WEEK 2 ASSIGNMENT

Data Systems in the Life Sciences – BIOL 51000 | Fall 2 2020

Christina Morgenstern

I. MAPPING SEQUENCES

1. How to begin mapping?

High-throughput sequencing (HTS) generates a vast amount of short sequencing reads from a variety of experiments, such as ChIP-seq, DNase-seq, RNA-seq and many more. In order to make sense of these data, the reads are aligned directly to a genome (or transcriptome) with the help of indexing. This approach is referred to as genome mapping. Before performing the mapping, the sequencing reads need to undergo a quality control step. The reads are assessed for their quality scores and low-quality reads are discarded. FastQC is a useful tool that helps in assessing quality parameters of the sequences [1]. FastQC not only reports average quality scores but also GC-content, average sequence length and many more parameters. Also, the adapters needed for building the sequencing library need to be trimmed before mapping. Several tools exist for this purpose, like Trimmomatic [2] or Cutadapt [3]. If the library was amplified extensively using Polymerase Chain Reaction (PCR) before sequencing, then removing PCR duplicates is also recommended as preprocessing step.
Depending on the type of experiment, the researcher has to decide whether the reads should be mapped to the full genome (genome mapping) or transcriptome (transcriptome mapping) of a reference species.
When mapping reads to a reference genome or transcriptome the researcher doesn´t need information about the transcribed regions or how exons are spliced together. Mapping allows for the discovery of new and unannotated transcripts [4].

2. How Python can be used to control the process of mapping

Python´s HTSeq library provides a framework for processing HTS data [5]. After installation, the HTSeq library can be imported into Python and used for a variety of analyses including getting statistical summaries about base-call quality scores and coverage data. The library also includes parsers for common file formats for a variety of input data [5], [6]. This includes FASTQ files from sequence reads, SAM files for the genome alignment results and GFF files which contain genome annotation information [7]. The sequence information is stored and accessed using the `GenomicArray` and `GenomicInterval` object classes.
Reads from an HTS experiment come in the form of FASTQ files which can be imported into Python using the `FastqReader` function and storing it in a file object. Looping through this object allows to extract each sequence record after the other. The sequence reads also contain quality scores which can be accessed and the mean quality scores plotted using the Matplotlib library [7].
Genome sequences for alignment are stored in the FASTA format and can be downloaded either through a web browser, via the command line or using a Python script. Several databases have genome files stored such as NCBI, ENSEMBL and UCSC.
Using the file transfer protocol (FTP), a Python function is able to fetch genomic data. Initially, the FTP-root and the link to the genome are encoded into variables `FTP_ROOT` and `GENOME`. Import of the libraries `os, gzip` and `fnmatch` provide functionality for operating interface

communicating with the remote server, dealing with compression and filename matching. One can build in a try and except statement to control for an `ImportError` when reading the file fails. When specifying a new function that is capable of downloading a genome file, the function takes two arguments, the remote file to acquire and the path to the file to be saved locally including a file name and the directory. Using the `urlopen()` function, the remote file is accessed and the `read` function reads the file into a data object. On the other end, the local file needs to be opened using the open function and the genome data written to the file using `write()` function. In the end the file is closed using `close()`. In order to follow the progress of the downloading a message can be printed while the function is executing [7].

An external program, like BOWTIE [8], BWA [9] or STAR [10] is then used to perform the mapping of the short reads to the genome.

3. How to index a genome

Before mapping the sequencing reads to the reference genome or transcriptome, the reference needs to be indexed. Indexing allows a more efficient and faster process of looking up sequences in the genome. This process takes a substantial amount of time but only have to be done for a new genome species or release thereof.

The Python `subprocess` module holds tools that allow to `call()` commands to run the indexing and the subsequent alignment using an external program. Importing the `call()` function from this module and specifying the path to the alignment program is done at first. The `indexGenome()` function specifies the genome name, the file name, the output directory and the quiet and pack arguments which can suppress text format and compress the data, respectively. When reading the genome FASTA file, a check can be built in to control for compressed data files.

Indexing can take a substantial amount of time, especially for large genomes like the human genome. Also, some aligners can take up a lot of memory while indexing, like STAR [10].

4. How to align “reads to a genome

The choice of sequence aligner depends on the type of experiment and the questions being addressed. For example, if the aligner needs to map reads in a splice-aware manner, a certain software needs to be chosen. Important factors in the alignment procedure are the length and the quality of the sequencing reads as well as how mismatches are dealt with. An alignment function will have as inputs at least the indexed genome, the sequencing reads and the name of the output file. Further arguments can adjust read quality and trimming parameters as well as set the maximum number of mismatches and the maximum number of tolerated alignments for a read. A full list of argument options can be seen in the manual of the respective aligner. Some aligners also allow to specify the number of CPU cores and thus allowing to maximize efficiency.

Python can be used to write a wrapper function that calls an external aligner software. For that a `genomeAlign()` function is defined that takes the necessary and available arguments and their specifications. Depending on the sequencing platform used and thus also different quality metrics the function can be customized.

References:

[1] ‘Babraham Bioinformatics - FastQC A Quality Control tool for High Throughput Sequence Data’. https://www.bioinformatics.babraham.ac.uk/projects/fastqc/ (accessed Nov. 07, 2020).

[2] ‘USADELLAB.org - Trimmomatic: A flexible read trimming tool for Illumina NGS data’. http://www.usadellab.org/cms/?page=trimmomatic (accessed Nov. 07, 2020).

[3] M. Martin, ‘Cutadapt removes adapter sequences from high-throughput sequencing reads’, *EMBnet.journal*, vol. 17, no. 1, Art. no. 1, May 2011, doi: 10.14806/ej.17.1.200.

[4] EMBL-EBI, ‘Read mapping or alignment | Functional genomics II’. https://www.ebi.ac.uk/training-beta/online/courses/functional-genomics-ii-common-technologies-and-data-analysis-methods/rna-sequencing/performing-a-rna-seq-experiment/data-analysis/read-mapping-or-alignment/ (accessed Nov. 07, 2020).

[5] S. Anders, P. T. Pyl, and W. Huber, ‘HTSeq—a Python framework to work with high-throughput sequencing data’, *Bioinformatics*, vol. 31, no. 2, pp. 166–169, Jan. 2015, doi: 10.1093/bioinformatics/btu638.

[6] ‘A tour through HTSeq — HTSeq 0.12.4 documentation’. https://htseq.readthedocs.io/en/master/tour.html (accessed Nov. 07, 2020).

[7] T. J. Stevens and W. Boucher, ‘Python Programming for Biology: Bioinformatics and Beyond’, *Cambridge Core*, Feb. 2015. /core/books/python-programming-for-biology/61762A9F672FDD8B2DD3FFF8773027B2 (accessed Nov. 07, 2020).

[8] B. Langmead, ‘Aligning Short Sequencing Reads with Bowtie’, *Curr. Protoc. Bioinform.*, vol. 32, no. 1, Dec. 2010, doi: 10.1002/0471250953.bi1107s32.

[9] ‘Burrows-Wheeler Aligner’. http://bio-bwa.sourceforge.net/ (accessed Nov. 07, 2020).

[10] A. Dobin *et al.*, ‘STAR: ultrafast universal RNA-seq aligner’, *Bioinformatics*, vol. 29, no. 1, pp. 15–21, Jan. 2013, doi: 10.1093/bioinformatics/bts635.