Impact of minimap2 parametrization on mapping rate and functional Annotations of Metagenomics samples

Introduction

Metagenomics samples are characterised by a high biodiversity and low purification. Metagenomics samples are taken to answer two main questions:

1.) Sample composition: abundance of species in the sample
2.) Sample function: what are the main tasks conducted by the species present in the sample

Answering both questions is approached by conducting sequencing experiments and downstream analysis. This project focuses solely on the second task. In sequencing workflow the sequncing experiment is followed by the steps base calling and mapping. Based on obtained mappings functional annotation can be conducted. There are several possible methods of approaching functional annotation. One of this options is querying pathway databases, such as KEGG. Especially for metagenomics samples, that contain less studied organisms these yields only few hits. Therefore, we are focusing on GO functional annotations and EC numbers. Both annotation types can be found in genbank files. We aim to find all annotations available for our mapping results.

Next to the characteristic of fewer annotations for less known species, nanopore sequencing of metagenomics samples itself is challenging already. Base calling models are trained on purified human samples, which might cause troubles in accuracy of reads, which consequently, impacts the mapping step. We aim to experiment with parameter settings of the mapping tools to see whether the described problems can be approached by different parameter settings. We want to find parameter settings, that change the mapping behaviour from prioritzing alignment scores to prioritizing better known genomes if possible. This approach should definetly help to increase the number of found annotations but might as well help to overcome the problems arising from lower accuracy. Penalizing low alignment scores less allows mapping of noisier reads.

This project yields big data and therefore requires specific methods to handle. Sequencing results (base calling already conducted) contain couple of 100.000 reads, which need to be mapped against the collection of all known (491.594) bacteria genomes [1]. The mapping step results in a file with mapping information for each read, regardless whether or whether not the read was mapped. For each original read the genome + annotation (genbank file) needs to be downloaded and analysed for the genes spanned by a read and the annotations for these genes. Sequential processing would be time consuming. Furthermore, the reads information is independent from each other meaning, several reads could be processed at the same time without loosing any kind of information.

Material

A read file was provided in fastq format. The reference genomes were available as fna format or as pre-index mmi file. The mapping algorithm used by minimap2 is based on the seed and extend approach, this parts the algorithm in two main parts: indexing the reference data conduct seed and extend alignment for the reads onto the indexed references. Using a pre-indexed files saves time, which is the reason I used the .mmi file.

Methods

The mapping of the reads onto the pre-indexed references sequences was executed in parallel fashion by exploiting minimap2's multithreading option. The process was split up into 16 processes and scheduled as SLURM task. The mappings were produced with nanopore sequencing standard parametrization (run 1: -ax map-ont -t 15) and an alternative parametrization (run 2: -ax map-ont -t 15 -k 12 -B 2) for comparison purpose.

Annotations were extracted, processing the mapping files using server client model setup using pythons multiprocessing library together with queue objects. At first, one server and 4 clients with 4 peons each were used. For the final two runs, the mapping files where sectioned into 4 parts per file. Each run was conducted with 4 servers, from which each server processed one partition of the mapping file by instructing 4 clients with 4 peons each. In total 8 server with 32 clients and 128 peons were used to process the two mapping files (4 server, 16 clients, 64 peons for each file). From the mapping file gi identifier was extracted to obtain the genbank files. Start position of the read mapping together with the length of either the raw read sequence or if missing length of the clipped cigar[1] string else 1 were used to get the stop position of a mapped read. Genbank files were scanned for the read start – stop range, to obtain annotations for spanned reads. The results (Annotations and their counts) were appended to a temporary file by each peon after completing a single job (one read). After completing a run, all temporary results were collected combined and written to final result files and final visualisation plots were produced and written to file.

Results

To find optimal parameter settings for metagenomics samples two different parametrizations of minimap2 were used. The mapping rate for standard parametrization (run 1) and an alternative (run 2) were not directly measured but could be determined from the original mapping files by counting the lines that have a gi-id (mapped) vs. the ones that do not have a gi-id (unmapped).

It was postulated that alternative parameter settings should as well result in more annotations because of prioritzing mapping to well known organisms over high alignment scores. Table 1 shows the number of unique annotations found per extracted annotation per run as well as the summed counts of all annotations found per run. Run 2 shows a slight increase in annotations.

Table 1: Annotation count statistics

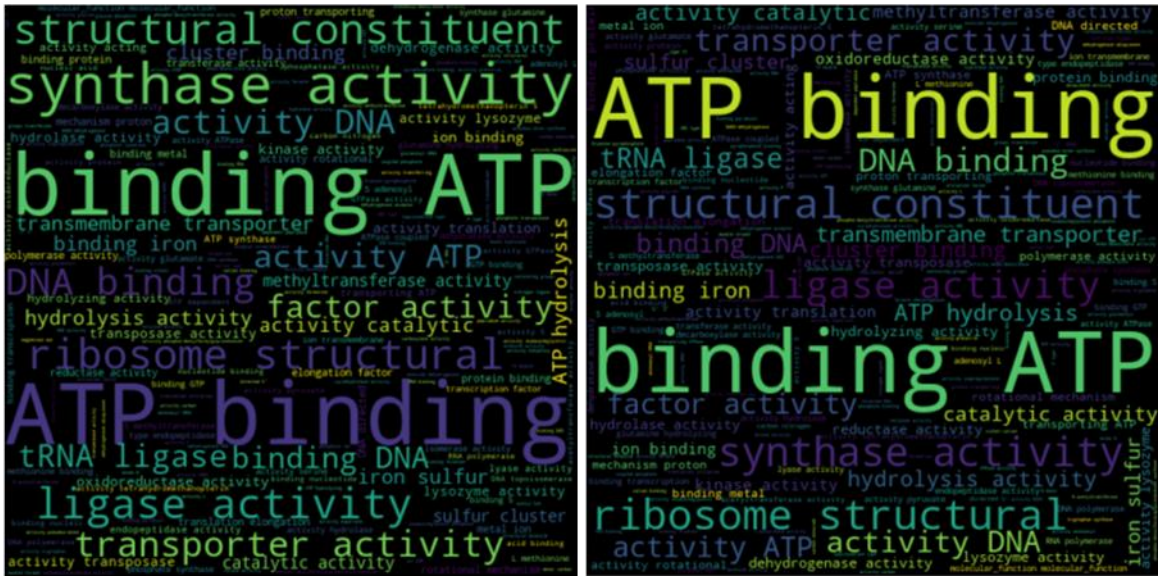
	Run 1		Run 2	
	Total unique annotations	Summed counts of annotations	Total unique annotations	Summed counts of annotations
GO numbers	1.294	533.652	1.312	586.668
GO descriptions	1.310	533.652	1.328	586.668
EC numbers	1.002	219.084	1.016	238.547

Further investigations of annotations comparing the two runs were conducted creating wordclouds for the GO descriptions to highlight the most abundant annotation terms. Figure 1 shows the wordclouds created for run 1 (subplot A) and run 2 (subplot B). The two most abundant terms are the same for the

[1] More on cigar string in appendix 1

two runs. For the second most abundant terms slight differences exist. Note, that the most abundant terms point to functions that are necessary for cell survival.

A) B)



The top 20 most frequent GO numbers and EC numbers are shown in table 2 and 3. The complete tables for GO and EC numbers can be found in the output directory together with the mapping files and log files. As table 1 already suggested, more GO annotations are present in genbank files than EC number annotations.

Table 2: top 10 GO numbers for run 1 and run 2

Run 1		Run 2	
GO number	Counts	GO number	Counts
GO:0005524	35.748	GO:0005524	38.781
GO:0003735	29.413	GO:0003735	37.583
GO:0003677	15.018	GO:0003677	16.404
GO:0016887	14.893	GO:0016887	15.898
GO:0003824	14.289	GO:0003824	15.221
GO:0004803	8.941	GO:0004803	9.018
GO:0003796	7.646	GO:0003796	7.642
GO:0051536	7.201	GO:0051536	7.482
GO:0051539	6.718	GO:0003746	7.302
GO:0003746	6.418	GO:0051539	7.172

The meaning per EC number was added to table 3. Comparing these results to the wordclouds (figures 1) suggestes, that both annotation systems point towards similar functional attributes of genes. Generally all highly abundant annotations point towards cell maintenance task.

Tabelle 3: Top 10 EC number for run 1 and run 2

Run 1			Run 2		
EC number	Meaning	Counts	EC number	Meaning	Counts

7.1.2.2	H(+)-transporting two-sector ATPase	5.882	7.1.2.2	H(+)-transporting two-sector ATPase	6.968
2.1.1.86	N(5)-methyltetrahydro-methanopterin--coenzyme M methyltransferase.	3.950	2.1.1.86	N(5)-methyltetrahydro-methanopterin--coenzyme M methyltransferase.	3.918
2.7.7.6	DNA-directed RNA polymerase	3.078	2.7.7.6	DNA-directed RNA polymerase	3.677
3.1.25.-	Hydroxymethylglutaryl-CoA hydrolase	2.523	3.1.25.-	Hydroxymethylglutaryl-CoA hydrolase	2.897
2.7.7.7	DNA-directed DNA polymerase.	2.337	1.6.5.9	NADH:ubiquinone reductase (non-electrogenic)	2.589
5.6.2.2	DNA topoisomerase (ATP-hydrolyzing)	2.210	6.3.5.5	carbamoyl-phosphate synthase (glutamine-hydrolyzing)	2.582
1.6.5.9	NADH:ubiquinone reductase (non-electrogenic)	2.144	2.7.7.7	DNA-directed DNA polymerase.	2.542
6.3.5.5	carbamoyl-phosphate synthase (glutamine-hydrolyzing)	2.051	5.6.2.2	DNA topoisomerase (ATP-hydrolyzing)	2.500
3.6.4.12	DNA helicase	1.982	4.2.1.20	tryptophan synthase	2.171
4.2.1.20	tryptophan synthase	1.950	3.6.4.12	DNA helicase	2.064

Technical Discussion

Minimap2 offers plenty parameter tuning option as well as output format options. I have received all reads processed by the tool. It would have been possible to only save mapped reads (parameter: --sam-hit-only)[2], which would have decreased the file size tremendously, concesquently resulting in less tasks for the Annotation extraction. For future usage, excluding unmapped reads could speed up the annotation routine.

For the annotation extraction routine I have considered two different set ups to conduct the task. One option would have been scheduling not only the mapping step but as well the extracting of annotations using SLURM. In this case SLURM would have received the server and clients with pre-defined workload per client as jobs. I have decided against this approach, due to the heterogenous runtime per task. In my approach a reference file is downloaded only if it is not present in a chache already. Furthermore, not every read was mapped. In both cases the processing time is significantly lower. For that reason I decided to keep the system dynamic, so that each peon can pick up a new job when completed the current job. I have kept every job to be exactly one read. Variation in job size by partitioning the mapping file looping per peon through the assigned lines conducting the same instructions as in the one line apporach would have been possible (function was developed see test folder). However, the one line job takes already quite some time, so the reporting back in one line packages does not lead to a bottleneck. Processing the mapping file as a whole took a lot of time, therefore I split the mapping file into several sections, which decreased the runtime per run from one week to two days.

At first, results were collected by the server and written to file once the run got completeed. This approach is risky for several reasons:

1.) if something goes wrong and the run gets terminated or stuck, for which the chances are increasing with increasing size of data, all data that has already been processed is lost.
2.) It is difficult to meet the termination criteria for the server to shutdown clients and itself.

For this routine there are several things that could go wrong such as invalid formats within the mapping file, missing genbank files or malformatted genbank files. Therefore, optimizing the system towards robustness should be the primary goal. I have done this by writing every result immediately to a temporary file maintained by the peon as well as being overly generous with exceptions, to minimize the risk of a job failing and causing problems with meeting termination criteria. For the future I should have considered, saving the results on the server side as well. Furthermore, I should have stored my logging elements instead of just printing them. This would allow to find failed jobs which could be fed in a "repair-run" to the system.

Research Discussion

While metagenomics studies often use minimap2 with standard parametrization, such as the study conducted by Sanderson et al. [3], there are other configurations reported. Parameter settings have been chosen basing on Oppelts settings choices. He decreased the -k paramter to higher the tolerance of noise by lowering the k mer size which leads to less specific minimizers. I decided to lower the penality for mismatches slightly. Both adjustments aim to allow less specific mappings, which might result in mapping to better known species. However the prioritization of model organisms could not be chosen directly by parameter tuning. Whether the chosen parametrizatin indeed shifted the mapping preferences remains unclear from the results collected. If further insight is required, a comparative frequency analysis of gi-ids between the two mapping runs could be considered. The second postulate of increasing the annotations by mapping to better annotated species does not really hold. The increase in annotations is only a small effect, which might not be due to the desired mapping to well-annotated species but just due to more mappings in general.

The chosen method for extracting annotations leads to an undercount of annotations in cases where neither the read sequence nor the cigar string is available. In these cases the read length is set to one. For these exceptional cases, the genes that would be spanned by the read are not considered, only the gene the read is located at. Inversely, an overcount arises in cases, where a read does not span a gene completely, regardless all annotations of the gene are counted. The assumption was, that such cases would be rare and therefore negligble. Since I have not collected data on the frequency of such cases it remains unclear if the assumption is correct.

The annotations and their counts collected, do not reveal any interesting result in the sense of special pathways or functions being revealed. Further analysis could be conducted to obtain the most interesting annotations for a sample, by executing enrichment analysis for instance. These analyses can be based on the final annotation count files produced by the implemented pipeline.

I have implemented a functioning and efficient pipeline, which allows further parameter tweeking to optimize the mapping step as well as being able to monitor changes by the produced result statistics and visualisations.

References

[1] All Bacterial genomes at NCBI: https://www.ncbi.nlm.nih.gov/genome/browse#!/prokaryotes/

[2] minimap2 man page: https://lh3.github.io/minimap2/minimap2.html

[3] Sanderson et al.: https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-018-5094-y#ref-CR16

[4] Jan Oppelt: https://janbio.home.blog/2020/07/14/minimap2-transcriptome-mapping/

Appendix 1

The cigar string encodes the mapping of a read in that sense, that each base is represented with encoding M for match/mismatch, I for insertion, D for deletion and S or H for soft or hard clipping. Extended cigar string format can be obtained, then M is replaced by X for mismatches and = for matches. The cigar string is a more condensed representation than the actual read sequence since neighbouring bases with the same encoding are summed and the sum is added to the cigar string as the prefixes for the encoding they refer to. Clipping is applied to the ends of the reads to increase the alignment score. The start position given for a mapped read is the first not clipped base of a read. There are R Bioconductor packages available to extract information from cigar strings. My own python script is reduced in functionality to calculate the read length.