Genome assembly

Dag Ahrén
Kalle Tunström

January 15, 2024

Contents

1 Conda introduction **2**
1.1 Introduction 2
1.2 Conda 2
1.2.1 Install and update conda 3
1.2.2 Configure Conda 3
1.2.3 Create a new Conda environment 3

2 Installation and data **5**
2.1 Introduction 5
2.2 Velvet 5
2.3 Spades 6
2.4 Minia 6
2.5 SRA Toolkit 6

3 Data download **8**
3.1 Bacterial reads 8
3.2 Fastq format 9

4 Bacterial Genome Assembly **11**
4.1 Introduction 11
4.2 Data 12
4.3 velvet 12
4.3.1 Assembly statistics 14
4.4 spades 14
4.5 Minia 14
4.6 Reference 14

Chapter 1

Conda introduction

1.1 Introduction

Conda is an open source package and environment system that is used to manage installations and dependencies of software and works for any programming language.

1.2 Conda

It is recommended to use Conda for the local installations for several reasons. For example:

- Conda makes installation and upgrades of software easier.
- It creates an isolated environment that can be tailored for a specific project
- It facilitates reproducibility across different computers and systems

OK, lets get started!

First check that conda is installed and take note of which version you are running:

```
conda -V
conda info
```

1.2.1 Install and update conda

If conda is not installed, the recommended approach is to install a local version of Miniconda from this web page: https://docs.conda.io/en/latest/miniconda.html

You can get full documentation of a command conda using help. For example to know more about the command update:

```
conda update --help
```

For example it could be good to update conda.

```
conda update conda
```

1.2.2 Configure Conda

Note! You only need to do the conda configuration once!
Configure the Conda environment so that you can install bioinformatics software, for example from the BioConda channel.

```
conda config --add channels defaults
conda config --add channels bioconda
conda config --add channels conda-forge
```

1.2.3 Create a new Conda environment

To list the currently available environments

```
conda env list
```

The * indicates which environment is active at the moment.

Create a new Conda environment to be used during the Genome assembly exercise.

```
conda create -n assembly
```

In order to activate the newly created environment we unsurprisingly use the command... **activate** followed by the environment we want to use., in this case **assembly**

```
conda activate assembly
```

You should now see that the prompt contain the word assembly in parenthesis. To deactivate you simply type:
conda deactivate

OK, great! Now, make sure you are in the assembly environment before continuing.

```
conda activate assembly
conda env list
```

Make sure that the assembly environment you just created is labelled with a * .

Chapter 2

Installation and data

2.1 Introduction

Each student will install the software to be used in their home directory using a conda environment. For this exercise, these software will be used:

velvet Another genome assembler (and transcriptome assembler).

spades *Preferred.* Yet another genome assembler.

SRA toolkit A software to extract sequences from NCBI sra files.

SCAFFOLD builder A web application for scaffolding by using a reference genome. No download required.

Minia *if time allows.* Yet another genome assembler.

2.2 Velvet

Velvet is a short read assembler that has been around for a looong time. We will test assemblers that are a bit more up to date shortly.

```
conda install velvet
https://www.overleaf.com/project/5ffed362d7828d64b0f31630
```

Test if it works:

```
cd
velvetg
velveth
```

2.3 Spades

Spades is one of the best genome assemblers for small to medium sized genomes and in addition it is developed to be able to assemble many different types of data, such as bacterial and viral genomes, transcriptomes, metagenomes as well as single cell genomes.

```
conda install spades
```

Voilá! You know have Spades installed! Chech that Spades is working properly by running this command:

```
spades.py --test
```

Now a test genome assembly should automatically start with a small test dataset. Once the assembly is finished use your BASH skills to check out the newly generated directory. One convenient command to investigate the directory is:

```
tree
```

2.4 Minia

Minia is a software for ultra low memory *de novo* assembly that has proved very useful for large genomes. It is an included in the GATB-Minia-Pipeline https://github.com/GATB/gatb-minia-pipeline that can assemble both genomes and metagenomes.

Install Minia in your assembly conda environment.

2.5 SRA Toolkit

Install the SRA Toolkit by searching for it in conda:

```
conda search sra*
```

Which tool do you think is the correct package to install using **conda install**?

```
Installation without conda is described below. Skip these
    steps if you managed to install SRA Toolkit using Conda!

If you think you need to install manually, please contact
    one of the instructors before you proceed.
```

Download to your `bin` directory the appropriate Linux 64 bit version from:
https://www.ncbi.nlm.nih.gov/sra

```
tar -xvzf sratoolkit.3.0.10-centos_linusx64.tar
cd sratoolkit.3.0.10-centos_linux64
ls
cd bin
ls -l  # Note that links link to links
```

As you see there are may executable files (programs or binaries). Linking all these to your own `bin` directory is too complicated. Instead try to add
~/bin/sratoolkit.3.0.10-centos$_linux64\_linux64/bin$
$toyourcurrent$PATH$variable.Instructionsarefoundinsection1.4inthe$Bash$handout.Remembertoinitiate$

```
bash -l # Opens a new shell in the current shell
# Alternatively:
source ~/.bashrc
# Alternatively: Open a new terminal
```

`Don't forget to test that at least one of the software works:`

```
cd
srf-load -h
```

`There is no need for copying these programs to the server.`

Chapter 3

Data download

3.1 Bacterial reads

First create a directory for the exercise:

```
cd
mkdir BacterialGenomeAssembly
cd BacterialGenomeAssembly
```

We will download sequence data from NCBI's Short Read Archive (SRA). To be allowed to publish studies where sequencing is a part of the study, all sequencing data has to be made public. Both EBI and NCBI have this service, but we concentrate on NCBI. Different type of sequences go into different databases. For example, annotated genes go into the nt database, draft genome sequences go into WGS (whole genome shotgun) and (some) read data from next generation sequencing go into SRA. The process of submitting annotated genome data is called "Submission hell". This type of submission is complicated and you will have a dedicated person at NCBI that helps you with the process. Go to:
http://www.ncbi.nlm.nih.gov/sra/SRX377522
Before processing data, take a look at the page.

1. What is the bacterium called and from where was it isolated?
2. The strategy used is WGS. What does shotgun mean (S in WGS)?
3. What sequencing technology has been used?
4. What kind of libraries (layout) have been used?
5. How long are the reads (you can double check this later)?
6. How many reads are there? One spot represents both the forward and the corresponding reverse read.

7. How many bases have been sequenced (take the results from the two questions above and multiply them)? Compare with what is stated on the page.

```
    echo 325997*300 | bc # An easy way to do it
```

8. This bacterium has a genome size of about 3.5 Mb. What is the sequencing depth (in average)?

We want to download the data in fastq format but navigating to the download link is not easy. Later, we will fetch data in a simpler way.

- Click on the run id SRR1029680 in the table at the bottom.
- Click on the ''FASTQ/FASTQ download'' tab.
- Tick "Filtered"
- Tick ''FASTQ''.
- Click the ''Download'' link.
- Choose the BacterialGenomeAssembly directory for the download.

The download takes some time. You can look at the progress:

```
ls -l
ls -l
```

You see a file called sra_data.fastq.gz.part (or similar) that is increasing in size. When the file is finally downloaded, the file is renamed to SRR1029680.fastq.gz. Why is the download processed like this? Uncompress the file. Before proceeding with the first assembly we should to take a look at the fastq format.

3.2 Fastq format

Normally each sequence entry consists of four lines. Take a look at the first two sequence entries:

```
head -8 SRR1029680.fastq
```

- The first line begins with a '@' character and is followed by a sequence identifier and an optional description (like a fasta title line). If you are interested how the sequence id looks like, go to https://en.wikipedia.org/wiki/FASTQ_format section ''Illumina sequence identifiers''.

- The second line contains the sequence. But be aware that in some rare cases the sequence can span several lines (like in fasta format). So always take a look first.
- The third line begins with a '+' sign and is optionally followed by the same sequence identifier (and any description) again. Often you only see a '+'.
- The last line encodes the quality values for the bases in the read sequence and must contain the same number of symbols as letters in the sequence.

The quality scale in "modern" fastq files is (starting with lowest value):

```
!%"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ [\]^_`abcdefghijklmnopqrstu
```

With bash check:

9. how many sequences there are
10. check how many base pairs there are
11. check how long a read is (you can assume that all reads have the same length)

You may have reflected on that there is only one fastq file. So this file contains both the forward and reverse reads, but many bioinformatic software assume two input files, one fastq for forward reads and one for reverse reads and that both of these have the same order when it comes to the sequence entries. It is possible to write your own program to do so, but we will use a simpler solution in the next assignment. To demonstrate how the sequence ids are built in the current file, do:

```bash
head -8 sra_data.fastq | grep @SRR
```

The only difference between the two lines is that a '1' is substituted to a '2'. '1' denotes forward read and '2' reverse read.

Chapter 4

Bacterial Genome Assembly

Please note, that copying text from a pdf-file into a text editor may cause some problems, especially when it comes to apostrophes and spaces.

4.1 Introduction

We will start with assembling a genome of a thermophilic bacterium, *Geobacillus thermopakistaniens*. We do that so we can be able to compare the observed genome size with the reported genome size from a related species (no genome has been reported from *Geobacillus thermopakistaniens*). After gene prediction we will also be able to compare "our" genes with the annotated genes.
The assemblers we will use are called minia, velvet and spades. Especially spades is commonly used for prokaryotic genome assembly. There is a bunch of assemblers out there and it's hard to evaluate which ones that make a better job. For this assembly minia, velvet and spades will work quite well, but when it comes to transcriptome assembly other assemblers are often preferred.

Fill in the following table when your work progresses:

#	Assembler	Assembly size	No. scaffolds	Optional: N50
0	Reference genome			
1	minia			
2	velvet			
3	spades			
4	Scaffold builder			

4.2 Data

For velvet assembly you will use the Illumina sequences that you downloaded from NCBIs Short Read Achive (SRA) in fastq format. The Illumina reads are paired end reads (as contrary to single end reads). With the aid of the SRA toolkit we can directly download data from NCBI's ftp-site. By default, fastq-dump, which we will use, outputs one single fastq file like the one you downloaded. Since most assemblers and mappers require two files when it comes to paired end reads, one with the forward sequences and one with the reverse, we add the option --split-files. We will use the two files when doing the velvet assembly.

```
# SRR1029680 is the run id we used earlier for download
fastq-dump --origfmt --split-files SRR1029680
```

We also added the -origfmt argument. NCBI reformats the sequence ids to an own format, but by giving this argument we will get the original Illumina format. Compare the new files with the old one:

```
ls -lh  # Note that M means megabytes and G means
   gigabytes.
wc -l *.fastq
head -2 *.fastq
```

1. Why is the size of the original file larger than the sum of the two new ones? Remember that normally one character corresponds to one byte.

2. What is the reason that there is one line more in the original file as compared to the two new ones? (Don't spend time on this one).

4.3 Assembly with velvet

velvet runs in two steps:

1. velveth which prepares a data set to be used with the assembler.

2. velvetg which makes the actual assembly.

velveth may require a lot of memory (RAM). To calculate how much memory that will be used, use the following approximation:

Required RAM = (-109635 + 18977*ReadSize + 86326*GenomeSize + 233353*NumReads - 51092*K)/1024

- Read size is in bases.
- Genome size is in millions of bases (Mb)
- Number of reads is in millions
- K is the kmer hash value used in velveth (default 21, but use 31)
- The answer is in MB.

3. What is the memory requirement?
4. Optional: Imagine that you want to assemble a bird genome. The genome size is 1 Gb. You will sequence the genome to 50 times coverage using 2x150 paired end reads. How many reads are required and what will the memory requirement be?
5. How much memory do you have on your computer?

Run velveth:

```
velveth Velvet 31 -fastq -shortPaired
   SRR1029680_1.fastq SRR1029680_2.fastq
# To understand the options:
velveth -h | less
```

Now we can assemble the reads (Velvet is the output directory from velveth):

```
velvetg Velvet
```

6. Why does velvet have two steps, velvetg and velveth, instead of one?
7. Compare the results with the assembly from other assemblers once you have generated them. The velvetg assembly output file is contigs.fa. For example, to find the longest contig:

```
cat contigs.fa | grep \> | cut -d _ -f 4 | sort -n
```

8. Optional: Improve the velvetg assembly by optimizing some of the options. You may take a look at for some tips on which settings to change:
http://evomics.org/learning/assembly-and-alignment/velvet/

4.3.1 Assembly statistics

Many tools have been made to generate assembly statistics and to make the process of evaluating their quality. If you want, you can try the stats.sh script that is included in the bbmap tool package (installation: https://anaconda.org/bioconda/bbmap, manual:https://jgi.doe.gov/data-and-tools/software-tools/bbtools/bb-tools-user-guide/statistics-guide/)

4.4 Assembly with spades

spades is a very popular tool for genome assembly. One of its advantages is that it alters the k-mer size depending on the coverage of the genome. For low coverage regions k-mer size should be low and for high coverage regions high (why?). It is easy to run. Run spades on your computer:

```
spades.py --pe1-1 SRR1029680_1.fastq --pe1-2
    SRR1029680_2.fastq -k 21,33,55,77 --careful -t 16 -o
    Spades
# Look what the options mean:
spades.py -h
```

The values were chosen by using the help on:
https://cab.spbu.ru/files/release3.15.3/manual.html#sec3.4

9. How big is the genome? Use the scaffolds.fasta file.

4.5 Minia

If time allows, try to run an assembly with minia as well. You can try out a few different kmer sizes (pair-up and exchange results!).

- Did you get an improvement in contig sizes?
- Which kmer size worked best for you?

4.6 Reference based assembly

Since we have a genome of a close relative, we can use that to improve our assembly. Again use https://www.ncbi.nlm.nih.gov/genome/1659. Do the following steps:

- Scroll down to the “Genome” section of the page.

  - Click the link to the reference genome assembly called ASM978v1.
  - In the right panel, click on ‘‘Download’’ button.
  - Select the refseq option and make sure only the ‘‘Genomic Sequences FASTA’’ checkbox is filled.

- Download ncbi$_{d}ataset_{d}ataset_{0}00009785.1\_genomic.fna.gz$

10. How big is the genome?

In order to imporve our assembly using this assembly we will use a tool called RagTag to scaffold our assembled contigs into larger scaffolds.
Go to https://github.com/malonge/RagTag and follow the installation instructions. (Please note that RagTag also require minimap2 or a similar aligner to function, so you will also need to install that https://github.com/lh3/minimap2=
Once Minimap2 and RagTag are installed, use ‘‘scaffold’’ command in RagTag to scaffold one of your genome assemblies against the ASM978v1 reference genome that you downloaded and compare the new scaffolded assembly against your other assemblies.

11. How big is the new assembly?

12. How many N:s have been inserted between contigs to build scaffolds?

13. Was the method successful?

Here we created scaffolds by using a reference genome. Another way is to have sequence data with long inserts, like 3 kb-20kb (mate pair data), long-read sequence data, or chromatin conformation data, such as Hi-C. The term used for creating scaffolds is *scaffolding*. Today genome assemblies are often carried out using another sequencing technologies, like PacBio and Nanopore. However, it’s more expensive, have lower throughput and still software development has not reached the same level as it has for Illumina. For small genomes Illumina is working very well but nowadays the recommendation for a high quality draft genome is to use PacBio sequencing.

Index

Assemblers
  minia, 11
  velvet, 12

fasta, 9
fastq, 9

minia, 11

scaffolding, 15
spades, 11
SRA toolkit, 7

velvet, 11
velvetg, 12
velveth, 12