Manual for the version 1.15

Contents:

Introduction

Elloreas is a targeted genome assembler, which means it takes a starter and then iteratively extends it with sequencing reads. This starter may be a contig from another assembly or just a random read. The principle of operation of Elloreas is similar to the principles of many other similar tools, like TASR, Mapsemsler2, NOVOPlasty or DistributedNucleatingAssembler. However, while those assemblers are designed for short reads, Elloreas was created for long reads, produced by sequencing machines of PacBio and Oxford Nanopore.

I created Elloreas to assemble the mitochondrial genome of Fagopyrum esculentum. A feature of this genome is that it consists of 10 circular chromosomes. Moreover, they can recombine, merging with each other or splitting into several smaller chromosomes. Multiple alternative forms of chromosomes coexist within a single plant, this is called "structural heteroplasmy". You can read more about this in an upcoming paper "Mitochondrial genome of Fagopyrum esculentum and the genetic diversity of extranuclear genomes in buckwheat" where Elloreas is described. I thought that Elloreas may be useful for other bioinformaticians, this is why I deposited it on GitHub.

How to run Elloreas.

A simple version, which will often be enough:

```
perl elloreas.pl --reads reads.fastq --starter starter.fasta
```

A slightly more complicated version:

```
perl elloreas.pl --reads reads.fastq --starter starter.fasta --
sequencing_technology oxford_nanopore --
minimum_length_of_aligned_read_part 8000
```

For a full list of parameters with suggestions on how to use them run

```
perl elloreas.pl --help
```

How Elloreas works.

1) A starter is taken. It can be a contig from another assembly or a random read.

2) Reads are mapped to the edge of the starter.

3) Elloreas finds clusters of reads with similar sequences. There will be several clusters represented by many reads if there are several alternative extensions on the edge. Usually only one alternative extension is represented by many reads, while other alternative extensions are scarcely represented as they arise from reads with many sequencing errors.

4) The starter is elongated using a consensus sequence of the cluster represented by the largest number of reads.

5) Go to the next iteration (point “2)”).

At each iteration Elloreas produces a list of alternative extensions.

A more detailed description of the algorithm is also present in this document (How Elloreas works (a detailed description)).

What is Elloreas used for?

Usually de novo genome assemblers cannot assemble each chromosome in a whole contig. They produce many contigs which are fragmented in places which are created by "forks" in the assembly graph. Forks are places with two or more alternative extensions, which usually arise because of repeats in genomes, like transposons. You can read more about forks, for example, here. Elloreas is useful to:

1. Understand which alternative extensions exist for your contig. Then you can choose the extension you like and assemble the genome by Elloreas further using this particular extension.
2. Assemble whole genomes. To do this, use a random read as a starter. Elloreas in not very fast - expect a rate of starter elongation of approximately 1000 bp/minute. Therefore, I don't recommend to use it for genomes larger than bacterial.

Requirements:

Perl
R
minimap2
samtools
BLAST+
bedtools
USEARCH
MAFFT
EMBOSS
LASTZ

I tested Elloreas with the following versions, but it should work with others as well: Perl 5.28, R 3.5.1, minimap2 2.17, samtools 1.9, BLAST+ 2.9.0, USEARCH 11.0.667, MAFFT 7.402, EMBOSS 6.6.0, LASTZ 1.04. Paths to binaries of all these programs should be available through the environment variable $PATH.

Output of Elloreas

The main files produced by Elloreas are:

1. elloreas_logs.txt - this file contains the basic information of how Elloreas worked. The first thing you may want to look at when Elloreas finishes is the end of this file - it provides the most important information of the Elloreas run.
2. history_of_alternative_extensions.txt - this file contains the list of alternative extensions present in each iteration of elongation. Only the top 5 most represented by reads extensions are listed there.
3. contig_from_the_final_iteration.fasta - this is the final contig produced by Elloreas.
4. coverage_plot_for_the_contig_from_the_final_iteration.jpeg and dotplot_for_the_contig_from_the_final_iteration.jpeg - these two figures represent a read coverage distribution and a dotplot for the final contig produced by Elloreas.

Caveats:

1. Long reads usually contain a large percent of sequencing errors, namely short indels and substitutions of one nitrogenous base by another. Some of these errors may persist in your assembly. Therefore, after you assemble the sequence, I highly recommend to polish it with programs such as Pilon or Racon. Don't forget to include all sequences

from your genome during polishing. Otherwise, there may be mistakes. For example, if you polish a plant's mitochondrial genome but don't add the plastid genome during polishing, some of plastid reads will map to the mitochondrial genome (because these two genomes have homologous sequences), which may produce a chimeric sequence. During the assembly stage, the plastid genome won't be a big problem, because those homologous sequences create forks, which are reported by Elloreas, so you can choose the mitochondrial extension and ignore the plastid one.

Frequently asked questions:

1. ### How to cite Elloreas?

   Cite the paper "Mitochondrial genome of Fagopyrum esculentum and the genetic diversity of extranuclear genomes in buckwheat" (currently under review) where Elloreas was first described.

2. ### What's the difference between this manual and the readme at https://github.com/shelkmike/Elloreas?

   The only difference is that this manual contains not only a short description of the algorithm of Elloreas, but also an additional detailed description (How Elloreas works (a detailed description)).

3. ### Elloreas extends the contig only in the 3' direction ("to the right"). Why not make it to extend the contig to the left too?

   One of the most important features of Elloreas is that it provides a user with a list of forks, when they are present. Making a user to examine forks on 3' and 5' ends simultaneously may confuse the user. If you need to extend the contig leftward, just take the reverse complement version of the contig, so left becomes right. If you're assembling a circular chromosome, you won't even need to do this, because during the extension the right end will join the left end.

4. ### Can Elloreas be used with short reads?

   Yes, it can. Minimap2, the read mapper which Elloreas uses, is capable of mapping short reads too. However, Elloreas will use paired-end reads as single-end reads. This is why I recommend to use dedicated targeted assemblers for short reads, like NOVOPlasty or TASR.

5. ### Long reads of PacBio and Oxford Nanopore usually contain many sequencing errors. Why not integrate a polishing process like in Racon into Elloreas, so it polishes the contig after the assembly finishes?

   The problem with contig polishing by read mapping is that if you take only a single contig for polishing, reads from paralogs will map onto it too, so the polished sequence

will become a chimera. This is why it is important to add all contigs of your genome to the contig created by Elloreas and perform polishing only after this, thus reducing the possibility of false mappings.

6. There was a fork at some iteration. Elloreas chose one of alternative extensions, but I want to try another one. What should I do?

   In the file history_of_alternative_extensions.txt find the respective iteration. You will see a list of alternative extensions for this iteration. Elloreas always chooses the extension supported by the highest number of reads. Take the file contig_from_the_final_iteration.fasta, remove everything starting with this most-supported extension and instead paste the extension you want to try. Then, use this newly formed contig as a starter.

7. Where can I ask questions about Elloreas?

   You can use the "Issues" section on GitHub (https://github.com/shelkmike/Elloreas/issues)

8. How fast is Elloreas?

   I tested it using 22 CPU cores for several datasets each containing about a million reads and the rate of contig elongation was on the order of 1 kilobase per minute.

9. Will Elloreas run on Windows or Mac?

   I didn't test it on Mac, but I think it will work if you can install all the required programs. To run Elloreas on Windows use a virtual Linux machine, for example Ubuntu run through VirtualBox (https://www.osboxes.org/ubuntu/).

10. Many de novo assemblers create assembly graphs. Why can't they be used to detect forks in the assembly instead of Elloreas?

    You can use them. For example, Spades produces FASTG files with assembly graphs which can be visualised with Bandage. However, in my experience, de novo assemblers often don't report all forks. Probably, this is because there are some inner requirements of how represented by reads a branch should be to report this branch in a FASTG file. Elloreas, instead, produces a more detailed list of alternative extensions arising during the assembly process.

11. Well, finally, will Elloreas be useful for me?

    I don't know. It definitely was useful for ME. You may give it a try.

How Elloreas works (a detailed description).

This description is more detailed than the one you can find in the paragraph "How Elloreas works", though it omits some peculiarities that are of little significance.

I. When Elloreas is started, first of all, it performs some checks. It checks whether:
1) All programs it requires are present in the $PATH environment variable.
2) The user hasn't made any mistakes in the command line.
3) The files with reads and starter exist.
4) The file with the starter is indeed a FASTA file and contains only one sequence.

II. Elloreas maps reads to the right region (aka "3' end") of the starter. Further I will call the starter just "contig", though the starter can be a read. The size of the region used for mapping is equal to the value provided by the user with the parameter "--contig_edge_size_to_use_for_mapping" or to the full contig sequence, whichever is smaller. This mapping is performed by minimap2. The parameter --sequencing_technology affects the mapping mode of minimap2. Namely, "--sequencing_technology typical_pacbio" is equivalent to the option "-ax map-pb" of minimap2, "--sequencing_technology oxford_nanopore" is equivalent to "-ax map-ont" and "--sequencing_technology hifi_pacbio" is equivalent to "-ax asm20". Also, if the user didn't specify the "--minimum_read_similarity" parameter (how similar a read should be to the contig), but did specify the "--sequencing_technology" parameter, Elloreas will automatically set "--minimum_read_similarity" to 80% for oxford_nanopore and typical_pacbio, and to 98% for hifi_pacbio.

III. Elloreas performs various checks of the mapped reads. You can approximately understand these checks by looking at the list of Elloreas' parameters ("perl elloreas.pl --help"). For example, it checks whether a read overlaps the right edge of the contig, whether the mapped part of the read has enough sequence similarity with the contig (the "--minimum_read_similarity" parameter), whether a sufficiently long part of the read mapped to the contig (the "--minimum_length_of_mapped_read_part") and some other checks.

IV. Elloreas determines the possible extension of the contig. To do this, it:
1) Calculates the median length of overhangs. By "overhangs" I mean the parts of reads that extend beyond the contig's edge.
2) All overhangs that are shorter than the median overhang length are dropped. All that are longer are shortened to that median length. This is important, because in the next step I do clustering of overhangs, and to perform a good sequence clustering it is preferable for sequences to be of equal length.
3) Elloreas clusters the overhangs by USEARCH. If the user set the minimum similarity of a read to the contig to X, then during the clustering only reads that have at least X^2 similarity will be allowed to form a cluster. For example, if the "--minimum_read_similarity" was set to 90%, then the threshold of read similarity for clustering with each other will be 81% (0.9^2=0.81). This is because if a read has sequencing errors such that it is only X% similar to the reference, then two reads (each of them has random errors) will be similar to each other approximately by X*X%.
4) For clusters that consist of only a single read (such clusters are usually formed by reads that have so many sequencing errors that they cluster with nothing), the cluster sequence is called "cluster consensus". For clusters that consist of many reads, a real

consensus calculation is required. To do this:

a) Elloreas aligns all overhangs that form a cluster by MAFFT.

b) Elloreas calculates the consensus for this cluster by the program "cons" from EMBOSS

c) The right 10% of bases of the consensus sequence are deleted. For example, if the consensus was 1500 bp, the right 150 will be deleted, thus retaining only 1350. This is because the procedure of multiple sequence alignment is error-prone at edges, thus the removal will make the consensus to contain less errors.

5) Elloreas prints consensus sequences of top-5 clusters most represented by reads (or less than 5, if there are less) to the file "history_of_alternative_extensions.txt"

V. Elloreas extends the contig with the consensus of the cluster that was represented by the largest number of reads. Then it goes to the next iteration of extension, using this extended contig as the starter for the next iteration (that means, Elloreas goes to point "II")).

Comment 1: at many stages of its work, Elloreas performs different checks. You can understand them by reading the full list of parameters printed by the command "perl elloreas.pl --help". For example, it checks whether the largest cluster of read overhangs is supported by enough reads (and Elloreas stops working, if the number is not big enough), or whether Elloreas hasn't yet reached the maximum number of iterations allowed by the user. These two checks correspond to the "--minimum_number_of_reads_in_the_most_popular_extension" and the "--maximum_number_of_iterations" parameters.

Comment 2: after Elloreas finishes working, it creates a read coverage plot (coverage_plot_for_the_contig_from_the_final_iteration.jpeg) and a dotplot (dotplot_for_the_contig_from_the_final_iteration.jpeg) for the final contig. These files can be useful to the user to understand the final contig better.

Comment 3: during its work, Elloreas creates a special folder for files of each iteration. Such folders are called like Iteration2 or Iteration137. The user can tell Elloreas to remove these folders by specifying the option "--delete_folders_of_iterations true". You can find the following files in those folders:

1) Sequence of the contig's right part, which was used to map reads to (called like "Edge_iteration137.fasta").
2) Sequences of read overhangs forming different clusters. They are produced by USEARCH and are called like "cluster0", "cluster1", "cluster2"... They are FASTA files. USEARCH numbers them in an arbitrary order, not in the order of decreasing size.
3) Alignments of these overhangs. Files are called like cluster0_alignment.fasta.
4) Consensuses of these alignments. Files are called like cluster0_consensus.fasta . Such files are created only for clusters that consist of more than 1 read (this is because the consensus for cluster consisting of 1 read is trivial - it is just the sequence of this read). Note that there are a lot of "n" letters in the consensus. They usually arise from sequencing errors that create improper insertions in reads (you can compare the consensus with the alignment in cluster0_consensus.fasta by eye to understand this). These n's are removed by Elloreas before attaching the consensus to the right edge of the contig.

5) Result of a BLAST alignment of contig after the extension with itself (called like megablast_results.it137.txt). This alignment allows Elloreas to understand whether there is a long repeat at the right edge. If it sees a repeat with 100% identity between its two units which is longer than the value provided with the "--minimum_length_of_mapped_read_part", this means that Elloreas will not be able to continue its work, because it will fall into an eternal loop, forever creating the sequence that lies between these two repeat units. This doesn't necessarily mean a problem, because if the repeat has one unit at the right edge of the contig and another unit at the left edge, this may mean that you have completely assembled a circular genome.
6) Contig after extension, which is created at the very end of the iteration. It is called like "starter_that_I_have_used.it137.fasta"

Comment 4: If you need more details about the algorithm, or just want me to explain something better, just write to https://github.com/shelkmike/Elloreas/issues or shelkmike@gmail.com and I will do it (and, if you want, I'll add the details or the explanation to this manual).