Analysis of *de novo* assembled targeted capture sequencing data and extract the possible intron sequence from contigs

Zhifan Wu (Corporate with Van Shen)

Supervisor: Belinda Chang

April 5th, 2018

Introduction:

Biology scientists were sequencing DNA using capillary DNA sequencer until the completion of Human Genome Project (HGP). After HGP, genome sequencing has moved forward to whole-genome sequencing (WGS), which focus on the paired-end reads sequencing [1]. But genomes sequences generated from WGS are neither highly polymorphic nor highly repetitive [2]. Thus, more advanced sequencing method is required. Here comes the Next-Generation sequence.

The advent massively parallel sequencing technologies, the so-called next-generation sequencing method (NGS) [1,2], is the holy grail of biology research field. There are quite a few next-generation platforms [1] that are used world widely: the Roche/454 FLX (http://www.454.com/enablingtechnology/the-system.asp), the Illumina/Solexa Genome Analyzer (http://www.illumina.com/pages.ilmn?ID=203), and the Applied Biosystems SOLiD$_{TM}$ System (https://www.thermofisher.com/ca/en/home/life-science/sequencing/next-generation-sequencing/solid-next-generation-sequencing.html). These sequencers are all sought to produce high-throughput reads of short lengths at a moderate cost [2] and they help move biology research forward, especially in genomics, gene expression analysis, noncoding RNA discovery, SNP detection and many other sequencing-related areas [1].

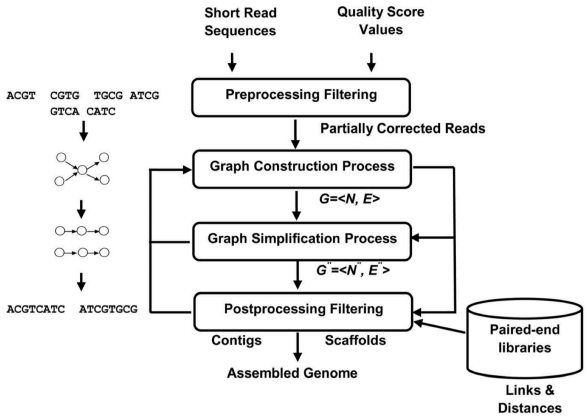

Figure 1. The four stages of Next Generation Sequence assembler [2].

Sara et al [2] construct a basic framework for next-generation genome sequence assemblers. The genome assembly pipeline (see Figure 1) contains the following four stages: preprocessing

filtering, graph construction process, graph simplification process, postprocessing filtering [2]. The pre- and post-filtering step is mainly used for error checking; thus, these two steps are usually performed in the different time in various NGS assemblers. Though the error correction step is performed at a different stage during assemblers, the concept behind this procedure is the same which reads with errors are infrequent and random. It utilizes the specific algorithm that counts the reads in the assembly pool to detect the reads with errors [2]. However, the performance of such algorithm will be extremely affected by the challenges of "high-frequency genomic repeats and non-uniform sampling of the genome, which lead to ambiguous results derived from multiple equal correction choices" [2].

Next-generation genome assembly begins with a set of short reads. And these reads are joined together to form contigs by the assembler. Then these contigs form longer contigs, named as scaffolds in the end [2]. There are two main methods used to generate contigs from reads: comparative approach and de novo approach. In the comparative approach, also known as reference-guided assemble, it uses existing contiguous sequences and sequence similarity between the target and reference species' genomes to assemble a genome [7]. De Novo assembly approach is especially useful in the case when reference sequences are unavailable or dealing with novel species [4]. This approach just joins input reads that overlap into contigs, then following the graph construction in assembler and then are separated to generate full-length RNA sequences [3]. Compare to the reference-guided approach, the de novo approach stands only on the input reads, and such advantage will advance the discovery of novel transcripts and splice variants in organisms that don't have well-annotated genomes [4]. De novo assemblers are slightly different in their algorithms implementation, Trinity and MIRA are currently used widely.

Intron sequences only exist in the Pre-mRNA and will undergo splicing and be removed to form mRNA. The sequences within the introns change far more rapidly than the exons during the evolution [5]. Nowadays, most biology scientists believe that both exons and introns acting a vital role in the splicing of the gene transcript. BS Jo et al [9] have discussed the functional benefits of introns, and they have concluded some direct and indirect roles of introns [9], for example, introns are involved in positive regulation of gene expression. Introns may be associated with mRNA transport and regulation of nonsense-mediated decay. Thus, introns sequences are quite worth looking into it.

Schott et al. [10] used a reference-guided approach to assemble the short reads into the complete coding sequences (exons). To characterize the contigs assembled by the first *de novo* assembly step, we examined the quality of the contigs file and whether these contigs contain complete exons sequences or intron sequences. Since the purpose of the contigs file is to assemble RNA sequence, there should be few introns reads and most exons reads. We mostly focused on extracting the intron sequences by analyzing the contigs file and the BLAST results [8]. We also examined whether these contigs contain other (contaminating) sequences and verified the introns sequences extracted from the contigs file.

Experiment Procedure:

We separated the experiment into three stages as shown in Figure 2 and I will talk about each stage below.

Stage 1: Utilizing BLAST [8] and Biopython package [11] to extract putative introns sequences from the contigs file.

In this stage, WE first fetched the contigs file assembled by Trinity [12] (file-name: Anolis_RSO6ex3_5_Trinity_NR.fasta) and the exons file from Schott's work [10] (file-name:

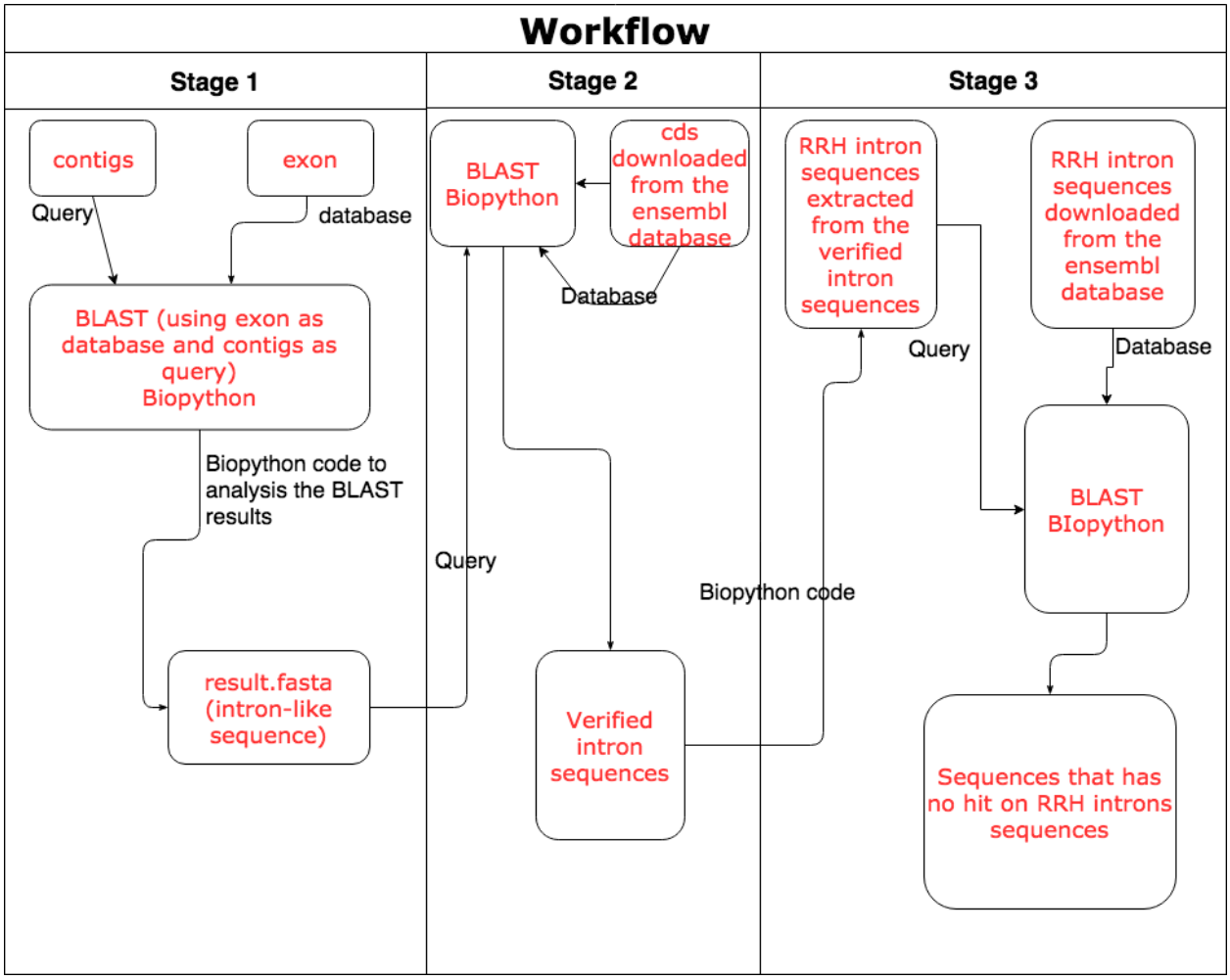

Figure 2. The 3 stages of the experiment.

Anolis_RSO6ex3_5_DeNovo_All_Trinity.fasta). Then we used local BLAST tool (note here using exons file as database and contigs file as query, since we wanted to find the intron sequences from the contigs file) in the lab computer with the following custom bash script (file-name: blast_contigs_aga_exome.sh). Then using custom python script (file-name: sequencing.py) to analysis the BLAST result. Thus, we have extracted the putative intron sequences from the contigs file. The concept behind my python script is:

1. Finding all the sequences that have hit with sequences in exons database, collect their names and hit sequences' lengths.

2. Using the above hit sequences' information, find all the hit sequences in the contigs file and find the regions that are not matched with the exons database. Collect these sequences into a file (file-name: result.fasta) and these sequences are the putative introns from the contigs file since they don't have hit with the exons database.

Stage 2: Verify the putative intron sequences extracted from the contigs file.

In this stage, we tried to verify the putative introns sequences extracted from the contigs file. Since in **stage 1**, we found out these introns sequences based on they had no hit with the exons database. Obviously, some contaminating sequences might also have no hits with exons database, thus would be included in our putative introns sequences file (result.fasta). Also, since the exons file we got was very small, we considered this exons file might contain part of the exons. To verify the introns sequences, we downloaded the complete coding sequences of Anole lizard from Ensembl [13] database. Then we utilized local BLAST tool (using cds.fasta as database and putative introns result.fasta as query) in the lab computer using the following custom bash script (file-name: blast_introns_aga_cds.sh). Then we analyzed the BLAST result using custom python script (file-name: verify.py). The concept behind this python script is similar with the previous one but with a few differences: Find all the sequences' names and their regions if they had hits with cds database and then delete these sequences. We extracted the remaining sequences and stored them in a file (file-name: verify.fasta).

Stage 3: Take RRH gene as an example to check the verified introns sequences quality.

In this stage, we wanted to further investigate our verified introns sequences. Thus, we took rhodopsin gene (RRH) as an example. We downloaded the intron sequences of RRH from Ensembl [13] database and extracted rrh introns sequences from our verified introns using

custom python script (file-name: find_introns_for_protein.py). Then we utilized local BLAST tool (using Ensembl rrh introns as database and our rrh introns as query) and analyzed the BLAST result using custom python script (file-name: verify.py).

Some technology we used:

BLAST: The Basic Local Search Tool (BLAST) [https://blast.ncbi.nlm.nih.gov/Blast.cgi] finds regions of local similarity between sequences [8]. This program compares the input sequence, could be nucleotide or protein, to the sequences databases in NCBI, and return the statistical significance of matches [8]. This tool allows scientists to infer the functional and evolutionary relationships about the input sequences with the similarity well-annotated sequences [8].

Result:

Figure 3. All the result files.

All the results are shown in figure 3. “result.fasta” is the putative introns sequences extracted from contigs file in Stage 1. “verified_introns.fasta” is the verified introns from Stage 2. “RRh_verify_introns.fasta” is the introns sequences from “verified_introns.fasta” from Stage 3, and the “verify_RRH_not_verify_introns.fasta” is the sequences that are from “verified_introns.fasta” and have no hits to the actual introns sequences from Ensembl database.

In Stage 1, the BLAST result showed only 2% (1065 sequences have hit, and total sequences are 52208) sequences in contigs have hit with the exons database. This result is surprising but consistent since the exons file we used is very small, it might only contain a certain part of exons (for example, only eye-gene) information. We cannot determine whether the sequences that have no hit with the exons database are exons or introns or other contaminating sequences, therefore we only tested the quality of all the hit sequences. And the analysis (Figure 4) of these sequences (1065) showed only 5 sequences (0.4%) have no hit, most sequences (858) have 2 introns. The data from Figure 4 showed these 2% sequences from contigs file is poor, with 94.4% sequences contains at least 2 introns region among the whole-length sequence.

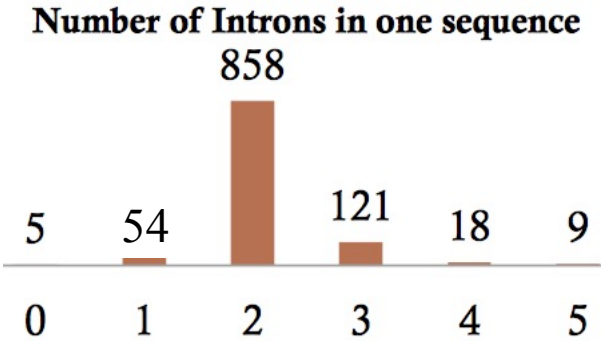

Figure 4. Analysis of hit sequences.

In Stage 2, we BLAST the putative introns (result.fasta) against with the cds (complete coding sequence) downloaded from Ensembl. The result was stored as verify.fasta. The analysis of this file revealed that 86% of the sequences in our putative introns has no hit with the cds database. This result confirmed that our putative introns' quality is good, it doesn't contain many exons. Such result supports that our approach to extract introns sequence from contigs file is reliable.

In Stage 3, we took RRH as an example to further test about our introns sequences quality. The analysis indicated that 7/9 of our RRH introns sequences have hit with the real RRH introns sequences downloaded from Ensembl, while 5/9 our sequences have whole-length hit. Furthermore, we calculated the nucleotide number of our hit sequences (3206) and the total nucleotide number of real RRH introns is 6352. Thus, our verified introns contain around 50% of

the real introns for RRH. Overall, such results suggested that our verified introns' quality is comparable.

Discussion:

One question remains in our project is that in Stage 1, the exons database we used is small which contains only parts of whole exons of Anole Lizard. Thus, we only analysis 2% of the contigs file while the remaining 98% sequences remain unknown, whether these sequences are introns or exons or other contaminating sequences. And the quality of the remaining sequences remains unclear. But we can further determine such issue with the proper exons database.

Since the final verified introns file (file-name: verify.fasta) still contains lots of genes, we only tested the RRH gene for a simple case investigation. The remaining introns could be tested using our Stage 3 method in the future. Another issue of our verified introns file is that we only excluded all the exons in Stage 2, there might be some contaminating sequences inside our verified introns file after Stage 2. These noise sequences affected the quality of our verified introns file and should be excluded in the future.

Though we only analyzed 2% of the contigs file, our results indicated that most sequences in this 2% contigs are intron sequences. Although 2% is quite small compared to the whole contigs file, we have concluded here that the quality of the whole contigs file needs to be tested before using it for the next step in de novo assembly.

Conclusion:

To summarize, we captured introns sequences from contigs file assembled from the first de novo assembly by utilizing BLAST [8], Ensembl [13] database and Biopython [11] packages. The quality of our extracted introns sequences file is highly dependent on the quality of contigs

file since eventually, all these introns sequences are sequences in contigs file. In our cases, the quality of introns sequences extracted from contigs file is quite comparable. And our results showed that our approach is consistent and reliable.

Acknowledgement:

I would like to thank Prof. Chang for supervising this project. Thanks to Ryan Schott for providing the initial idea about this project. Thanks to Matthew Preston for the help to find the data file in lab computers and for the help with the code challenges. Also, I would like to thank everyone in Chang's lab for discussions and advices for this project.

Reference:

[1] Mardis ER. Next-generation DNA, sequencing methods. Annu Rev Genomics Hum Genet. 2008;9:387–402

[2] El-Metwally S, Hamza T, Zakaria M, Helmy M (2013) Next-Generation Sequence Assembly: Four Stages of Data Processing and Computational Challenges. PLoS Comput Biol9(12): e1003345. https://doi.org/10.1371/journal.pcbi.1003345

[3] De Wit P, Pespeni MH, Ladner JT, Barshis DJ, Seneca F, Jaris H, Overgaard Therkildsen N, Morikawa M and Palumbi SR (2012) The simple fool's guide to population genomics via RNA-Seq: an introduction to high-throughput sequencing data analysis. Molecular Ecology Resources 12, 1058-1067.

[4] Bhawan P NGA data analysis.

[5] Fedorov A, Merican AF, Gilbert W. Large-scale comparison of intron positions among animal, plant, and fungal genes. Proc Natl Acad Sci U S A. 2002;99(25):16128–33

[6] LI, C., RIETHOVEN, J.-J. M. and NAYLOR, G. J. P. (2012), EvolMarkers: a database for mining exon and intron markers for evolution, ecology and conservation studies. Molecular Ecology Resources, 12: 967–971.

[7] Card DC, Schield DR, Reyes-Velasco J, Fujita MK, Andrew AL, Oyler-McCance SJ, et al. (2014) Two Low Coverage Bird Genomes and a Comparison of Reference-Guided versus De Novo Genome Assemblies. PLoS ONE9(9): e106649.

[8] Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. 1997;25(17):3389–402.

[9] Jo B. S. & Choi S. S. Introns: The Functional Benefits of Introns in Genomes. Genomics & informatics 13, 112–118, doi: 10.5808/GI.2015.13.4.112 (2015).

[10] Schott RK, Panesar B, Card DC, Preston M, Castoe TA, Chang BSW. 2017. Targeted capture of complete coding regions across divergent species. Genome Biol Evol . 9:398–414.

[11] Cock PA, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, Friedberg I, Hamelryck T, Kauff F, Wilczynski B and de Hoon MJL (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25, 1422-1423

[12] Grabherr MG, Haas BJ, Yassour M, Levin JZ, Thompson DA, Amit I, Adiconis X, Fan L, Raychowdhury R, Zeng Q, Chen Z, Mauceli E, Hacohen N, Gnirke A, Rhind N, di Palma F, Birren BW, Nusbaum C, Lindblad-Toh K, Friedman N, Regev A. Full-length transcriptome assembly from RNA-seq data without a reference genome. Nat Biotechnol. 2011 May 15;29(7):644-52. doi: 10.1038/nbt.1883. PubMed PMID: 21572440.

[13] Zerbino, D.R.; Achuthan, P.; Akanni, W.; Amode, M.R.; Barrell, D.; Bhai, J.; Billis, K.; Cummins, C.; Gall, A.; Girón, C.G.; et al. Ensembl 2018. Nucleic Acids Res. 2018, 46, D754–D761.