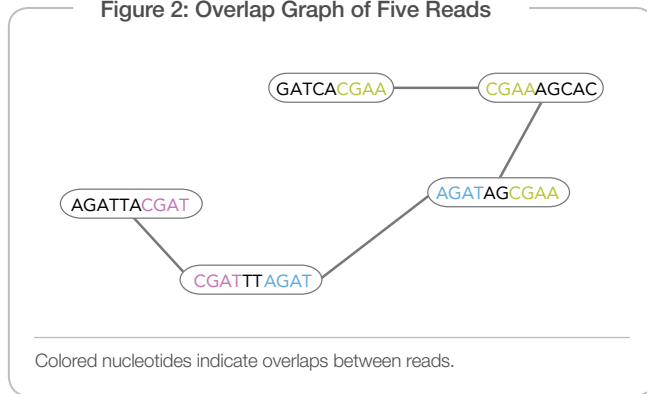


Figure 2: Overlap Graph of Five Reads



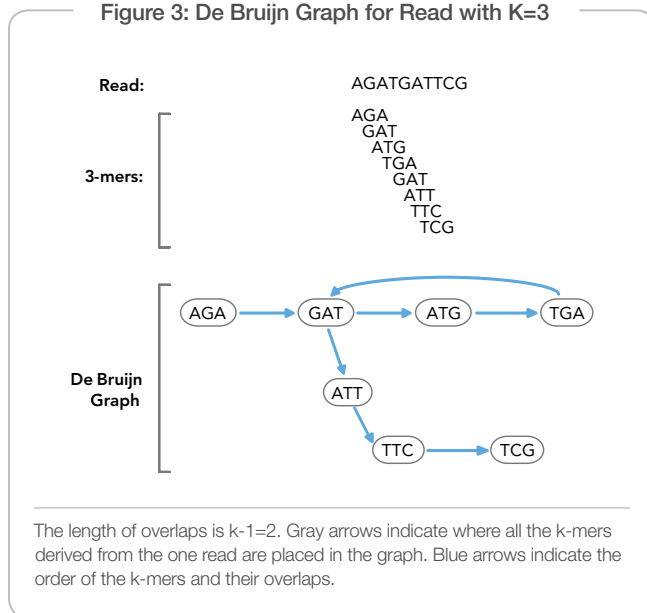
Colored nucleotides indicate overlaps between reads.

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 Velvet¹ 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 genomes¹.

SOAPdenovo² 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.

Forge⁵ 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 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 150x 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 Assemblers

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 50x 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

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%

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.

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 30x 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.

