**Training Module**

**Next Generation Sequencing (NGS) Analysis**

**(De novo RNA-Seq Data Analysis)**

**Course Content:**

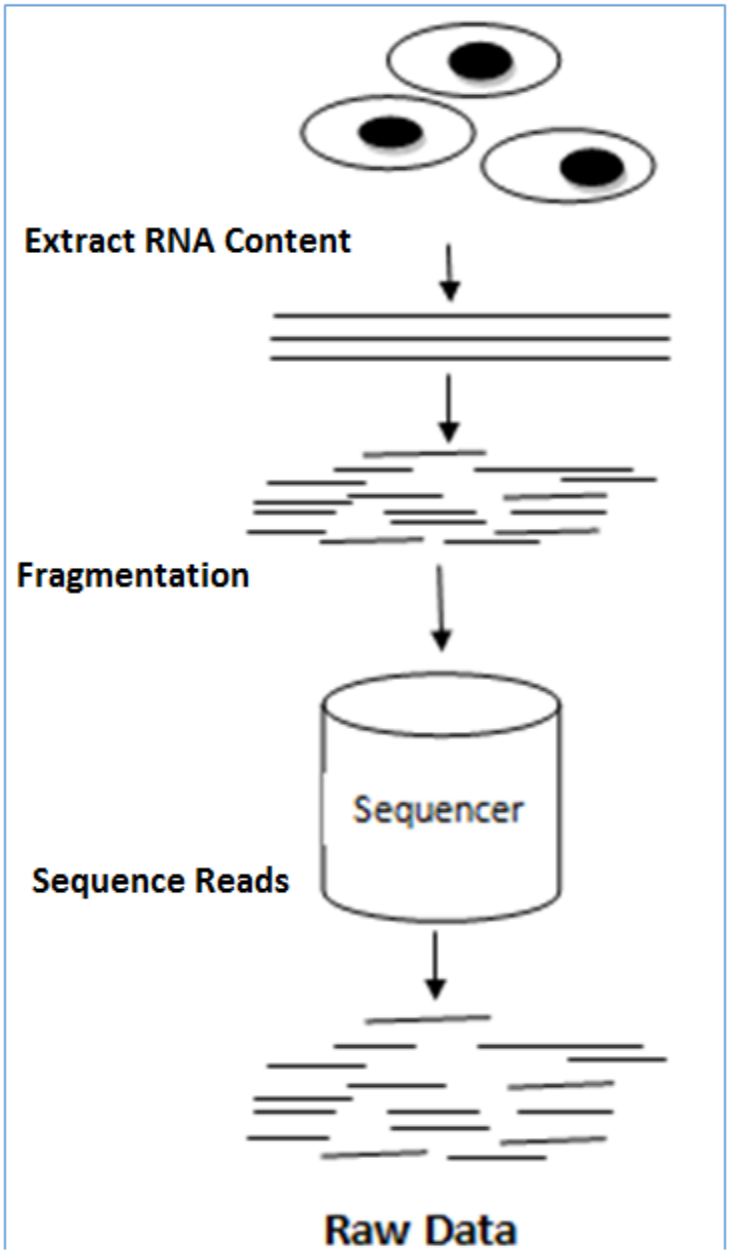
- Overview of NGS & detailed Understanding Denovo Assembly
- Data Retrieval (NCBI SRA) & Introduction to data types
- Read Quality Check (FastQC & Cutadapt)
- Assembly of Reads (MaSuRCA)
- Assembly Statistics (Assembly-stats)
- Gene Prediction & Annotation (Prokka)
- Assembled Genome and gene Visualization(Circos)

*Genome assembly refers to the process of taking a large number of short DNA sequences and putting them back together to create a representation of the original chromosomes from which the DNA originated. De novo genome assemblies assume no prior knowledge of the source DNA sequence length, layout or composition.*

*The goal of a sequence assembler is to produce long contiguous pieces of sequence (contigs) from these reads. The contigs are sometimes then ordered and oriented in relation to one another to form scaffolds. The distances between pairs of a set of paired end reads is useful information for this purpose.*

*The mechanisms used by assembly software are varied but the most common type for short reads is assembly by de Bruijn graph.*

*Genome assembly is a very difficult computational problem, made more difficult because many genomes contain large numbers of identical sequences, known as repeats. These repeats can be thousands of nucleotides long, and some occur in thousands of different locations, especially in the large genomes of plants and animals.*




Raw Data will be in **.fastq** format (standared file format)

There are other file format like **.sff / .abi/ .sra** etc which needs to be converted into .fastq format as for all NGS analysis tools mostly accept .fastq format

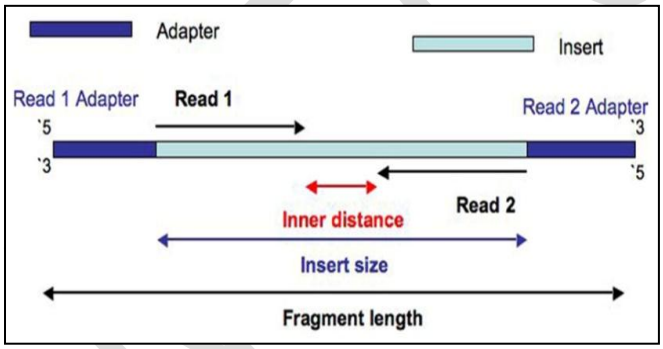
**Raw data can be of two types**

### ***Single End***

Single end sequencing involves sequencing DNA from only one end. In this less sequencing is required and it is used for general purposes like differential expression analysis.

### ***Paired End***

Paired-end sequencing allows users to sequence both ends of fragment and generate high quality, alignable sequence data. So, in this you will get two files one stores all the forward sequences and the other stores the reverse sequences in the exact same order as the first file. It is used for detection of genomic rearrangements and repetitive sequence elements, as well as gene fusion and novel transcripts.



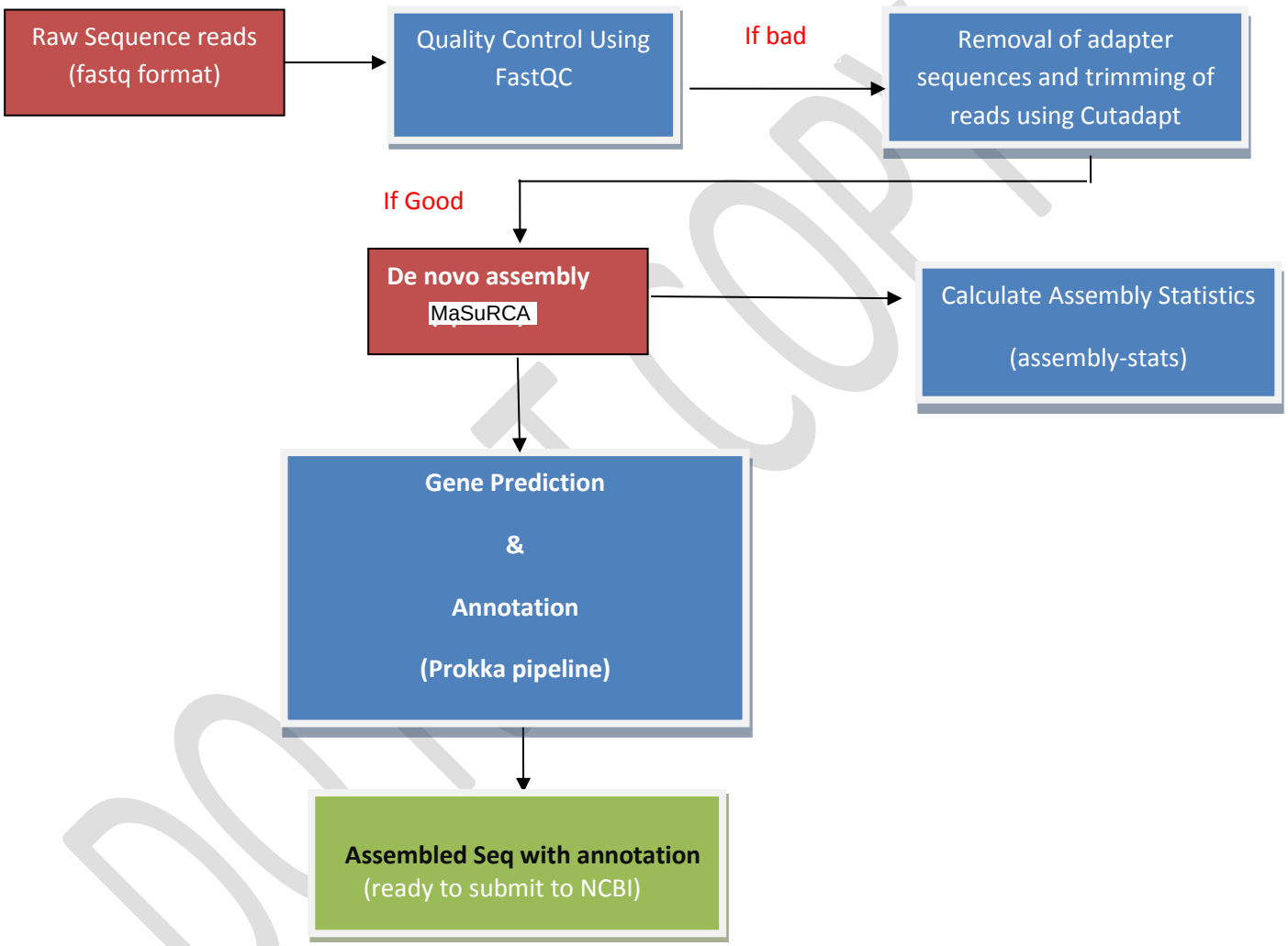

Paired-end Sequencing

**Applications of NGS**

1) DNA-Seq/ Genome Variant Detection/ Exome Sequencing
2) DenovoRNA-Seq (Transcriptome Assembly & Differential Expression profiling using sequencing)
3) ChIP-Seq
4) ***De novo* Genome Assembly**
5) Metagenomic analysis
6) Methylome-Seq etc.

In this module, we will be covering only **Denovo Genome Assembly analysis** in detail

# De novo Genome Assembly Pipeline



## As other short read assemblers, works in K-mer space (K=25)

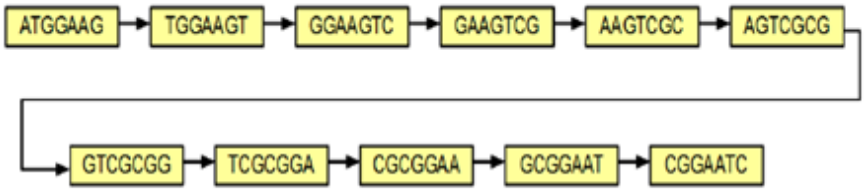
What are K-mers?

sequence ATGGAAGTCGCGGAATC

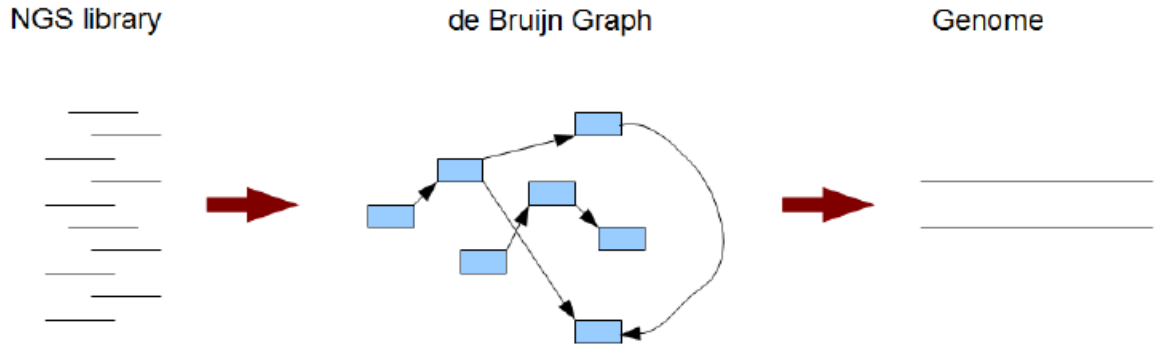
7mers

ATGGAAG
TGGAAGT
GGAAGTC
GAAGTCG
AAGTCGC
AGTCGCG
GTCGCGG
TCGCGGA
CGCGGAA
GCGGAAT
CGGAATC

de Bruijn graph




## What an assembler is trying to do



## Raw Data Download

Download raw data from below link

**NCBI SRA:** **https://www.ncbi.nlm.nih.gov/sra/?term=SRR088897**

**You can also use below link to download the fastq raw data**

**ENA:** **https://www.ebi.ac.uk/ena/data/view/SRR088897**

## Tools & Installation

- **Fastqc**
- **Cutadapt**
- **MaSuRCA**
- **Assembly-stats**
- **Prokka**
- **Circos**

## Fastq file format

FASTQ format is a text based format for storing both a biological sequence and its corresponding quality scores.
A FASTQ file contains four lines per sequence.

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

The last line contains ASCII value of the quality of the base in the second line. Quality scores of the identification of the nucleotide bases generated by automated DNA sequencing are measured in Phred quality score. Phred scale is the error probability of having called the base wrong. The most important use of Phred quality scores is the automatic determination of accurate, quality-based consensus sequences.

The sequence letter and quality score are each encoded with ASCII character for brevity. ASCII Values are as given in the table.

**ASCII printable code chart**

| Binary | Oct | Dec | Hex | Glyph |
|---|---|---|---|---|
| 010 0000 | 040 | 32 | 20 | (space) |
| 010 0001 | 041 | 33 | 21 | ! |
| 010 0010 | 042 | 34 | 22 | " |
| 010 0011 | 043 | 35 | 23 | # |
| 010 0100 | 044 | 36 | 24 | $ |
| 010 0101 | 045 | 37 | 25 | % |
| 010 0110 | 046 | 38 | 26 | & |
| 010 0111 | 047 | 39 | 27 | ' |
| 010 1000 | 050 | 40 | 28 | ( |
| 010 1001 | 051 | 41 | 29 | ) |

| Binary | Oct | Dec | Hex | Glyph |
|---|---|---|---|---|
| 100 0000 | 100 | 64 | 40 | @ |
| 100 0001 | 101 | 65 | 41 | A |
| 100 0010 | 102 | 66 | 42 | B |
| 100 0011 | 103 | 67 | 43 | C |
| 100 0100 | 104 | 68 | 44 | D |
| 100 0101 | 105 | 69 | 45 | E |
| 100 0110 | 106 | 70 | 46 | F |
| 100 0111 | 107 | 71 | 47 | G |
| 100 1000 | 110 | 72 | 48 | H |
| 100 1001 | 111 | 73 | 49 | I |

| Binary | Oct | Dec | Hex | Glyph |
|---|---|---|---|---|
| 110 0000 | 140 | 96 | 60 | ` |
| 110 0001 | 141 | 97 | 61 | a |
| 110 0010 | 142 | 98 | 62 | b |
| 110 0011 | 143 | 99 | 63 | c |
| 110 0100 | 144 | 100 | 64 | d |
| 110 0101 | 145 | 101 | 65 | e |
| 110 0110 | 146 | 102 | 66 | f |
| 110 0111 | 147 | 103 | 67 | g |
| 110 1000 | 150 | 104 | 68 | h |
| 110 1001 | 151 | 105 | 69 | i |

## FastQC

FastQC provides a simple way to do quality control checks on raw sequence data coming from high throughput sequencing pipelines. FastQC will work on data generated from Illumina, ABI Solid Technologies.

To download FastQC, go to this link. (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/)

Follow instruction to install it.

Or install using linux terminal

How to run FastQC?

```
sudo apt-get install fastqc
```

To get the Quality report of raw data

```
fastqc   SRR0088897_1.fastq SRR0088897_2.fastq
```

To get the help about fastqc

```
fastqc --help
```

If the quality of reads is good, then you can proceed with the mapping procedure otherwise use Cutadapt software to clean the data.

**For further detail documentation please visit at**
**http://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/**

## Cutadapt

Cutadapt finds and removes adapter sequences, primers, poly-A-tails and other types of unwanted sequence from the high-throughput sequencing reads.

For reference see https://cutadapt.readthedocs.org

How to Download and install cutadapt

To download and install Cutadapt

```
sudo pip install cutadapt
```

Cutadapt searches for the adapter in all reads and removes it when finds it. Basic commands in Cutadapt are as given below:

To trim a 3' adapter, AACCGGTT is the adapter we want remove, reads are read from input. fastq and results are stored in output. fastq

```
cutadapt  -a  AACCGGTT   -o  cutout_read1.fastq   read1.fastq
```

To trim low quality ends from reads before adapter removal, by default only 3' end of each read is quality trimmed

```
cutadapt   -q   10  -o cutout_read1.fastq  read1.fastq
```

To throw away processed reads shorter than N bases

```
cutadapt  -m  N  -o cutout_read1.fastq  read1.fastq
```

For paired end reads we need to follow two steps –

As the data selected is paired end sequencing hence need to follow the below two steps for the trimming based on phred score and adapter content.

```
cutadapt -q 30,30 -m 20 -b GATCGGAAGAGCACACGTCTGAACTCCAGTCACGCCAATATCTCGTATGC -o tmp.1.fastq -p tmp.2.fastq SRR088897_1.fastq SRR088897_2.fastq
```

```
cutadapt -q 30,30 -m 20 -b GATCGGAAGAGCACACGTCTGAACTCCAGTCACGCCAATATCTCGTATGC -o trimmed.2.fastq -p trimmed.1.fastq tmp.2.fastq tmp.1.fastq
```

**For further detail documentation please visit at**

**https://cutadapt.readthedocs.org/en/stable/guide.html**

Now perform fastqc with trimmed output from cutadapt again to check whether the raw data is good

```
fastqc trimmed.1.fastq trimmed.2.fastq
```

Now the raw data looks good and we can further process for assembly.

# Denovo Assembly

We have used MaSuRCA assembler tool for the Denovo genome assembly.

The MaSuRCA (Maryland Super Read Cabog Assembler) assembler combines the benefits of deBruijn graph and Overlap-Layout-Consensus assembly approaches. Since version 3.2.1 it supports hybrid assembly with short Illumina reads and long high error PacBio/MinION data.

To install, first download the latest distribution from ftp://ftp.genome.umd.edu/pub/MaSuRCA/. Then untar/unzip the package MaSuRCA-X.X.X.tgz, cd to the resulting folder and run './install.sh'. The installation script will configure and make all necessary packages. In the rest of this document, '/install_path' refers to a path to the directory in which './install.sh' was run.

## Running the assembler

**Overview**

**Step-1**

Create new directory Masurca and Copy in your assembly directory the template configuration file '/install_path/sr_config_example.txt' which was created by the installer with the correct paths to the freshly compiled software and with reasonable parameters.

**Setting Configuration file as below-**

```
# example configuration file

# DATA is specified as type {PE,JUMP,OTHER,PACBIO} and 5 fields:
# 1)two_letter_prefix 2)mean 3)stdev 4)fastq(.gz)_fwd_reads
# 5)fastq(.gz)_rev_reads. The PE reads are always assumed to be
# innies, i.e. --->.<---, and JUMP are assumed to be outties
# <---.--->. If there are any jump libraries that are innies, such as
# longjump, specify them as JUMP and specify NEGATIVE mean. Reverse reads
# are optional for PE libraries and mandatory for JUMP libraries. Any
# OTHER sequence data (454, Sanger, Ion torrent, etc) must be first
# converted into Celera Assembler compatible .frg files (see
# http://wgs-assembler.sourceforge.com)
DATA
PE= pe 90 90  /home/arraygen/Desktop/Denovo-Assembly/trimmed.1.fq
/home/arraygen/Desktop/Denovo-Assembly/trimmed.2.fq
#JUMP= sh 3600 200  /FULL_PATH/short_1.fastq  /FULL_PATH/short_2.fastq
#pacbio reads must be in a single fasta file! make sure you provide absolute
path
#PACBIO=/FULL_PATH/pacbio.fa
#OTHER=/FULL_PATH/file.frg
END
```

**PARAMETERS**

```
#set this to 1 if your Illumina jumping library reads are shorter than 100bp
EXTEND_JUMP_READS=0
#this is k-mer size for deBruijn graph values between 25 and 127 are
supported, auto will compute the optimal size based on the read data and GC
content
GRAPH_KMER_SIZE = auto
#set this to 1 for all Illumina-only assemblies
#set this to 1 if you have less than 20x long reads (454, Sanger, Pacbio) and
less than 50x CLONE coverage by Illumina, Sanger or 454 mate pairs
#otherwise keep at 0
USE_LINKING_MATES = 0
#specifies whether to run mega-reads correction on the grid
USE_GRID=0
#specifies queue to use when running on the grid MANDATORY
GRID_QUEUE=all.q
#batch size in the amount of long read sequence for each batch on the grid
GRID_BATCH_SIZE=300000000
#coverage by the longest Long reads to use
LHE_COVERAGE=30
#this parameter is useful if you have too many Illumina jumping library
mates. Typically set it to 60 for bacteria and 300 for the other organisms
LIMIT_JUMP_COVERAGE = 60
#these are the additional parameters to Celera Assembler.  do not worry about
performance, number or processors or batch sizes -- these are computed
automatically.
#set cgwErrorRate=0.25 for bacteria and 0.1<=cgwErrorRate<=0.15 for other
organisms.
CA_PARAMETERS = cgwErrorRate=0.25
#minimum count k-mers used in error correction 1 means all k-mers are used.
one can increase to 2 if Illumina coverage >100
KMER_COUNT_THRESHOLD = 1
#whether to attempt to close gaps in scaffolds with Illumina data
CLOSE_GAPS=1
#auto-detected number of cpus to use
NUM_THREADS = 16
#this is mandatory jellyfish hash size -- a safe value is
estimated_genome_size*estimated_coverage
JF_SIZE = 200000000
#set this to 1 to use SOAPdenovo contigging/scaffolding module.  Assembly
will be worse but will run faster. Useful for very large (>5Gbp) genomes from
Illumina-only data
SOAP_ASSEMBLY=0
END
```



We need to calculate read statistics using below command

```
for i in *.fq; do echo "$i   "; awk 'BEGIN { t=0.0;sq=0.0; n=0;} ;NR%4==2
{n++;L=length($0);t+=L;sq+=L*L;}END{m=t/n;printf("total %d avg=%f stddev=%f\n",n,m,sq/n-m*m);}'
$i; done
```

**Run the above command in the same folder where the both fastq file exists.**

**Now we have to run Masurca with configuration file**

```
/home/arraygen/NGS_Tools/masurca-master/MaSuRCA-3.2.6/bin/masurca   sr_config_example.txt
```

**Change the path of Masurca tool as your path**

**With the above command another script will be generated assemble.sh and run the script as below for further Denovo assembly**

```
./assemble.sh
```

**Output of Masurca**

There are many files and folders generated by Masurca assembler but there are two files which will have final assembled sequence

---- Go to CA folder

--- check for

1.scaffolds.ref.fa --- file contains all scaffolds sequences

2.final.genome.scf.fasta --- file contains all final assembled genome sequences from scaffolds sequences

Further generate the genome assembled assembly statistics using assembly stats and the program can be downloaded using https://github.com/sanger-pathogens/assembly-stats

**To get the assembly statistics run the below command -**

```
assembly-stats  final.genome.scf.fasta
```

**Assembly stats Output**

```
arraygen@arraygen-System-Product-Name:~/Desktop/Denovo-Assembly/Masurca/CA$ '/ho
me/arraygen/NGS_Tools/assembly-stats-master/build/assembly-stats'  final.genome.
scf.fasta
stats for final.genome.scf.fasta
sum = 2874599, n = 123, ave = 23370.72, largest = 143163
N50 = 54947, n = 17
N60 = 46065, n = 22
N70 = 37199, n = 29
N80 = 27868, n = 38
N90 = 16669, n = 51
N100 = 331, n = 123
N_count = 0
Gaps = 0
```

**All the assembled sequences were further rearranged using reference genome https://www.ncbi.nlm.nih.gov/nuccore/NC_007795.1**

**For the reference-based assembly we will be using Ragout (Reference-Assisted Genome Ordering UTility)**

It is a tool for chromosome assembly using multiple references. Given a set of assembly fragments (contigs/scaffolds) and one or multiple related references (complete or draft), it produces a chromosome-scale assembly (as a set of scaffolds).

The approach is based on the analysis of genome rearrangements (like inversions or chromosomal translocations) between the input genomes and reconstructing the most parsimonious structure of the target genome.

**Download: https://github.com/fenderglass/Ragout**

**To run the ragout tool, we need to make cnf.rcp file where all details of the assembled sequence as well as reference genome needs to be provided-**

---

**.references = NC_007795_Ref**

**.target = final.genome.scf**

**NC_007795_Ref.fasta = /home/arraygen/Desktop/Denovo-Assembly/Ref_Based_Assembly/NC_007795_Ref.fasta**

**final.genome.scf.fasta = /home/arraygen/Desktop/Denovo-Assembly/Ref_Based_Assembly/final.genome.scf.fasta**

**********

```
/home/arraygen/NGS_Tools/Ragout-master/ragout.py -o Reference cnf.rcp
```

**Output:**

| | |
|---|---|
| **Scaffolds:** | **1** |
| **Used fragments:** | **91** |
| **Scaffolds length:** | **2835462** |

**Output File: contig_scaffolds.fasta (This is the final complete genome assembled sequence)**

**Further assembled complete genome contig_scaffolds.fasta were searched for the gene prediction and functional annotation using PROKKA tool.**

**Prokka (rapid prokaryotic genome annotation)**

Whole genome annotation is the process of identifying features of interest in a set of genomic DNA sequences and labelling them with useful information. Prokka is a software tool to annotate bacterial, archaeal and viral genomes quickly and produce standards-compliant output files.

**Download: https://github.com/tseemann/prokka**

**Download the Prokka tool from above link and you need to build database first before you run for genome annotation.**

```
/home/arraygen/NGS_Tools/prokka-master/bin/prokka --setupdb
```

**Now run the Prokka with assembled genome fasta sequence contig_scaffolds.fasta**

```
/home/arraygen/NGS_Tools/prokka-master/bin/prokka  contig_scaffolds.fasta
```

**OUTPUT from Prokka:**

**## Output Files**

**| Extension | Description |**

**| .gff | This is the master annotation in GFF3 format, containing both sequences and annotations. It can be viewed directly in Artemis or IGV. |**

**| .gbk | This is a standard Genbank file derived from the master. gff. If the input to prokka was a multi-FASTA, then this will be a multi-Genbank, with one record for each sequence. |**

**| .fna | Nucleotide FASTA file of the input contig sequences. |**

**| .faa | Protein FASTA file of the translated CDS sequences. |**

**| .ffn | Nucleotide FASTA file of all the prediction transcripts (CDS, rRNA, tRNA, tmRNA, misc_RNA) |**

**| .sqn | An ASN1 format "Sequin" file for submission to Genbank. It needs to be edited to set the correct taxonomy, authors, related publication etc. |**

**| .fsa | Nucleotide FASTA file of the input contig sequences, used by "tbl2asn" to create the .sqn file. It is mostly the same as the .fna file, but with extra Sequin tags in the sequence description lines. |**

**| .tbl | Feature Table file, used by "tbl2asn" to create the .sqn file. |**

**| .err | Unacceptable annotations - the NCBI discrepancy report. |**

**| .log | Contains all the output that Prokka produced during its run. This is a record of what settings you used, even if the --quiet option was enabled. |**

**| .txt | Statistics relating to the annotated features found. |**

**| .tsv | Tab-separated file of all features: locus_tag,ftype,len_bp,gene,EC_number,COG,product |**

**Annotation summary**

**contigs: 1**

**bases: 2835462**

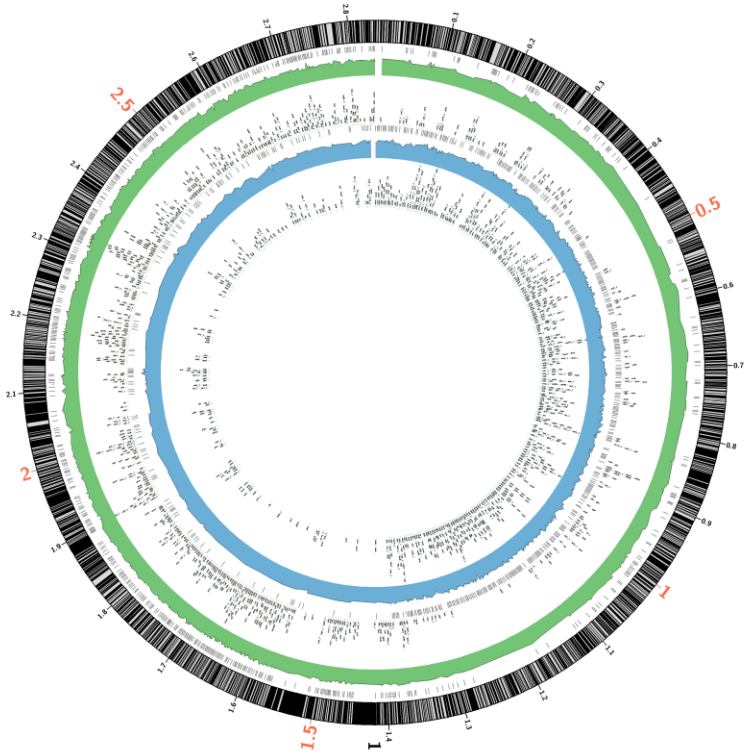
**CDS: 2592**

**tRNA: 45**

**tmRNA: 1**

**Further we can represent the annotation and gene information on complete assembled genome using Circos vizualization tool.**

**Please request you to follow manual for Circos vizualization-**




**We have also attached all the input and output results along with this manual.**

************************************Thank you **************************************