Published online 20 12 2020

Nucleic Acids Research, 2020, Vol. xx, No. xx ***1–2***
doi:10.1093/nar/gkn000

SeqEMA: A seq implementation of EMerAid mapping

Rahul Nawani [1,*], **Daniel Kinahan** [2 *]

ABSTRACT

New advancements in long read sequencing have allowed us to read areas of the genome that were previously unobtainable. One of these advancements is 10x's Genomic Chromium sequencing, which consists of short, 16bp barcodes that refer to a long read(1). These barcodes can be aligned using traditional NGS alignment algorithms, but this approach is inefficient and produces high error rates(2). SeqEMA is a seq reimplementation of EMerAid, a barcode aligner for short-read, 10x-genomic data. SeqEMA attempts to replicate the preprocessing steps of EMA, which sorts and bins barcoded reads for mapping, with increased efficiency and abstracted Pythonic code.

INTRODUCTION

Recent improvements in the throughput of the genome sequencing process combined with the drive for evolution of NGS short-read sequencing data have given rise to "third generation sequencing" techniques. Where most sequencing types use long reads with high error rates, 10x genomics have created a type of sequencing in which the long read (150,000bp) is referenceable by a 16bp barcode that is sequenced as a short read/cite1. With this technique, long reads are achievable with comparable accuracy to NGS short read data with a similar price point, and offer many advantages. The resulting reads form read clouds - groups of reads sharing the same barcode that originate from the same long read fragment(3).

EMA is a tool that performs probabilistic interpretation of these clouds. It uses a latent variable model (a statistical model that relates a set of observable variables to latent or inferred variables) to dynamically map reads to their probable alignment locations and statistically bins them rather than choosing the most similar read for a location.

EMA is originally written in C and C++. For this implementation we have chosen seq for its ability to write concise, optimized programs in computational genomics research.

METHODS

There are four steps to the EMA preprocessing stage. They follow similar steps to a 10x Genomics' WGS software suite, Long Ranger(4). SeqEMA begins by reading in barcodes from whitelist and storing them in a library. After which it reads

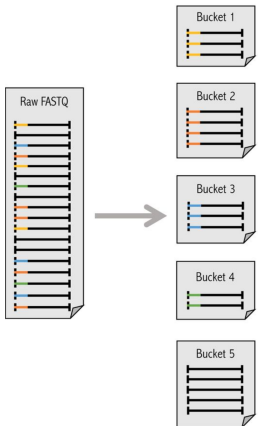

Figure 1. FASTQ processing in EMA and SeqEMA

FASTQ paired-end reads from an input file and creates two items: a dictionary with barcodes that match the whitelist and a list of records that mismatch to be corrected.

Once complete, it corrects reads by checking barcodes with a hamming distance of 1 for each read. If multiple reads are within the hamming distance, it will choose the barcode with the highest probability (priority) and quality score. Each read is then linked to its barcode in a dictionary for faster processing. From there, barcodes are sorted into buckets with approximately N/B reads per bucket where N = total number of reads used after correction and B is the number of buckets. Each barcode in a bucket is then assigned its barcode, name, sequence, and quality score with the first 16 bases removed to account for the barcode from each sequence

Output

The purpose of this preprocessing step is to create buckets that contain a range of barcodes that enables us to align the reads in parallel. A "cloud" is constructed by grouping nearby reads with the same barcode. In the EMA alignment process, a latent variable model is employed to optimize the cloud construction. This grouping of barcodes with the alignment process is an approach that is similar to Lariat.

*To whom correspondence should be addressed. Tel: Email: danielkinahan@uvic.ca, rnawani27@gmail.com

© 2020 The Author(s)
This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/2.0/uk/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

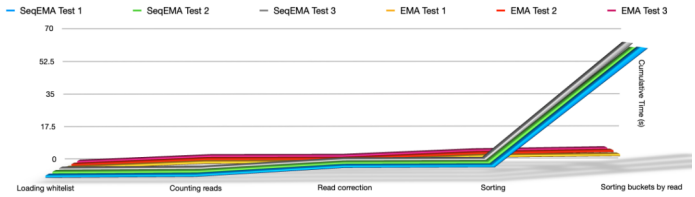

Figure 2. Comparison of SeqEMA and EMA pre-processing

	SeqEMA Test 1	SeqEMA Test 2	SeqEMA Test 3	EMA Test 1	EMA Test 2	EMA Test 3
Loading whitelist	3.1343	3.1906	3.1617	1.9000	1.9000	1.9000
Counting reads	0.3155	0.3226	0.3022	2.8000	2.8000	2.8000
Read correction	3.9770	3.9046	3.9479	0.0100	0.0100	0.0110
Sorting	0.1361	0.1336	0.1363	2.8000	2.8000	2.8000
Sorting buckets by read	54.1719	53.8962	56.0549	0.6030	1.1040	1.0030

Figure 3. Computational Time Comparison of SeqEMA and EMA pre-processing

RESULTS

SeqEMA runs significantly worse than its counterpart. Over 3 tests on the same 50 000 read FASTQ file, using a 6-core/12-thread processor, SeqEMA took on average 53 seconds longer to complete. The largest performance difference between the two comes from sorting the buckets by their reads from barcodes, which takes 54x longer or 500x longer without parallelization. This result is expected given the scope of our project and the time constraints, as well as general familiarity with the topic.

Data preprocessing in EMA consists of extracting the barcode from the read sequence, correcting the barcode errors based on quality scores and a list of known barcode sequences, and grouping reads by barcode into "barcode buckets" to enable parallelism during alignment. EMA does H2 correction as well as H1 correction where SeqEMA only does H1 correction. However we did not ignore the barcode if the quality score was below a threshold provided. In EMA the mapping of barcodes was done in an unordered map, which was later sorted to an ordered map. In SeqEMA it was done using a dictionary and sorted in place with iteratives which caused a long computational time

FUTURE WORK

Much more work needs to be done to make this a viable alternative. Performing H2 corrections to complete the scope of the EMA functions is top priority. Beyond this, there are still many optimizations that can be done to improve the speed of SeqEMA.

As previously discussed, the computational time of SeqEMA is significantly slower than the original EMA pre-processing pipeline. In order to achieve similar results the logic behind mapping each barcode in a bucket to a read must be improved. An unsorted map was used in the original sorting, however SeqEMA uses a dictionary to map each barcode and iterates through each read to sort the buckets. This can be improved by using an unsorted map which is sorted later similarly in the original concept. Learning seq as a pythonic language can hinder ones ability to learn the nuances and new framework of the language. As such, I believe we could have taken better advantage of the pipeline functionality that seq offers in our project to further optimize SeqEMA.

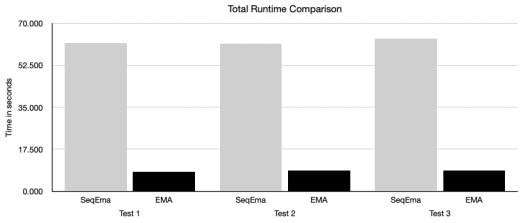

Figure 4. Comparison of total runtime

CONCLUSION

With SeqEMA we tried to re-implement the preprocessing steps of EMA in a simpler language with comparable speeds. Unfortunately, due to timing constraints we were unable to provide full functionality of EMA nor the same runtime speed. We did, however, achieve much higher code abstraction, and the program is significantly shorter than the same pieces written in C. This tool and others tailored to barcoded long read data will be the future of genomics and the importance of accurate mapping of these genomes is tantamount.

REFERENCES

1. Goodwin, S., McPherson, J. & McCombie, W. (2016) Coming of age: ten years of next-generation sequencing technologies. *Nat Rev Genet* https://doi.org/10.1038/nrg.2016.49
2. Shajii, A., Numanagić I., Berger B. (2017) Latent variable model for aligning barcoded short-reads improves downstream analyses *Cold Spring Harbor Laboratory* https://doi.org/10.1101/220236
3. Serafim's Lab Read Cloud Technologies https://www.serafimb.org/read-cloud-technologies.html
4. 10x Genomics (2017) What is Long Ranger https://support.10xgenomics.com/genome-exome/software/pipelines/latest/what-is-long-ranger/