Introduction:

Next Generation Sequencing (NGS) is a term that applies to many new sequencing technologies. The drastic increase in speed and cost of these novel methods are changing the world of sequence data. A plethora of raw sequence data is currently available to be studied and new datasets are generated by these new sequencing methods faster than the raw data can be thoroughly analyzed (Bubnoff, 2008). Each method of sequencing has its own set of anti-limitation and limitations, yet no technology currently developed can generate a full genomic sequence. The genome is produced in fragments or short reads and then must be "assembled" (through computational methods) in order to piece together the fragments (Nagarajan and Pop, 2013). Sequence assembly is computationally intensive and it is almost nearly impossible to verify accuracy. There are many algorithms used for sequence assembly and many, if not all depend on the characteristics of the input data. However, simply joining the fragments together is not the only step to sequence assembly. There are three main steps—data quality control, assembly, and assembly verification (Magoc, et al. 2013). Many sequence assembly programs do not conduct all three steps, yet the data quality control is just as critical to a study as the sequence assembly or the assembly verification. The process of these three steps collectively is referred to as an assembly pipeline or workflow and in order to complete the workflow, two and sometimes three separate programs are often required. NGS technologies have greatly increased the need and use of assembly pipelines. Scientists conducting the experiments that generate the sequence data rarely have the computational experience to run the programs required to conduct their full study. This is due to the fact that most of the developed programs are a command-line interface and require significant skills in programming or computational science. As one can imagine, this creates a critical need for pipeline interfaces. The proposal of the overall NGS pipeline project is to create pipelines for four different computationally intensive processes required for scientific studies—genome assembly, genome annotation, RNA-seq, and variant calling. This report focuses on the pipeline intended for genome or sequence assembly and the different ways in which assembly can be refined. Eventually, the three steps needed for a complete sequence assembly (see Appendix B, figure 3) will be combined into a single script with an interface that allows researchers to efficiently and easily complete their assembly projects.

Methods:

Assembly was tested using genome data from the bacteria, *Vibrio gazogenes.* The data was collected by the Smithsonian Institution (Dikow, et al. 2013). As provided, the data sets were too large for a successful run through any assembler program and in order to obtain a usable dataset, the *V. gazogenes* genome files were trimmed using either Trimmomatic or BBtools. Trimmomatic is a sequence preprocessing tool developed especially for paired-end reads and BBtools is explained in the BBtools section below. A difficulty in processing paired-end reads is the need for the positional order of each read to be conserved throughout all preprocessing steps. Paired-end data is typically found in two FASTQ files, the forward reads and the reverse reads. (See Appendix A for more information regarding Paired-end data.) The reads would have been generated in a specific order that aligns the forward reads with their respective reverse reads.

Resources:

Software:

The following software programs were used throughout the project.

Trimmomatic:

Trimmomatic is a special preprocessing tool that is successfully and efficiently able to identify the adapter sequences and filter the reads based on quality while still maintaining the position and order of the reads (Trimmomatic article). Once the trimmed datasets were generated, they were run on multiple assembler programs. This report focuses on the results from SPAdes and SOAPdenovo2 with k-mer sizes of 21, 33, 55, and 71. A random 50 % of the trimmed database was selected as a "subset." This subset was run through both SPAdes and SOAPdenovo2 with k-mer sizes of 51, 61, 71, 81 and 91. The assembled outputs were run through QUAST which returns statistics used for a comparison of output (Gurevich, et al. 2013). Statistics chosen for comparison at this point in the project are number of contigs, genome size N50, and Guanine/Cytosine content.

BBtools:

Another set of tools were also used for reducing dataset size, BBtools. BBtools has many programs that do a variety of processing on a dataset. (BBtools reference) For the purpose of our project, we used a combination of *bbnorm* and *bbtrim*. *Bbnorm* trimmed the data by normalization and reduction of coverage before *bbtrim* trimmed the low-quality reads. Normalization removes any redundancy within the data. Genome coverage is the total length of all reads divided by the number of base pairs in the genome. Reducing the redundancy of reads would in directly reduce the coverage (Illumina paper). The smaller dataset output from *bbnorm* and *bbtrim* was also run through the different assembler programs, SPAdes and SOAPdenovo2 with k-mer sizes of 21, 33, 55, and 71. The assemblies were run through QUAST and the resulting statistics were compared to assess the quality of the assembly. For examples of how to run bbnorm and bbtrim, see Appendix C. The differing methods of trimming the dataset will be compared to determine if one trimming method produces better assemblies. This outcome could vary among assembler programs, as some are known to not work well with normalized data.

SPAdes:

SPAdes is a sequence assembler program that uses a novel form of the de Bruijn graph—*multisized de Bruijn*. There are four explicit stages to SPAdes assembly which include graph construction, k-mer adjustment, paired assembly graph, contig construction. SPAdes can detect the k-mer size for best utilization of assembly after it constructs the initial graph. The new multisized de Bruijn graph implements new methods for "error correction" of constructed graph (Bankevich, et al. 2012). Graph error correction typically refers to removing pieces of the graph that do not have a high probability of being an accurate assembly. Error correction is a tough balance; too vigorous error correction can remove parts of the graph that were accurate representations of the assembly while too subtle error correction does not provide an easily readable and usable assembly (Li, et al. 2010). SPAdes construction method of the multisized de Bruijn graph presents the user with the option to backtrack all the steps of graph construction. For an example of a call to SPAdes, see Appendix C, figure 6.

SOAPdenovo2:

The SOAPdenovo2 assembler was built for *de novo* assembly of large, mammalian genomes (Magoc, et al. 2013). De novo assembly means that there is no reference genome used to assemble the data. This is especially helpful when the data is from a species that does not have a full genome sequence available (Nagarajan and Pop, 2013). SOAPdenovo also uses the de Bruijn graph method but does require each new edge in the graph be linked to an existing sequence (Li, et al. 2010). For an example of a call to SOAPdenovo2 and the required configure file, see Appendix C, figure 7 and 8, respectively.

QUAST:

QUAST is a program the uses multiple quality-check algorithms that are typically only found in separate programs. Some of the algorithms produce statistics that need a reference genome in order to compute while others like Guanine/Cytosine (GC) content do not. QUAST produces many different tables and plots but this project focuses on the following four values: number of contigs, genome size, N50, and GC content. The number of contigs is self-explanatory and represents the total number of contigs in the assembly. This value is compared to the number of chromosomes found in the sample organisms' genome. Genome size is represented as the total number of bases in the assembly. If available, the genome size is compared to the reported genome size of the sample organism. If the genome of the organism has never been studied, the genome size will be compared to the genome size of closely related organisms. The N50 value represents a contig length that 50% of all contigs fall above. GC content is simply a count of all Guanine or Cytosine bases found in the assembly divided by the total number of bases. The higher percentage of G or C bases produces a more stable molecule (Gurevich et al. 2013).

Hardware:

The Nautilus super computer located at Oak Ridge National Laboratory was used due to its large amount of available memory. Nautilus uses an SGI Altix UV system and has one UV1000 node containing 128 Intel processors (1024 cores) with 4 terabytes of global shared memory and 8 GPUs (NICS).

Results:

The results for all runs of this project can be found in Tables (a)-(d) in Appendix D. None of the runs produced results close to the ideal number of contigs, 36. Thirty-six is the number of chromosomes found in the genome of *V. gazogenes* and therefore with 1 contig per chromosome being the best possible outcome, 36 contigs is the ideal result. The assembled genome size was steady around 4.5 million base pairs for most runs. The GC content also remained steady around 45% for most runs. SOAPdenovo2 runs of the Trimmomatic trimmed data k-mer sizes of 21 and 33 had results that varied significantly from the rest of the results with less than 20 contigs and a genome size of only about 11 thousand base pairs. The reason behind this difference is still being investigated.

Conclusion and Continuation:

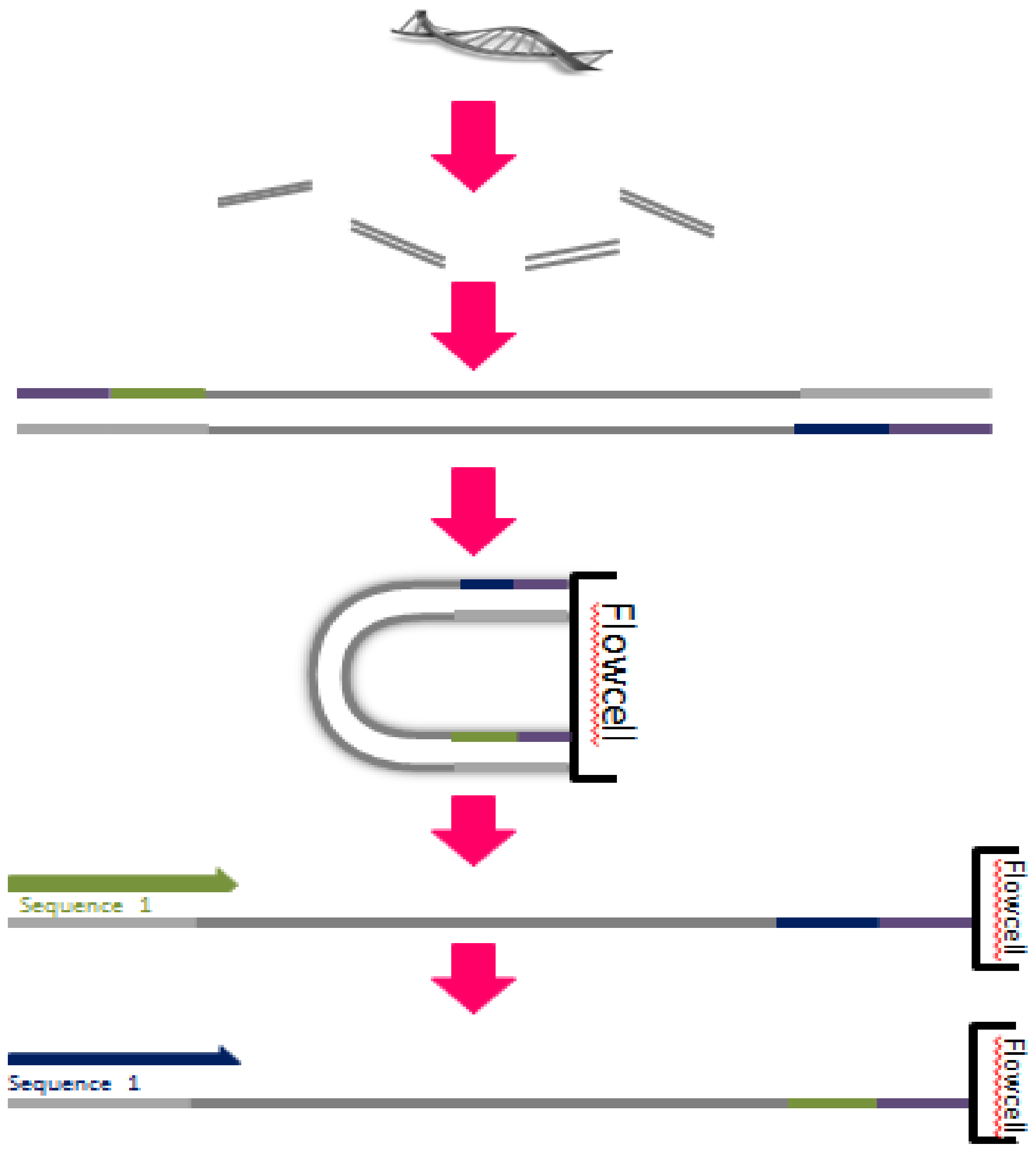
Despite Dikow, et al. (2013) reporting a genome size of over 6 million base pairs for the organism, we suggest that the genome of *Vibrio gazogenes* is between 4 to 4.5 million base pairs. Both normalized and quality trimmed data produced genomes of roughly this size. Also, species closely related to *V. gazogenes* typically have genomes of between 4.5 to 5 million base pairs. It is unlikely that species closely related to one another have genome sizes that differ in over a million base pairs. The stability of the results produced from

Trimmomatic processed sequences with the results from BBtools processed sequences suggests that neither method negatively affected the outcome of assembly. Once the method of using BBtools for pre-processing of sequences was successful, a collective bash script that runs *bbnorm, bbtrim*, SPAdes and/or SOAPdenovo2, and QUAST was written. This script does not remove the command line interface of each program but does greatly increase the ease of use for scientists who might need these programs used together. The amount of time spent on figuring out the right environment set up and parameters for each program has also been significantly decreased by the development of this collective script. Alongside, the script there is a general configure file for the SOAPdenovo2 runs which contains documentation on how to correctly set up said file. This collective script will continue to be developed into a script that can run multiple pipelines of assembly with varying parameters, not just the set of 4 programs currently incorporated. Eventually, these collective and general files will be used to remove the command line interface presented to the user and replace it with a web-based graphical interface.

References

1. Bankevich, Anton, Sergey Nurk, Dmitry Antipov, Alexey A. Gurevich, Mikhail Dvorkin, Alexander S. Kulikov, Valery M. Lesin, Sergey I. Nikolenko, Son Pham, Andrey D. Prjibelski, Alexey V. Pyshkin, Alexander V. Sirotkin, Nikolay Vyahhi, Glenn Tesler, Max A. Alekseyev, and Pavel A. Pevzner. "SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing." *Journal of Computational Biology* 19.5 (2012): 455-77. Web.
2. Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: A flexible trimmer for Illumina Sequence Data. *Bioinformatics*, btu170.
3. Bubnoff, Andreas Von. "Next-Generation Sequencing: The Race Is On." *Cell* 132.5 (2008): 721-23. Web.
4. Dikow, Rebecca B., and William Leo Smith. "Genome-level Homology and Phylogeny of Vibrionaceae (Gammaproteobacteria: Vibrionales) with Three New Complete Genome Sequences." *BMC Microbiology* 13.1 (2013): 80. Web.
5. Gurevich, A., V. Saveliev, N. Vyahhi, and G. Tesler. "QUAST: Quality Assessment Tool for Genome Assemblies." *Bioinformatics* 9.8 (2013): 1072-075. Web.
6. Illumina. "Genomic Sequencing." *Data Sheet:* Sequencing (2010): n. pag. Web.
7. Li, Yingrui, Yujie Hu, Lars Bolund, and Jun Wang. "State of the Art De Novo Assembly of Human Genomes from Massively Parallel Sequencing Data." *Human Genomics* 4.4 (2010): 271. Web.
8. Luo et al.: SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. GigaScience 2012 1:18.
9. Magoc, T., S. Pabinger, S. Canzar, X. Liu, Q. Su, D. Puiu, L. J. Tallon, and S. L. Salzberg. "GAGE-B: An Evaluation of Genome Assemblers for Bacterial Organisms." *Bioinformatics* 29.14 (2013): 1718-725. Web.
10. Nagarajan, Niranjan, and Mihai Pop. "Sequence Assembly Demystified." *Nature Reviews Genetics* 14.3 (2013): 157-67. Web.
11. NICS. “Nautilus.” http://www.nics.utk.edu/computing-resources/nautilus.

Appendix A



Caption for Figure 1: Diagram showing process of collecting paired-end reads. The genomic DNA is sequences into fragments which adaptors and primers are attached to (Green, Blue, and Purple ends). A cluster is formed and the sequences are read starting from both adaptors, producing the paired-end read. (Modified visual from Illumina Data Sheet: Sequencing).

Appendix B

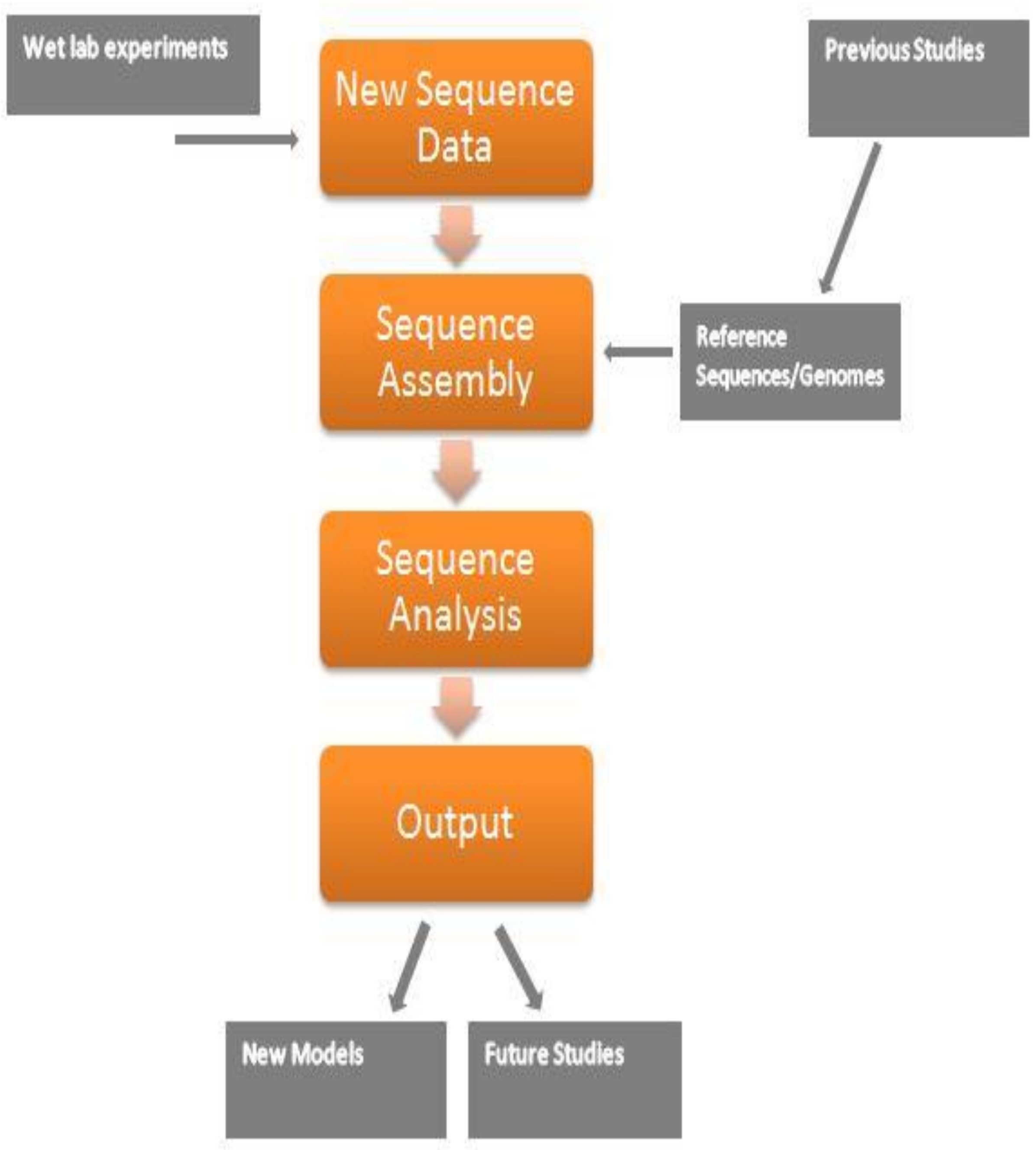


Figure 2: Diagram for the complete data analysis process. Orange rectangles are the actual analysis steps while the gray rectangles represent input from outside sources.

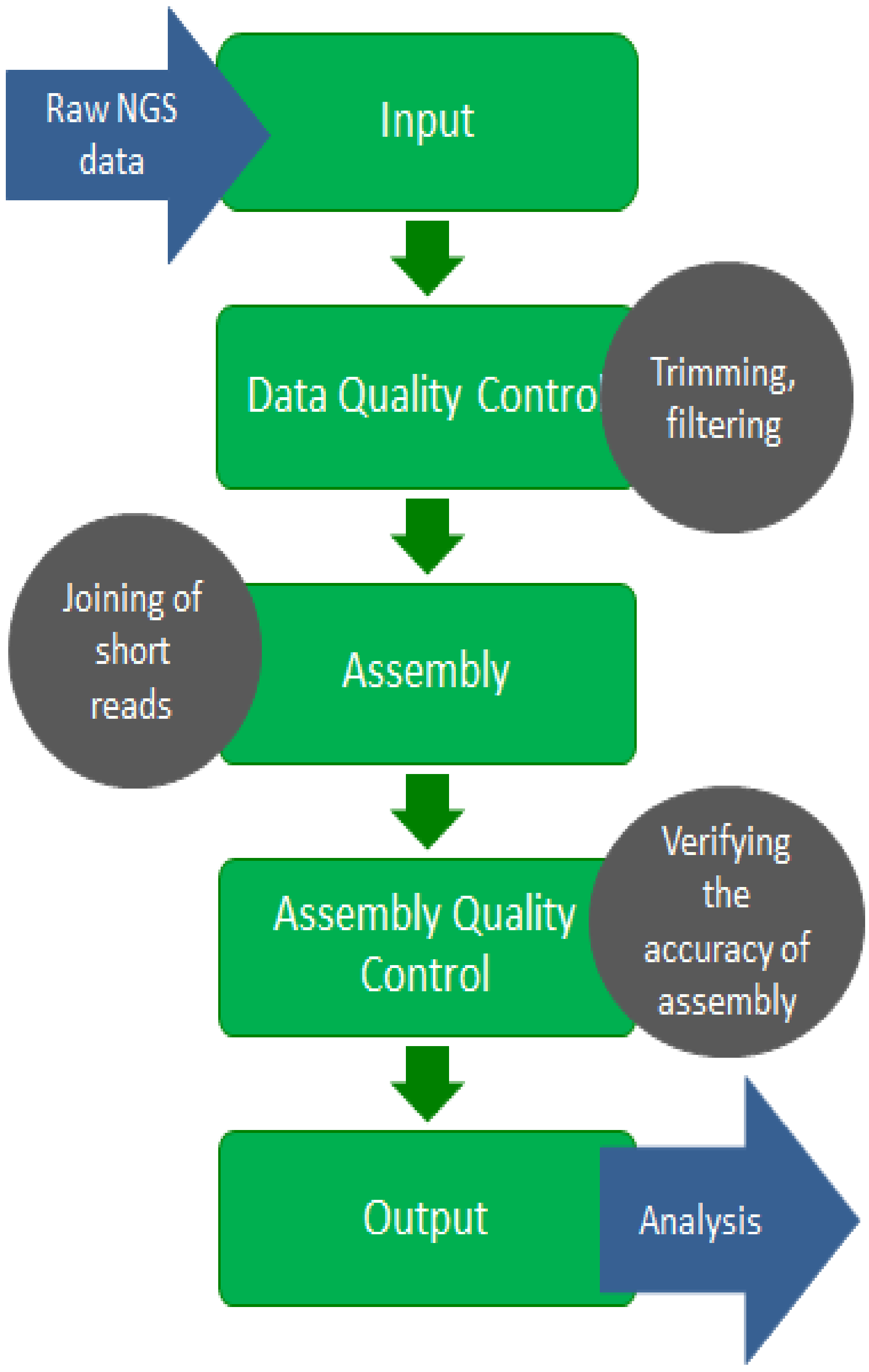
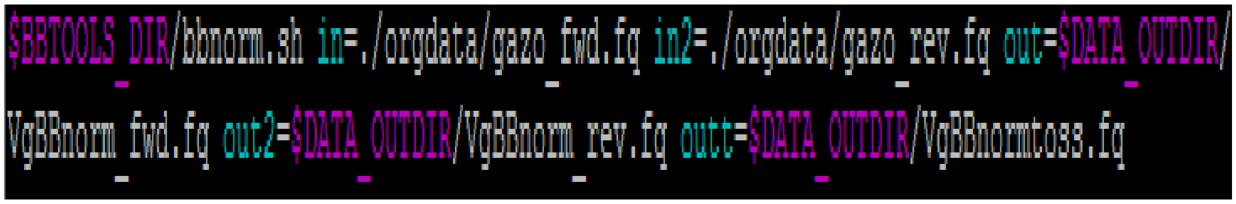


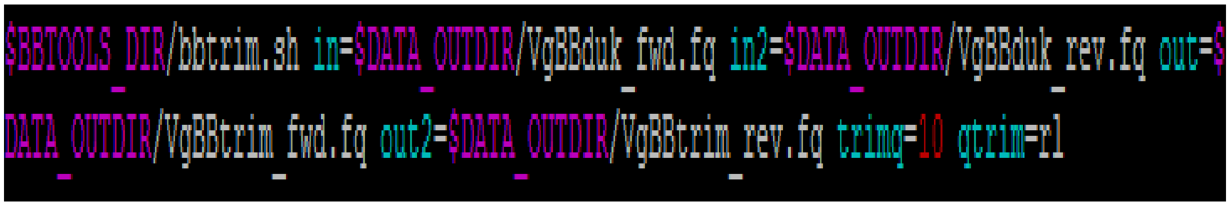
Figure 3: Diagram for the complete assembly process, beginning with raw sequence data. The assembled sequences must be checked for accuracy– a difficult step. Green rectangles are the steps, gray circles a short description. And blue arrows are steps that have their own process.

Appendix C



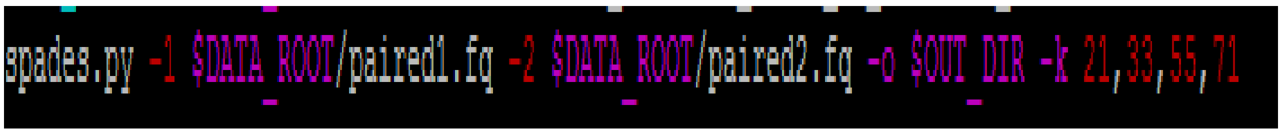
```
$BBTOOLS_DIR/bbnorm.sh in=./orgdata/gazo_fwd.fq in2=./orgdata/gazo_rev.fq out=$DATA_OUTDIR/
VgBBnorm_fwd.fq out2=$DATA_OUTDIR/VgBBnorm_rev.fq outt=$DATA_OUTDIR/VgBBnormtoss.fq
```

Figure 4: *bbnorm* call for a paired-end data set (gazo_fwd.fq and gazo_rev.fq)



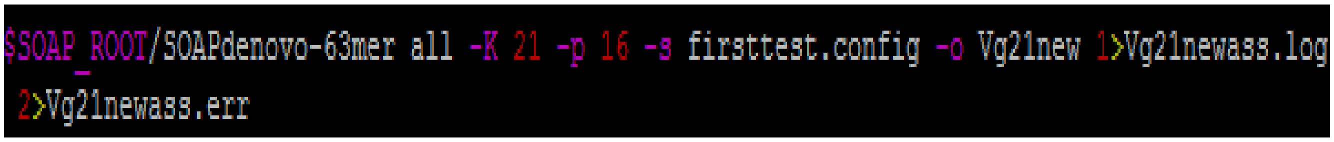
```
$BBTOOLS_DIR/bbtrim.sh in=$DATA_OUTDIR/VgBBduk_fwd.fq in2=$DATA_OUTDIR/VgBBduk_rev.fq out=$
DATA_OUTDIR/VgBBtrim_fwd.fq out2=$DATA_OUTDIR/VgBBtrim_rev.fq trimq=10 qtrim=rl
```

Figure 5: *bbtrim* call for normalized paired-end data set (VgBBnorm_fwd.fq and VgBBnorm_rev.fq)



```
spades.py -1 $DATA_ROOT/paired1.fq -2 $DATA_ROOT/paired2.fq -o $OUT_DIR -k 21,33,55,71
```

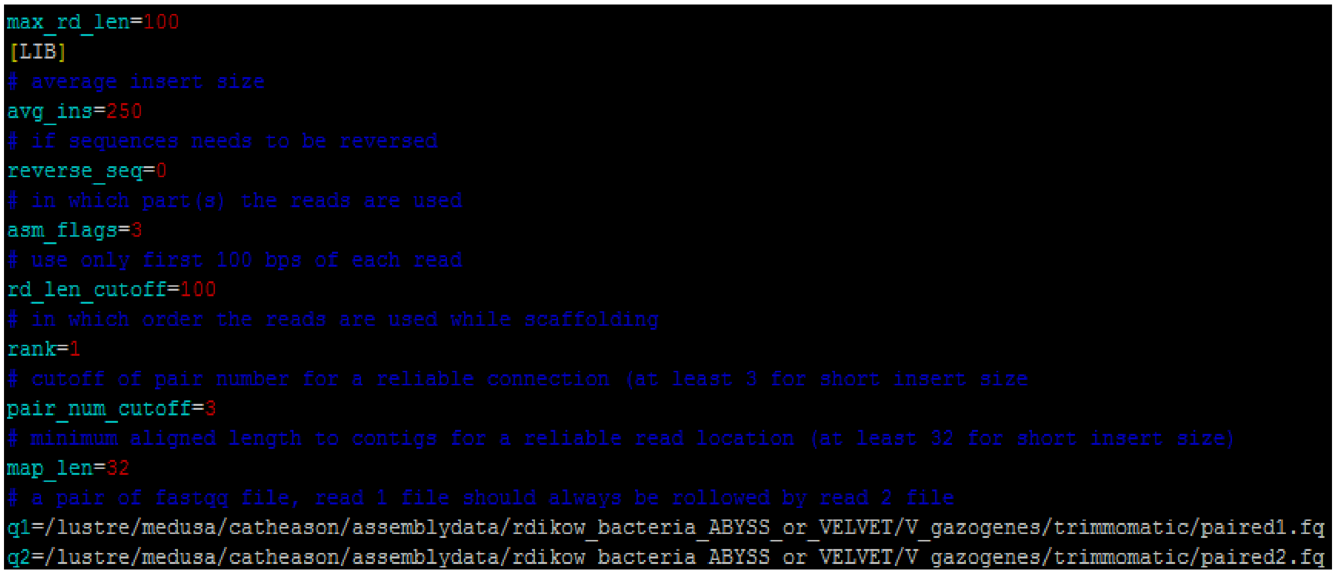
Figure 6: SPAdes call for paired-end data set (paired1.fq and paired2.fq)



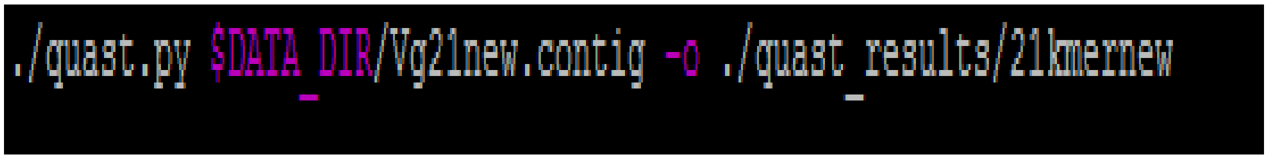
```
$SOAP_ROOT/SOAPdenovo-63mer all -K 21 -p 16 -s firsttest.config -o Vg21new 1>Vg21newass.log
 2>Vg21newass.err
```

Figure 7 (above): SOAPdenovo call with k-mer size K=21 for paired-end data set (paired1.fq and paired2.fq)

Figure 8 (below): Example of SOAPdenovo config file for paired1.fq and paired2.fq



```
max_rd_len=100
[LIB]
# average insert size
avg_ins=250
# if sequences needs to be reversed
reverse_seq=0
# in which part(s) the reads are used
asm_flags=3
# use only first 100 bps of each read
rd_len_cutoff=100
# in which order the reads are used while scaffolding
rank=1
# cutoff of pair number for a reliable connection (at least 3 for short insert size
pair_num_cutoff=3
# minimum aligned length to contigs for a reliable read location (at least 32 for short insert size)
map_len=32
# a pair of fastqq file, read 1 file should always be rollowed by read 2 file
q1=/lustre/medusa/catheason/assemblydata/rdikow_bacteria_ABYSS_or_VELVET/V_gazogenes/trimmomatic/paired1.fq
q2=/lustre/medusa/catheason/assemblydata/rdikow_bacteria_ABYSS_or_VELVET/V_gazogenes/trimmomatic/paired2.fq
```



```
./quast.py $DATA_DIR/Vg21new.contig -o ./quast_results/21kmernew
```

Figure 9: QUAST call for SOAPdenovo 21 k-mer results

Appendix D

(a)

SPAdes (Trimmomatic)				
Kmer size	# of contigs	Genome Size	N50	GC %
21	514	4,430,394	17,374	45.27
33	282	4,467,765	54,782	45.27
55	215	4,496,327	68,126	45.27
71	120	4,555,395	246,573	45.32
Subset 51	201	4,468,133	61,386	45.30
Subset 61	193	4,485,523	68,843	45.31
Subset 71	180	4,499,332	79,631	45.32
Subset 81	173	4,510,565	88,093	45.33
Subset 91	88	4,545,153	262,031	45.36

(b)

SOAPdenovo2 (Trimmomatic)				
Kmer Size	# of contigs	Genome Size	N50	GC %
21	16	11,398	690	42.96
33	17	11,766	690	41.00
55	1,385	968,669	685	46.87
71	444	4,448,857	18,563	45.33
Subset 51	1,481	4,321,140	4,296	45.39
Subset 61	309	4,459,372	29,329	45.30
Subset 71	206	4,481,934	55,249	45.30
Subset 81	172	4,499,317	75,768	45.32
Subset 91	159	4,519,076	100,098	45.34

Tables (a) and (b): (a) is for the assembly of Trimmomatic trimmed data through SPAdes while table (b) is for the same but using SOAPdenovo2. Both table (a) and (b) show number of contigs, genome size, N50, and GC % statistics for k-mer sizes 21,33,55,71 and a random 50% subset of data's statistics for k-mer sizes 51,61,71,81, and 91.

(c)

SPAdes (BBtools)				
Kmer Size	# of contigs	Genome Size	N50	GC %
21	506	4409861	17,893	45.29
33	263	4445712	49,223	45.30
55	190	4474737	65,281	45.30
71	106	4532943	167,499	45.35

(d)

SOAPdenovo2 (BBtools)				
Kmer Size	# of contigs	Genome Size	N50	GC %
21	770	4389210	9,940	45.29
33	379	4430953	24,090	45.28
55	202	4467392	62,696	45.30
71	169	4488672	81,399	45.31

Tables (c) and (d): (c) is for the assembly of Bbtool trimmed data through SPAdes while (d) is for the same but through SOAPdenovo2. Both tables (c) and (d) show number of contigs, genome size, N50, and GC % statistics for k-mer sizes 21, 33, 55, and 71.