mitoGenome Assembly

Ian G. Brennan

14 January 2020

Contents

Install the software

1. MITObim[1]: https://github.com/chrishah/MITObim
    - make sure to follow install instructions, including dependecies: MIRA, PERL, etc.
2. MUSCLE[2]: https://www.drive5.com/muscle/
3. MAFFT[3]: 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.

[1]Hahn, 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).

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

[3]Katoh, 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
|-- bbmap
|   |-- reformat.sh
|   |-- et al.
|-- 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")
```

---

mitoAssemble

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 can be either run sequentially (one mitogenome assembled after another), or in parallel (multiple mitogenomes assembled simultaneously), and requires as input just a few things:

```
mitoAssemble(num.iter, reference.name, project.name,
             write.shell, ncores, combine = c("cat", "bbmap"))
```

- *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)
- *write.shell* if you'd like the function to write a shell script so that you can run MITObim in parallel
- *ncores* designate the number of cores to use if you want to run in parallel
- *combine* tells the function whether to use *cat* to combine raw read files, or *bbmap* to create an interleaved file that can be used with the '–paired' flag in MITObim

If run sequentially (*write.shell* = *FALSE*), 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
```

Depending on how many you need to assemble, this could take a while.

If run in parallel (*write.shell* = *TRUE*), the function will spit out a shell script for you to drop into your terminal (outside R suggested), and provide instructions:

```
Your shell script for running MITObim in parallel is written to:
/Users/Ian/MITObim/Assa/Assa_parallel.txt
Execute the command in parallel by copy/paste to your terminal:
parallel -j 14 :::: /Users/Ian/MITObim/Assa/Assa_parallel.txt
```

This requires you have *parallel* installed to your machine, which you can easily do with *$ brew install parallel*, assuming you have *homebrew* installed (highly recommended).

---

mitoCollate

If you have used the parallel option *write.shell = T* for the function *mitoAssemble*, then you'll need to pull together the best assembly for each sample into a single location. This function will:

- find each completed assembly
- rename the assembly according to the sample name
- rename the file according to the sample name
- copy it to a folder which will hold all the assemblies

The function requires only the name of the project:

```
mitoCollate(project.name)
```

- *project.name* is what you'd like the generated directories to be called (where the assemblies are stored)

---

mitoAlign

The next 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.

At the moment, I'd probably suggest using the MAFFT aligner with the reference option if possible. Aligning the assembled mitogenomes without a reference can result in some poor behavior from poorly assembled genomes.

---

mitoCheck

The function *mitoCheck* will:

- read in your alignment, and provide information about missing/gaps in your data

- remove taxa with low amounts of sequence if you'd like

You just have to give it the alignment and a little info:

```
mitoCheck(project.name, alignment, count.gaps=T, missing.threshold=NULL)
```

- *project.name* is again, just what you've been calling the project so far (name of the directory)

- *alignment* is the name of the output .fasta file from the *mitoAlign* step

- *count.gaps* is just a TRUE/FALSE statement about whether to count gaps as missing data

- *missing.threshold* if you want to remove taxa that exceed a certain percentage of missing data, indicate that here (0 < x < 1)

If you do decide to trim taxa based on missing data, the function will produce a new alignment and tell you the name:

```
your reduced alignment is now called:
/Users/Ian/MITObim/Assa/Assa_mtGenomes/Assa_Aligned_Assemblies_Reduced.fasta
```

I thought about using a different metric such as base composition/content, but I haven't thought about it enough to implement it well. Instead, it always identifies the outgroups as being funky, so I've just stuck with missing data.

---

mitoChop

The last 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=NULL)
```

- *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 respository ("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!