

mitoGenome Assembly

Ian G. Brennan

26 July 2019

Contents

Install the software	2
Multiple mtGenome assembly/alignment using MITObim from R	3
Why Bother?	3
File Preparation	3
Executing the Code	4

Install the software

1. MITObim¹: <https://github.com/chrishah/MITObim>
 - make sure to follow install instructions, including dependencies: MIRA, PERL, etc.
2. MUSCLE²: <https://www.drive5.com/muscle/>
3. MAFFT³: <https://mafft.cbrc.jp/alignment/software/>

If you're on a mac, downloading this repository will include MUSCLE and MAFFT suited for a mac, so you can plug 'n' play. If you're on a Windows or Linux machine, you'll have to do this separately, and adjust the call to the programs in 'mitoAlign.R'. Sorry, I'm just too dumb.

¹Hahn, C., Bachmann, L., Chevreux, B. Reconstructing mitochondrial genomes directly from genomic next-generation sequencing - a baiting and iterative mapping approach. *Nucleic Acids Research* 41:13. doi:10.1093/nar/gkt371 (2013).

²Edgar, R.C. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research* 32(5). doi:10.1093/nar/gkh340 (2004).

³Katoh, Standley. MAFFT multiple sequence alignment software version 7: improvements in performance and usability. *Molecular Biology and Evolution* 30:772-780 (2013).

Multiple mtGenome assembly/alignment using MITObim from R

Why Bother?

As a result of the exon capture process for Anchored Hybrid Enrichment projects, we're getting a considerable amount of mitochondrial bycatch in our raw reads. This mtDNA can provide a lot of information including a separate phylogenetic history for the group of interest, or even past introgression events. Also, *::free data::*.

File Preparation

We'll use the MITObim pipeline, run from R to assemble our mtGenomes against a reference (find one on GenBank, distantly related is OK). We'll then pull out the final assembly for each sample and align them against each other using MUSCLE.

Start by creating a directory to hold:

- the MITObim pipeline ('MITObim.pl')
- the MIRA sequence assembler and mapper directory ('mira.4.0.2...')
- the MUSCLE executable ('muscle3.8.31...')
- the MAFFT folder ('mafft-mac')
- a project-specific directory holding all your mtDNA reads for each sample in subdirectories ('Frogs'), and your reference mtGenome as a fasta file, labelled '[taxon]_mtGenome.fasta'

Here's a quick schematic of what the structure should look like:

```
/PATH_TO_PARENT_DIRECTORY/mtGenomes
|-- MITObim.pl
|-- mira_4.0.2_darwin13.1.0_x86_64_static
|   |-- bin (et al.)
|-- muscle3.8.31_i86darwin64
|-- Frogs
|   |-- [taxon]_mtGenome.fasta (reference genome)
|   |-- Taxon1
|       |-- Taxon1_R1.fastq.gz
|       |-- Taxon1_R2.fastq.gz
|   |-- Taxon2
|       |-- Taxon2_R1.fastq.gz
|       |-- Taxon2_R2.fastq.gz
```

Executing the Code

Open a new R script, and source the functions we'll need

```
source("/PATH_TO_CODE/mtGenome_Assembly.R")
```

Then set your working directory to the project-specific directory (I've called this directory 'Frogs'). This holds a subdirectory for each sample, and your reference mitogenome.

```
setwd("/PATH_TO_DIR/mtGenomes/Frogs")
```

The first function (*mitoAssemble*) assumes that in the directory 'Frogs', there are one or more subdirectories (1 per sample), with one or more gzipped fastq files of the raw reads. For each sample (subdirectory), it will:

- merge (concatenate) any *.fastq.gz files together.
- copy your reference genome (fasta file)
- call MITObim and attempt to assemble the mtGenome
- copy the assembled mtGenome to a directory of all assemblies

The function requires as input, three things:

```
mitoAssemble(num.iter, reference.name, project.name)
```

- *num.iter* is the number of assembly iterations MITObim should try before it times out and moves on to the next sample. I usually leave this set to 100, sometimes it takes 4 iterations, sometimes 60, hard to know.
- *reference.name* is the unique name of your reference genome fasta file which precedes '_mtGenome.fasta'
- *project.name* is what you'd like the generated directories to be called (where the assemblies are stored)

The function *mitoAssemble* will spit out all the assemblies to a new directory, and tell you where it is:

```
Assembly(s) completed and saved to: /Users/Ian/MITObim/Assa/Assa_mtGenomes
```

The second function *mitoAlign* will:

- combine all assembled mtGenomes into a single fasta alignment ([project.name]_Assembly_Alignment.fasta)
- align the assemblies using MUSCLE or MAFFT, into a single final alignment ([project.name]_Aligned_Assemblies.fasta)

The function *mitoAlign* will spit out the final alignment into your new directory, and tell you where it is:

your alignment of mtGenome assemblies is called:

```
/Users/Ian/MITObim/Assa/Assa_mtGenomes/Assa_Aligned_Assemblies.fasta
```

The function requires little information to do its job well. Bare minimum it should be given the project name, and your choice of aligner ("MUSCLE" or "MAFFT"). You can also designate the reference genome for aligning purposes:

```
mitoAlign(project.name, "MUSCLE", reference.name)
```

- *project.name* what you've been calling the project, so it can find the folder with all your mitoGenome assemblies,

- *reference.name* if you'd like to improve your multi-mitoGenome alignment, you can align them back to your reference genome. By default, this is NULL, to specify it, just give it the name you provided in the *mitoAssemble* step.
-

The third function *mitoChop* will:

- take the whole mitoGenome alignment and split it up by locus
- this function requires the R packages *ape* and *seqinr* to read and write the files.

The function will kick out the alignment for each mitochondrial locus (CDS/gene, rRNA, tRNA, *et al.*) separately into a new folder:

```
your separated mitochondrial loci alignments are in a folder called::  
/Users/Ian/MITObim/Assa/Assa_mtGenomes/Assa_mitoLoci
```

You just have to provide a character set file as a .CSV so it knows where to cut it up

```
mitoChop(project.name, alignment, character.sets)
```

- *project.name* is again, just what you've been calling the project so far (name of the directory)
 - *alignment* the name of the output .fasta file from the *mitoAlign* step
 - *character.sets* is the .CSV file with three columns "Name", "Minimum", and "Maximum". These correspond to the locus name, the starting position (*Minimum*), and final position (*Maximum*) for each locus. An example is included in the repository ("mitoGenome_Annotations_Example.csv").
-

There's probably a bunch of other slick things I could do after this, like have it plot the individual assembly lengths, but I'm not sure it's worth it at the moment. Let me know if there's something specific you're looking for though.

Good luck!
