Chapter 22

Genome Sequencing and Analysis Methods in Chronic Lymphocytic Leukemia

Víctor Quesada, Miguel Araujo-Voces, José G. Pérez-Silva, Gloria Velasco, and Carlos López-Otín

Abstract

The genomic sequencing of chronic lymphocytic leukemia (CLL) samples has provided exciting new venues for the understanding and treatment of this prevalent disease. This feat is possible thanks to high-throughput sequencing methods, such as Illumina sequencing. The interpretation of these data sources requires not only appropriate software and hardware, but also understanding the biology and technology behind the sequencing process. Here, we provide a primer to understand each step in the analysis of point mutations from whole-genome or whole-exome sequencing experiments of tumor and normal samples.

Key words Bioinformatics, Genomics, Cancer, Next-generation sequencing, Leukemia

1 Introduction

Genomic studies on hematological neoplasias have provided important insights into the molecular mechanisms driving initiation and evolution of these diseases [1]. This is particularly the case of chronic lymphocytic leukemia (CLL), which has benefited enormously from Cancer Genomic initiatives aimed at elucidating the mutational landscape of this prevalent disease [2–4]. Multiple programs exist for the interpretation of high-throughput sequencing (HTS) data, including graphical [5] and commercial tools. The search for somatic mutations in paired tumor/normal samples can be roughly divided into three phases with dedicated tools: alignment of reads (frequently using BWA [6]), mutation discovery (using for instance GATK [7] or SomaticSniper [8]), and mutation characterization (using for instance VEP [9]). Although not exclusively, this type of analysis is mainly used with whole-genome (WGS) or whole-exome (WES) sequencing.

In the near future, HTS is very likely to become a fixture in research laboratories and clinical institutions. A foreseeable

Sami N. Malek (ed.), *Chronic Lymphocytic Leukemia: Methods and Protocols*, Methods in Molecular Biology, vol. 1881, https://doi.org/10.1007/978-1-4939-8876-1_22,

consequence of this trend will be the tight integration and standardization of every step of HTS analysis. While this will allow non-specialists to benefit from these powerful techniques, it also means that users will be separated from the analytical process. However, in our experience, the understanding of the challenges posed by HTS improves the interpretation of results, independently of which tools are used. For this reason, we provide here a typical analysis with Sidrón, our mutation discovery pipeline [10, 11]. To simplify this primer, only one sample will be considered, and the existing variants will be obtained.

2 Materials

All the necessary files to follow this pipeline are provided at http://github.com/vqf/sidron. These files are designed for Unix-based systems. Most executables are written in Perl, and therefore can be run in Windows-based systems. However, adapting the pipeline to Windows systems requires some programming experience. This tutorial includes small input (fastq and bam) files, so it does not require special hardware. Actual work with WES and particularly WGS files requires at least large memory storage capacities, and in practice also multiple CPUs and access to large RAM. As a reference, each full WGS file will require permanent memory in the order of hundreds of Gb.

In addition, other external programs are necessary to follow the procedure. The first part of the tutorial includes the alignment of the sequences, which is performed with BWA. Once the reads are aligned, Sidrón uses Samtools to extract information from the BAM files. Finally, one of the filtering procedures uses a second aligner, named BLAT. All these programs are public and free:

1. BWA installation files and procedures can be found at http://bio-bwa.sourceforge.net/.
2. To install Samtools, follow the instructions at http://www.htslib.org/. You will also need the corresponding Perl library (install distribution *LDS/Bio-SamTools-1.43.tar.gz* from CPAN).
3. To install BLAT, download the file https://users.soe.ucsc.edu/~kent/src/blatSrc.zip and follow the instructions within. Also download http://hgdownload.soe.ucsc.edu/admin/exe/linux.x86_64/faToTwoBit.
4. The example uses the human genome as a template. The corresponding FASTA sequence can be downloaded from ftp://ftp.ensembl.org/pub/release-91/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.toplevel.fa.gz. Download the file and decompress it with *gzip –d Homo_sapiens.GRCh38.dna.toplevel.fa.gz* at the download folder.

5. Index the reference genome with *faToTwoBit*. *Homo_sapiens.GRCh38.dna.toplevel.fa Homo_sapiens.GRCh38.dna.toplevel.fa.2bit*. Start a BLAT server with the script provided in the example (*./startHumanServer Homo_sapiens.GRCh38.dna.toplevel.fa.2bit [port_number]*). If *port_number* is not specified, the script will use 9006.
   The tutorial assumes that the executables *bwa*, *samtools*, *gfClient*, *gfServer*, and *faToTwoBit* are available from any folder. You can create softlinks to those executables in a folder inside the PATH environment or change the corresponding commands to include the path to the executable.

3 Methods

The files provided at http://github.com/vqf/sidron include two small input fastq files (*ex_1.fastq.gz* and *ex_2.fastq.gz*). You can see the format of this file by typing *zcat ex_1.fastq.gz | head* at the download folder. Although this file is small, its format is identical to that of the typical output from HTS machines.

3.1 Alignment

1. Create a custom alignment file. In the folder where Sidrón was downloaded, run *perl align.pl*. The script will ask for several pieces of data:
   (a) Basename: name of the output file (i.e., align_mreads).
   (b) Ref_genome: Path to the fasta file containing the reference genome.
   (c) Read_folder: Path to the folder containing the fastq files to align (single-end or paired-end). You can accept the default value pointing at the current folder.
   (d) File_pattern: Common part of the fastq file names (e.g., *ex*). If reads are paired, the script searches for files whose names match this pattern and afterward contain "_1" and "_2." This will identify the input files *ex_1.fastq.gz* and *ex_2.fastq.gz*.

   If all data are correct, the script will create an executable file called *align_mreads.sh*. Its contents automate the alignment process. At the beginning of the file, two lines provide the names of the executables for *bwa* and *samtools*. These can be changed manually.

2. Execute the custom alignment file with. */align_mreads.sh*. This will create a folder called *align_mreads* with two files named *align_mreads.sorted.bam* and *align_mreads.nodups.bam*. The second file contains the alignments with duplicates removed (**Note 1**). In this example, we will use *align_mreads.sorted.bam*.

3.2 Extraction and Calculation of Putative Variants

1. Enter the folder containing the bam file (*cd align_mreads*).
2. Use samtools to feed the pileup of the bam file into the perl script (*samtools mpileup -f* path_to_genome_fasta *align_mreads.sorted.bam | perl ../extract_mq.pl>mreads.mq*) (**Note 2**). The file *mreads.mq* contains all the positions in the alignment that show any change that may suggest that there is a variant. Most of these positions will in fact not contain variants, but sequencing or alignment errors. It is important to notice that for the rest of the positions there is no indication of variant. Even if other variants exist, the current sequencing data cannot find them.
3. Use Sidrón to assign a score to each putative variant (*perl ../sidron.pl mreads.mp ../table.hsh>mreads.sidron*) (**Note 3**). The output file contains three additional columns: genotypes considered, Sidrón score, and reserved (**Note 4**).

3.3 Filtering of Variants

1. Get positions with high S values (*../downstream_onesample.sh mreads.sidron*) (**Note 5**). This will create a file called *mreads.sidron.variants*.
2. Filter by nonlocal criteria (bad alignment, repetitive regions, etc.) with polyfilter (*perl ../polyfilter.pl mreads.sidron.variants align_mreads.sorted.bam > mreads.polyfilter*) (**Note 6**).
3. Repeat steps in Subheading 2, **item 3** and Subheading 3.1 with the filtered positions:
   perl ../sidron.pl mreads.polyfilter ../table.hsh > mreads.polyfilter.sidron
   ../downstream_onesample.sh mreads.polyfilter.sidron.
   At the end, we obtain a file called *mreads.polyfilter.sidron.variants* with the filtered variants.

3.4 Exploration of Variants

1. Create files with the genomic coordinates to explore. For instance, we can run *head mreads.polyfilter.sidron.variants>ex*. This will create a file named *ex* with the first ten variants. The only columns needed are the first and the second (chromosome and position), the script will not read the rest of the columns.
2. Create snapshots of the interesting positions with *perl ../snapshot.pl align_mreads.sorted.bam ex*. Each position will yield an html file that can be examined with any web browser (**Note 7**). The reference genome appears at the top in green, and each read appears aligned below. When the read base is the same as the corresponding base in the reference genome, we have points (read in the +strand) or commas (read in the −strand). High-confidence bases are in blue, and low-confidence bases are in red. Each read is clickable for more information (**Notes 8–9**).

4 Notes

1. Depending on the type of sequencing, we may want to remove duplicates (whole-genome, whole-exon) or not (pooled sequencing). In general, we want to remove duplicates when the read depth is relatively low and the probability of independently getting exactly the same DNA fragment more than once is low. If we sequence a part of the genome at very high read depth, this probability is much higher, and most duplicates will not be artifacts.
2. By default, *samtools mpileup* cuts the read depth at 250. If necessary, this limitation can be circumvented by adding the –d option (e.g., *samtools mpileup -d 1e8 -f* path_to_genome_fasta *align_mreads.sorted.bam...*).
3. The *table.hsh* file contains the expected rates of error for each base the sequencer reads (i.e., the probability that the machine reads A when in fact it should read C). Ideally, one should determine those error rates with orthogonal methods, such as genotyping microarrays. However, we have also developed specific methods to estimate those error rates directly from the reads.
4. The genotypes in the Sidrón file are given as a pair of bases N_1N_2. The first base is the most represented in the pileup, and the second base is the second most frequent base in the same position. Sidrón considers and compares two genotypes: homozygous (N_1 N_1) and heterozygous (N_1 N_2). The S score is defined as

   S=log10pcHetpcHz

   Here, $\log_{10}$ is the logarithm in base 10, *c* is the configuration (which bases were read and with which qualities), *Het* is the heterozygous genotype, and *Hz* is the homozygous genotype. Each probability is computed from the configuration and the error table (*table.hsh*). For instance, the probability that a configuration contains 3 As and one G given a *Hz* genotype is the probability that 3 bases were correctly read and in one the machine gave a G when it should have given an A.
5. The criteria to filter variants with *downstream_onesample.sh* are complex. An explanation of the cutoff points can be found inside the script. The cutoff values depend on the read depth of each position, as high-depth positions contain more information and allow finer distinctions. Each parameter in this file can be overridden by creating a file called *config.txt* in the run folder with the definitions. The operations and values are stored in a file called *log.txt*.

6. As the name implies, polyfilter contains several filters. The most important ones consider the probability that the variant occurs only at specific positions in each read and the possibility that the reads with the variants can be aligned to different positions in the genome. The first of those filters finds the position of each variant base inside its read. Then, it calculates the maximum distance between those positions (d). The probability that n bases chosen at random in a read of length l yield a maximum distance of d or less is

$$p(n,d)=(l-d)(d+1)^n-(l-d-1)d^n,\forall d\in(0,l-1)$$

If the computed probability for a position is lower than 0.2%, the configuration is considered spurious and filtered out. The second filter performs a BLAT alignment of each read containing a variant and filters the read out if it can be aligned with the same or higher quality at some other place in the genome. Since BLAT has a slightly different algorithm than BWA, this filter can improve the sensitivity to misaligned reads. On the other hand, this filter does not remove positions, only reads. This is the reason why Sidrón must be executed again after this step. Filtered positions are written to a file called *filtered_out*.

7. We developed *snapshot.pl* when few alternatives existed to examine a genomic position, and we still use it as a lightweight tool. Currently, other more sophisticated tools exist, such as IGV (http://software.broadinstitute.org/software/igv/).
8. This primer only explores how to get variants from a single sample. To compare tumor and normal samples, the procedure can be followed with these techniques. First, we obtain the *mq* file from the tumor sample. Then, we extract the corresponding positions from the normal sample with a different script (not provided). The Sidrón script then computes the S values for each position in both the tumor and normal sample. The rest of the procedure is similar to the one described above, with the only distinction that we will look for high S values in the tumoral sample (Het) and low S values in the normal sample (Hz).
9. We have only considered point mutations in this procedure, where Sidrón adds resolving power. For small insertions and deletions, other considerations make this technique insufficient. For a primer on how to find insertions and deletions, *see* http://samtools.sourceforge.net/cns0.shtml.

Acknowledgment

We thank J.M.P. Freije and X.S. Puente for helpful comments and advice. The Instituto Universitario de Oncología is supported by Fundación Bancaria Caja de Ahorros de Asturias. V.Q. is supported by Ministerio de Economía y Competitividad and Gobierno del Principado de Asturias, including FEDER funding. C.L.-O. is supported by grants from European Research Council (DeAge, ERC Advanced Grant), Ministerio de Economía y Competitividad, Instituto de Salud Carlos III (RTICC) and Progeria Research Foundation.

References

1. Ferrando AA, López-Otín C (2017) Clonal evolution in leukemia. Nat Med 23 (10):1135–1145
2. Puente XS, Beà S, Valdés-Mas R, Villamor N, Gutiérrez-Abril J, Martín-Subero JI et al (2015) Non-coding recurrent mutations in chronic lymphocytic leukaemia. Nature 526 (7574):519–524
3. Landau DA, Tausch E, Taylor-Weiner AN, Stewart C, Reiter JG, Bahlo J et al (2015) Mutations driving CLL and their evolution in progression and relapse. Nature 526 (7574):525–530
4. Valdés-Mas R, Gutiérrez-Abril J, Puente XS, López-Otín C (2016) Chronic lymphocytic leukemia: looking into the dark side of the genome. Cell Death Differ 23(1):7–9
5. Afgan E, Baker D, van den Beek M, Blankenberg D, Bouvier D, Čech M et al (2016) The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. Nucleic Acids Res 44(W1): W3–W10
6. Li H, Durbin R (2010) Fast and accurate long-read alignment with Burrows-Wheeler transform. Bioinformatics 26(5):589–595
7. McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A et al (2010 Sep) The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res 20 (9):1297–1303
8. Larson DE, Harris CC, Chen K, Koboldt DC, Abbott TE, Dooling DJ et al (2012) SomaticSniper: identification of somatic point mutations in whole genome sequencing data. Bioinformatics 28(3):311–317
9. McLaren W, Gil L, Hunt SE, Riat HS, Ritchie GRS, Thormann A et al (2016) The Ensembl variant effect predictor. Genome Biol 17 (1):122
10. Puente XS, Pinyol M, Quesada V, Conde L, Ordóñez GR, Villamor N et al (2011) Whole-genome sequencing identifies recurrent mutations in chronic lymphocytic leukaemia. Nature 475(7354):101–105
11. Puente XS, Quesada V, Osorio FG, Cabanillas R, Cadiñanos J, Fraile JM et al (2011) Exome sequencing and functional analysis identifies BANF1 mutation as the cause of a hereditary progeroid syndrome. Am J Hum Genet 88(5):650–656