De Novo Assembly Using Illumina Reads

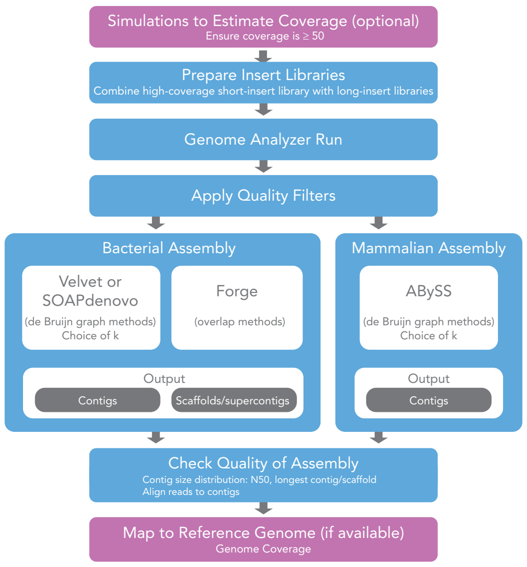
High quality de novo sequence assembly using Illumina Genome Analyzer reads is possible today using publicly available short-read assemblers. Here we summarize the results of several de novo assembly experiments. We show that for E. coli, genome coverage of as high as 99.4% (N50 = 82,595 bp) and 99.72% (N50 = 97,333 bp) can be achieved with two of the assemblers tested and short inserts. The latter assembly job completed in 15 minutes on a 32 bit Windows desktop with 3 GB of RAM, indicating that in some instances, *de novo* assemblies can easily be performed with existing computer resources in the laboratory.

Introduction

Recent technological advances have dramatically improved next-generation sequencing throughput and quality. Illumina's Genome Analyzer (GA) produces a significant larger volume of sequence data than traditional Sanger sequencing. Compared to just a few years ago, it is now much easier and cheaper to sequence entire genomes, and a wide variety of species are being studied using these advanced genetic analysis tools. Because of the rapid improvements in cost and quality of sequencing data, de novo sequencing and assembly is possible not only in large sequencing centers, but also in small labs.

In parallel with the technological improvements that have increased the throughput of the next-generation short-read sequencers, many algorithmic advances have been made in de novo sequence assemblers for short-read data. High quality *de novo* assembly using Illumina Genome Analyzer reads is possible today using many of these assemblers. Here we summarize the results of several de novo sequencing experiments using E. coli and human data.

Figure 1: De Novo Assembly Workflow


Simulations to Estimate Coverage (optional)
Ensure coverage is ≥ 50
Prepare Insert Libraries
Combine high-coverage short-insert library with long-insert libraries
Genome Analyzer Run
Apply Quality Filters
Bacterial Assembly
Velvet or SOAPdenovo
(de Bruijn graph methods)
Choice of k
Forge
(overlap methods)
Output
Contigs
Scaffolds/supercontigs
Mammalian Assembly
ABySS
(de Bruijn graph methods)
Choice of k
Output
Contigs
Check Quality of Assembly
Contig size distribution: N50, longest contig/scaffold
Align reads to contigs
Map to Reference Genome (if available)
Genome Coverage

Assembling a genome using the reads generated by the Genome Analyzer requires a different approach than the overlap methods that were developed for the long reads produced by Sanger sequencing. For example, the software packages that assemble the reads into a genome need to be able to process a large number of short reads. A critical step during assembly is the optimization of parameters such as coverage, paired-end insert length, and data quality filtering. After proper optimization, we show that, for Escherichia coli, we are able to achieve a genome coverage of up to 99.4% (N50 = 82,595 bp; largest scaffold = 482,333 bp) and 99.72% (N50 = 97,333 bp; largest scaffold = 233,793 bp) with two assemblers tested. While larger genomes require large amounts of memory, the latter assembly was replicated on a 64 bit Linux desktop with a 2 Ghz processor and 4 Gb of RAM in 24 minutes. Additionally, by randomly removing sequencing reads such that the sequence data covered the genome at 50× depth, we were able to replicate the numbers above on a 32 bit Windows desktop with 3 Gb of RAM in 15 minutes processing time, using the Linux-like environment Cygwin. This demonstrates that in some instances, *de novo* assembly can be performed with minimal computational resources.

In this technical note, we provide guidance for designing studies and filtering data to produce high quality assemblies (see Figure 1 for a *de novo* assembly workflow). In addition, we test various publicly available packages in assembling a bacterial and a human genome.

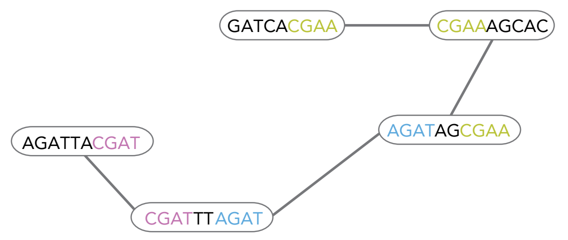
Assemblers

There are two basic approaches in algorithms for short-read assemblers: overlap graphs and de Bruijn graphs. These approaches are described below.

Overlap Graphs

Most established assemblers that were developed for Sanger reads follow the overlap-layout-consensus paradigm. They compute all pair-wise overlaps between the reads and capture this information in a graph. Each node in the graph corresponds to a read, and an edge denotes an overlap between two reads (Figure 2). The overlap graph is used to compute a layout of reads and a consensus sequence of contigs. This method works best when there is a limited number of reads with significant overlap.

Figure 2: Overlap Graph of Five Reads



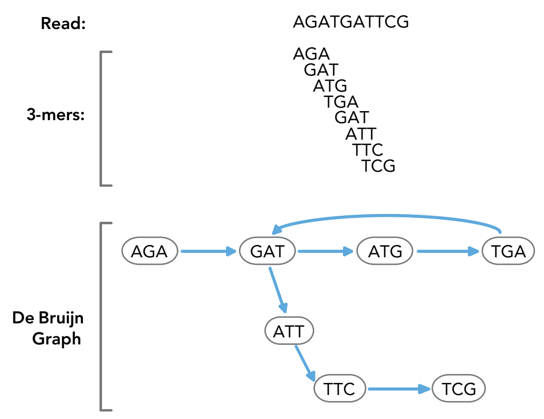
Colored nucleotides indicate overlaps between reads.

Some assemblers for next-generation sequence data use overlap graphs, but this traditional approach is computationally intensive: even a *de novo* assembly of simple organisms needs millions of reads, making the overlap graph extremely large.

De Bruijn Graphs

Because overlap graphs do not scale well with increasing numbers of reads, most assemblers for next-generation sequencing use de Bruijn graphs. De Bruijn graphs reduce the computational effort by breaking reads into smaller sequences of DNA, called k-mers, where the parameter k denotes the length in bases of these sequences. The de Bruijn graph captures overlaps of length k-1 between these k-mers and not between the actual reads (Figure 3).

Figure 3: De Bruijn Graph for Read with K=3



The length of overlaps is k-1=2. Gray arrows indicate where all the k-mers derived from the one read are placed in the graph. Blue arrows indicate the order of the k-mers and their overlaps.

By reducing the entire data set down to k-mer overlaps the de Bruijn graph reduces the high redundancy in short-read data sets. The maximum efficient k-mer size for a particular assembly is determined by the read length as well as the error rate. The value of the parameter k has significant influence on the quality of the assembly. Estimates of good values can be made before the assembly, but often the optimal value is best found by testing a small range of values. We provide more details in the “Recommendations” section.

Another attractive property of de Bruijn graphs is that repeats in the genome can be collapsed in the graph and do not lead to many spurious overlaps, although this does not mean that they can be more easily bridged or resolved. The maximum size of the de Bruijn graph is independent of sequence depth with an upper bound of 4k. Depending upon the genome being sequenced and the value of k, the de Bruijn graph may not reach the theoretical maximum, but in the presence of sequencing errors or biological variation, the memory footprint of the graph increases. Nevertheless, it has been our experience that reasonable error rates do not significantly increase the memory requirement.

A Sampling of Assemblers for Short Reads

The software package Velvet[1] was among the first assemblers for short reads and is now widely used. It implements an approach based on de Bruijn graphs, uses information from read pairs, and implements various error correction steps after building the graph. Velvet has successfully been used to assemble bacterial genomes[1].

SOAPdenovo[2] also implements a de Bruijn graph approach. In contrast to Velvet, error correction is performed before the actual graph is built.

The assemblers ABySS3 also uses the de Bruijn graph method. Its advantage is that it can be run in a parallel environment and thus has the potential to assemble much larger genomes. For example, Simpson et al. demonstrate the assembly of a human genome using ABySS3. SOAPdenovo also implements a parallel assembly algorithm based on de Bruijn graphs but details of this tool are not yet published.

Forge[5] implements an overlap-layout-consensus approach with various changes to accommodate Illumina reads. It distributes the computational and memory consumption on various nodes and has therefore the potential to assemble much larger genomes, despite not being a de Bruijn graph method.

An overview of the tested assemblers is given in Table 1.

Note

The analysis presented here represents a snapshot in time of a subset of the currently available assemblers. For example, much of our analysis was performed using Velvet version 0.7.31 but several releases have occurred since we downloaded and tested this software. Assemblers evolve constantly and we anticipate that new methods will be developed to allow mammalian genomes to be more rapidly and efficiently assembled.

Comparing Assembly Outcomes

The outcome of an assembly is a set of contigs. A contig is a contiguous assembled piece of DNA sequence. Some assemblers also compute scaffolds, which is a set of contigs for which the relative orientation and distance is known. An alternative to scaffolds are supercontigs: contigs in which gaps are allowed.Gaps are usually denoted by the letter ‘N’ in the DNA sequence.

Table 1: Overview of Tested Assemblers

Algorithm	Description	Strength	Genomes Assembled
Velvet	De Bruijn graph based Error corrections after graph is built	Fast (~30 mins) Easy to use Larger supercontig N50	Bacterial (Ref. 1; this technical note)
SOAPdenovo	De Bruijn graph based Error correction before graph is built	Easy to use Multi-threaded mode	Panda, Bacterial (Ref. 11; this technical note)
ABySS	De Bruijn graph based Can be run in parallel Distributed memory model (efficient)	Easy to use Largest contigs/scaffolds Best suited for large genomes	Human (Ref. 3; this technical note)
Forge	Overlap-layout-consensus method Modifications to accommodate Illumina reads	Largest contigs/supercontigs Good "long read" assembler	Bacterial (this technical note)

The following metrics are often used to compare the quality of assemblies:

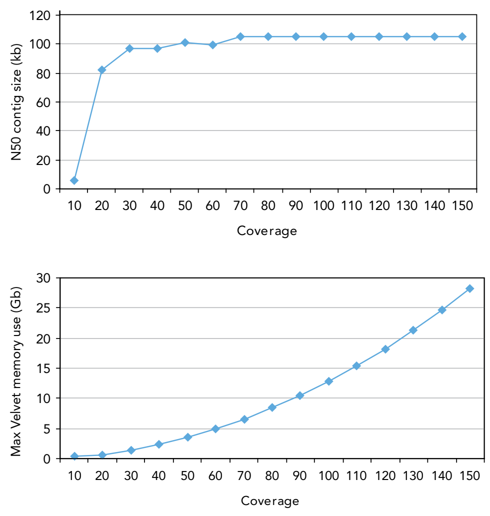
- **N50—The contig length such that 50% of the *de novo* assembled genome lies in blocks of this size or larger. N80 or N60 are also used.**
- **Genome coverage—The percentage of bases in the reference covered by the assembled contigs. This can only be computed if a reference genome exists, and the way the assembled contigs are mapped to the reference can have a significant influence on this parameter. Popular tools for mapping include BLAST[6], MUMmer[7], and SSAHA[8]. Contigs aligning with large internal gaps or low confidence indicate either structural variants or misassemblies, and should be carefully monitored. If paired reads are available, mapping them back onto the contigs is another source of information. Large deviations in the mapping distances or read pairs in which only one read maps against the contig hint at misassemblies rather than structural variants.**
- **Maximum/median/average contig size—Usually calculated after removing the smallest contigs (for example removing all contigs less than 150 bp in length).**

The contig sizes and their distribution are convenient and straightforward quality metrics but they do not contain all of the information needed to judge quality. For example, an assembly comprising several medium-sized contigs can be high-quality if these contigs cover the inter-repeat regions of the genome with high accuracy. On the other hand, an assembly consisting of one large contig with roughly the length of the genome being assembled is useless if the contig is incorrectly assembled.

Sequencing Parameters

The most important sequencing parameters are discussed in this section. We also provide examples of assemblies of E. coli data using Velvet to illustrate the influence of the parameters on the quality of the assembly. We performed several assemblies with Velvet, using different values of k, and determined that Velvet works best with k=31 for the E. coli data set. Note that higher values of k may perform better but Velvet 0.7.31 only supports values up to 31. This value of k was used throughout the work described here.

Figure 4: Effect of Coverage



Effect of coverage on N50 contig size and memory requirements in an E. coli de novo assembly.

Coverage

A high-quality de novo assembly cannot be achieved unless there is a sufficient number of error-free reads covering the entire genome. To achieve this and thus produce a high-quality assembly, a high depth of coverage is essential. The coverage needed will depend on the organism, its genome size, and the repeat content. To give an example, the Beijing Genomics Institute sequenced the Giant Panda genome using 75 bp reads at a coverage of 50×[11].

To evaluate the influence of coverage on a bacterial assembly, we created simulated data sets based on the E. coli genome using error-free reads. We simulated 75 bp paired-end data sets with a 200 bp insert size assembled using Velvet 0.7.31 (Figure 4) and different coverages.

As can be seen from Figure 4, a coverage of more than 50× does not yield a significant improvement in terms of the contig sizes. Since Velvet's memory usage increases with sequencing depth, higher coverage can require a significantly larger amount of memory with little or no improvement in assembly quality.

To further examine whether 50× coverage is sufficient, we used real sequencing data: a single GA lane from a 200 bp insert library of E. coli with 75 bp paired reads. After removing reads that did not pass the GA analysis software Failed_Chastity filter (described later) or containing Ns, the coverage is 320×. From this starting data, we randomly removed reads to generate samples with lower coverage, and assembled using Velvet 0.7.31.

Table 2 shows that the contig sizes drop as expected when the coverage drops under 50×, but contig sizes remain stable at higher coverage. We expect similar results for genomes with sizes, base compositions, and repeat contents similar to E. coli. In principle, it is possible to obtain several good assemblies from a single GA lane using multiplexing to sequence several bacterial samples.

Table 2: Effect of Coverage on Assembly Quality

Coverage	N50 contig size	Largest contig	Genome coverage
320×	95,313 bp	215,645 bp	99.47%
160×	95,368 bp	209,234 bp	99.72%
50×	97,333 bp	223,793 bp	99.72%
21×	35,828 bp	119,071 bp	99.38%

In general, the coverage threshold above which no improvement in N50 is possible depends on the size and repeat content of the genome to be sequenced. Still, even for larger genomes, it is expected that after a certain coverage is reached, adding more short-insert reads will not improve the assembly.

Read Length

The Genome Analyzer can generate paired reads with a length of 100 bp and more. There are, however, alternate technologies such as pyrosequencing that produce longer unpaired reads. In this experiment, we investigate the influence of read length on an assembly.

Chaisson et al.12 used simulations to show that reads with more than 36 bp and 60 bp do not improve assemblies of E. coli and S. cerevisiae, respectively. They restricted their experiment to paired reads and the two abovementioned organisms with relatively simple genomes.

We simulated paired reads of 100 bp length (with a 400 bp fragment size) and 400 bp unpaired reads from the E. coli genome and human chromosome 20, respectively. All reads were simulated as error-free and at 50× base coverage. We used Velvet 0.7.31 with a k-mer size of 31 to assemble all four data sets.

Table 3 shows the contig sizes for each assembly. The assemblies using the 100 bp paired reads yield far larger contigs as compared to the 400 bp single read assemblies. This is in part due to the way a de Bruijn graph assembler such as Velvet works: the reads are split into smaller pieces, the k-mers. In this experiment, we used a k-mer size of 31 which is the maximum for Velvet 0.7.31. Therefore we compared in principle k-mers of length 31 with and without pairing information. The strong impact of read pairs as opposed to single-ended reads is evident.

To further explore the advantage of paired-end reads, we performed an assembly of the 100 bp paired reads in which we ignored the read pairing information. This means that Velvet treats the reads essentially as 100 bp single reads. Table 4 shows that the assembly quality decreases strongly when not using paired-ends. The 100 bp single-read contig sizes match surprisingly well the 400 bp assembly of both E.coli and chromosome 20 (Table 3). This illustrates the critical importance of read pairs for obtaining high-quality assemblies.

There are newer versions of Velvet which support, in theory, k-mers of unlimited size. In practice, the maximum k-mer size is restricted by the available RAM since longer k-mers need to be stored in a different data structure with higher memory requirements. Furthermore, for real data sets, sequencing errors and base coverage limit the maximum k-mer size as well. In the next experiment, we repeated the 400 bp assemblies with the most recent version of Velvet, 0.7.55, and a k-mer size of 119 (Velvet allows only odd k-mer sizes). The k-mer size increased significantly the memory footprint and the RAM usage peaked at ~80 GB for the assembly of the Human chromosome. Even more RAM will be required for real (non-perfect) data and this is why k-mers of this size or larger will be of limited practical use.

Table 5 shows the results of this experiment. The N50 of resulting assemblies are still smaller than the ones obtained with 100 bp paired reads (Table 3) but are of the same order of magnitude.

These experiments show that 400 bp reads, even at high coverages, do not provide an advantage over shorter paired reads even for Human chromosomes. Longer k-mers in a de Bruijn graph assembly increase the contigs obtained from 400 bp reads but in practice there is a limit on the maximum k-mer size that can be employed due to memory requirements and sequencing errors.

Table 3: Effect of Read Length

Sample	N50 contig size	Largest contig	Genome coverage
E. coli, 100 bp pe	132,786 bp	326,886 bp	99.87 %
E. coli, 400 bp sr	22,902 bp	127,976 bp	99.87 %
Chr. 20, 100 bp pe	70,744 bp	484,312 bp	92.69 %
Chr. 20, 400 bp sr	2,319 bp	22,823 bp	92.65 %

Table 4: Effect of Pairing Reads

Sample (100 bp reads)	N50 contig size	Largest contig	Genome coverage
E. coli, paired-end	132,786 bp	326,886 bp	99.87 %
E. coli, single read	23,326 bp	127,976 bp	99.87 %
Chr. 20, paired-end	70,744 bp	484,312 bp	92.69 %
Chr. 20, single read	2,320 bp	22,823 bp	92.43 %

One might argue that instead of de Bruijn graph methods, one should use traditional overlap-layout-consensus assemblers for 400 bp reads. But, in contrast to Genome Analyzer reads, there are only few publicly available tools that support long pyrosequencing reads and their characteristic sequencing errors.

Long-Insert Libraries

Even relatively simple genomes such as E. coli contain a significant number of repeats. The largest exact or near exact repeat in E. coli is about 5.4 kb. An assembly with only short inserts will result in a set of contigs with gaps at each repeat that are longer than the insert sizes.

Table 5: Effect of Long K-Mers (K=9)

Sample	N50 contig size	Largest contig	Genome coverage
E. coli, 400 bp sr	132,476 bp	326,884 bp	99.25 %
Chr. 20, 400 bp sr	68,659 bp	645,179 bp	92.78 %

To bridge these gaps, libraries with longer inserts are essential. We advise using a combination of long-insert libraries with a high-coverage short-insert library to obtain sufficient coverage.

To examine how long inserts can improve the assembly we sequenced E. coli with one GA lane using a 6 kb insert library and another lane using a 10 kb insert library, then combined these with a 200 bp insert library and repeated our assembly. All of the de novo assemblers that we tested have the ability to use long-insert libraries. Table 6 shows results obtained using Velvet 0.7.31 with a variety of long-insert libraries. The 6 kb inserts already help bridge most of the repeats in E. coli if combined with the short-insert library, and result in an assembly with an N50 of 1,303,210 bp. The two largest contigs cover a region of 3,378,195 bp, which corresponds to 70% of the E. coli genome. The rest of the genome is contained in smaller contigs.

Table 6: Effect of Insert Size on Assembly

Inserts	Reads (bp)	Coverage	N50 supercontig	Largest supercontig	Genome coverage
200 bp	2×75	50×	97 kb	223 kb	99.58%
200 bp + 6 kb	2×75 2×35	50× 28×	1.3 Mb	2.1 Mb	99.07%
200 bp +10 kb	2×75 2×35	50× 28×	4.5 Mb	4.5 Mb	99.69%

If we combine the short-insert library with the 10 kb insert library, we obtain an assembly with one single supercontig covering ~98% of the E. coli genome. While the genome coverage as a whole does not increase significantly, the assembly is much less fragmented and will be more useful in, for example, analysis of structural variations.

In general, libraries with larger insert sizes will result in less fragmented assemblies and larger contigs. The maximal insert size needed will depend on the repeat structure of the organism to be sequenced. Note that there is a trade-off between the mean insert size and the variance of the insert size: libraries with larger mean insert size usually have a larger variance. This might hamper the scaffolding step in an assembly algorithm. For large-scale assemblies, a mixture of different long-insert libraries will give the best result.

Data Quality

Filtering read data (i.e. removing low-quality reads) can improve the assembly. We recommend removing reads that do not pass the GA analysis software Failed_Chastity filter before attempting to assemble the sequence. The chastity of a base call is the ratio of the intensity of the greatest signal divided by the sum of the two greatest signals. Reads do not pass the quality filter if there are two or more base calls with chastity of less than 0.6 in the first 25 cycles. These reads have an "N" in the last column of the GA analysis software export file. You can apply more stringent filtering criteria, but there is a trade-off between obtaining higher quality data and reducing coverage.

To illustrate the influence of filtering, we used sequencing reads of a single GA lane from a 200 bp insert library of E. coli with 75 bp paired reads, using Velvet 0.7.31. We applied the following sequential filtering criteria:

- **Turned off filtering.**
- **Removed all reads that did not pass the Failed_Chastity filter (PF).**
- **Removed all reads that contained ambiguous characters (Ns). Note: Some assemblers, such as ABySS, do this automatically, while others, such as Velvet, simply replace the N with a random nucleotide (A,C, G or T).**
- **Removed reads that did not contain at least 25 Q30 bases among the first 35 cycles (s35). Q30 refers to the Phred score of the base call and is defined as an error probability of 0.001.**

Table 7: Effect of Filtering on Assembly Quality

Filtering	Read Coverage	N50 contig size (bp)	Largest contig (bp)	Genome coverage
No filtering	420×	12,083	62,228	99.37 %
Only PF	328×	95,351	209,222	99.63 %
PF + Ns removed	320×	95,313	215,645	99.47 %
PF + Ns + s35 removed	203x	95,338	268,040	99.58 %

As can be seen from Table 7, removing reads not passing the Failed_Chastity filter greatly improves the quality of the build. Additional filtering steps increase the size of the largest contig, but do not improve the overall assembly. The genome coverage is high for all assemblies and differences between the filtered assemblies are within the expected variability.

Other filtering methods such as a k-mer-based error correction4,9 can make sense depending on the organism and assembly algorithm used.

Table 8: Comparison of Contig Assembly

Software package	N50	Largest contig	Genome coverage
Velvet 0.7.31, k=31	61,802 bp	115,666 bp	99.72%
ABySS 1.0.8, k=42	45,171 bp	140,706 bp	99.64%
Forge 1.0, k=15	70,447 bp	444,471 bp	99.4%
SOAPdenovo 1.0	3,026 bp	20,258 bp	99.51%

Error Rate

We created a series of simulated data sets based on the E. coli genome to investigate the influence of sequencing errors. We simulated different error rates in sequencing reads, and used Velvet 0.7.31 to perform an assembly at 150× coverage (Figure 5).

The results for an error rate less than ~4% match the contig sizes we obtained using real E. coli reads. There is a sharp drop in contig sizes as soon as the error rates surpass 4%. This error rate is well above the average error rate for a good GA run, indicating that sequencing error does not usually limit the assembly quality (as shown in Table 2).

Testing Assembelers

Comparison of Assemblers on a Bacterial Genome

We compared currently available assemblers for Illumina reads using a single GA lane from 200 bp insert library of E. coli with 75 bp paired reads, down-sampled to 50× coverage.

It is difficult to compare the results of assemblers directly, since they produce different outputs: ABySS computes only contigs without gaps whereas Velvet, Forge, and SOAPdenovo compute sets of contigs, "sequence-connected-supercontigs (SCSS)" in Velvet, supercontigs in Forge and SOAPdenovo. To make the results comparable, we generated two tables.

Table 8 shows a comparison based on the contig sizes, where supercontigs/scaffolds for Velvet, Forge, and SOAPdenovo were split whenever at least one gap character ('N') occurs. Table 9 shows a comparison based on the supercontig/scaffold sizes. Since ABySS does not generate supercontigs, it is omitted from this table.

Comparing supercontig/scaffolds, Velvet produced the largest N50 statistic in the E. coli assembly using short inserts, but Forge and SOAPdenovo computed assemblies of similar quality and contig size distribution (Table 9). In fact, Forge produced a much longer scaffold, but took ~50 times longer than Velvet to run (~30 minutes for Velvet versus ~24 hours for Forge). We executed Velvet on a machine with 60 GB of RAM and 16 CPUs with 2.4 GHz. Note that Velvet does not make use of multiple CPUs. Forge was executed in a parallel fashion on a cluster with 20 CPUs and 4 GB RAM per CPU. Most computing time was spent on building the scaffold and traversing the overlap graph.

Table 9: Comparison of Supercontig/Scaffold Assembly

Software package	N50	Largest scaffold	Genome coverage
Velvet 0.7.31, k=31	97,333 bp	223,793 bp	99.72%
Forge 1.0,k=15	82,595 bp	482,322 bp	99.4%
SOAPdenovo 1.0	95,472 bp	223,876 bp	98.61%

Assembly of Larger Genomes

Since ABySS uses parallelization and de Bruijn graphs, it can be used for de novo assembly of larger genomes. The other assemblers have limitations that become prohibitive when assembling a large genome, such as a mammalian genome.

We tested ABySS using reads from a Yoruba male (child of the individual published in Bentley et al.[10]) with the HapMap reference number NA18506. The data set consisted of 100 bp paired reads sampled at 30× coverage with an insert size of 600 bp. We first assembled chromosomes 1 and 20, to serve as medium-sized genome test cases. After that, we assembled the whole human genome.

Medium-Sized Genome Assemblies

We aligned all reads from the Yoruba male against the NCBI human reference genome and used reads aligning to chromosome 1 and 20 to assemble both chromosomes. These chromosomes have a size of 247 Mb and 62 Mb respectively and thus fall into the gap in genome size between E. coli and mammalian genomes.

After assembly, we discarded contigs with less than 100 bp to make the results comparable to previously published data3 (Table 10).

Table 10: Assembly of Human Chromosome 1 (K=55) and Chromosome 20 (K=62) by Abyss 1.0.8

Chromosome	Size (bp)	N50 contig size (bp)	Largest contig (bp)	Bases in contigs (Mb)
Chr. 20	62,435,965	4,743	48,538	64
Chr. 1	247,199,719	2,879	32,516	197

ABySS assembles both data sets into reasonably sized contigs. Contigs of this size can be useful for characterizing Single Nucleotide Polymorphisms (SNPs) and small to medium-sized structural variants. Further improvements in contig size can be obtained by adding long-insert libraries.

Whole Human Genome Assembly

We also performed a prototype assembly of the whole genome. The first stage of assembly, which was performed without the read pairing information, took ~20 hours on a cluster with 150 cores. Joining and error correcting the resulting contigs required an additional three days.

Due to the high repeat content and the small insert size, this assembly is highly fragmented. The largest contig had a size of 27,534 bp, but the N50 is much lower than the N50 that we achieved for chromosome 1. Whole-genome assembly of a mammalian genome with ABySS may therefore provide a starting point, but requires significant hands-on assembly afterwards. However, we expect that assemblies of whole mammalian genomes will improve with further improvements in algorithms and the application of long-insert libraries.

Recommendations

The assembly of bacterial genomes using current Illumina sequence reads can be performed with a number of publicly available assemblers, such as Velvet, Forge, and SOAPdenovo. With a combination of short- and long-insert reads and sufficient coverage, it is possible to assemble the E. coli genome in one supercontig covering almost the entire genome. We anticipate that comparable results can be obtained for other bacterial genomes with a repeat content and a base composition similar to E. coli. Larger genomes can be assembled successfully if coverage is large enough and long-insert libraries are used.

Parameter Optimization

When performing a whole genome sequencing project, take care to optimize the following parameters:

- **Size of k-mers: The size of k and thus the size of k-mers from which the graph is built is crucial for all de Bruijn graph based assemblers. The right choice for k depends on coverage, read length, and error rates and is hard to determine in advance. Anecdotal recommendations indicate that the size of k should not be lower than half of the read length. If time allows, we recommend performing several assemblies over a small range of k and choosing the one that yields the best assembly for the desired application.**
- **Quality filtering: Remove low-quality reads that fail the Failed_ Chastity filter. These reads carry an "N" in the last column of the GA analysis software export file. Additional filtering steps do not necessarily improve the overall assembly.**
- **Insert size: In general, libraries with larger insert sizes will result in less fragmented assemblies and larger contigs, depending on the repeats of the organism. We also recommend combining long-insert libraries with a high-coverage short-insert library to obtain sufficient coverage.**
- **Coverage: The coverage threshold above which no improvement in N50 is possible will depend on the size and base composition of the genome to be sequenced. We have observed few improvements in the assembly at sequencing deeper than 50×.**
- **Sequencing error rate: The results for an error rate less than 4% are acceptable. There is a sharp drop in contig sizes as soon as the error rate surpasses 4%. Optimally, to create the largest supercontigs, the insert sizes should be large enough to span the largest repeats. If feasible, we recommend using simulations to obtain a reasonable error rate. If not, we recommend at least 30–40× coverage of good-quality sequence data.**
- **Aligning the contigs back to the reference genome should be done to check the quality of the resulting assembly and search for structural variants, if applicable. Recommended tools include BLAST[6], MUMmer[7], or SSAHA[8].**
- **Aligning reads to the assembled contigs will help to determine insert sizes which differ from the expected one and to detect misassemblies.**

Assemblers

Velvet stands out as the algorithm producing the largest N50 statistic in the E. coli assembly using short inserts. Forge and SOAPdenovo computed assemblies of similar quality and contig size distribution. While by far the fastest assembly algorithm for small genomes, Velvet requires a lot of RAM for larger genomes.

To give an example, we used Velvet to assemble the human chromosome 20 (~ 64.5 Mb) from high-quality reads on a machine with 64 GB RAM. At its peak memory usage, Velvet used most of the available RAM. Assemblies of larger genomes or assemblies with error-prone reads will require significantly more memory. Velvet is easy to use and has a well-written manual. Velvet requires some Linux skills and knowledge of the command line, just like the other assemblers presented here. Overall, Velvet is the software of choice for small to medium-sized bacterial genomes and can be executed using an affordable compute setup.

Forge's limitation is its speed: the size of the contigs and scaffolds is comparable to Velvet, but the assembly process takes an order of magnitude longer. Due to its distributed approach, it can be executed on a cluster and has the potential to assemble large genomes. Nevertheless, this ability is constrained by its use of a traditional overlap graph.

ABySS seems to be the only algorithm that is currently able to compute an assembly of mammalian-sized genomes from short reads. This is due to its implementation of a distributed de Bruijn graph. But the obtained assembly is highly fragmented, in part because the tested version of ABySS does not perform scaffolding. We expect this to change with the further improvement of assembly algorithms and the application of long-insert size libraries for mammalian genomes.

For E. Coli, SOAPdenovo created assemblies of similar quality to Forge and does not have the large memory footprint of Velvet. The Beijing Genomics Institute has created and used a newer version of SOAPdenovo (publication pending) to assemble the Panda11 and human genomes sequenced as part of the 1000 Genomes Project. When available, the new version of SOAPdenovo will provide researchers with multiple software options to assemble mammalian-sized genomes from next generation sequencing data.

Conclusion

One can perform high quality de novo sequence assembly using Illumina Genome Analyzer reads and publicly available short-read assemblers. In many instances, existing computer resources in the laboratory are enough to perform *de novo* assemblies. The recommendations provided in this technical note can be used to perform genomic assemblies efficiently.

References

1. Zerbino DR, Birney E (2008) Velvet: algorithms for *de novo* short read assembly using de Bruijn graphs. Genome Research 18: 821–829.
2. BGI (Beijing Genome Institute). SOAP: Short Oligonucleotide Analysis Package. http://soap.genomics.org.cn.
3. Simpson JT, Wong K, Jackman SD, Schein JE, et al. (2009) ABySS: A parallel assembler for short read sequence data. Genome Research, 19: 1117-23.
4. Butler J, MacCallum I, Kleber M, Shlyakhter IA et al. (2008) ALLPATHS: *de novo* assembly of whole-genome shotgun microreads. Genome Research, 18: 810-820.
5. Platt D, Evers DJ Forge (2009) Unpublished Manuscript. See http://forge.sourceforge.net/
6. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. J Mol Biol 215: 403-10.
7. Kurtz S, Phillippy A, Delcher AL, Smoot M et al. (2004) Versatile and open software for comparing large genomes. Genome Biology 5:R12.
8. Ning Z, Cox AJ, Mullikin JC (2001) SSAHA: a fast search method for large DNA databases. Genome Research 11: 1725-9.
9. Pevzner PA, Tang H, and Waterman MS (2001) An Eulerian path approach to DNA fragment assembly. PNAS 98, 9748-53.
10. Bentley DR, Balasubramanian S, Swerdlow HP, Smith GP et al. (2008) Accurate whole human genome sequencing using reversible terminator chemistry. Nature 456: 53-59.
11. Application note available at: http://www.illumina.com/downloads/BGIdenovo_AppNote.pdf
12. Chaisson MJ, Brinza D, Pevzner PA (2008) *De novo* fragment assembly with short mate-paired reads: Does the read length matter? Genome Research. 19:336-46

Additional Information

Visit our website or contact us at the address below to learn more about Illumina sequencing products and software solutions.

Illumina, Inc. • 9885 Towne Centre Drive, San Diego, CA 92121 USA • 1.800.809.4566 toll-free • 1.858.202.4566 tel • techsupport@illumina.com • illumina.com

FOR RESEARCH USE ONLY