The Pennsylvania State University

The Graduate School

VIRAMP: A GALAXY-BASED VIRUS GENOME ASSEMBLY PIPELINE

A Thesis in

Integrative Biosciences

by

Yinan Wan

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2014

The thesis of Yinan Wan was reviewed and approved* by the following:

Istvan Albert
Associate Professor, Bioinformatics, Biochemistry and Molecular Biology
Thesis Advisor

Moriah L. Szpara
Assistant Professor of Biochemistry & Molecular Biology

Cooduvalli S. Shashikant
Associate Professor of Molecular and Development Biology
Co-Director, IBIOS Graduate Program Option in Bioinformatics and Genomics

Peter Hudson
Willaman Professor of Biology
Director, Huck Institutes of the Life Sciences

*Signatures are on file in the Graduate School

ABSTRACT

Background

Advances in next generation sequencing make it possible to obtain high-coverage sequence data for large numbers of viral strains in a short time. However, since most bioinformatics tools are developed for command line use, the selection and accessibility of computational tools for genome assembly and variation analysis limits the ability of individual scientist to perform further bioinformatics analysis.

Findings

We have developed a multi-step viral genome assembly pipeline named VirAmp that combines existing tools and techniques and presents them to end users via a web-enabled Galaxy interface. Our pipeline allows users to assemble, analyze and interpret high coverage viral sequencing data with an ease and efficiency that previously was not feasible. Our software makes a large number of genome assembly and related tools available to life scientists and automates the currently recommended best practices into a single, easy to use interface. We tested our pipeline with three different datasets from human herpes simplex virus (HSV).

Conclusions

VirAmp provides a user-friendly interface and a complete pipeline for viral genome assembly and analysis. We make our software available via an Amazon Elastic Cloud disk image that can be easily launched by anyone with an Amazon web service account. A demonstration version of our system can be found at http://www.viramp.com. We also maintain detailed documentation on each tool and methodology at http://docs.viramp.com.

TABLE OF CONTENTS

List of Figures vi

List of Tables vii

Acknowledgements viii

Chapter 1 Introduction 1

Chapter 2 Background and Related Work 3

- Whole-genome variation analysis strategy 3
  - Mapping against a reference genome 3
  - *de novo* Assembly 4
- Viral Genome Related Assemblers 5

Chapter 3 Assembly Pipeline Description 7

- Data preprocessing 8
- Error correction and Coverage Reduction 9
- Two-step *semi- de novo* assembly 11
  - *de novo* assembly 11
  - Reference-based schaffolding 12
- Information Recovery 13
- Assembly Assessment and Variation Analysis 14
  - Assembly Assessment 14
  - Assembly-Reference Comparison Report 15
  - Circos Graph for genome alignment visualization 15
  - Variation Analysis 16
- Recognising Errors in Assemblies using Paired Reads 17

Chapter 4 Example analysis 20

- Data and Computational Resources Description 20
- Comparing the performances at each step of the assembly pipeline 21
- Comparing the final assemblies to the reference genome 23
- Comparing VIRAMP with other popular assemblers 26
- REAPR reference-free evaluation of genome assembly 28

Chapter 5 Discussion 31

- Final Gap Filling 31
- Repetitive Region 33
- Assemble genome using single-end reads 33

Chapter 6 Conclusion 35

Bibliography 36

Appendix Demonstration of VirAmp platform 38

LIST OF FIGURES

Figure 3-1 Overview of the VIRAMP pipeline 15

Figure 3-2 Demonstration of Read coverage before and after digital normalization 18

Figure 3-3 Demonstration of SSPACE scaffolding and extension..... 21

Figure 3-4 Circos graph projecting the comparison between assembly and reference genome..... 24

Figure 3-5 Demonstration of REAPR core algorithm FCD calculation [18] 25

Figure 3-6 Gap identification in REAPR[18]..... 25

Figure 4-1 Comparing the final assembly with the reference genome..... 31

Figure 4-2 VIRAMP visualization of the alignment between the final assembly and reference genome. 33

Appendix Figure 1 Settings to run the VirAmp pipeline..... 44

Appendix Figure 2 A published Galaxy workflow connects individual tools in the platform and demonstrates VirAmp paired-end assembly. 46

Appendix Figure 3 VirAmp with result file at pipeline complete status..... 47

LIST OF TABLES

Table 3-1 Example of Assembly-Reference Comparison Report 15

Table 4-1 Statistics of assemblies at each step of the VIRAMP pipeline 21

Table 4-2 The contigs coordinates in reference gnome..... 21

Table 4-3 Comparison report between the final assembly and reference genome..... 23

Table 4-4 Comparing VirAmp results with three other popular Assembly pipelines..... 27

Table 4-5 Reapr evaluation of assemblies from different pipelines..... 29

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Istvan Albert, who generously supported me during my graduate education. His guidance has helped me in all aspect of my research as well as in developing my verbal and written skills. At the same time he has also offered me advice and support beyond the academic while helping me through various difficult situations.

I would like to thank Prof. Moriah Szpara for starting the collaboration that lead to this thesis project, sharing her insights and thoughts while providing me with the training that allowed me to complete this project.

I would also like to thank Prof. Cooduvalli Shashikant for being my committee member. In his role as the co-director in Bioinformatics & Genomics program, Dr.Shashikant also offered me help and support all the way along for the past three years, helping me go through a lot of hard choices and decisions.

Special acknowledgement goes to Yunfei Li and Aswathy Sebastian, as lab colleagues, whose daily discussion and collaborations have helped me greatly learn the technical know how as well as the best approaches to communicate and interact with other scientists. I would like to thank the members of the Szpara Lab, for helping me improve my understanding of the challenges of virology as well as general biology. My sincere thanks goes to all my fellow classmates in Bioinformatics & Genomics program. I have spent three incredible years in this program with help and support from all of you.

Finally I would like to thank my parents, who always support me and provide me a warm home in various complicated situations. I will do better.

Chapter 1

Introduction

In this work we present VirAmp, a fast and efficient virus assembly pipeline that is able to handle high throughput sequencing datasets with highly variable coverage. VirAmp is a multilayered, semi- *de novo* assembly pipeline that makes use of multiple *de novo* assemblers that are integrated with a reference-guided scaffolding procedure as well as a k-mer based error correction model. The VirAmp project's goals are to create a methodology that allows non-technical users to easily assemble genomes of DNA viruses. We have validated our pipeline using data from multiple herpes simplex virus 1 (HSV-1) strains, that encompass a wide range of mutations, including SNPs, small and large INDELs, short sequence repeats (SSRs) that might occur in all virus genomes. We demonstrate that our approach can produce genomes with a quality that is better than that created with previously published methods while requiring substantially fewer computational resources (~90 min using 4GB ram one single CPU). In addition we also include other additional functionality to allow researchers to assess the assembly quality as well as visualize their results. Our pipeline is built upon the Galaxy system [5], a web-based computational platform packaged via the Amazon cloud in a format that can be deployed by other groups or institutions.

The rest of the thesis is organized as below:

Chapter 2 provides the necessary background information of the assembly algorithms.

Chapter 3 describes the novel viral genome assembly pipeline VirAmp that we have developed.

Chapter 4 demonstrates the performance of VirAmp by using a typical example of an HSV genomic dataset. This chapter compares the performance of VirAmp with several other popular assembly pipelines with both reference-based and reference-free evaluation.

Chapter 5 discusses several problems related to viral genome assembly of high coverage data, such as dealing with gaps and repetitive regions.

Chapter 6 highlights the conclusion of the study and proposes future work.

The Appendix will provide technical details of the programming and tool parameter tuning and testing.

Chapter 2

Background and Related Work

Viral diseases have always had a significant impact on all living organisms. From human perspective viruses such as HIV, influenza, HSV and many others have affected the lives of millions of individuals and thus societies as whole. As a result, the study of viral genetic composition is critical for downstream pathology analysis, as it holds the potential for assisting in the development of vaccines and may provide alternatives to therapeutic approaches. With the fast development and spread of high-throughput sequencing, genome-wide variation analyses of these genomes have become possible. In this chapter, major variation analysis strategies will be discussed, with a review of published assemblers as applied to viral genome data.

Whole-genome variation analysis strategy

Two approaches for detecting genomic variation are:

1. Mapping against a reference genome and
2. *de novo* genome assembly.

Mapping against a reference genome

Mapping against a reference genome for variation analysis is frequently used in humans and other complex organisms. This strategy is suitable for the analysis of SNPs, small insertions and deletions (INDELs) and other mild mutations. However, when compared to human genome, the viral genome is mutating at a much faster rate and the variation rate is also much higher. This means that the viral genomes we are sequencing may be too genetically distant from the reference we are comparing to. This could lead to the problem that results are biased to reference,

specifically like low quality mapping, information loss, and most importantly, mapping to reference cannot be used for structural variation discovery. [1]

de-novo Assembly

One possible solution to the issues mentioned above is *de novo* assembly. Because of its reference-free strategy, *de novo* assembly may be able to solve the bias problem; and at the same time it is suitable for structural variation (SV) discovery and annotation.

There are two major strategies for performing *de novo* assembly. A more traditional approach is known as the overlap-layout-consensus algorithm. This algorithm starts by comparing the reads and attempts to overlap pairs by computing the similarity (overlap step). It then applies a multiple sequence alignment (MSA) algorithm to position the reads in the right order with respect to one another (layout step). Finally the MSA information is used to produce the consensus sequence (consensus sequence). [2] This strategy has been proven to produce fairly good results through numerous early assembly projects using Sanger sequencing. However, a critical weakness of this strategy is that it is not suitable for high throughput sequencing, which produces large amounts of relatively short reads. With reads commonly shorter than 200bp, the algorithm is unable to deal with the repetitive regions and decision which one really overlaps. This results in a much higher error rate. In addition, the huge amount of high-throughput sequencing data compared to traditional Sanger sequencing, requires huge computational resources.

The *de Bruijn* graph algorithm is a completely different approach. The methodology breaks down reads down into nucleotide sequence of length k, or k-mers, then sets the first (k-1) bases of one k-mer as prefix and last (k-1) bases as suffix, the algorithm then chains the k-mers according to prefix and suffix and constructs a graph. Within the graph the most efficient path is

identified. In this way the high redundancy problem is solved by handling k-mers rather than reads, which shortens the computational time as well as saving other computational resources. In this strategy, the k-mer size becomes the most crucial parameter. Smaller k-mer sizes usually result in contigs of small size, and repeats longer than k will tangle the graph; while k-mers that are too large often tend to contain errors which later may produce errors in the genome. In general, assemblies generated from de Bruijn graph based assemblers tend to have contigs with smaller sizes compared to the traditional overlap-layout-consensus algorithms.

As recent assembler evaluations like GAGE[3] and Assemblathon 2[4] point out, depending on the variety of the input data and organisms to be assembled, the results can be very different. This indicates that there's hardly an assembler that can perform "well" universally, even when allowing for parameter tuning. This, on the other hand, suggests that it is more advisable to create entire pipelines and methodologies for different categories.

Viral Genome Related Assemblers

The size of viral genomes can range from two kilobytes to two megabytes, and though this seems to span an order of magnitude, it can still be considered to be much smaller than the genome of other higher order species. With today's sequencing technology, a single run can generate shot-gun sequencing data with surprisingly high coverage. Taking the Illumina HiSeq2500 machine as an example, one standard run can generate about 14 billion 150bp x 150bp paired-end reads, that on average can produce at least ~100,000x fold coverage for one virus sample. Taking the advantage of the fast-increasing capacity of the sequencing machine, researchers tend to sequence a large number of strains of the same virus to conduct population difference studies and variation analysis. An extra advantage that this high sequencing capacity brings is that now scientists can perform very deep sequencing of one single strain, to ensure that

those regions in the viral genome that are difficult to sequence are covered with an adequate number of reads.

Because of the potentially high diversity in the genome, together with the technical difficulties of sample preparation, the resulting sequencing reads tend to be highly variable in coverage across the genome, and may potentially contain a large amount of contamination from host cells and artifacts. Because of these issues, an efficient error correction step is necessary before applying the assembly process.

VICUNA[10] is a *de novo* assembler designed for genome analysis of high mutation rate virus populations. It assembles one set of sequences as the linear presentation of the mixed populations. VICUNA is an *overlap-layout-consensus* algorithm based assembler aimed at non-repetitive genomes. As a result, though VICUNA is optimized for high mutation data by aggressively merging similar sequences, for situations as herpes simplex virus (HSV) which contains small and large repeats, and where the deep-sequencing depth varies greatly due to high GC content, VICUNA may be computationally inefficient.

SPAdes[11] is another increasingly popular *de novo* assembler built upon the *de Bruijn* graph algorithm. SPAdes implements a multisized *de Bruijin* graph by utilizing multiple k-mers to remove bulges and thus detect and fix the chimeric reads problem. SPAdes is primarily designed for the genome assembly of single-cell multiple displacement amplification (MDA) bacteria data, but also works well for standard multi-cell isolates of bacterial data and even smaller genomes. Besides the core assembler, SPAdes also comes with an option of running the whole assembly pipeline, including a read error correction (BayesHammer) and a mismatch correction using Burrows-Wheeler Aligner (BWA) [11] by mapping the reads back to new assemblies to improve polymorphism detection rates.

Chapter 3

Assembly Pipeline Description

Our major assembly pipeline contains three steps: Coverage Reduction, Genome Assembly and Information Recovery. Figure 1 demonstrates the standard pipeline of VirAmp.

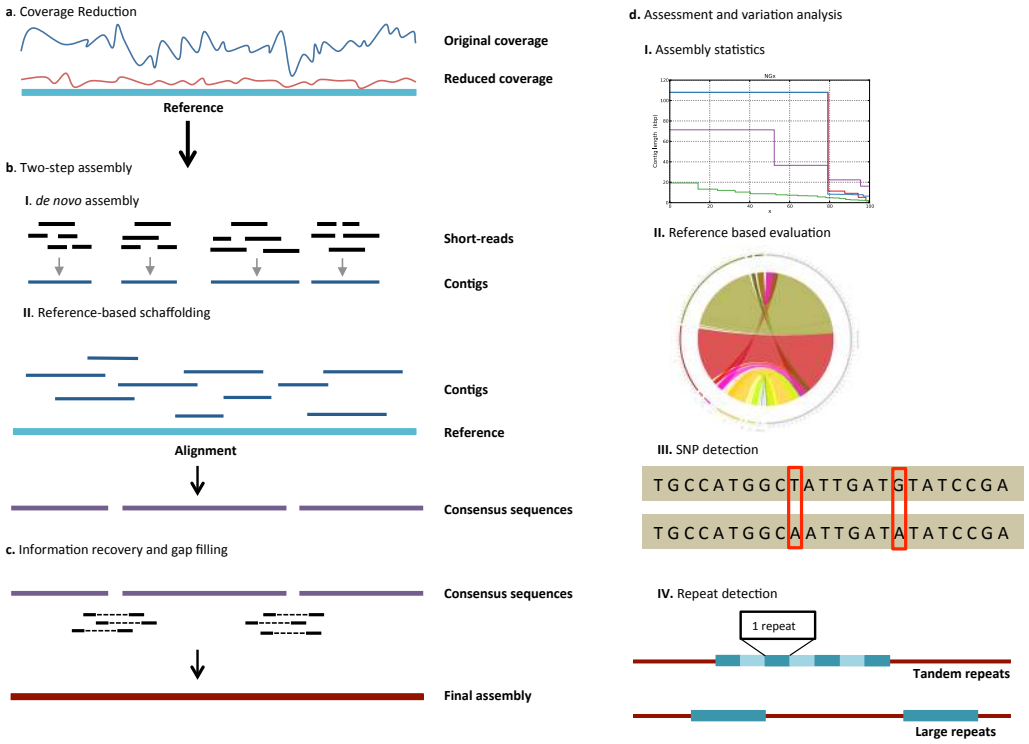


Figure 3-1 Overview of the VirAmp pipeline

As stated before, one of the major issues that virus genome sequencing is facing is the uneven and extreme high coverage data. The situation of coverage being too high often causes normal assemblers to fail, because they are unable to handle the entire dataset. We have developed a reduction-recovery strategy especially for this situation. We apply this coverage

reduction model before the assembly step, and add an information recovery step after assembling.

For the core assembly, we formulated a two-step, semi- *de novo* assembly strategy including both *de novo* assembly and reference-based assembly to help capture most of the variations while optimizing the running time to the least amount needed. By using *de novo* assembly we assemble those short reads into short contigs using tools that implement the *de Bruijn* graph algorithm, which further reduce the number of sequences while still retaining the variations from the sequenced genome. We further scaffold the short contigs assembled from the *de novo* assemblers into a draft genome using a reference-based assembler AMOScmp[8].

Our two-step assembly strategy combining the two mainstream algorithms solves the drawbacks from both sides: Major structural change will be captured by the de novo assembly in the first step; while in the second step, the AMOScmp alignment process will orient the sequences into proper position and connect them by overlapping the ends of each neighboring sequence. Further, the second-step reference-guided assembly also helps solve the high-error rate problem from *de Bruijn* graph algorithm. During the consensus step in AMOScmp, one multiple sequence alignment will be applied to incorporate most of the shorter contigs into longer ones.

Data preprocessing

Raw shot-gun data directly from the sequencing instruments usually contain multiple errors, so a quality control process needs to be applied before further downstream analysis. Depending on various factors including sample preparation, type and quality of the instrumentation, reagent performance etc, different quality control strategies may need to be employed. By default we assume the datasets used in our assembly pipeline have been already filtered and have met the quality control criteria designed by individual lab. Due to the

differences in sample preparation and sequencing platform, the raw data can be in dramatically different shape, which normally requires human supervision and curation during quality control step. We do however provide a separate section containing a collection of common quality control tools, including polyA and adaptors clipping, base quality trimming and contaminating host genome filtering, for users to choose from.

Error Correction and Coverage Reduction

The cost of high-throughput sequencing has decreased remarkably in the recent years and for shorter genomes it is now possible to perform very deep sequencing at reasonable prices. Sequencing coverage however can deviate greatly from the theoretical averages and occasionally extremely high read coverage is required to cover certain, so called “hard to sequence”, regions of a genome. In turn this very high coverage causes new problems. Assemblers may fail because they cannot handle the entire dataset. We employ a method called Digital Normalization, or Diginorm to reduce very high coverage regions to more manageable values. Diginorm breaks the reads down into k-mers by utilizing the median k-mer abundance to estimate the coverage. This allows the algorithm to reduce high coverage to a predefined cutoff, while retaining most of the reads covering low coverage regions. The reduction also helps to reduce the sequencing error since it redefines the coverage by breaking down the reads into smaller k-mers.[6]

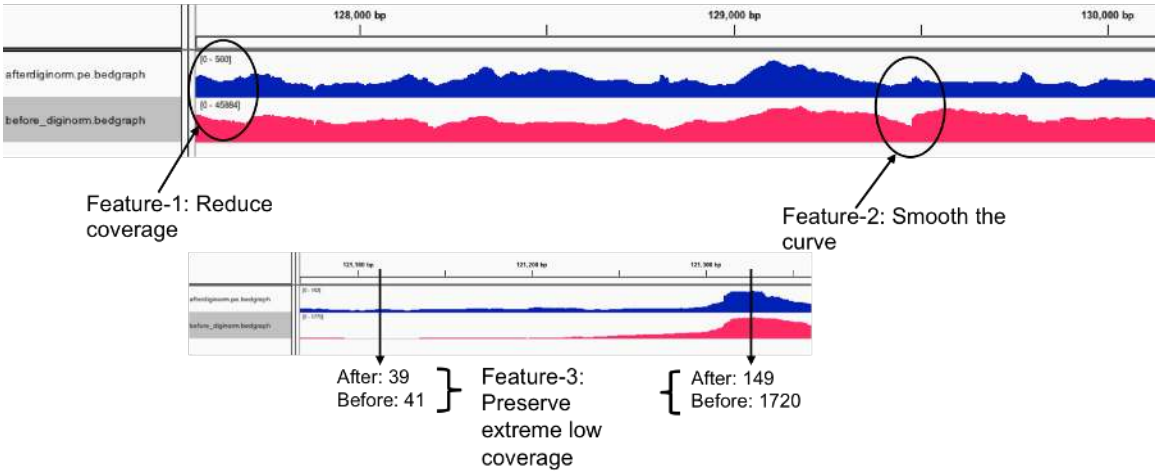


Figure 3-2 Demonstration of Read coverage before and after digital normalization

Figure 3-2 shows the three main features of Diginorm's performance. The figure demonstrates the dataset coverage when mapping the reads back to the reference. The blue curve on the top is after Diginorm and the red below is before Diginorm. Feature-1 shows the scale for the blue curve is 0-580bp while for the red graph is 0-45884bp, suggesting an overall large reduction at coverage. Feature-2 indicates a curve at around 129500bp position for the original data, but after Diginorm the curve disappeared, indicating Diginorm's ability to scale the variation of coverage into a uniformly low but balanced level. Feature-3 actually points out two extreme situations. At the position to the right of 121,300bp there's a large peak at both graphs, but this is demonstrated at different scale. After Diginorm the coverage is only 149bp while before it was 1720bp. This suggests Diginorm's ability of reducing high coverage into desired level. While another position about 200bp upstream in the same view has low coverage when sequencing, but Diginorm is still able to keep 39bp out of 41bp after coverage reduction, this suggests the algorithm's ability to retain reads at those extremely low abundant regions, which most time is the hard-to-sequence region. These three features demonstrate that Diginorm is different from random subset and

produces a smart coverage reduction process that can reduce the redundant data while keeping the critical information.

Two-step *semi- de novo* assembly

We introduce a two-step assembly strategy that includes both *de novo* assembly and reference-based assembly methods to help capture most of the variations while optimizing the running time. In addition our assembly strategies include both *de novo* and reference-guided assembly correction steps.

De Novo Assembly

As stated, *de novo* assembly is used to assemble a new genome from the sequencing dataset without reference information. Viral genomes tend to evolve at a faster rate eukaryotic genome and thus a particular strain could be very distant from the standard reference in the database. Processing the sequencing reads by aligning to the reference genome could bias the results and lead to the reads being oriented incorrectly or be mapped to the wrong location when structural variations such as large insertions and deletions exist. By using *de novo* assembly we combine the short reads into contigs using tools that implements *de Bruijn Graph* algorithm. With this approach we both reduce the number of sequences while keeping the variations inside the sequence. Moreover, as mentioned above, larger k-mer size in *de Bruijn* graph often results in longer contigs but also incorporate errors in the assembled contigs; while small k-mer size can achieve more precise local assemblies but often collapse repeats and results in smaller contigs. Here we run multiple rounds of the *de novo* assemblers with different k-mers sizes and combine them into a single set that will serve as the input for the second-step assembly. It is possible that

multiple small contigs corresponding to one position or region are stored in the dataset. We provide choices of three popular assemblers for this step:

> **Velvet** [7] is one of the earliest assemblers implementing the *de Bruijn* graph algorithm. Velvet is designed as a general assembler for shot-gun sequencing with no specialization on any better performance of particular organisms or datasets. Since we are only expecting contigs with criteria of larger than 1kb, Velvet is set as default assembler at this step for its stable performance as well as its high performance in running speed.
>
> **SPAdes**[11] is an increasingly popular assembler designed for both standard isolates and single-cell MDA bacteria assemblies. SPAdes uses an iterative approach to implement a multi-sized *de Bruijn* graph algorithm with multiple sizes of k-mer. It is reported to retrieve good results in small size genomes such as bacteria. SPAdes has its own pipeline, although here we only integrate the core assembler into our platform. Moreover, since SPAdes already makes use of multiple k-mers, we only run SPAdes once with multiple k-mer size fed in. SPAdes is set as an alternate assembler in parallel to Velvet.
>
> **VICUNA**[10] uses a modified OLC algorithm and specifically targets high mutation rate virus genome assembly. It can handles deep sequencing data with a high variation rate but takes longer than the above two *de Bruijn* graph assemblers. Besides the assembler, VICUNA also has a full pipeline from contamination removal (reference sequence required) to *de novo* assembly. Here the VICUNA core assembler is also set as an alternate assembler.

Reference-based scaffolding

AMOScmp[8] is used to assemble the contigs output from the *de novo* assembly step. AMOScmp is part of the AMOS consortium projects (v 3.1.0), and was originally designed as a reference-guided assembler using an *alignment-layout-consensus* algorithm similar to the traditional *overlap-layout-consensus* assembly algorithm. Multiple steps of AMOScmp may be adjusted to make the algorithm better suited for contig scaffolding. The first step is orienting contigs' position by aligning them to the reference genome or genome of related species using MUMmer [9] aligner. Then a layout refinement step **casm-layout** carried out to deal with partial matches caused by large INDELs and other structural rearrangements. Repetitive reads are then placed randomly into one of the copy locations (-r option) to allow the assembly of genomes with

large repetitive regions (such as the complete strain of HSV-1). Finally a **make-consensus** step performs a multiple alignment to generate consensus sequences; this step is particularly important as this will help incorporate variations that were assembled with small k-mers into large contigs. This multiple sequence alignment result will provide the final consensus sequences with more accurate structural information. The minimum overlapping bases (-o option) is set to be 10.

Information Recovery

To ensure that all the information is devoted to the assembly, an extra scaffolding & contig extension tool SSPACE [17] is used at the end of the pipeline. SSPACE is a stand-alone scaffolding tool using Bowtie [23] as the underlying aligner to map the reads back to the contigs.

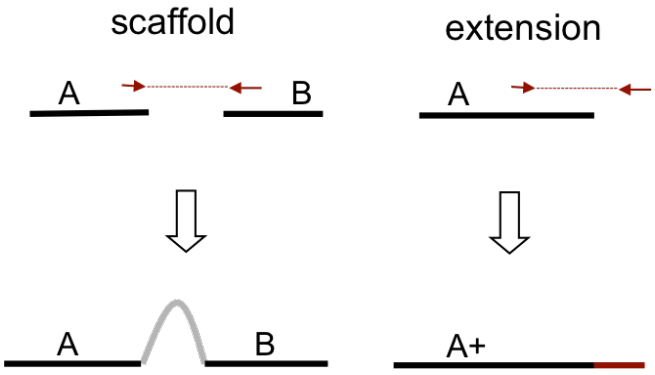


Figure 3-3 Demonstration of SSPACE scaffolding and extension

As shown in Figure 3-3, SSPACE uses the information from the paired-end reads that are properly mapped to the end of the sequences to scaffold two contigs. The insert size of the paired reads is used to estimate the gap size in between. An optional step is to extend the contigs when only one read in the pair can be properly mapped. SSPACE uses the original dataset before Coverage Reduction step (Digital Normalization) to make up any potential information loss during previous steps. The extending or scaffolding of contigs is called the “recovery step”.

The final assembly is created at the end of this step. These contigs are listed in one multi-fasta file as the order of the genome, forming a linear genome potentially with several gaps. An optional step is provided to connect the contigs into one sequence by adding Ns in between. The number of Ns is estimated based on the alignment to reference genome. This step is provided for the convenience of downstream functional analysis and is advised to be taken at the end of the whole analysis, after evaluation and variation analysis.

Assembly Assessment and Variation Analysis

Upon the completion of assembly, we also provide draft genome assessment and variation discovery, to help researchers understand and interpret the assembly results, as well as provide information and directions for further analysis.

Assembly Assessment

Basic reference-free assembly evaluation metrics, such as contig length and N50, are provided as a data summary. When a reference is present, reference-based evaluation, such as genome fraction and NG50, are also listed in the summary. The evaluation matrices are powered by QUAST[12], a genome assemblies evaluation tool using MUMmer[13] as the underlying aligner to compute various metrics. A full version of QUAST report is also available for downloading.

Assembly-Reference Comparison Report

A table-formatted report comparing the new assembly with reference genome is provided for the whole genome comparison and analysis. This is done by using the MUMmer[13] package and generates pairwise local alignments between the final assembly and reference sequence.

As shown in Table 3-1, the coordinates and percentage identity information of the aligned regions between new assembly and reference genome are given in the table. This helps the users to figure out any structural variations like large insertions and deletions, as well as other complex genome structure which is hard to identify.

Table 3-1 Example of Assembly-Reference Comparison Report

Ref_start	Ref_end	Contig_start	Contig_end	% Identity	References	Contigs
1	62457	288	62722	99.46	JN555585_ truncated	Ctg_1
53191	53334	53597	53454	100.00		Ctg_1
62638	108009	62722	108094	99.15		Ctg_1
108111	108301	186	1	92.75		Ctg_1

Circos Graph for genome alignment visualization

Circos [11] has become increasingly popular in comparative genomics visualization, as it provides a straightforward way to show structural and other variations of large datasets, which are usually hard to visualize. In VirAmp we provide this assembly visualization step using Circos, projecting the assembled draft genome to the aligned part of the reference to create a straightforward visualization for large structural variation. Users only need to submit a draft genome and a reference genome via the web interface. As shown in Figure 3-4, the assembly and reference are listed as two halves of a circle, represented by colored and grey bars respectively.

The colored ribbons represent the projected alignment, and the color of each ribbon corresponds to the contig in the new assembly.

Circos can also be used for visualization of two sets of contigs to assess the differences and similarities between two assemblies.

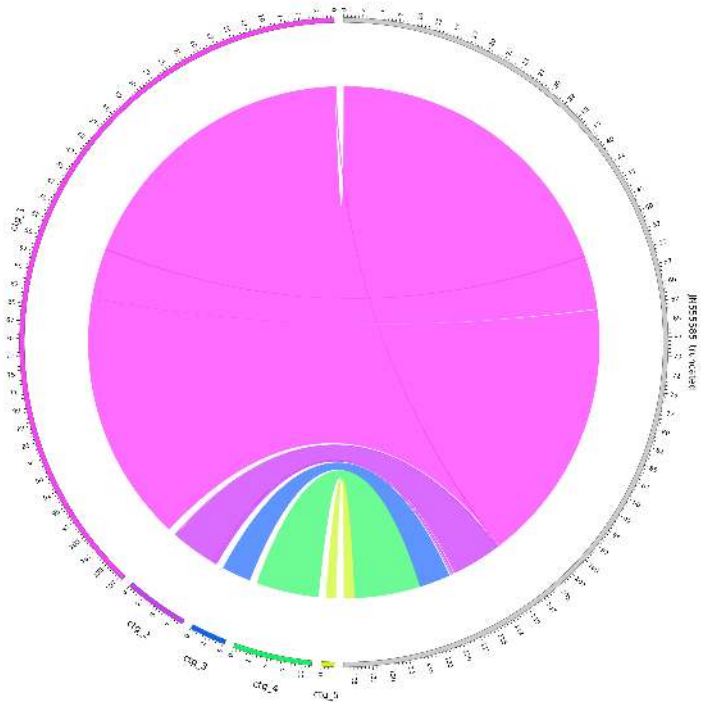


Figure 3-4 Circos graph projecting the comparison between assembly and reference genome

Variation Analysis

VirAmp also provides a collection of tools built upon the MUMmer aligner for variation identification. SNP analysis produces a VCF formatted SNP records between the reference and new assembly. Repeats and tandem repeats can also be identified using tools we provided. BWA is integrated in the platform both for data preprocessing (for example, host contamination removal) and to detect minor variations.

Recognising Errors in Assemblies using Paired Reads

Recognising Errors in Assemblies using Paired Reads, or REAPR [18], is a tool that evaluates the accuracy of a genome assembly, specifically focused on the detection of mis-assembly, without using a reference genome for comparison. REAPR first uses aligners to map the paired-end reads back to the new assembly to test each base of a genome sequence for both small local errors and structural errors. The errors are detected based on the changes to the expected distribution of inferred sequencing fragments, termed Fragment Coverage Distribution (FCD).

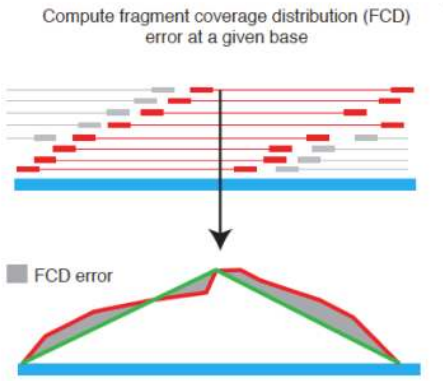


Figure 3-5 Demonstration of REAPR core algorithm FCD calculation [18]

Figure 3-5 shows the core algorithm of REAPR FCD calculation. REAPR locates all the fragments (fragments in red) constructed by the paired-ends reads covering one particular base (pointed by the black arrow), then flanks the sequence to both directions that any of those fragments covered (blue line), then computes the total coverage (red curve) and comparing with the theoretical coverage (green lines), which is the fragment coverage distribution (FCD). The difference, which is the grey area between the two curves, is the FCD error. The bases having higher FCD errors than the predefined cutoff are determined as wrong bases.

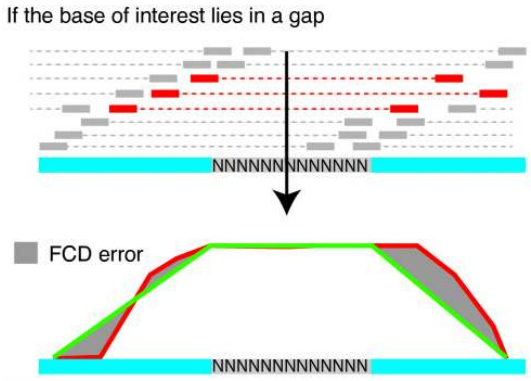


Figure 3-6 Gap identification in REAPR[18]

Gaps are identified when the coverage is similar as shown in Figure 3-6. In this way, a correction will be applied and enable the scaffolding error identification.

REAPR summary is given as an evaluation report for each assembly, this includes the original and broken N50, number of gaps and total gap length, and scaffold errors called by REAPR. Particularly, the broken N50 is the N50 of the newly generated contig set after REAPR correcting the mis-assemblies.

Finally a REAPR summary score is computed for multiple assembly comparison, using REAPR version 1.0.16. The first step is to map the original paired-end reads back to the assembly using smaltmap function, which integrates SMALT[24] version 0.7.5 as map engine. The option –y is set to 0.9, which requires 90% percent of the reads to map perfectly.

Since on average the insert size of our library is about 280bp, in the second step we chose the perfectmap function to generate perfect and uniquely mapping read coverage. The perfectmap function is designed specifically for small insertion sizes (<300bp) thus this filter will keep only those reads that are properly paired. The reads that pass the cutoff will have to fulfill other criterions like minimum Smith-Waterman alignment score and mapping quality scores.

The final step is to run the pipeline function using the default setting. The REAPR summary score is computed as following:

$$REAPRscore = NumberOfErrorFreeBases * \frac{(brokenN50)^2}{originN50} \quad [17]$$

The three parameters: Number of Error Free Bases, broken N50 and origin N50 are all related to score ranging from 0 to 1, which for each statistics, the assembly of the largest value is given 1 and the remaining is calculated as the percentage of that value. All the original value of these three parameters are given in the REAPR pipeline result files for each assembly.

Chapter 4

Example analysis

Data and Computational Resources Description

We evaluated our protocols by assembling data obtained from a lab strain of Herpes Simplex Virus 1 (HSV-1). Among all the human-infecting viruses, HSV-1 is one of the most common human pathogens in terms of its global distribution, longevity in the host, and its generally mild symptoms[19]. The reference strain HSV-1 17 has a genome of 152 kb (GenBank Accession JN555585). The genome consists of a 108 kb unique long (UL) and a 13 kb unique short (US) region, with each unique region flanked by inverted copies of large structural repeats (termed repeat long (RL) and repeat short (RS), with lengths of 9.2 kb and 6.6 kb respectively)[20]. For evaluation purposes, we used a modified or trimmed version of this reference, where the terminal copies of RL and RS have been removed, leaving a sequence of 136 kb. The removal of terminal repeats facilitates alignment of *de novo* assembled contigs to the reference genome.

For evaluation, most of the analysis is built upon one virulent HSV-1 laboratory strain which contains 33 million reads. In the assembly pipeline comparison section we selected two more datasets, a variant laboratory strain with a fluorescent protein inserted into the genome, and a clinical isolate of HSV-1. These two datasets as well as the first datasets contain from 33 to 87 million Illumina HiSeq reads of paired-end, 100 bp x 100 bp sequence. Using previously published approaches[21, 22], we de-multiplexed these sequence reads, trimmed off adaptor sequences, removed low quality bases and sequencing artifacts, and removed sequences matching the genome of the host cells used for growing viral stocks. The computational resources applied

here is an Amazon instance of one CPU of 4Gb RAM. The completion of the whole pipeline process ranges from 90 to 240 min.

Comparing the performances at each step of the assembly pipeline

To demonstrate the necessity and contribution of each step during the assembly, we performed an assembly assessment using QUAST[9].

We used NG50 as our assessment metric, since as shown in the Assemblathon 2 paper, NG50 is a better measurement when reference genome is known[17]. The N50 metric is an integer number that corresponds to shortest length for which the sum of all contigs longer than this length contain at least half of the total number of bases in all contigs. NG50 is calculated in the similar way as N50, the only difference is that it uses reference genome length for the value that determines half coverage rather than the sum of all assembled contigs.

Velvet is used for the *de novo* assembly step, and multiple k-mer sizes (k=35,45,55,65) are used for providing the contigs for the second-step assembly. Statistics for this step is from one best assembly of the above four sets (k=65), using NG50 as evaluation.

Table 4-1 Statistics of assemblies at each step of the VirAmp pipeline

# step	#contigs (>=0)	#contigs (>=500)	Total length	Largest contig	NG50	#INDELs
#1 de novo assembly	52	47	126990	14080	4197	91
#2 reference-guided assembly	14	9	135040	62437	32695	86
#3 scaffolding	5	5	135772	108094	108094	98

Table 4-2 The contigs coordinates in reference genome

Ref_start	**Ref_end**	**Contig_start**	**Contig_end**	**Contig_ID**
Contig coordinates using step#3 contigs (after SSPACE scaffolding)				
1	108009	288	108094	Ctg_1
108111	117634	5	9181	Ctg_2

117897	123237	1	5261	Ctg_3
123245	134543	1	11337	Ctg_4
134593	136376	1	1754	Ctg_5
Selected contig coordinates using step#2 contigs (after AMOScmp assembly)				
75278	108004	14	32695	AMOSctg_3
108111	108281	9	174	AMOSctg_4
… …				
117476	117634	1	159	AMOSctg_10
117897	119760	1	1864	AMOSctg_11
1119846	123229	1	3387	AMOSctg _12
123251	134537	1	11325	AMOSctg_13
134595	136376	1	1752	AMOSctg_14
Selected contig coordinates using step#1 contigs (after velvet assembly)				
101276	108004	6723	1	NODE_25
109208	109807	1	597	NODE_45
… …				
115768	116021	252	1	NODE_50
120582	120928	347	1	NODE_52
121397	122313	1	915	NODE_34
123396	128815	4	5429	NODE_21
… …				
131850	134423	1	2579	NODE_32
134595	136376	1791	40	NODE_26

As shown in Table 4-1, the total length, largest contig, and NG50 are all increasing after each step, and after step3, the largest contig size is raised up to 108kb. This is similar size as the Unique Long region (UL) in the reference genome, and makes up about 80% of the whole reference.

Table 4-2 displays the starting and ending coordinates of selected contigs in each step. The table is in the ascending order of the coordinates, the records next to each other reflect the physical location when mapped back to the reference. From Table 4-2, we can figure out that the gaps (distance between the ending coordinates and next starting coordinates) have been generally closed or narrowed, indicating those gaps in the last version are indeed coming from previous assembly. This again proves that our reference-guided assembly (step#2) does not influence the

assembly by integrating any information from reference, but is more likely using the reference genome only as a guide to orient the contigs.

Also there is an improvement from step#2 to step#3, where we further close 9 out of 13 gaps, and narrow down the rest of them, showing the necessity of the last step. With the Diginorm step before assembly and scaffolding after the assembly, we are able to integrate the most information from the short read sequencing data into the assembly.

When mapping the contigs back to reference genome, we can see after the final step, the five contigs have very short overlaps, suggesting this is almost a linear genome. We do provide an option to connect those contigs into one linear genome. This will be helpful for downstream functional analysis; on the other hand, we strongly recommend users to keep the contigs separately, or at least take an extra look at those gaps. More information and concerns are provided in the discussion section.

The number of insertions and deletions (INDELs) stays very similar at each stage, suggesting the reference-guided assembly step does not remove the variations when using reference genome as assistance to orient the contigs into the right position.

Comparing the final assemblies to the reference genome

The final assembly is a multi-fasta file containing five contigs, with lengths ranging from 1.7 kb to 108kb. MUMmer is then used to align the new assembly back to the reference genome to evaluate the quality of the new assembly.

Table 4-3 Comparison report between the final assembly and reference genome

Rec#	Ref_start	Ref_end	Contig_start	Contig_end	% Identity	References	Contigs
1	1	62457	288	62722	99.46	JN555585_t	Ctg_1

2	53191	53334	53597	53454	100.00	runcated	Ctg_1
3	62638	108009	62722	108094	99.15		Ctg_1
4	108111	108301	186	1	92.75		Ctg_1
5	108299	108496	5	193	92.96		Ctg_2
6	108497	116585	296	8423	97.10		Ctg_2
7	116652	116924	8421	8687	93.04		Ctg_2
8	117090	117383	9718	9042	86.50		Ctg_2
9	117497	117634	9044	9181	100.00		Ctg_2
	117897	123237	1	5261	97.37		Ctg_3
	123245	134543	1	11337	98.74		Ctg_4
	134593	136376	1	1754	97.42		Ctg_5

Table 4-3 shows the comparison report generated by VirAmp and indicates alignment between the reference and new assembly. Each row is one record of alignment between one piece of sequence in reference to one part in one contig; the six columns from left to right contains the following information: starting and end coordinates of the reference region, starting and end coordinates of one region in the new assembly, percentage identity between two sequence, name of the reference sequence and contig name in the new assembly.

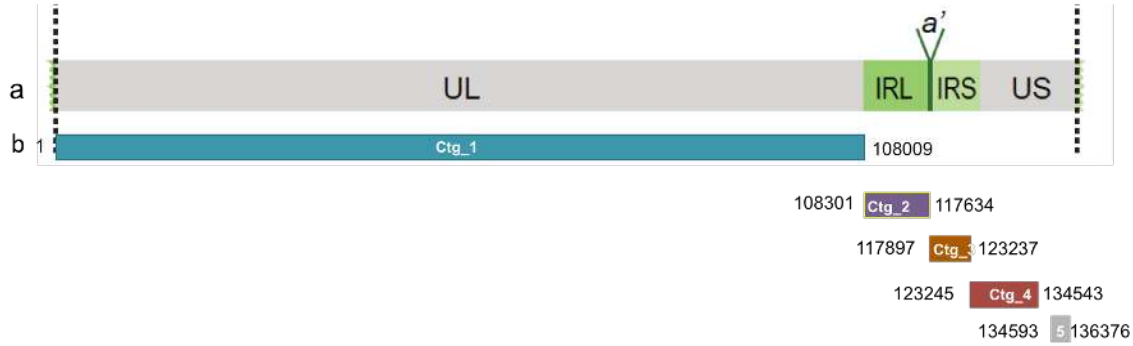


Figure 4-1 Comparing the final assembly with the reference genome

Figure 4-1 further visualizes the alignment in Table 4-1's alignment. Figure 4-1-a is a model of the reference genome, which is a modified HSV-1 reference (JN555585) made by

truncating out the Repeat Long and Repeat Short regions at the beginning and end of the sequence. The modified reference is mainly composed of four elements, a 108kb long Unique Long region (UL) and a 13kb Unique Short region (US); the UL is flanked by 9.2kb Inverted Repeat Long Region (IRL), which is the inverted copy of the Terminal Repeat Long region (TRL) at the beginning of the sequence and truncated for analysis convenience; similarly, US is flanked by the 4.4kb long terminal and inverted copies of Repeat Short regions (TRL/IRL).

Figure 4-1-b shows the start and end coordinates where the new assembly aligns to the reference genome. As shown in the figure, among the five contigs of the final assembly, the first three each cover about one whole element of the reference, with contig-4 (Ctg_4) and contig-5 (Ctg_5) adding up together covering the US region. As stated, the sequences between each element are hard to sequence and assemble, the gap between contig-1/contig-2 and contig-2/contig-3 both occur in those regions, with contig-3 actually flanking over the gap and ending up inside the US1 gene region. This suggests VirAmp's ability to rescue some information in those hard-to-retrieve regions.

VirAmp allows users to visualize alignments. Our pipeline can make use of the Circos tool to generate circular graphs that can display the alignment between the final assembly and the reference. The grey bar on the right indicates the reference whereas the colored bars on the left represent the contigs in the final assembly. Each color indicates one contig, and the ribbon with the same color inside the circle demonstrates alignment between the specific contig and reference genome. This makes it easy to read out the start and end coordinates of each contig from Table 4-3. When shown in the Circos graph, users can easily identify structural variations that occur from genomic rearrangements. First of all, there's a very thin ribbon connecting the starting region of contig-1 with the regions between UL and IRL in the reference. This represents the record 4 in Table 4-3, because the reference is a truncated sequence while the reads for assembly is from the full sequence, so we are able to flank the sequence at the beginning into the TRL. During the

alignment phase, the assembled TRL region will be aligned to the IRL, creating record#4 in Table 4-3 and the ribbon. There is another line inside contig-1 among the coordinates 53191-53334 when aligning to the reference, corresponding to record#2 in the table 4-3. This region can be aligned both forward and reverse to the reference, suggesting it could be a palindrome. This assumption is further validated in the reference sequence. The last interesting region to be pointed out is at 62457-62638 in the reference. This corresponds to an 181bp deletion in the new assembly, as we can only identify a gap at the reference side, but no breaks at contig-1.

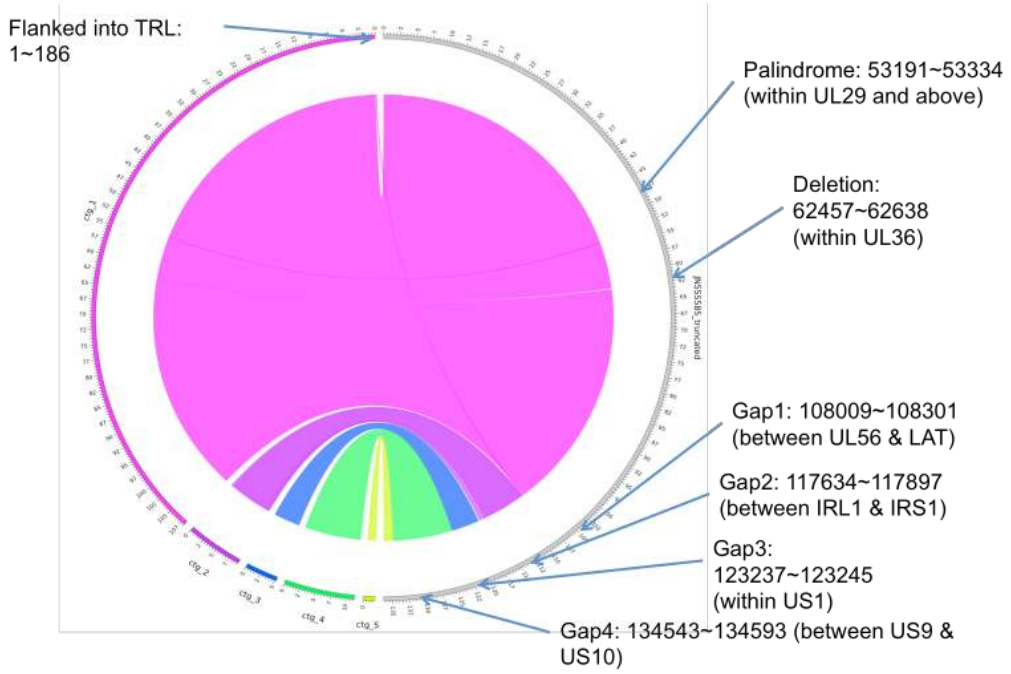


Figure 4-2 VirAmp visualization of the alignment between the final assembly and reference genome.

Comparing VirAmp with other approaches

To assess the performance of our assembly pipeline, we use three different sets of sequencing data of HSV-1 to run through our pipeline and compare the results to that of two other assembling pipelines, SPAdes and VICUNA. The SPAdes assembler utilizes multiple k-mer

sizes and takes advantage of both small and large kmer, to improve the performance; while the assembler of VICUNA, on the other hand, is a *de novo* assembler developed by the Broad Institute specifically for virus genome assembly. One of the advantages in the VICUNA pipeline is that it does a pre-filtering step before feeding into the assembler to keep only reference-genome like reads. This is extremely useful in contamination-rich samples(though our testing data is after our manual filter with by mapping back to the reference). Also VICUNA assembler uses a modified *overlap-layout-consensus* algorithm, which though it takes a longer time, can be applied to the mix-population with relatively high mutant rate.

Single-Cell Mode is applied in SPAdes, as this is the most similar situation as virus genome sequencing, with k=21, 33, 55 given as the kmers according to the recommendation from the manual. We did multiple rounds of VICUNA assembly and chose the best assembly with kmer=21 for comparison.

Table 4-4 Comparing VirAmp results with three other popular Assembly pipelines

Virus[1]	# reads (x 10^6)	Pipeline	#contigs	Largest contig (bp)	N50	NG50	#fully un-aligned contigs	Run time (h)	# thread (4GB/ CPU)
HSV-1 **laboratory strain**	33	VirAmp	5	108,094	108,094	108,094	0	1.5	1
		SPAdes	9,609	107,857	258	107,857	9582	6	4
		VICUNA	266	19,285	5,654	8,704	163	8	6
HSV-1 w/ **fluorescent insert**	37	VirAmp	4	63,109	49,971	49,971	0	2.5	1
		SPAdes	5,946	39,441	273	13,888	5898	7	4
		VICUNA	101	33,391	9,822	7,644	60	13	6
HSV-1 **clinical isolate**	87	VirAmp	3	117,134	117,134	117,134	0	4	1
		SPAdes	74,927	93,771	256	82,041	74,608	21	4
		VICUNA	424	23,611	2,786	7,136	383	30	6

We compared the assemblies back to the trimmed HSV-1 reference genome (136 kb), and used several commonly accepted metrics to evaluate performance of each assembly method, as recommended by Assemblathon 2 [4]. We considered any contigs longer than 500 bp as a valid

assembly output. All basic statistics except REAPR are done by using QUAST[12], and our pipeline produces a full version of the QUAST report at the end of the assembly.

According to the evaluation statistics, the VirAmp pipeline achieves the highest NG50 in all three HSV-1 datasets (Table 4-4), further demonstrating the rationale for this project that applies judicious use of multiple tools that can radically improve the quality of the results. In two of the three datasets, the largest VirAmp contig coveres about 80% of the whole genome. SPAdes retrieved one large contig with a length similar to the longest contig of VirAmp, but in all three testing datasets more than 95% of the SPAdes contigs cannot be properly aligned back to the reference. This causes SPAdes to receive the lowest N50 and REAPR score among the three assemblers. VICUNA retrieves an assembly with a size similar to the reference and an acceptable number of contigs, but the largest contig it produced is only around 20kb, which is much shorter than the other two assemblers.

In terms of computational resources, VirAmp analyzed the above datasets on a single 4 GB RAM CPU machine, while neither SPAdes nor VICUNA could finish the job successfully using the same machine. For a dataset with ~20,000x coverage on average (e.g. HSV_1 lab strain, Table 2), VirAmp finished the assembly within 1.5 hours, while the other two assemblers ran the same dataset with multiple CPUs (4 for SPAdes and 6 for VICUNA) with 4GB RAM and took more than double the time to complete.

REAPR reference-free evaluation of genome assembly

We use REAPR to describe the assemblies produced in the previous chapter, the tool produces additional details about the assembly quality including statistics like per-base errors and mis-assemblies and REAPR scores.

Table 4-5 REAPR evaluation of assemblies from different pipelines

Assemblers	Original Assembly					REAPR corrected Assmebly				
	Total length	#seq	N50	#gaps	Gap length	Total length	#seq	N50	#gaps	Gap length
VirAmp	135722	5	108094	8	145	135722	6	108094	8	599
SPAdes	2715404	9609	258	60	3107	2678194	9083	259	96	13745
VICUNA	212586	266	5654	7	62	212524	266	5654	12	5522

Table 4-5 continue

Assemblers	Error free base	FCD error within a contig	FCD error over a gap	Wrong read orientation	REAPR score	Computationa resources
VirAmp	92.19%	1	1	18	1	1 CPU; 90mir
SPAdes	5.6%	18	0	36	0.016	4 CPU; 6h
VICUNA	58.06%	11	1	5	0.029	6 CPU; 8h

Table 4-5 shows the REAPR evaluation of assembly results from the three pipelines listed. REAPR uses the fragment coverage distribution (FCD) algorithm to evaluate per-base correctness and mis-assemblies. Both original and REAPR corrected assembly metrics are listed in the table. As it's shown, VirAmp achieves the largest N50 both before and after REAPR correction, VirAmp only has a single minor mis-assembly in the shortest contig and just one gap with mis-estimated length FCD error over a gap. Similar situations are present for the error free bases, where 92.19% in VirAmp is considered to be correct, followed by VICUNA at 58.06%. A poor performance of SPAdes in REAPR evaluation suggests too many mis-assembled sequences even though it retrieves a compatible longest contig of 107kb. All the three pipelines have fewer than 50 reads mapped in wrong orientations, compared to the original datasets of millions of reads, this is considered to be an acceptable rate. The REAPR score at the end can be viewed as a summary of the above information, as indicated before, it contains comparison information of original and corrected N50, as well as per-base accuracy.

The last column of Table 4-5 lists the minimal computational resources that each pipeline needs to complete the assembly task; VirAmp shows a huge advantage with only 1 CPU and

finishing in 90 min. This means assembly tasks similar to the testing dataset can be accomplished with a desktop using VirAmp, a task that is impossible for all the other assemblers.

Chapter 5

Discussion

Final Gap Filling

We provide two kinds of draft genome as output. By default the draft genome is normally a multi-fasta file that usually contains 4-10 contigs in the orientation according to the references. The alternative is a single fasta file containing only one linear genome sequence. The single fasta file simply connects the contigs by Ns according to the alignment result towards reference genome. Single fasta file is provided for users who are not familiar with computation especially alignment assessment and text manipulation. The single fasta will provide convenience for them to do downstream functional analysis. However, we strongly recommend more experienced users to take the multi-fasta file and look into those gaps and then decide how to close them. There are several common reasons why the final gaps are not closed.

First, at the last step of reference-guided scaffolding, when generating the consensus genome, even though the gap region is overlaid by multiple short contigs, the multiple sequence alignment receives a very low score at the region and thus can not pass the threshold if the program can not generate consensus bases, then the program will set a break point, and run two multiple sequence alignment for two regions at the left and right side of the break point. So even though in the end, the gap might be covered by the flanking sequences from both sides, those two sequences may not be connected together if the program cannot generate a high consensus score.

Second, the presence of repetitive regions is known to cause difficulties when aligning sequences. Though our two-step assembly improves the accuracy in those regions, issues still remain in some mostly low complexity regions. Contigs from *de novo* assembly tend to end at those repetitive regions. Then in reference-guided scaffolding, the coverage of those regions is significantly lower than normal regions, and alignment scores tend to be lower due to the fact that these regions, even when assembled tend to contain more errors. As a result repetitive regions often introduce gaps in the final assembly.

The gaps can also be introduced by low sequencing quality or insufficient read coverage caused by the low complexity of the region. For example, regions that contain repeat sequences are not only hard to assemble, but also hard to sequence with the current instrumentation. Highly repetitive regions tend to use up the labeled nucleotide or cause enzyme slippage, which both lead to system bias and low sequencing quality. Moreover, those low quality sequences are likely not pass the quality control step of the pipeline; which may result in even lower coverage for those regions in the assembling process.

The gaps do not affect the assessment and variation discovery of our pipeline, but one may want to create one single sequence for the conveniences of downstream analysis. For quick assessment, users could assign Ns to the gaps to connect the contigs into a single draft genome sequence. A more prudent way would be to further investigate the gaps via alignments and/or database searches or even conduct experimental validation. We did not implement these steps in our main pipeline because these require human decision making and do not lend themselves to automation. But we do provide the BWA[14] aligner in the utility section, a tool that can be used to produce alignments against reference genomes. A more comprehensive approach would require experimental validation via a PCR verification on those specific regions to look into the real structure and biological meaning.

Repetitive Region

Large repetitive regions cause substantial difficulties for genome assembly. Take the HSV-1 genome as an example, one of the most significant structural features are the two sets of large repeating regions, Repeat Long (RL) and Repeat Short (RS), residing in the middle and two ends of the genome. Normally researchers will simply trim off the set of the RL and RS at both ends to create a truncated genome without large repetitive regions. This simplifies the process and also fits our pipeline. Since the structural variation discovery depends on the genome alignment between draft genome and references, or against itself, including large repeat regions could create additional problems for the alignment as the large repeats will also align to the other repeat regions, and, depending on the situation may lead to competing alignment scores that the programs need to account for. However, for demonstrative reason, or if users want to specifically study the repeat regions, the pipeline can construct the repeats when large repeats are provided. As we already discussed above, *de novo* assemblers will produce small contigs for repetitive regions, then during the reference-guided scaffolding step, if a contig can be aligned to multiple positions, the AMOScmp program will randomly assign it to one of them. One advantage of feeding *de novo* assembly contigs instead of short reads into AMOScmp, is that longer contigs are more likely to contain unique regions.

Assembling genomes using single-end reads

We also provide an alternative assembly pipeline for single-end reads. In this pipeline, the SSPACE scaffolding step will be missing, since that program requires the paired-end information for connecting contigs. All the other modules will be running in a single-end module (diginorm, velvet and AMOScmp). Since single-end sequencing is normally about 2/3 the price

of paired-end, it is commonly used when sequencing simple organisms like viruses. However, we still strongly recommend paired-end sequencing for all projects as it typically leads to higher quality assemblies. For deep sequencing datasets, like the one demonstrated in the paper, we keep about 20x coverage from the dataset using diginorm. This is because most programs cannot handle high coverage over 1000x, which will also utilize extremely high computational resources and time. Therefore, paired-end reads with slightly lower coverage may preserve more information, e.g. large structural variations, than trying to cover every corner of the genome. Furthermore, when using single-end reads, any information missing from the diginorm process will not be retrieved, since no paired-end information can be provided to SSPACE.

Chapter 6

Conclusion

We have described a web-based virus genome assembly platform, VirAmp, that can be used to assemble high throughput sequencing data. Our pipeline makes use of several existing programs and connects them in a convenient interface. The pipeline makes use of recommended practices and can assemble extremely high coverage viral genome data with minimal computational resources. In addition we provide a series of reporting and genome assembly analytic tools for evaluating the assemblies. All of our tools are wrapped into a Galaxy instance that individual research groups can run themselves. The Galaxy platform and default pipeline makes the approach usable by researchers without advanced programming skills or limited access to high-performance computing clusters.

Future work will first focus on improving the assembly process. We plan to integrate more evaluation tools like REAPR as well as downstream analysis tools, that can assist with functional annotation.

Bibliography

1. Henn, Matthew R., et al**. "Whole genome deep sequencing of HIV-1 reveals the impact of early minor variants upon immune recognition during acute infection."***PLoS pathogens* 8.3 (2012): e1002529.

2. Deonier, Richard C., Simon Tavaré, and Michael Waterman. **Computational genome analysis: an introduction**. Springer, 2005.

3. Salzberg, Steven L., et al. **"GAGE: A critical evaluation of genome assemblies and assembly algorithms."** *Genome research* 22.3 (2012): 557-567.

4. Bradnam, Keith R., et al. "**Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species**." *GigaScience* 2.1 (2013): 1-31.

5. Goecks, Jeremy, Anton Nekrutenko, and James Taylor. "**Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences.**" *Genome Biol* 11.8 (2010): R86.

6. Brown, C. Titus, et al. "**A reference-free algorithm for computational normalization of shotgun sequencing data**." arXiv preprint arXiv:1203.4802(2012).

7. Zerbino, Daniel R., and Ewan Birney. "**Velvet: algorithms for de novo short read assembly using de Bruijn graphs**." *Genome research* 18.5 (2008): 821-829.

8. Pop, Mihai, et al. "**Comparative genome assembly**." *Briefings in bioinformatics* 5.3 (2004): 237-248.

9. **Toolkit for processing sequences in FASTA/Q formats**: https://github.com/lh3/seqtk

10. Yang, Xiao, et al. "**De novo assembly of highly diverse viral populations**." *BMC genomics* 13.1 (2012): 475.

11. Bankevich, Anton, et al. "**SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing**." *Journal of Computational Biology* 19.5 (2012): 455-477.

12. Gurevich, Alexey, et al. "**QUAST: quality assessment tool for genome assemblies**." *Bioinformatics* 29.8 (2013): 1072-1075.

13. Kurtz, Stefan, et al. "**Versatile and open software for comparing large genomes**." *Genome biology* 5.2 (2004): R12.

14. **Circos Circular visualization**: http://circos.ca/

15. Sanger, Fred, and Alan R. Coulson. "**A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase**." *Journal of molecular biology* 94.3 (1975): 441-448.

16. Li, Heng, and Richard Durbin. "**Fast and accurate short read alignment with Burrows–Wheeler transform**." *Bioinformatics* 25.14 (2009): 1754-1760.

17. Boetzer, Marten, et al. "**Scaffolding pre-assembled contigs using SSPACE**." *Bioinformatics* 27.4 (2011): 578-579.

18. Hunt, Martin, et al. "**REAPR: a universal tool for genome assembly evaluation**."*Genome Biol* 14 (2013): R47.

19. Roizman B SE: **Herpes simplex viruses and their replication**. In Fundamental Virology. 3rd edition. Edited by Fields BN KD, Howley PM. Philadelphia, PA: Lippincott-Raven; 1996: 1043-1107

20. McGeoch DJ: **The genomes of the human herpesviruses: contents, relationships, and evolution**. *Annual review of microbiology* 1989, 43:235-265.

21. Szpara ML, Parsons L, Enquist LW: **Sequence variability in clinical and laboratory isolates of herpes simplex virus 1 reveals new mutations**. *Journal of virology* 2010, 84:5303-5313

22. Hunt M, Kikuchi T, Sanders M, Newbold C, Berriman M, Otto TD: **REAPR: a universal tool for genome assembly evaluation**. *Genome biology* 2013, 14:R47.

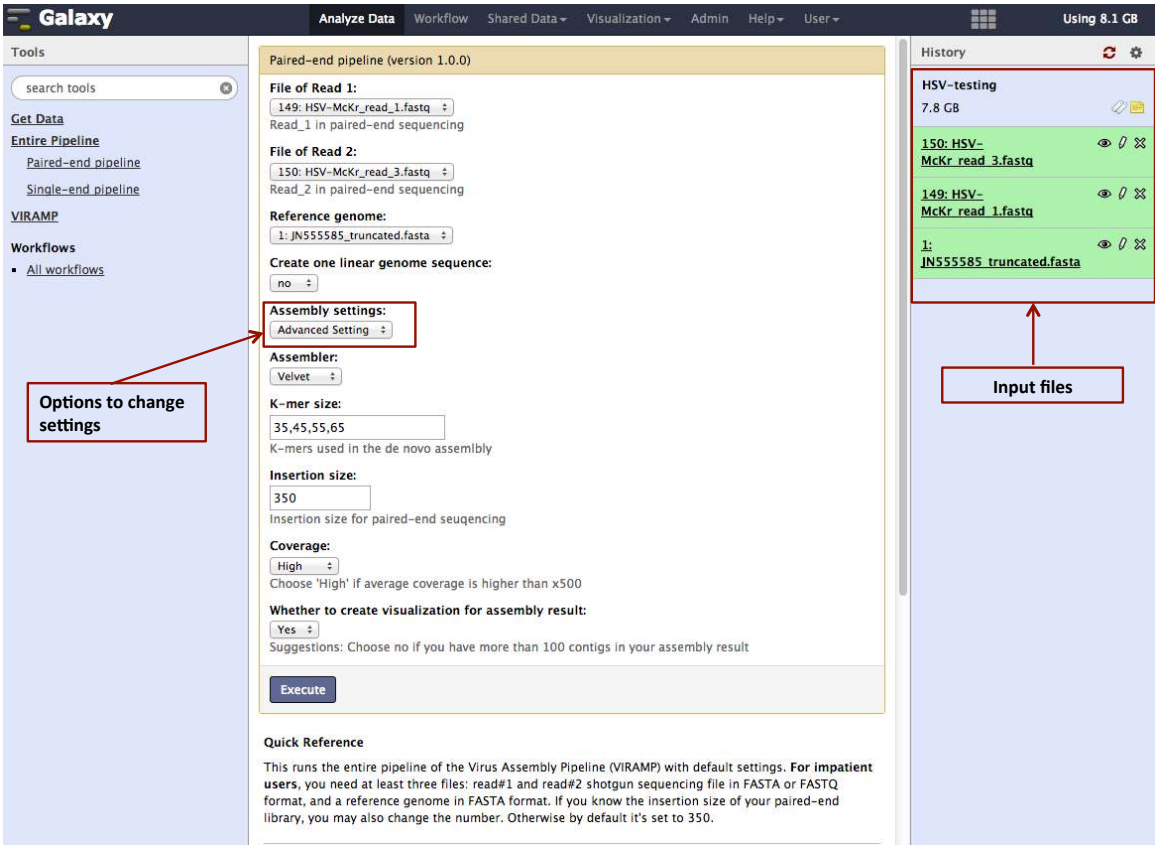
23. Langmead, B., Trapnell, C., Pop, M., & Salzberg, S. L. (2009). Ultrafast and memory-efficient alignment of short DNA sequences to the human genome.Genome Biol, 10(3), R25.

24. http://www.sanger.ac.uk/resources/software/smalt/

Appendix

Demonstration of VirAmp platform

VirAmp is stored as an Amazon image and deployed into the Amazon Cloud. For demonstration purposes, we also have a public VirAmp platform at http://viramp.com/. One demo dataset is placed under 'Shared Data -> Data Libraries -> HSV-McKr', which is a paired-end read dataset in two separate files; a results gallery from the demo dataset is placed under 'Shared Data-> Data Libraries -> Results Gallery'. We provide two ways to run the VirAmp pipeline: working via the pre-packaged tool and via published workflows. Appendix Figure 1 shows the parameter setting panel for running the whole VirAmp pipeline.



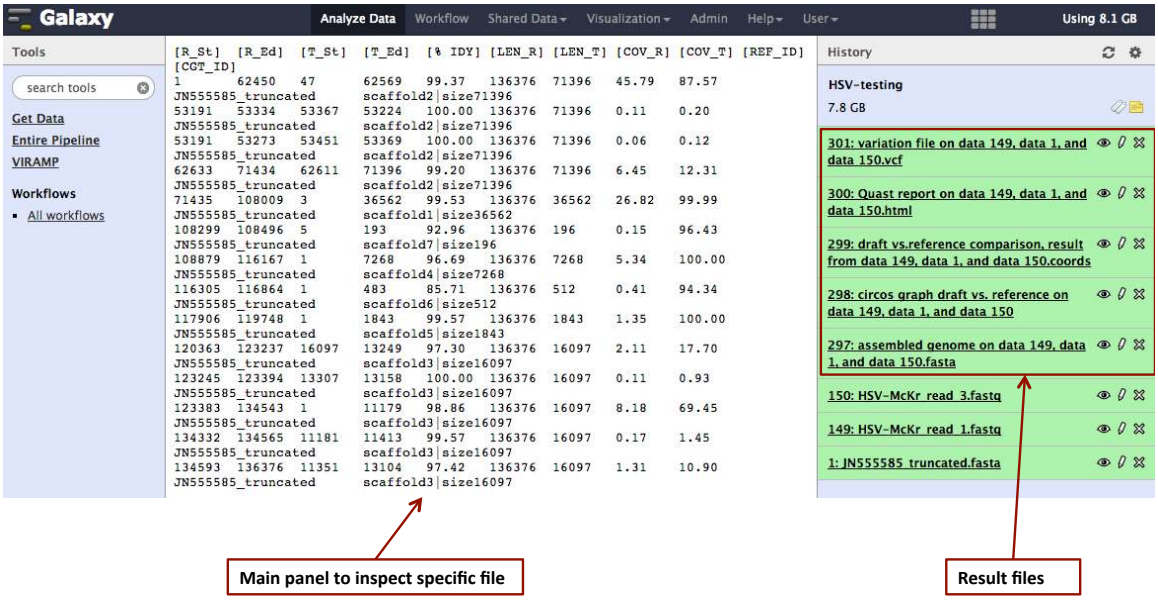
Appendix Figure 1 Settings to run the VirAmp pipeline

With default settings, users only need to upload the paired-end sequencing data in FASTQ format and a reference genome in FASTA format. If users feel it necessary to navigate the pipeline to create more suitable settings for their data, they can simply change the option "Assembly Setting" to "Advanced Setting" to set other parameters.

Appendix Figure 2 shows the second way to run the pipeline, which is via existing workflows. We have one published workflow under 'Shared Data -> Published Workflows -> viramp_workflow_pe', which works for paired-end read assembly. Users can simply click on the workflow and choose 'run' on the dropdown menu, and a screen similar as Appendix Figure 2 will show up. One only needs to select paired-end read files in step-1 and step-2 respectively; select the proper reference genome in step-3 and hit run. Then the system will run the necessary tools in the workflow sequentially using results generated from previous steps. At the very end, a final result files the same as the pre-packaged tool will be generated. To be noted, since this is actually running the individual tools listed in the platform, in addition to the final results as in the prepackaged pipeline, the results generated between tools will also show up in the history, which is the green panel located at the right.

Appendix Figure 2 A published Galaxy workflow connects individual tools in the platform and demonstrates VirAmp paired-end assembly.

The default results of the pipeline will produce five files listed in the green panel on the right. One can hit the “view data” button to inspect the specific result file in the main panel or hit “download” button to save it to local disk.



Appendix Figure 3 VirAmp with result file at pipeline complete status

For more details, please refer to the VirAmp online document: http://viramp.readthedocs.org/en/latest/