Genome Assembly (June 2016)

Alexander Jueterbock, Martin Jakt*

PhD course: High throughput sequencing of non-model organisms

Contents

After removing adapters and bad-quality reads, we are ready for *de novo* assembly of the sequenced genome libraries. The number of available genomes is increasing, also for non-model species (for marine species, see for example The IMAGO Marine Genome projects). Many analyses are only possible with a reference genome. Thus, *de novo* assembly is a first important step for many follow-up analyses, such as SNP-discovery for population-genomics or differential-expression analysis based on RNAseq data.

1 MIRA - assembler

The choice of *de novo* sequence assemblers is wide (overview). Some of the better known open-source assemblers include SPAdes, Velvet, SOAPdenovo, and MIRA. Have a look on GAGE, which compares the performance of major assembly strategies.

We use MIRA in this tutorial because it can handle sequencing data from different platforms, including Illumina and Ion Torrent. Please find its documentation here. Use the following command to get an overview of the parameters:

```
mira --help
```

Before we start, we will create a folder that contains the data we want to assemble:

```
mkdir GenomeAssembly
cd GenomeAssembly
mkdir data
```

*University of Nordland, Norway

copy your quality-trimmed fastq file into the `data` directory with:

```
cp SOURCEFILE TARGETDIRECTORY
```

Here, you have to replace `SOURCEFILE` with your trimmed fasta files and `TARGETDIRECTORY` with the path to the `data` directory that you just created. In case of paired-end sequencing, you copy both the fasta file with trimmed forward reads and the fasta file with trimmed reverse reads into this folder.

The configurations for `mira` are specified in a so-called `manifest` file. In this tutorial and will choose the most simple settings for a *de novo* genome assembly.

To create the manifest file in your `GenomeAssembly` directory, use the command `touch`:

```
touch manifest.conf
```

Now, to edit this file, we can use the command-line program `nano`. This allows you to open and edit small text files from the command line (no graphical user interface needed). To open the `manifest.conf` file, just type:

```
nano manifest.conf
```

Once you hit ENTER, `manifest.conf` will be opened. For now, it is still empty. You can edit the content of the file by deleting and adding text. At the bottom of the terminal window you see some shortcuts for certain actions. For example `^O WriteOut` or `^X Exit`. The `^` indicates that you need to press CTRL+O or CTRL+X.

Depending on whether you have sequencing data from Illumina or from Ion Torrent, your `manifest.conf` file should have different content. Each case is described in the folllowing two sections.

1.1 manifest file for Ion Torrent data

```
# Manifest file for de novo genome assembly with Ion Torrent single reads

project = IonTorrentDeNovoAssembly
job = denovo,genome,accurate

readgroup = UnpairedIonTorrentReadsFromHTSCourse2015
data = fastq::data/YOURINPUTFILE.fq
technology = iontor
```

You can copy these lines and paste them into your file by pressing SHIFT+CTRL+V. Change the name `YOURINPUTFILE.fq` to the name of the fastq-file that contains your quality-trimmed reads. Then save the file and exit with CTRL+O and CTRL+X.

That's all you need before you can start `mira` with:

```
nohup mira manifest.conf >log_assembly.txt &
```

The analysis will take at least 1 hour to finish.

1.2 manifest file for Illumina data

The illumina runs produced paired end reads. To assemble these reads with MIRA, we have to specify two additional parameters as compared with single end reads:

1. `segment_placement` The orientation of the forward and backward reads to each other. In our case it is `---> <---`

2. `template_size` The total length the fragments. The total length of the reads is 600bp. The length of the fragments depends on the overlap between the reads. If they overlap by 200bp, then the total length of the fragments is 400bp. We can specify a range, like `500 700` (for 500bp-700bp). If we are unsure about this, we can increase the range and use the `autorefine` feature of MIRA to automatically refine this range during the assembly based on real, observed distances of read pairs.

Here, we specify the manifest file as:

```
# Manifest file for de novo genome assembly with paired end Illumina data
project = IlluminaDeNovoAssembly
job = denovo,genome,accurate
parameters = -NW:cmrnl=warn

readgroup = PairedIlluminaReadsFromHTSCourse2015
data = fastq::data/YOURINPUTFILE_1.fq fastq::data/YOURINPUTFILE_2.fq
technology = solexa
template_size = 100 1000 autorefine
segment_placement = ---> <---
```

The Illumina reads generally have names longer than 40 characters. MIRA stops running in this case because several programs have restrictions concerning the length of the read name. In our case we let MIRA give us a warning about this but avoid that the program stops completely by providing the argument `-NW:cmrnl=warn`.

You can copy these lines and paste them into your file by pressing SHIFT+CTRL+V. Change the name `YOURINPUTFILE_1.fq` `YOURINPUTFILE_2.fq` to the name of the fastq-files that contain your quality-trimmed forward and reverse reads. Then save the file and exit with CTRL+O and CTRL+X.

If you do not know these parameters at all, you could simply add the parameter `autopairing` to your manifest file, like here:

```
# Manifest file for de novo genome assembly with paired end Illumina data - the lazy way;)

project = IlluminaLazyDeNovoAssembly
job = denovo,genome,accurate
parameters = -NW:cmrnl=warn


readgroup = PairedIlluminaReadsFromHTSCourse2015
autopairing
data = fastq::data/YOURINPUTFILE_1.fq fastq::data/YOURINPUTFILE_2.fq
technology = solexa
-NW:cmrnl=warn
```

That's all you need before you can start `mira` with:

```
nohup mira manifest.conf >log_assembly.txt &
```

The analysis will take at least 1 hour to finish.

2 Results and assembly metrics

MIRA creates a directory named `PROJECTNAME_assembly` (you defined the PROJECTNAME in the manifest file) and several subdirectories. We are primarily interested in the following two subdirectories:

- 1. `PROJECTNAME_d_results`: this directory contains all the output files of the assembly in different formats. Here we are specifically interested in the following fasta files:
  - `PROJECTNAME_out.padded.fasta`. This file contains the assembled contigs. Gaps are denoted by an asterisk.
  - `PROJECTNAME_out.unpadded.fasta`. This file also contains the assembled contigs, but with positions containing gaps removed.
  - `PROJECTNAME_LargeContigs_out.fasta`. This file contains the longer contigs of your assembly, which are of particular interest. To be included in this file, a contig generally needs to be at least 500bp long and must have a coverage of at least 1/3 of the average coverage.
- 2. `PROJECTNAME_d_info`: this directory contains files describing the properties of the final assembly. We are particularly interested in:
  - `PROJECTNAME_info_assembly.txt`. This file contains summary statistics and information about problematic areas in the results. Here, 'Consensus bases with IUPAC' refers to positions that are not clearly 'A', 'C', 'T', or 'G', but where two or more bases were equally likely. For example, 'R' refers to 'A or G', and 'K' refers to 'G or T'.
  - `PROJECTNAME_info_contigstats.txt`. This file contains statistics about the contigs themselves, their length, average consensus quality, number of reads, maximum and average coverage, average read length, number of A, C, G, T, N, X and gaps.

Search for the following information in `PROJECTNAME_info_assembly.txt`:

- Number of contigs in the assembly
- Maximum contig coverage
- Largest contig
- N50 contig size

Reminder on the N50 metric (see Fig. 1):

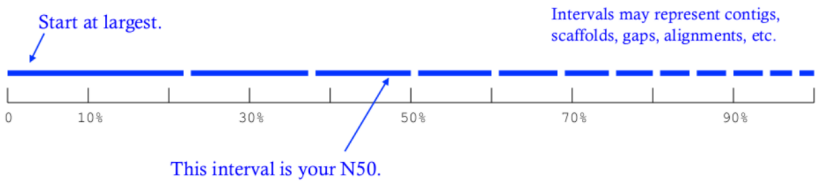


Figure 1: From Kane, N.C.

N50 measures the median contig length in a set of sequences. The larger it is, the closer your assembly gets to the real genome. N50 is obtained by:

- 1. Sorting contigs in descending length order.
- 2. Identifying the size of the contig above which the assembly contains at least 50% of the total length of all contigs.

We can use the program R to create histograms of the contig lengths and coverages from the file `PROJECTNAME_info_contigstats.txt`. If you are in the directory named `PROJECTNAME_assembly` (if you are not in this directory, you can move to it with the `cd` command), you can copy and paste the following commands into your terminal window to plot histograms of the contig lengths and coverages:

Replace here PROJECTNAME with the name of your own project. For this, you can first copy all of the following commands besides the last line (`R CMD BATCH...`) and past it in your terminal. Then you can use `nano Rplothistogram.r` to change the contents of this file.

```
rm Rplothistogram.r # Use this if the file Rplothistogram.r already exists.

cat >> Rplothistogram.r << 'EOF'
contigs <- read.table(
  file="PROJECTNAME_d_info/PROJECTNAME_info_contigstats.txt",
  sep="\t", header=FALSE)

png(filename = "ContigLengths.png",
  width = 480, height = 480, units = "px", pointsize = 12,
  bg = "white")
hist(contigs$V2,main="Histogram of contig lengths",
  xlab="Contig length (bp)",ylab="Frequency",col="blue",breaks=100)
dev.off()

png(filename = "ContigCoverages.png",
  width = 480, height = 480, units = "px", pointsize = 12,
  bg = "white")
hist(log10(contigs$V6),main="Histogram of average log10 contig coverages",
  xlab="Average log10 contig coverage",ylab="Frequency",col="blue",breaks=100)
dev.off()

EOF

R CMD BATCH Rplothistogram.r
```

Alternatively you can use R interactively by starting an R session (just type `R` and return) and pasting the commands one by one into the R session. In this case you can omit the `png(...)` and `dev.off()` commands; these are used to create exportable images of plots (see below for more).

To open the figures, you can use the `eog` command, which is the Eye of Gnome graphics viewer

program:

```
eog ContigLengths.png
eog ContigCoverages.png
```

Example histograms of contig lengths and coverages are shown in Fig. 2 and 3.

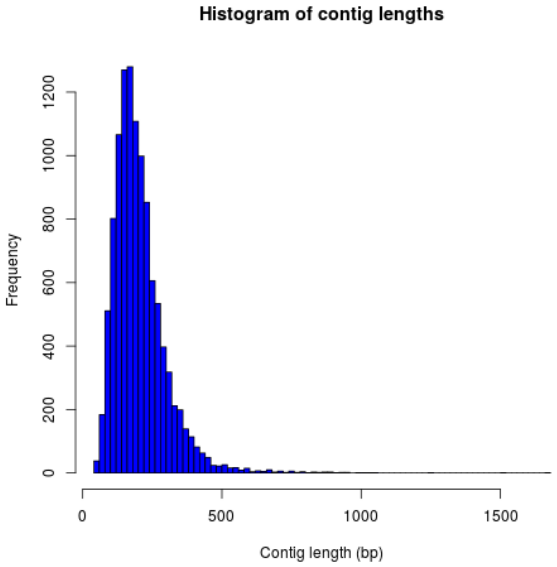


Figure 2: Histogram of contig lengths

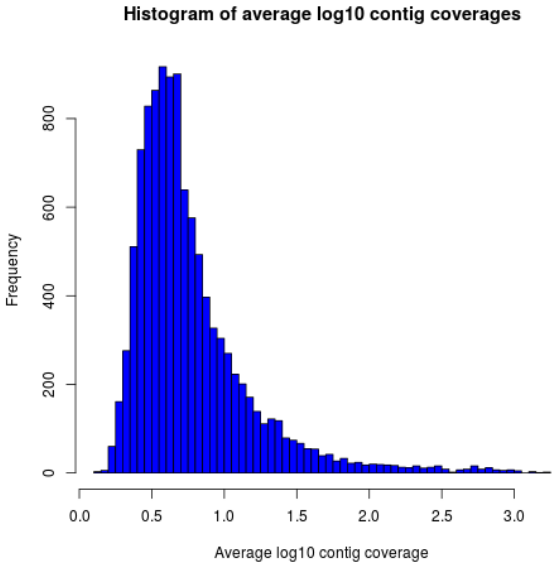


Figure 3: Histogram of contig coverages

You can also extract the number of contigs>500bp and the sum of bases in these contigs with R. Until now you have used R scripts with the `R CMD BATCH` command, just like the created script `Rplothistogram.r` above.

Instead of running `R` scripts from the shell command line, you can also open an `R` command-line window where you can execute commands directly. To start `R`, just type `R` in the terminal and hit enter. All that comes after this command will be executed in the R console. Lines preceded with a `#`-sign will be ignored and serve only as non-executed comments. You should be in the directory named `PROJECTNAME_assembly`

Again, replace here PROJECTNAME with the name of your own project.

```
R

# open the output file from MIRA
contigs <- read.table(
file="PROJECTNAME_d_info/PROJECTNAME_info_contigstats.txt",
  sep="\t", header=FALSE)

# Extract only those contigs that are longer than 500bp
contigs.above500 <- contigs[contigs[,2]>500,2]

# Count the number of contigs that are longer than 500bp
length(contigs.above500)
# Output for example: 156


# Count the number of bases in these contigs
sum(contigs.above500)
# Output for example 102297

# leave R again
q()
n
```

MIRA does not only assemble your reads but it comes with a command line tool named `miraconvert`, which allows you to extract contigs based on, for example, contig length and coverage (see in the MIRA documentation for further details and options).

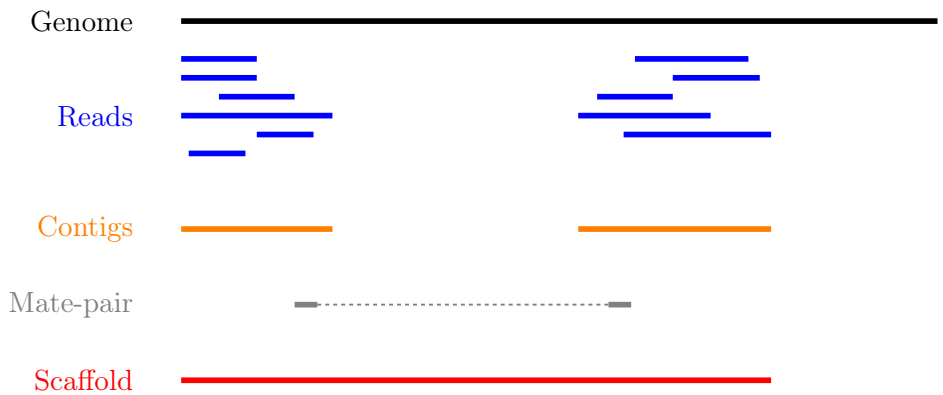
3 Next steps to consider

Hint: to identify the proportion of contigs that are protein-coding and the proportion that may result from bacterial contamination, you can use the Basic Local Alignment Search Tool (BLAST) to align the contigs to databases with known genes and proteins.

Proper annotation of a *de novo* genomes is a challenging task. An overview of the process for eukaryotes is given in Yandell, Mark, and Daniel Ence. "A beginner's guide to eukaryotic genome annotation." Nature Reviews Genetics 13.5 (2012): 329-342. If you work with prokaryotes, have a look at Stothard, Paul, and David S. Wishart. "Automated bacterial genome analysis and annotation." Current opinion in microbiology 9.5 (2006): 505-510.

For protein annotation it helps also to sequence the transcriptome. The assembled contigs of the transcriptome will only consist of exons; all introns are cut out. Annotation of a *de novo* transcriptome assembly is described in The Simple Fool's Guide to Population Genomics via RNA-Seq. Be aware that the annotation can take several weeks to finish.

MIRA assembles the reads to so-called contigs, which are based on overlapping sequences. Contigs can be joined with mate-pair libraries into longer fragments (often referred to as scaffolds, which are basically contigs that were connected by gaps, see figure below). MIRA does not perform scaffolding. This can be done with the stand-alone SSPACE software.



Emacs 24.5.1 (Org mode 8.3beta)