Introduction to Genome Assembly with SPAdes

Overview

In this tutorial, we will guide you through the process of assembling genomic reads using the SPAdes assembler. Genome assembly is a critical step in bioinformatics, as it involves reconstructing the original genome from short DNA sequences generated by sequencing technologies. A well-assembled genome is essential for various applications, including comparative genomics, functional genomics, and evolutionary studies. For this practical we will be using the cleaned reads that we generated from our read_cleaning tutorial so if you haven't don't that tutorial i would advise that you do in order to complete this section.

Importance of Genome Assembly

Accurate genome assembly is vital for understanding the structure, function, and evolutionary relationships of genomes. High-quality genome assemblies allow researchers to:

- Identify genes and regulatory elements.
- Understand genetic variation within populations.
- Perform annotation and functional studies on the genome.
- Facilitate downstream analyses, such as variant calling and phylogenetic studies.

This tutorial emphasizes the basic operation of SPAdes and serves as a stepping stone toward more complex genome assembly tasks. For a comprehensive understanding of SPAdes features and capabilities, we encourage users to consult the official documentation and relevant scientific publications.

Key Steps Covered in the Tutorial

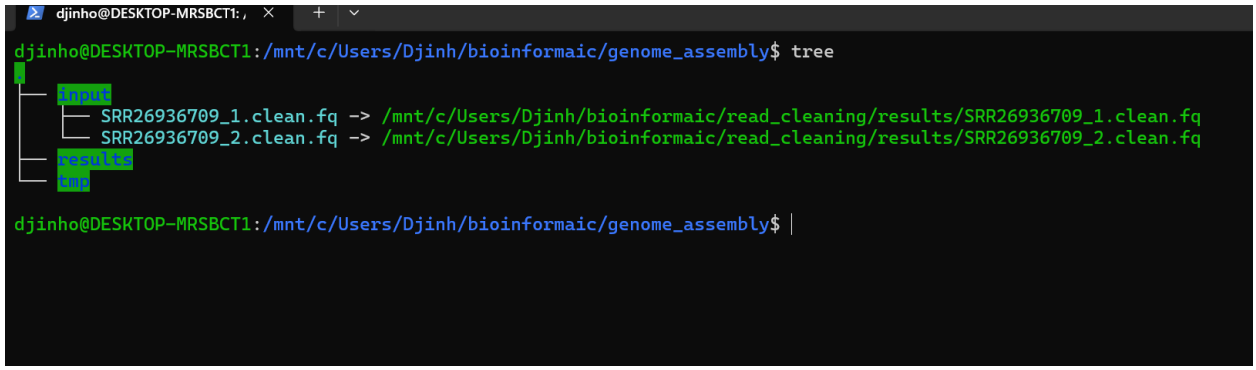
1. **Setting Up Directories**: We will create a structured directory to organize input data, temporary files, and results:
   mkdir genome_assembly

```
mkdir input tmp results
 cd genome_assembly
```

2. Create a systemic link between the clean reads and the input directory for the genome assembly:

```
cd /mnt/c/Users/Djinh/bioinformaic/genome_assembly/input
ln -s
/mnt/c/Users/Djinh/bioinformaic/read_cleaning/results/SRR26936709
_1.clean.fq .
ln -s
/mnt/c/Users/Djinh/bioinformaic/read_cleaning/results/SRR26936709
_2.clean.fq .
```

After running the tree command the output should be this :



3. **Assembling Reads**: Use the SPAdes command to assemble cleaned reads. Replace the paths with the correct ones from your read-cleaning tutorial. The image below is the output you should get. The command:
spades.py -o tmp -1 input/SRR26936709_1.clean.fq -2 input/SRR26936709_2.clean.fq

```
djinho@DESKTOP-MRSBCT1: ,

===== Read error correction started.


== Running read error correction tool: /usr/lib/spades/bin/spades-hammer /mnt/c/Users/Djinh/core_bioinformatics_workflows/genome_asse
mbly/tmp/corrected/configs/config.info

  0:00:00.001     4M / 4M    INFO    General                 (main.cpp                  :  75)   Starting BayesHammer, built from N/A
, git revision N/A
  0:00:00.002     4M / 4M    INFO    General                 (main.cpp                  :  76)   Loading config from /mnt/c/Users/Dji
nh/core_bioinformatics_workflows/genome_assembly/tmp/corrected/configs/config.info
  0:00:00.014     4M / 4M    INFO    General                 (main.cpp                  :  78)   Maximum # of threads to use (adjuste
d due to OMP capabilities): 4
  0:00:00.014     4M / 4M    INFO    General                 (memory_limit.cpp          :  49)   Memory limit set to 1 Gb
  0:00:00.014     4M / 4M    INFO    General                 (main.cpp                  :  86)   Trying to determine PHRED offset
  0:00:00.052     4M / 4M    INFO    General                 (main.cpp                  :  92)   Determined value is 33
  0:00:00.053     4M / 4M    INFO    General                 (hammer_tools.cpp          :  36)   Hamming graph threshold tau=1, k=21,
 subkmer positions = [ 0 10 ]
  0:00:00.053     4M / 4M    INFO    General                 (main.cpp                  : 113)   Size of aux. kmer data 24 bytes
      === ITERATION 0 begins ===
  0:00:00.063     4M / 4M    INFO   K-mer Index Building     (kmer_index_builder.hpp    : 301)   Building kmer index
  0:00:00.063     4M / 4M    INFO    General                 (kmer_index_builder.hpp    : 117)   Splitting kmer instances into 64 fil
es using 4 threads. This might take a while.
  0:00:00.066     4M / 4M    INFO    General                 (file_limit.hpp            :  32)   Open file limit set to 1024
  0:00:00.066     4M / 4M    INFO    General                 (kmer_splitters.hpp        :  89)   Memory available for splitting buffe
rs: 0.0830078 Gb
  0:00:00.066     4M / 4M    INFO    General                 (kmer_splitters.hpp        :  97)   Using cell size of 174080
  0:00:00.325   376M / 376M  INFO   K-mer Splitting          (kmer_data.cpp             :  97)   Processing /mnt/c/Users/Djinh/core_b
ioinformatics_workflows/read_cleaning/results/SRR26936709_1.clean.fq
  0:00:05.902   388M / 392M  INFO   K-mer Splitting          (kmer_data.cpp             : 107)   Processed 52028 reads
  0:00:10.865   388M / 392M  INFO   K-mer Splitting          (kmer_data.cpp             : 107)   Processed 109829 reads
  0:00:12.452   388M / 392M  INFO   K-mer Splitting          (kmer_data.cpp             :  97)   Processing /mnt/c/Users/Djinh/core_b
ioinformatics_workflows/read_cleaning/results/SRR26936709_2.clean.fq
```

4) As with most assemblers, SPAdes generates multiple output files. One key file you'll likely use in further analyses is the scaffolds.fasta file. To prepare for the next steps, copy this file to the results directory:

```
cp tmp/scaffolds.fasta results/
```

5) **Examining the Assembly Output**: Check the contents of the scaffolds.fasta file to understand its structure. You can view the first or last ten lines using:

```
head results/scaffolds.fasta
```

Or, to check the last 10 lines, use:

```
tail results/scaffolds.fasta
```

```
djinho@DESKTOP-MRSBCT1:/mnt/c/Users/Djinh/core_bioinformatics_workflows/genome_assembly$ head results/scaffolds.fasta
>NODE_1_length_449_cov_2.930108
CAGCAGCCGCGGTAATTCCAGCTCCAATAGCGTATATTAAAGTTGTTGCAGTTAAAAAGC
TCGTAGTCGAACTTCGGATCTGGCGGGATGGTCCGCCTTACGGTGTGTACTGTCCGGCCG
GATCTTACCTCTTGGTGAAGCCGTATGCCCTTTACTGGGTGTGCAGTGGAACCAGGAATT
TTACCTTGAGAAAATTAGAGTGTTCAAAGCAGGCAATTGCCCGAATACATTAGCATGGAA
TAATAGAATAGGACGTGCGGTTCTATTTTGTTGGTTTCTAGGATCGCCGTAATGATTAAT
AGGGACGGTCGGGGGCATTAGTATTCCGTTGCTAGAGGTGAAATTCTTAGATTTACGGAA
GACTAACATCTGCGAAAGCATTTGCCAAGGACGTTTTCATTGACCAAGGACGAAGGTTAG
GGGATCAAAAACGATTAGATACCCGCGTA
>NODE_2_length_448_cov_4.749326
djinho@DESKTOP-MRSBCT1:/mnt/c/Users/Djinh/core_bioinformatics_workflows/genome_assembly$
```

5) Assembly software typically provides statistics on the process, but the format of these outputs can vary across different assemblers. To generate a standardized report, you can use **Quast** (Quality Assessment Tool for Genome Assemblies). Run Quast on the scaffolds.fasta file without any additional options to obtain basic assembly statistics:

So install Quast first:
sudo apt install python3-pip
sudo pip3 install quast

6) Next, run Quast. The default Quast command uses a minimum contig length of 500 bp, but since these reads are shorter, we opted to use 400 bp instead. Use the following command:
/usr/local/bin/quast.py scaffolds.fasta --min-contig 400

Conclusion

By following this tutorial, you will gain practical experience in genome assembly using SPAdes, equipping you with essential skills for bioinformatics research. For a deeper understanding, consider exploring the official documentation and recent studies that provide insights into assembly strategies and quality assessments.