das análises, foi realizada a reimplementação do código utilizando um gerenciador de *workflow*, o Nextflow, tornando o *pipeline* modular, escalonável e portátil, além de arquivos com o resumo de todo o processamento graficamente.

2. Material e Métodos

Para a reimplementação do pipeline, foi necessário identificar e entender os passos que foram implementados no *script* original escrito na linguagem Perl [Fresnedo-Ramírez et al. 2019]. Este realiza o processamento de dados produzidos pelo sequenciamento de *short reads* de DNA pela tecnologia Illumina, após PCR e multiplexação. Sua entrada é um conjunto de dois arquivos com extensão ".fastq"por amostra, uma pertencente à fita direta e outra à reversa do DNA. Através dele é possível a obtenção de diferentes haplótipos[1] referentes às regiões desejadas, a fim de explorar polimorfismos existentes.

A Figura 1 descreve os passos do pipeline: (i) é realizado o corte dos adaptadores utilizados durante o sequenciamento, além do descarte de leituras (sequências) com qualidade abaixo do especificado pelo usuário. Para o corte, é utilizado o *software* Trimmomatic [Bolger et al. 2014]; (ii) há a verificação dos pares das leituras, por tratar-se de sequenciamento *paired-end*, descartando pares que não possuem sobreposição entre si. Para encontrar as sobreposições, é aplicado o *software* Flash [Magoč and Salzberg 2011]; (iii) as leituras em cada arquivo são separadas de acordo com as sequências dos *primers*. Este pré-processamento é feito através de uma das ferramentas do FASTX-Toolkit, o Barcode Splitter [Gordon et al. 2010]; (iv) efetua-se uma divisão de acordo com cada uma das amostras e suas leituras. Também é utilizado o FASTX-Toolkit, porém com a ferramenta Collapser; (v) é realizada uma segunda divisão das leituras, porém agora de acordo com os *primers*, novamente aplicando o Collapser; (vi) são criadas duas tabelas para

[1]Grupos de sequências similares que apresentam os mesmos polimorfismos referentes a regiões desejadas

cada região de interesse (especificada por cada *primer*), uma com as sequências de cada haplótipo encontrado e o total de ocorrências, e outra com quantidade de haplótipos por amostra. Esta etapa é feita inteiramente no *script* do próprio *pipeline*; (vii) por fim, são criados arquivos fasta com os haplótipos mais recorrentes, havendo um alinhamento entre eles a fim de especificar as regiões discrepantes. Também são criadas duas tabelas com informações sobre os dois haplótipos mais recorrentes em cada amostra. Para a criação das tabelas, o desenvolvimento é no *script* do *pipeline*, e o alinhamento é feito através do *software* Clustal O [Sievers et al. 2011].

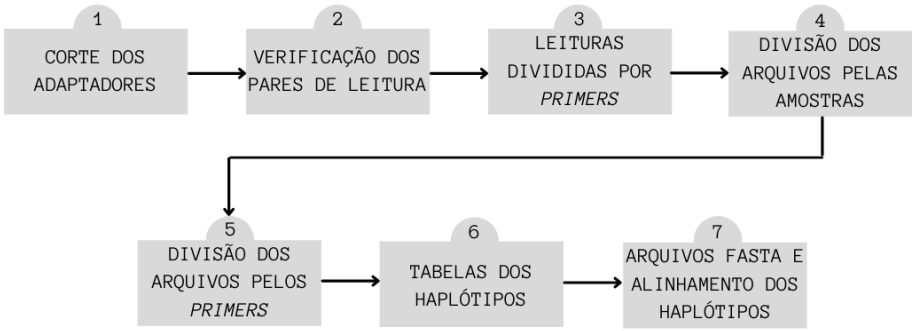

Figura 1. Fluxograma do *pipeline*

3. Resultados

Com a refatoração proposta, a média de linhas de código em cada etapa do *pipeline* foi reduzida de 139 para 37. Isto porque com o Nextflow a identificação dos arquivos de entrada é substituída pelos canais, e o processamento e utilização das linhas de comando são simplificados na escrita.

Por fim, por meio do Nf-core são gerados gráficos referentes à execução do *pipeline* e aos resultados. Há gráficos referentes a cada processo quanto a utilização de CPU, porcentagem de memória alocada, tempo de duração, quantidade de número de bytes de leitura e escrita, entre outros. Também, é produzida uma tabela com informações sobre cada execução realizada no *workflow* em relação a cada arquivo, permitindo a visualização dos comandos executados.

A Tabela 1 mostra o resultado de uma das saídas principais do *pipeline*. Para cada *primer* há um conjunto de haplótipos e a frequência em que ocorrem, além da quantidade destes em cada uma das amostras de entrada.

Tabela 1. Resultado de uma das saídas do *pipeline* refatorado

Primer	Haplótipos	Amostra
Ren1_SC47_6	1(0.75);2(0.17);4(0.08);	2/2:3841
Ren6_PN9_068	3(0.50);1(0.27);2(0.23);	./.:0
Ren7_PN19_018	1(0.67);2(0.33);	2/2:152
Ren7_VVin74	2(0.50);1(0.42);3(0.08);	2/2:144
Rpv12_UDV014	1(0.38);2(0.25);17(0.17); 11(0.08);21(0.08);3(0.04);	1/1:36
Rpv12_UDV370	1(0.58);3(0.33);5(0.08);	1/1:7976

4. Conclusão

O objetivo deste trabalho foi a refatoração de um *pipeline* voltado para a análise de *amplicons* provindos de sequenciamento genético. Foi possível organizar o código ao encapsular etapas mais trabalhosas ou com grande número de linhas, tornando-o mais legível e resultando em um código mais simples. Adicionalmente, o arquivo de entrada referente ao nome dos arquivos pôde ser suprimido, e a chamada aos programas não necessita mais ser editada no código-fonte. Além disto, os processos agora são executados paralelamente quando possível, devido à paralelização que o Nextflow provê automaticamente. Por fim, foi possível obter gráficos e tabelas referentes às etapa de execução. O trabalho encontra-se no GitHub do projeto (`https://github.com/aalinebini/ampliconseq`).

Referências

Bolger, A. M., Lohse, M., and Usadel, B. (2014). Trimmomatic: a flexible trimmer for illumina sequence data. *Bioinformatics*, 30(15):2114–2120.

Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., and Notredame, C. (2017). Nextflow enables reproducible computational workflows. *Nature biotechnology*, 35(4):316–319.

Durham, A. M., Kashiwabara, A. Y., Matsunaga, F. T., Ahagon, P. H., Rainone, F., Varuzza, L., and Gruber, A. (2005). Egene: a configurable pipeline generation system for automated sequence analysis. *Bioinformatics*, 21(12):2812–2813.

Fresnedo-Ramírez, J., Yang, S., Sun, Q., Karn, A., Reisch, B. I., and Cadle-Davidson, L. (2019). Computational analysis of ampseq data for targeted, high-throughput genotyping of amplicons. *Frontiers in plant science*, 10:599.

Garlan, D. and Shaw, M. (1993). An introduction to software architecture. In *Advances in software engineering and knowledge engineering*, pages 1–39. World Scientific.

Goecks, J., Nekrutenko, A., and Taylor, J. (2010). Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology*, 11(8):1–13.

Gordon, A., Hannon, G., et al. (2010). Fastx-toolkit. *FASTQ/A short-reads preprocessing tools (unpublished) http://hannonlab. cshl. edu/fastx_toolkit*, 5.

Köster, J. and Rahmann, S. (2012). Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522.

Magoč, T. and Salzberg, S. L. (2011). Flash: fast length adjustment of short reads to improve genome assemblies. *Bioinformatics*, 27(21):2957–2963.

Richards, M. and Ford, N. (2020). *Fundamentals of Software Architecture: An Engineering Approach*. O'Reilly Media, Inc.

Sadedin, S. P., Pope, B., and Oshlack, A. (2012). Bpipe: a tool for running and managing bioinformatics pipelines. *Bioinformatics*, 28(11):1525–1526.

Sievers, F., Wilm, A., Dineen, D., Gibson, T. J., Karplus, K., Li, W., Lopez, R., McWilliam, H., Remmert, M., Söding, J., et al. (2011). Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular systems biology*, 7(1):539.