Assembly and Data Quality

5.1 Data Quality and Filtering – 82

5.2 Assembly Strategies – 84

5.2.1 Greedy Assemblies – 87

5.2.2 Overlap-Layout-Consensus (OLC) Assemblies – 88

5.2.3 K-mer Assemblies Using de Bruijn Graphs – 90

5.3 Comparing Assemblies – 94

5.4 De Novo Assembly of Genomes – 96

5.4.1 Scaffolding – 96

5.4.2 Hybrid Assemblies – 97

5.5 De Novo Assembly of Transcriptomes and Metagenomes – 97

References – 100

© Springer International Publishing AG 2017
C. Bleidorn, *Phylogenomics*, DOI 10.1007/978-3-319-54064-1_5

- The outputs of a sequencer are sequence reads, and each of its nucleotides receives a quality score, indicating the error probability.
- Overlapping sequence reads can be assembled into contiguous stretches of DNA called contigs, which can further on be ordered into scaffolds.
- Three main types of assembly strategies are in use, based on greedy algorithms, overlap-layout-consensus approaches or *k-mer* graphs.
- Different strategies are used for genome, transcriptome and metagenome assemblies, and all of them greatly benefit from the inclusion of long sequence reads.

5.1 Data Quality and Filtering

Sequence reads can be of either good or bad quality. To measure the error probability for a given base in a given sequence read, quality scores have been developed already back in the 1990s for Sanger sequencing. Based on sequence chromatograms, error probabilities were calculated for each position resulting in a quality score named *Phred* (Ewing and Green 1998).

$$Q_{\text{Phred}} = -10\log_{10}(P) \quad \textbf{(5.1)}$$

In this formula, P is the expected error probability for a given base call and Q_{Phred} specifies the according, logarithmically linked *Phred* score (Table 5.1). For example, a base call having a probability of 1/1000 to be wrong receives a *Phred* score of 30. High *Phred* scores correspond to low base-calling error probabilities, whereas low scores indicate higher ones. *Phred* quality values are always rounded to the nearest integer. *Phred* scores can handily be used to estimate the number of expected errors in sequence projects. Let us assume we sequenced a 70 Kb insert of a BAC clone with an average *Phred* score of 32 for every base. Given the formula above, this translates to an error probability of 0.00063, and one would have to expect ~44 wrongly called bases in this sequence.

Phred qualities are predicted without reference to a «true» sequence, but they were shown to correspond well with observed error rates. Moreover, it has been demonstrated that *Phred* scores have a high sensitivity to discriminate between correct and incorrect base calls. Due to their usefulness, these scores have been incorporated into Sanger sequencing machine analysis software early on. As such, *Phred* scores were routinely used to make decision regarding double peaks in the chromatogram or for trimming the ends of sequences to get rid of low-quality regions.

Table 5.1 *Phred* score and error probabilities

Phred score	Probability of incorrect base calls	Accuracy of base calling (%)
10	1 in 10	90
20	1 in 100	99
30	1 in 1000	99,9
40	1 in 10000	99,99
50	1 in 100000	99,999

Next-generation sequencing (NGS) techniques like Illumina, 454 or PacBio usually produce sequence reads with much higher error rates than Sanger sequencing. For each of these sequencing techniques, separate quality scores to assess sequencing reads were introduced. Unfortunately, the first quality scores introduced for Illumina (then called Solexa) sequencing were different from the established *Phred* scores and calculated as follows:

$$Q_{\text{Solexa}} = -10\log_{10}(p/1-p) \tag{5.2}$$

Using a rearrangement of this formula and an equation of the estimated error, it is possible to transform Q_{Solexa} values into *Phred* values:

$$Q_{\text{Phred}} = -10\log_{10}(10^{\text{QSolexa}/10}+1) \tag{5.3}$$

In 2009, Illumina also introduced the commonly used *Phred* score system as quality scores, which is also used for 454 and Ion Torrent. A *Phred* mimicking quality score is available for PacBio (Hackl et al. 2014). The usual format to store this quality information is the fastq format (Cock et al. 2010). Nanopore sequencing data using the MinION does not follow *Phred* scores (Laver et al. 2015).

Sequence reads in the fastq format are organized in four lines (Fig. 5.1). The first line always starts with an @ and contains information regarding the sequencing description and a unique identification. The second line contains the raw sequence. Usually the third line only contains the «+» sign or the «+» sign and a repetition of the sequence identification. The fourth line contains the quality values and must contain the same number of characters as the sequence line. As *Phred* scores are often values higher than 10 (and therefore needing two characters for description), they are encoded using the ASCII code, which comprises 128 specified characters including alphabetic letters, numbers and symbols. Each of these characters specifies one integer of the *Phred* score. To make things complicated, different ranges of values had been historically used for Sanger and some

Fig. 5.1 Example of the fastq format. **a** Sequence reads are organized in four lines. The first line indicates a name, the second line comprises the sequence data, the third line only contains a + plus (sometimes also the name again), and the fourth line bears the *Phred* quality information coded in ASCII. **b** ASCII code encoding *Phred* scores ordered from low to high scores

Illumina fastq formats, which has to be taken into account when analyzing this quality data. For Sanger and starting with Illumina 1.8 (and recent versions), the ASCII characters 33–126 are used, indicating a range of the *Phred* quality score from 0 to 93. In contrast, for some Illumina versions (1.3 to 1.8) the ASCII characters 64–126 are used, indicating a *Phred* quality score range from 0 to 62. Obviously, using the wrong ASCII translation might result in the strong over- or underestimation of error rates.

Base calling can be conducted using various methods. Moreover, different sequencing strategies are prone to different error types. As such, 454 and nanopore sequencing often produce errors due to the misspecification of the number of bases in homopolymers. PacBio sequences seem to be especially prone to contain chimeras (Hackl et al. 2014). In contrast, for Illumina sequence data, it might be difficult to distinguish between A and C and G and T, as both pairs of bases show similar emission spectra. Another problem for Illumina sequencing is generated due to a phenomenon called phasing. In this case, the incomplete removal of the 3′-blocking and fluorophore leads to the detection of the wrong signal during the next cycle (Kircher et al. 2011). Moreover, inverted repeats seem to be a problem caused by PCR amplification of single-stranded DNA during library preparation (Nakamura et al. 2011). All these error sources have to be taken into account by base-calling programs. Several programs exist besides software distributed with the analysis pipeline of the Illumina machines, like IBIS and FREEIBIS (Kircher et al. 2009; Renaud et al. 2013). For MinION nanopore sequencing, NANOCALL is a freely available open source base caller (David et al. 2017).

After base calling, a first quality filtering of the sequence data is usually conducted. In this step, adapter sequences and barcodes (or index primer regions) are removed. In the next step, it is recommended to remove low-quality regions or discard such reads completely. For Illumina reads, often an exponential increase in error probabilities from the 5′- to 3′-end is observed. The stringency of the filter procedure is chosen by the user and different parameters might be exploited as part of the analysis. For example, using filtering sequence reads which contain more than 5% of bases under a specified quality score (e.g. *Phred* 20) could be removed. As also known for Sanger sequences, the 5′- and 3′-ends of reads often show lower quality than the rest of the sequence read and might be completely discarded. This process is called trimming. Initial rigorous quality checks of sequence reads clearly increase the quality and minimize the number of artefacts of subsequent analysis steps, as assembly or mapping. Several programs are available to visualize the distribution of error probabilities across reads, as, for example, the freely available software fastqc report (► http://www.bioinformatics.babraham.ac.uk/projects/fastqc/). TRIMMOMATIC is a widely used software for Illumina read trimming (Bolger et al. 2014), a package of perl scripts called CONDETRI is another easy to use for trimming and read filtering tools (Smeds and Kunstner 2011), and error correction can be conducted with the program QUAKE (Kelley et al. 2010). As long reads from PacBio or nanopore sequencing are often more error prone, they can be corrected in a hybrid approach using Illumina short reads (Salmela and Rivals 2014; Koren et al. 2012; Goodwin et al. 2015).

5.2 Assembly Strategies

Usually two different main strategies analyzing sequence reads are employed: assembly and mapping. Mapping describes a procedure to align sequence reads or assembled contigs on a given reference sequence. This reference might be a genome or transcriptome of

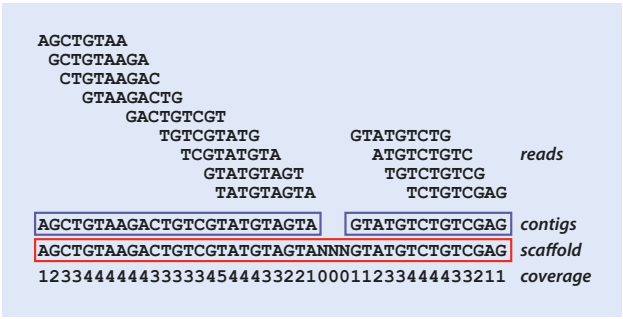


Fig. 5.2 Important terms for understanding sequence assembly methods. Reads are assembled into contigs, which can be ordered into scaffolds. The coverage shows the number of reads covering a certain position in the contig

the target species or from closely related species. As this strategy is basically an application of alignment methods, it will be introduced in the corresponding chapter. Assembly refers to the procedure of generating longer, continuous stretches of sequences by using combinations of shorter sequence reads (Miller et al. 2010). Some basic terms are important to know before digging deeper into how different assembly strategies and methods work (Fig. 5.2).

An assembly is a set of contigs computed from sequence reads. A sequence derived from assembling several sequence reads is called contig. Some methods further work with unitigs, which are basically high-confidence contigs (Myers et al. 2000). Evolutionary studies based on NGS data usually work with contigs, and it is important to remember that these do not refer to observations derived from a sequencing technique but are products of the ongoing analysis. Using different assembly strategies, different assembling parameters or even different quality procedures as described above may lead to different contigs. In the ideal case for whole-genome shotgun data, a single contig refers to a single chromosome, which might already constitute the complete genome in case of bacteria or organelle genomes. However, usually more and smaller contigs compared with the number of sequenced chromosomes are the result of the assembly. By using additional information (e.g. positional information from paired-end or mate pair reads), these contigs can be ordered into scaffolds, which further resolve the orientation of contigs to each other. In scaffolds, nonoverlapping stretches between contigs are marked by stretches of N's (unspecified bases). As part of the analysis, sequence reads can be mapped onto contigs. How often a given sequence position is covered by sequence reads is called coverage. A coverage of 10x means that any sequence position of an assembly is covered on average by ten sequencing reads. Before starting a sequencing project, it is useful to estimate the genome size of the target organism. For example, in case of the genome size of humans (~3 Gb), a single lane of paired-end sequencing (100 cycles) by the Illumina HiSeq sequencer would already produce an expected data volume of ~60 Gb, corresponding with a theoretical coverage of 20x (note that the practical coverage will be considerably lower). An oversampling of the genome in terms of coverage is important to have overlapping reads for assembly.

Assemblies are often compared with solving puzzles, and in case of de novo assemblies, even the desired picture is unknown. If a genome of the target species or from a closely related species is already available, it can be used to guide the assembly. Three main types of assembly methods are currently in use: greedy, overlap-layout-consensus methods and *k*-mer assemblies (Miller et al. 2010). Numerous assemblers are available for all these methods (Table 5.2).

Table 5.2 Overview of some current widely used sequence assemblers

Program name	Citation/source	Strategy	Remarks
ABruijn	Lin et al. (2016)	*k*-mer	Long-read assembly for PacBio and nanopore data
AbySS	Simpson et al. (2009) Robertson et al. (2010)	*k*-mer	Versions for genome and transcriptomes available
ALLPATHS	MacCallum et al. (2009)	*k*-mer	Hybrid assemblies using short and long reads
ARACHNE	Batzoglou et al. (2002)	OLC	Whole-genome assembly
Canu	Koren et al. (2016)	OLC	Long-read assembly for PacBio and nanopore data
CAP3	Huang and Madan (1999)	greedy	Useful for Sanger data
Celera	Myers et al. (2000)	OLC	Strictly a variant of OLC using so-called string graphs. Used to assemble the genomes of *Drosophila melanogaster* and humans
CLC	► https://www.qiagenbioinformatics.com/products/clc-genomics-workbench/	*k*-mer	Commercial software package for genomic applications. Easy to use and memory efficient
Edena	Hernandez et al. (2008)	OLC	Fast and memory efficient
Euler	Pevzner et al. (2001)	*k*-mer	First *k*-mer assembler
FALCON	► https://github.com/PacificBiosciences/FALCON-integrate	OLC	Long-read assembly of PacBio data
MEGAHIT	Li et al. (2016)	*k*-mer	Fast and memory-efficient metagenome assembler
Minia	Chikhi and Medvedev (2014)	*k*-mer	Memory efficient, usable in low memory environments
Miniasm	Li (2016)	OLC	Ultrafast long-read assembly for PacBio and nanopore data
MIRA	Chevreux et al. (2004)	hybrid	Swiss army knife of sequence assembly, useful for combining different technologies
Newbler	Distributed with 454 (Roche) sequencing platforms	OLC	Standard for 454 data

Table 5.2 (continued)

Program name	Citation/source	Strategy	Remarks
IDBA	Peng et al. (2010) Peng et al. (2011) Peng et al. (2013)	*k*-mer	Iterative *k*-mer size, versions for genomes, transcriptomes and metagenomes
Oases	Schulz et al. (2012)	*k*-mer	De novo transcriptome assembly, splice variants
PHRAP	▶ http://www.phrap.org/	greedy	Useful for Sanger data
SOAPdenovo	Luo et al. (2012)	*k*-mer	Assembly of the first eukaryotic genome solely based on short reads, different modules
SPAdes	Bankevich et al. (2012)	*k*-mer, hybrid	Assembler for bacterial genomes, hybrid module to include PacBio reads
TruSPAdes	Bankevich and Pevzner (2016)	*k*-mer	Assembler for synthetic long reads (e.g. TruSeq, 10x Genomics)
Trinity	Grabherr et al. (2011)	*k*-mer	De novo transcriptome assembler, splice variant detection
Velvet	Zerbino and Birney (2008)	*k*-mer	For genomes, included in some transcriptome assemblers

5.2.1 Greedy Assemblies

Assemblers using the greedy algorithm represent the most simple and intuitive approaches. In this case, sequence reads are iteratively joined to build contigs, starting with those showing the highest score for an overlap. These scores measure the amount of matching bases and the length of the overlap region, and parameters can usually be defined by the user. The operation of joining reads and/or contigs is repeated using the same rules till no more steps are possible. The term greedy refers to the fact that due to the search for best overlaps, only local optimal solutions are analysed, which leads to a result comparatively fast. However, the best overall (global) assembly might be missed using this strategy. An assembly of sequences from PCR experiments consisting of several overlapping fragments were usually constructed with this method, as implemented in widely distributed software, e.g. CAP3 (Huang and Madan 1999).

5.2.2 Overlap-Layout-Consensus (OLC) Assemblies

The OLC assembly can be divided into three steps. In the first step, pairwise alignments of all sequence reads are conducted, where overlaps for each pair of sequence reads are analysed. This overlap might be perfect (a match of corresponding nucleotides in every overlapping position) or contains few mismatches. However, only overlaps of ends of sequencing reads are allowed (Fig. 5.3).

The information of overlapping reads is stored in graphs, so-called overlap graphs (Fig. 5.4). Mathematical graphs are of central importance of many genomic methods and will appear in several chapters of this book. In the case of overlap graphs, the nodes in the graph represent sequence reads, whereas the branches (or alternatively called edges) indicate which reads are connected by an overlap (Fig. 5.4b). Due to the fact that edges should only be traversed in one direction (as indicated by arrows), the result is a directional graph. The number of subgraphs produced in this step will correspond to the number of contigs which are resolved.

The second step is the layout. During this step, the relative position of sequence reads (nodes) of every overlap graph is determined and arranged accordingly into an alignment. This is conducted by searching for a mathematical path describing a way to go over every

Fig. 5.3 Overlap of sequence reads. **a** Only overlaps of ends of sequence reads will be included in the subsequent graph. The minimum number of overlapping nucleotides has to be specified by the user and mismatches might be allowed. **b** When overlapping regions are in the middle of one of the sequences, they will not be used in subsequent analysis

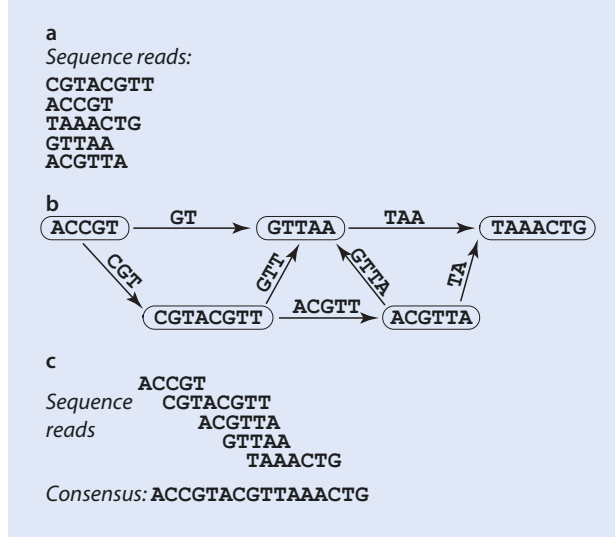


Fig. 5.4 Overlap-layout-consensus. **a** Sequence reads for assembly. **b** Overlap graph. **c** Alignment of reads after layout step, in which a Hamiltonian path was searched for in the overlap graph. The consensus sequence is the resulting contig

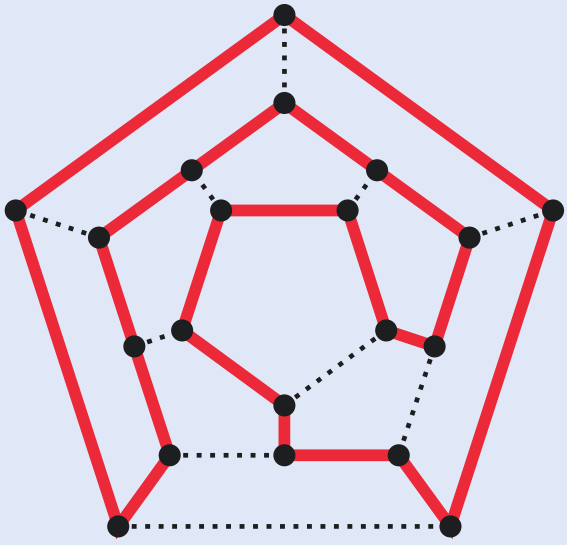
node exactly once, the Hamiltonian path (▶ see also Infobox 5.1). In the last step, the resulting alignments are used to determine consensus sequences which represent contigs (◘ Fig. 5.4c). Coverage information can be used to correct base calling or sequencing errors. For example, in case of resolving a certain nucleotide for a given position in a contig, the alternative supported by most of the reads covering this position is chosen. For example, when for a certain sequence position with a 10x coverage eight times an A is read, but only two times a C, then A will be chosen.

Infobox 5.1

The Century-Old Origin of Short-Read Genome Assembly Algorithms

Solutio problematis ad geometriam situs pertinentis was the name of an article by the Swiss mathematician and physicist Leonhard Euler (1707–1783) which described a solution for the so-called Bridges of Königsberg problem presented to the St. Petersburg Academy in 1735. This ground-breaking work not only solved an old mathematical problem but also was one of the first contributions to graph theory, including an idea that is now part of *k*-mer assembly strategies. The former Eastern Prussian and now Russian city of Königsberg (Kaliningrad) is located at the opposing sites of the river Pregel, as well as on two river islands. The four parts of Königsberg were joined by seven bridges. The «Bridges of Königsberg problem» asked the question if it would be possible two visit all four parts of the city by crossing every bridge exactly once, while returning to the starting point. Euler's brilliant idea to solve this problem was to represent each part of the city as a node and each bridge of the city as an edge and to connect them appropriately within a graph. Euler described a method that finds a path traversing the graph while visiting each edge exactly once, and this path is still known as the Euler path. Unfortunately, there is no way to cross the seven bridges of Königsberg exactly once and visiting all parts of the city.

Another important mathematical path is named after William R. Hamilton (1805–1865), an Irish mathematician and physicist. The Hamiltonian path visits each node of a graph exactly once. A graph that contains a Hamiltonian path that forms a cycle is called Hamiltonian cycle. Hamilton used such cycles to invent the icosian game. The aim of this game is to find a Hamiltonian path along the edges of a dodecahedron, a geometrical figure which might also be described as a cube with 12 flat faces (◘ Fig. 5.5).



◘ **Fig. 5.5** Hamiltonian Path through a Dodecahedron by Christoph Sommer - Own work. Licensed under Creative Commons Attribution-Share Alike 3.0 via Wikimedia Commons

```
true genome sequence:
AAGACTGTCGTATGTATATATACCAAGGTTCCTATATATATGTCTGTCGAGCGTC
AAGACTGTCGTATGTATATATA                          read_1
              TATATATATGTCTGTCGAGCGTC           read_2
AAGACTGTCGTATGTATATATATGTCTGTCGAGCGTC           assembly
```

Fig. 5.6 Example of a wrong assembly of a repetitive region. The repeat motive is given in *red*, a stretch of the true sequence which is missing in the resulting assembly is given in *blue*

OLC assemblers were originally developed for the analysis of Sanger sequence data. Accordingly, they are well suited when analysing moderate amounts of larger sequence reads (>500). The first sequenced animal and plant genomes were assembled with methods based on this strategy (Myers et al. 2000). A widely used assembler for 454 data, a sequencing technique which usually generates longer but less reads than Illumina sequencing, is NEWBLER (Table 5.2). In this software, two subsequent OLC steps are performed. In the first step, so-called unitigs are generated by the process described above. Unitigs are high-confidence contigs composed of reads which do not bear overlap with reads in any other unitig. In the second step, unitigs are joined into longer contigs based on pairwise overlap between unitigs.

However, some problems remain with OLC methods. Firstly, finding the Hamiltonian path is a mathematically NP-hard problem. Nondeterministically, polynomial-time hard problems are those which are not efficiently solvable by algorithms. This basically means that computers are and will always be too slow to calculate this kind of mathematical paths. As always in these cases, heuristic solutions are used in hope to find the best path for the problem. Secondly, the first step of finding overlaps by pairwise alignments becomes too time and memory intensive with NGS data. Originally developed for hundreds to rarely up to millions of longer sequence reads of very high quality, OLC becomes problematic to unusable when dealing with millions to billions of short reads of often lower quality. Moreover, usage of OLC methods can be problematic to resolve long repetitive regions and may produce misassemblies in this case (Fig. 5.6).

To avoid these problems, low-complexity regions (e.g. long stretches of a single nucleotide) and repetitive regions are often masked and discarded before assembly. Alternatively, an error correction step can be performed before starting the assembly. However, the availability of long-read sequences (e.g. from PacBio and nanopore sequencing) revived the OLC approach, and many assemblers dealing with this kind of data have been recently published (see Table 5.2).

5.2.3 K-mer Assemblies Using de Bruijn Graphs

Assemblies using de Bruijn graphs based on *k*-mers are composed of two steps: In the first step, the sequence reads are fragmented into smaller pieces called *k*-mers, which are used to construct a de Bruijn graph. In the second step, the contigs are derived from the de Bruijn graph (Schatz et al. 2010).

Every sequence, reads from a sequencer as well as complete genomes downloaded from GenBank, can be fragmented into *k*-mers. The *k* in *k*-mers denotes the size of the fragment, and after choosing this, the sequence is fragmented in all possible *k*-1 overlapping fragments of this size (Fig. 5.7).

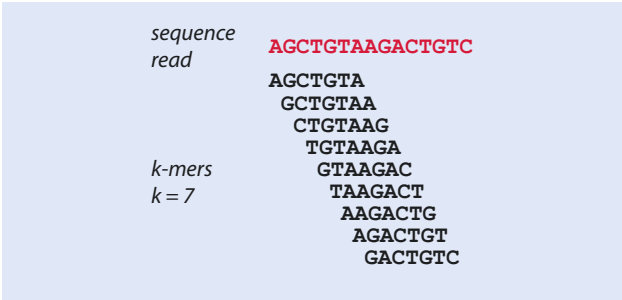


Fig. 5.7 An example sequence and all its *k*-mers of the size 7

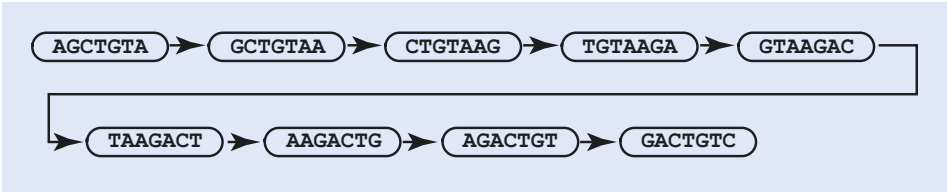


Fig. 5.8 De Bruijn graph of *k*-mers from Fig. 5.7

Surprisingly, the most common method to deal with de novo assemblies of short-read data is to fragment these short reads into even smaller pieces. The resulting *k*-mers are then connected via a de Bruijn graph. In the case of assemblies, the de Bruijn graph is a graph where the nodes represent sequences (*k*-mers) which are connected by edges in case they show a *k*-1 overlap. Arrows are used to indicate the direction of the overlap from the *k*-mer where the last *k*-1 nucleotides overlap to the *k*-mer with the first *k*-1 nucleotides (Fig. 5.8).

To reconstruct contigs, the de Bruijn graph has to be traversed by finding a Euler path (▶ see also Infobox 5.1). The Euler path goes exactly once over every edge of the graph. Reconstructing the contig derived from perfect *k*-mers of a single short sequence is a simple problem. However, real genomic data is usually more complex, including repetitive regions. Further on, real sequencing data usually contains errors. Both errors and repeat regions lead to more complex de Bruijn graphs which are much more difficult to resolve. The presence of repeats can introduce loops into the graph as illustrated by a simple example (Fig. 5.9).

Likewise, sequencing errors lead to more complex graphs. Errors in the middle of a sequence can lead to bubbles in the graph, whereas errors at the end of sequences may introduce dead ends (tips) into the graph (Fig. 5.10).

Both repeats and number of errors introduced are directly influenced by the chosen *k*-mer value. Unfortunately, this choice represents a trade-off (Chikhi and Medvedev 2014). Larger number of *k* reduces the number of repeats which can tangle the graph and break up contigs. Obviously, a repetitive region or sequence motive which is longer than the chosen *k* cannot be resolved. This would argue for choosing the highest possible value for *k*, which is limited by the length of the sequence reads. However, with longer *k*-mers, the probability that these *k*-mers contain sequencing errors increases. For example, given an error in the middle of a 100 bp sequencing read and a chosen *k*-mer size of 25, up to 25 erroneous *k*-mers are included into the analysis. When choosing a *k*-mer size of 13 for the same data, only up to13 erroneous *k*-mers are created. The choice of *k* also directly influences the size of the de Bruijn graph, as lower *k*-mer values decrease the number of edges stored in the graph which at the same time reduces the amount of memory needed to store

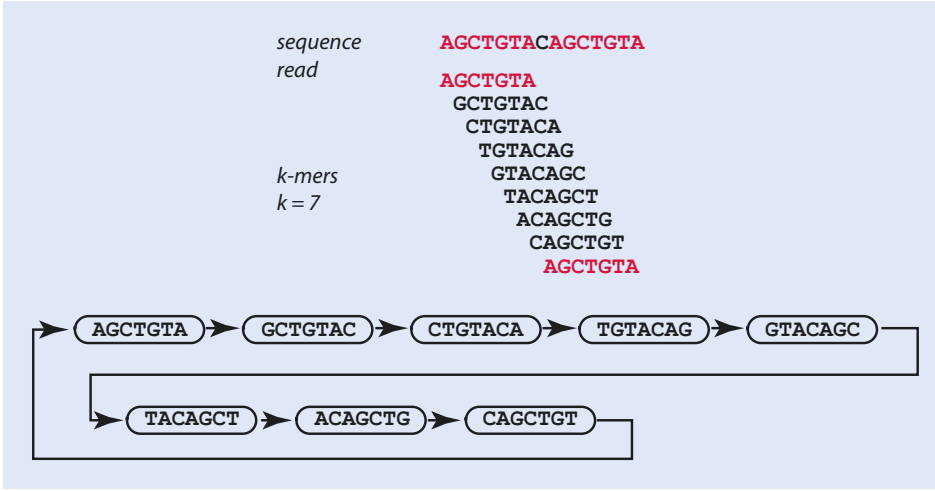


Fig. 5.9 Repetitive sequences can lead to loops in a de Bruijn graph. The repetitive motive is indicated in *red*

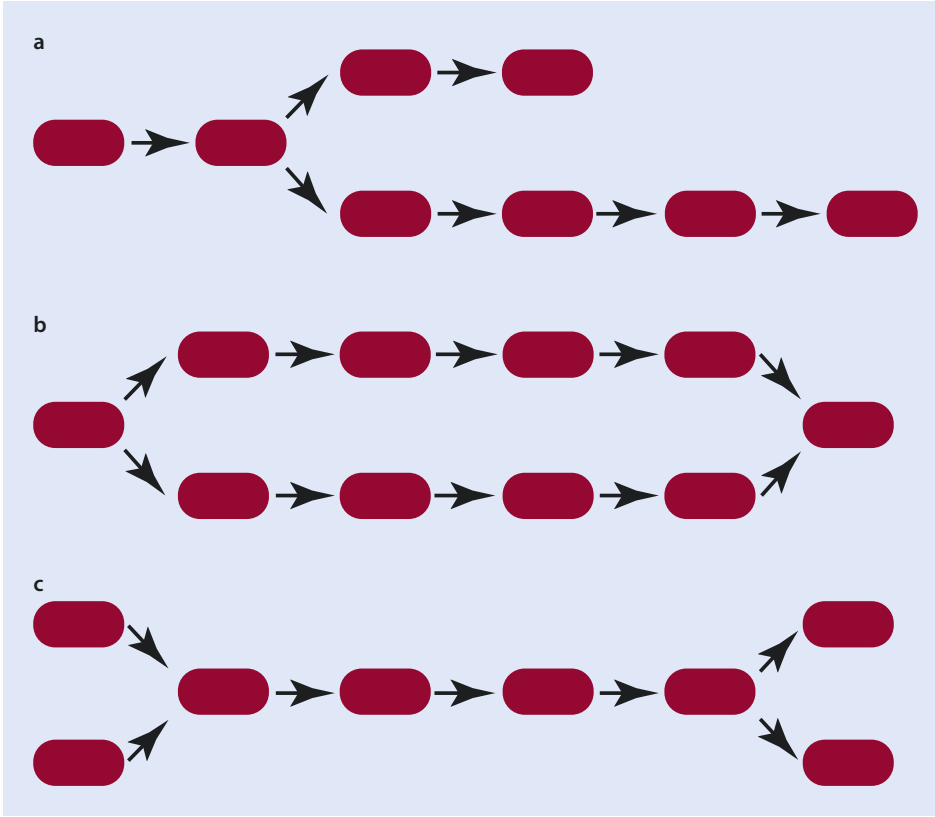


Fig. 5.10 Sequence errors and repeats lead to more complex *k*-mer graphs. Nodes representing *k*-mers are indicated by red boxes. **a** Errors at the end of sequence introduce dead ends into the graph **b**. Errors in the middle of sequences introduce bubbles into the graph. **c** Repeat sequences lead to a pattern of convergent and divergent paths (After Miller et al. (2010))

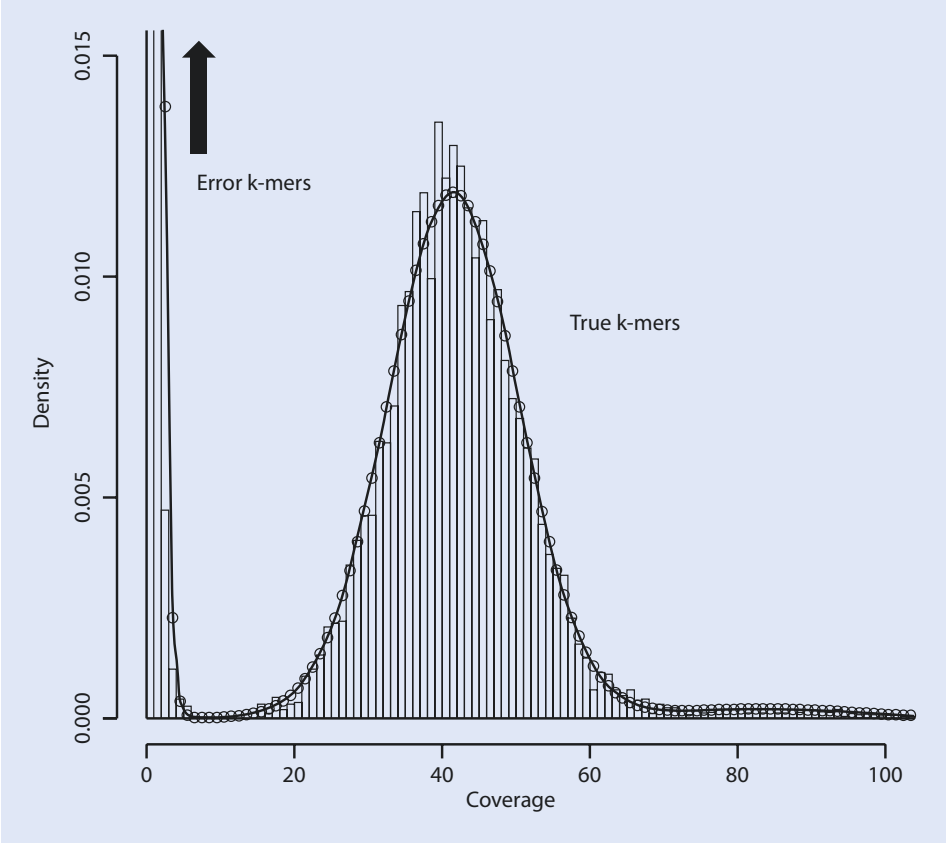


Fig. 5.11 Typical *k*-mer distribution for genomic data originating from one individual/species. The coverage of *k*-mers is plotted against its frequency (density). Low-coverage *k*-mers likely represent sequencing errors (Reprinted from Kelley et al. (2010))

this information. For this reason, several *k*-mer-based assemblers have an upper limit for *k*, as otherwise the computation becomes too memory intensive. On the other hand, larger *k*-mers are more informative and the numbers of nodes in the graph which can be traversed are decreased, making it easier finding paths through it. Consequently, the first step of any *k*-mer assembly is the careful choice for *k*. The number of *k*-mers generated per read can be estimated by a simple formula.

$$\text{Formula 5.4: } N_{\text{kmers}} = L_{\text{read}} - k + 1$$

In this formula, N_{kmers} refers to the calculated number of *k*-mers per read, with the read length defined by L_{read}. However, when counting *k*-mers for subsequent assemblies, only unique *k*-mers are stored together with the information how often they occurred. This can be done for a range of different *k*-values. Usually only uneven integers are used for *k*, as in case of even-numbered *k*-mers the occurrence of palindromic sequences can introduce further complexity into the graph, leading to shorter contigs. Plotting the coverage of *k*-mers against its frequency (Fig. 5.11) can be used to choose the optimal *k*-mer value for assembly. These plots are generated for many *k*-mers sizes, and the one leading to the highest number of distinct non-erroneous *k*-mers is chosen (Chikhi and Medvedev 2014).

Moreover, *k*-mer counting is used for error correction before assembly and can be used to detect repeated sequences, e.g. transposons (Marçais and Kingsford 2011).

K-mer frequencies can further be used to get a rough estimate of the genome size. A first step is to plot the *k*-mer coverage against the frequency as indicated in Fig. 5.11. Such outputs can be quite easily generated, for example, with the software JELLYFISH (Marçais and Kingsford 2011). The histograms can be used to distinguish between erroneous *k*-mers and potentially true *k*-mers (Fig. 5.11). The peak of the true *k*-mer distribution gives an estimate of the coverage of the genome (~40× in the example in Fig. 5.11). In the last step, the total number of true *k*-mers is divided by the coverage estimate to get an estimate of the total genome size. However, this number can be a huge underestimation if the genome bears a high percentage of repeat regions.

Several approaches can be used to choose the best *k*-value. An obvious way would be to generate assemblies for every possible *k* and choose the best after comparison of the assemblies. However, assemblies are usually very memory-intensive computations, and especially in case of large *k*'s, this way is not suitable. An intuitive (and heuristic) way to choose the best *k* is to generate abundance histograms (as shown in Fig. 5.11) for many values of *k* and to choose the value which generates the highest number non-erroneous *k*-mers (Chikhi and Medvedev 2014). A different approach is used by the IDBA assembler, which uses an iterative *k*-mer optimization, thereby de facto using different *k*-mer size for one assembly (Peng et al. 2010).

After choosing a value for *k* and fragmentation of sequence reads into *k*-mers, rare *k*-mers should be discarded. The logic is that in case of high-coverage genome sequencing, the abundance of *k*-mers should correlate with the expected coverage. Rare *k*-mers likely arose from sequencing errors. Likewise, overabundant *k*-mers are assumed to originate from high copy number regions of the genome (e.g. ribosomal cluster, transposons). This assumption might not work for transcriptome assemblies (▶ see Sect. 5.4). The resulting *k*-mer graph will be finally used to generate contigs by finding the Euler paths. Many software applications are available for generating *k*-mer assemblies (Table 5.2).

Compared with the OLC strategy, *k*-mer approaches bear some advantages for assembling huge numbers of short reads. Firstly, as no step for initial pairwise alignments is involved, *k*-mer strategies are much more time and memory efficient. Moreover, efficient algorithms are available for finding the Euler path within a de Bruijn graph. However, *k*-mer assemblies are usually less robust against sequencing errors, and the number of potential Euler paths is exponential to the number of repeats in the genome. Not surprisingly, especially the de novo assembly of eukaryotic genomes remains a challenge.

5.3 Comparing Assemblies

Different assembly strategies, programs or parameters can lead to vastly different sets of contigs. This raises the important question of how to judge different assemblies. A straightforward assessment of the accuracy of an assembly would be to compare it with an independent sequencing project of the same organism. This has been partly conducted for the assembly of the panda genome, which was the first assembled complex eukaryotic genome solely based on short-read data (Li et al. 2010). For validation, extra Sanger sequencing was performed to assess the quality of the short-read assembly. However, such an experimental setup is time intensive and costly and usually not practicable. When a reference genome is available, this can be also used for comparison, and BLAST searches

could be conducted to find assembled contigs. However, in many cases no reference of the same or even closely related species is available. Some metrics are available to describe assemblies without referring to a reference, and the most widely used is the N50 (Fig. 5.12). The N50 of an assembly is a weighted median and means that half of the assembled bases of all contigs are represented by contigs of this length or longer (Salzberg et al. 2012). To calculate the N50, after assembly all contigs are ordered according to their size, starting with longest one. In the next step, the overall contig size summing up all contigs is calculated. Lastly, starting with the longest contig, the next longest one is added until the sum of these combined contigs reaches 50% or more of the size of all contigs. The length of the contig added in this last step defines the N50 value for an assembly. For example, if the N50 is 15 mb, it means that contigs which contain 50% of the nucleotides of the complete assembly are at least 15 mb or longer. For genome assembly, it is usually desired to have higher N50 values. In an equal way, values for N75 or N80 could be calculated, containing the information for the respective percentages of the total contig size. The N50 is part of the output of many assemblers, but for comparison, it is important that the total contig size is calculated comparable, as often contigs with a size under a certain threshold (e.g. 500 bp) are not included in the calculation. A convenient way for multiple assembly comparison is to use the *QUAST* (quality assessment tool for genome assemblies) software, which could be either installed locally or used on an online server (Gurevich et al. 2013).

How well does the N50 describe the quality of a genome assembly? And which is the best assembler for my problem? To get an answer for these questions, the scientific community organized a competition for de novo short-read assemblers, the Assemblathon (► www.assemblathon.org). For the first competition, interested groups of software developers got sequence reads of a simulated eukaryotic genome, including contaminations and a typical sequence error profile. The genome had to be assembled de novo (even though information of a simulated reference genome could have been included), and the

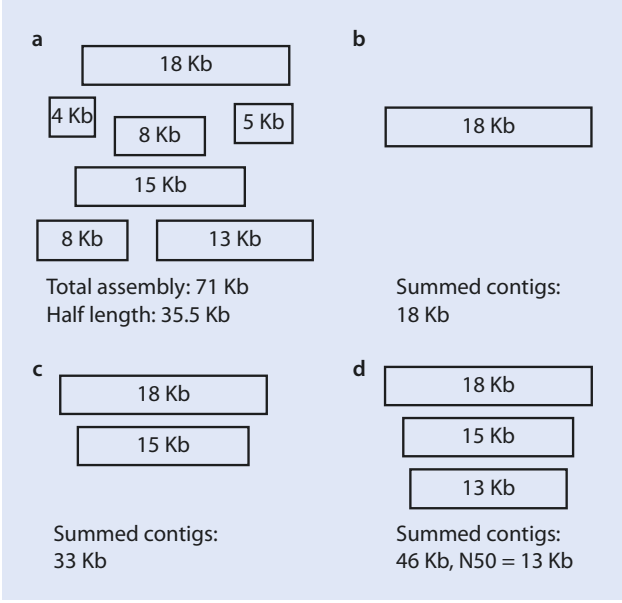


Fig. 5.12 Calculation of the N50 metric for sequence assemblies. **a** The assembly size if all contigs is 71 Kb. The cut-off value for the N50 is 35.5 Kb. **b** Sequences are summed step by step, starting with the longest. **c** Adding the second longest contig sums up to a total size of 33 Kb, still under the cut-off for the N50. **d** Adding the third longest contig result in a total sum of 46 Kb, which is over the cut-off (35.5 Kb). The length of this contig equals the N50 (13 Kb)

outcome was compared to the «true» simulated genome. Using this data, metrics relying on a reference genome could be compared with those working without any reference. One of the most important outcomes was that the N50 indeed seems to be a good way to describe assembly quality (Earl et al. 2011). Moreover, large differences could be shown between different assemblers. Consequently, the second iteration of the competition targeted this issue. This time, real genomic sequence data of three vertebrates was analysed. Overall, the tested genome assemblers produced useful assemblies, providing a significant representation of genes and overall genome structure. However, it was found that approaches which work well in assembling the genome of one species may not necessarily work well for another (Bradnam et al. 2013). The practical advice is to use different assemblers (Table 5.2) and to choose the best assembly based on available metrics like the N50 afterwards.

It is well known for phylogenetic tree reconstruction that many equally or similarly good solutions can be the outcome of the analysis. The same is true for assemblies, where due to uncertainty alternative solutions for contig building may be present. In a Bayesian framework, each assembly alternative could be given a probability, making it possible to evaluate different assemblies in a statistical framework (Howison et al. 2014; Howison et al. 2013). The development of software implementing these strategies is at the beginning, but ideas like this will open new future directions for choosing the best assembly.

5.4 De Novo Assembly of Genomes

Genomes can differ hugely in size and content of repetitive regions and so differ in their degree of difficulty to be assembled. Genomes (or chromosomes) also dramatically exceed the length of sequence reads generated by any sequencing technique actually used. Therefore, strategies like whole-genome shotgun sequencing are used, where the genome is fragmented in small pieces, and later these sequenced fragments are puzzled into the complete genome by assembly programs. The most widely used technique today is Illumina sequencing. However, as the recovered reads are usually not longer than 150 bp (HiSeq) or 250 bp (MiSeq), especially the assembly of complex eukaryotic genomes including many repeat regions remains challenging. Different strategies are applied to improve the initial assembly and to combine contigs into longer pieces.

5.4.1 Scaffolding

Contigs can be linked together into longer pieces via scaffolding. The information to bring contigs into an order usually comes from paired-end reads or mate pairs. As for assembly, graph theory can be used to solve scaffolding. In this case, the assembled contigs represent the nodes of the graph, and they are linked through read pairs as represented by edges (Hunt et al. 2014). Moreover, whereas the ordering information (and orientation) of contigs can be retained from such graphs, the expected length can be deduced from the approximate distance of read pairs known due to library preparation. This information can be included in the length of the edges connecting nodes in the graph. In scaffold sequences, this distance is given by N's inserted between two linked contigs. The first step for scaffolding is always the mapping of the actual sequence reads onto the contigs, to know the location of paired-end or mate pair reads. In the second step, this information is

comprised into a graph as described above. The available scaffolders use different strategies to resolve the graph into contigs, from exact mathematical solutions to heuristic approaches simplifying the graph into subgraphs. Some fast solutions are based on greedy algorithms where those contigs are joined first which are linked by the highest number of edges. Comparable to what has been found for assembly programs, scaffolding software can vary strongly in their performance and the required analysis time, based on the complexity of the analysed genome (Hunt et al. 2014). Several assembly programs also include scaffolding modules. Stand-alone scaffolding tools are, for example, SSPACE (Boetzer et al. 2011), SCARPA (Donmez and Brudno 2013) or OPERA (Gao et al. 2011). As in the case of assemblers, the performance of several scaffolders should be compared to choose the best suited for the task at hand.

5.4.2 Hybrid Assemblies

Assembling complex genomes solely with short reads is a difficult task, especially when no reference genome from a closely related organism is available. Not surprisingly, the difficulty of genome assembly is reduced with increasing sequence read length. Whereas Illumina sequencing is by far the most widely used technique, other high-throughput sequencing techniques generating considerably longer reads are available (e.g. PacBio and nanopore sequencing). The caveat with long reads of current sequencing techniques is their high error rate. For example, by using PacBio, an error rate of approximately 15% is expected, which can additionally vary dramatically across positions (Chin et al. 2013). Current assembly strategies are not equipped to directly deal with these high error rates (Sović et al. 2016). For example, when using OLC it is easily conceivable that perfect overlaps are hard to find. In contrast, *k*-mer-based assemblers need to find exact-matching k-mers between reads, which is an limiting factor when dealing with high error rates. To deal with these problems, hybrid methods have been developed taking advantage from the fact that Illumina short-read data, which has a much lower error rate than PacBio or nanopore long reads, is perfectly suited for error correction of long reads (Koren et al. 2012). Using hybrid assembly strategies, high-coverage (50x and higher coverage of the genome) short-read data (100–150 bp per read) is combined with low-coverage (10–20x) long reads (reads >5000 bp). The first step would be to assemble all short reads into contigs. These contigs are mapped onto the long reads for error correction. Error-corrected long reads can then be overlapped to get long contigs. In the last step, scaffolding as described above using short-read paired-end or mate-pair data might be performed. Alternatively, Illumina short reads can be assembled first, and then long reads are incorporated to bridge coverage gaps and resolve repeats, e.g. using the assembler ALLPATHS-LG (MacCallum et al. 2009).

5.5 De Novo Assembly of Transcriptomes and Metagenomes

Transcriptomes comprise the total RNA or mRNA expression data of isolated cells or tissue. Genes with a high expression will be represented by many sequence reads, whereas genes with low expression will yield few reads and non-expressed genes will obviously be missed totally (Wang et al. 2009). As such, a mixture of full-length and partial transcripts at various levels of abundance is expected. Consequently, huge differences in coverage of

different transcripts will be found in the final assembly. To further complicate things, different forms of alternatively spliced genes might be recovered (at least in eukaryotes). This is a challenge for all currently used assembly algorithms, and assembly quality decreases as transcriptome complexity increases (Chang et al. 2014).

As for genome assemblies, major strategies for transcriptome assemblies are reference based, de novo or a mixture of both. Reference-based assemblies consists of three steps: first, alignment/mapping of reads against a reference genome is conducted; second, overlapping reads of one locus are used to build a graph; and, third, the resulting graph is traversed to resolve isoforms (Martin and Wang 2011). De novo transcriptome assembly of short-read data is usually facilitated by de Bruijn graph-based assembly methods. It has been shown that lower *k*-mer sizes yield more sensitive assemblies, recovering also lowly expressed variants, whereas higher *k*-mer sizes result in more specific assemblies, leading to a more accurate assembly of abundant transcripts (Nagarajan and Pop 2013). It is important to keep in mind that the interpretation of *k*-mer abundance is less straight forward than for genome assemblies. Whereas in the latter case rare *k*-mers are probably originated from sequencing errors, rare *k*-mers may alternatively originate from lowly expressed transcripts in the case of transcriptome sequencing. A common idea of many transcriptome assemblers is to use different *k*-mer values. The resulting assemblies are merged afterwards, with redundant contigs being removed (Robertson et al. 2010). For many phylogenetic studies, researchers are only interested in the most reliable transcripts, which will be used for further analyses, and such approaches are well suited in this case. However, if isoforms and splice variants are targets of the study, more refined methods should be used. Two widely used transcriptome assemblers for this task are OASES (Schulz et al. 2012) and TRINITY (Grabherr et al. 2011). Both these methods firstly reconstruct contigs using *k*-mer-based de Bruijn graphs and subsequently explore transcript variants by connecting contigs or by retrieving contigs which represent different paths through the graph, but share the same starting and end point. Assessment of the quality of de novo transcriptome assembly is less established as for de novo genome assemblies. The N50 can be still used as a measure, but as transcriptome assemblies usually represent a set of thousands of medium-sized contigs (transcripts), the ultimate goal is not to get as few and large contigs as possible. Some approaches to assess transcriptome assemblies are in use which do not depend on a closely related reference. Completeness of transcripts can be described using reference alignments with homologous genes and check for start codons and – if applicable – presence of signal peptides. Further on, all organisms rely on a core set of housekeeping genes which are generally expected to be expressed in most cells. For eukaryotes, a set of such core proteins is well established and can be automatically detected using the BUSCO pipeline (Simão et al. 2015). Missing genes of this set could indicate improper assembly or lack of sequencing depth. A similar approach is implemented in the software DOGMA, which performs a fast and easy quality assessment of transcriptome assemblies based on conserved protein domains (Dohmen et al. 2016). Other proposed quality metrics can be derived from read mapping and include descriptions of accuracy, fragmentation, incompleteness, redundancy or chimerism (Fig. 5.13), e.g. implemented in TRANSRATE (Smith-Unna et al. 2016). If a reference genome is available, several metrics can be inferred based on its comparison, e.g. completeness or contiguity (Martin and Wang 2011). However, the biggest hope for the future is the availability of low-error long reads originating from single transcripts which could solve the assembly problem in this field completely.

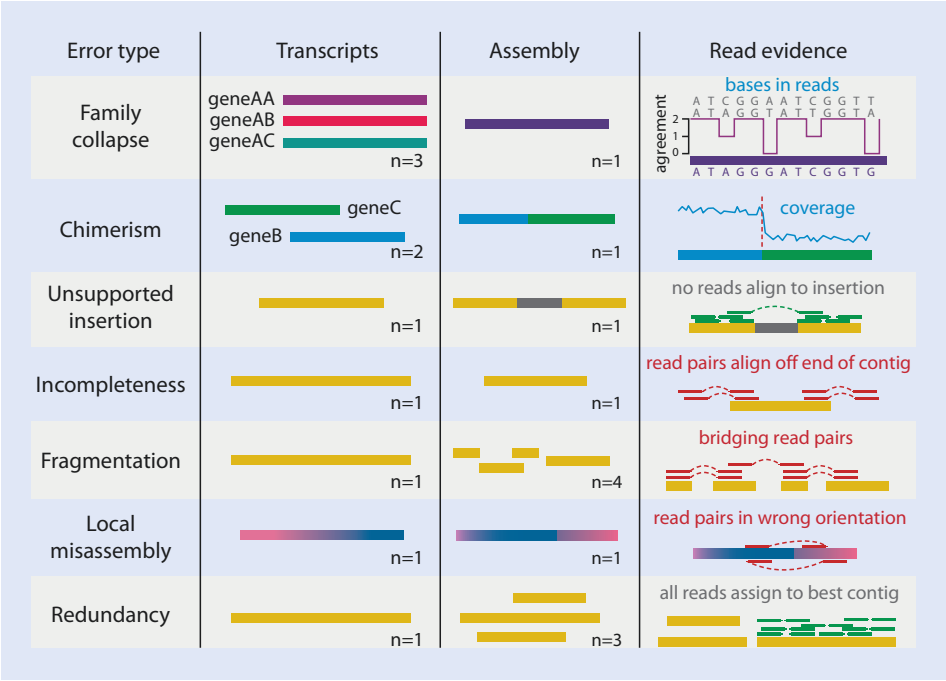


Fig. 5.13 Typical errors in transcriptome assemblies which can be assessed based on read evidence (Reprinted from (Smith-Unna et al. 2016))

The properties of metagenomic data are similar in some aspects to those of transcriptomes. Metagenomes comprise data originated from many different genomes from different individuals (Coughlan et al. 2015). As for transcriptomes, differences in abundance and therefore coverage through sequence reads are expected. Moreover, different organisms may harbour identical sequences, e.g. in the case of easily horizontally transmitted retrotransposons. Some assemblers are available which have been optimized for metagenome assembly, e.g. METAVELVET (Namiki et al. 2012) or METAIDBA (Peng et al. 2011). Additionally to the steps of «normal» *k*-mer-based genome assemblers, *k*-mer abundance information is used during assembly. It is assumed that sequences from different organisms differ in their coverage due to the individual abundance of the organisms in the sample. For example, highly abundant bacteria from a soil sample will be covered by more sequence reads than rare bacteria. This information can be used to refine the resolution of the *k-mer* graph. Nodes of these graphs which are included in the final contig should be joined by *k-mers* of similar coverage. A path through chimeric nodes, which represent the identical *k*-mers from different species, can be resolved according to this information (Fig. 5.14). The main problem of NGS-based metagenomic studies is to trace the origin of short reads back to different organisms (especially when there are no reference genomes available). In the future, techniques generating long-read data with lower error probabilities will be a key to enhance the accuracy of metagenomic assemblies.

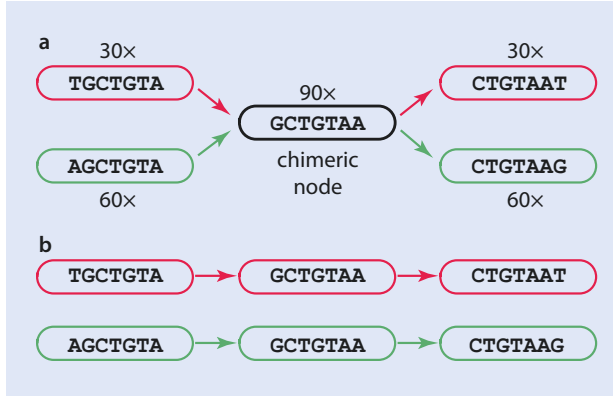


Fig. 5.14 Metagenomic assembly and chimeric nodes. **a** A *k*-mer assembly produces a chimer node, as indicated by the coverage information given by the number above the nodes. **b** Assembly of contigs can be resolved using the coverage information

5

References

Bankevich A, Nurk S, Antipov D, Gurevich AA, Dvorkin M, Kulikov AS, Lesin VM, Nikolenko SI, Pham S, Prjibelski AD, Pyshkin AV, Sirotkin AV, Vyahhi N, Tesler G, Alekseyev MA, Pevzner PA (2012) SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. J Comput Biol 19:455–477

Bankevich A, Pevzner PA (2016) TruSPAdes: barcode assembly of TruSeq synthetic long reads. Nat Methods 13:248–250

Batzoglou S, Jaffe DB, Stanley K, Butler J, Gnerre S, Mauceli E, Berger B, Mesirov JP, Lander ES (2002) ARACHNE: a whole-genome shotgun assembler. Genome Res 12:177–189

Boetzer M, Henkel CV, Jansen HJ, Butler D, Pirovano W (2011) Scaffolding pre-assembled contigs using SSPACE. Bioinformatics 27:578–579

Bolger AM, Lohse M, Usadel B (2014) Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics 30:2114–2120

Bradnam K, Fass J, Alexandrov A, Baranay P, Bechner M, Birol I, Boisvert S, Chapman J, Chapuis G, Chikhi R, Chitsaz H, Chou W-C, Corbeil J, Del Fabbro C, Docking T, Durbin R, Earl D, Emrich S, Fedotov P, Fonseca N, Ganapathy G, Gibbs R, Gnerre S, Godzaridis E, Goldstein S, Haimel M, Hall G, Haussler D, Hiatt J, Ho I, Howard J, Hunt M, Jackman S, Jaffe D, Jarvis E, Jiang H, Kazakov S, Kersey P, Kitzman J, Knight J, Koren S, Lam T-W, Lavenier D, Laviolette F, Li Y, Li Z, Liu B, Liu Y, Luo R, MacCallum I, MacManes M, Maillet N, Melnikov S, Naquin D, Ning Z, Otto T, Paten B, Paulo O, Phillippy A, Pina-Martins F, Place M, Przybylski D, Qin X, Qu C, Ribeiro F, Richards S, Rokhsar D, Ruby J, Scalabrin S, Schatz M, Schwartz D, Sergushichev A, Sharpe T, Shaw T, Shendure J, Shi Y, Simpson J, Song H, Tsarev F, Vezzi F, Vicedomini R, Vieira B, Wang J, Worley K, Yin S, Yiu S-M, Yuan J, Zhang G, Zhang H, Zhou S, Korf I (2013) Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. GigaScience 2:10

Chang Z, Wang Z, Li G (2014) The impacts of read length and transcriptome complexity for *De Novo* assembly: a simulation study. PLoS One 9:e94825

Chevreux B, Pfisterer T, Drescher B, Driesel AJ, Müller WEG, Wetter T, Suhai S (2004) Using the miraEST assembler for reliable and automated mRNA transcript assembly and SNP detection in sequenced ESTs. Genome Res 14:1147–1159

Chikhi R, Medvedev P (2014) Informed and automated k-mer size selection for genome assembly. Bioinformatics 30:31–37

Chin C-S, Alexander DH, Marks P, Klammer AA, Drake J, Heiner C, Clum A, Copeland A, Huddleston J, Eichler EE, Turner SW, Korlach J (2013) Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. Nat Methods 10:563–569

Cock PJA, Fields CJ, Goto N, Heuer ML, Rice PM (2010) The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. Nucleic Acids Res 38:1767–1771

Coughlan L, Cotter P, Hill C, Alvarez-Ordóñez A (2015) Biotechnological applications of functional metagenomics in the food and pharmaceutical industries. Front Microbiol 6:672

David M, Dursi LJ, Yao D, Boutros PC, Simpson JT (2017) Nanocall: an open source basecaller for Oxford nanopore sequencing data. Bioinformatics 33:49–55

Dohmen E, Kremer LPM, Bornberg-Bauer E, Kemena C (2016) DOGMA: domain-based transcriptome and proteome quality assessment. Bioinformatics 32:2577–2581

Donmez N, Brudno M (2013) SCARPA: scaffolding reads with practical algorithms. Bioinformatics 29:428–434

Earl D, Bradnam K, St John J, Darling A, Lin D, Fass J, Hung On Ken Y, Buffalo V, Zerbino DR, Diekhans M, Ngan N, Ariyaratne PN, Sung W-K, Ning Z, Haimel M, Simpson JT, Fonseca NA, Birol I, Docking TR, Ho IY, Rokhsar DS, Chikhi R, Lavenier D, Chapuis G, Naquin D, Maillet N, Schatz MC, Kelley DR, Phillippy AM, Koren S, Yang S-P, Wu W, Chou W-C, Srivastava A, Shaw TI, Ruby JG, Skewes-Cox P, Betegon M, Dimon MT, Solovyev V, Seledtsov I, Kosarev P, Vorobyev D, Ramirez-Gonzalez R, Leggett R, MacLean D, Xia F, Luo R, Li Z, Xie Y, Liu B, Gnerre S, MacCallum I, Przybylski D, Ribeiro FJ, Yin S, Sharpe T, Hall G, Kersey PJ, Durbin R, Jackman SD, Chapman JA, Huang X, DeRisi JL, Caccamo M, Li Y, Jaffe DB, Green RE, Haussler D, Korf I, Paten B (2011) Assemblathon 1: a competitive assessment of de novo short read assembly methods. Genome Res 21:2224–2241

Ewing B, Green P (1998) Base-calling of automated sequencer traces using *Phred*. II. Error probabilities. Genome Res 8:186–194

Gao S, Sung W-K, Nagarajan N (2011) Opera: reconstructing optimal genomic scaffolds with high-throughput paired-end sequences. J Comput Biol 18:1681–1691

Goodwin S, Gurtowski J, Ethe-Sayers S, Deshpande P, Schatz MC, McCombie WR (2015) Oxford nanopore sequencing, hybrid error correction, and de novo assembly of a eukaryotic genome. Genome Res 25:1750–1756

Grabherr MG, Haas BJ, Yassour M, Levin JZ, Thompson DA, Amit I, Adiconis X, Fan L, Raychowdhury R, Zeng Q, Chen Z, Mauceli E, Hacohen N, Gnirke A, Rhind N, di Palma F, Birren BW, Nusbaum C, Lindblad-Toh K, Friedman N, Regev A (2011) Full-length transcriptome assembly from RNA-seq data without a reference genome. Nat Biotechnol 29:644–U130

Gurevich A, Saveliev V, Vyahhi N, Tesler G (2013) QUAST: quality assessment tool for genome assemblies. Bioinformatics 29:1072–1075

Hackl T, Hedrich R, Schultz J, Förster F (2014) *proovread*: large-scale high-accuracy PacBio correction through iterative short read consensus. Bioinformatics 30:3004–3011

Hernandez D, François P, Farinelli L, Østerås M, Schrenzel J (2008) *De novo* bacterial genome sequencing: millions of very short reads assembled on a desktop computer. Genome Res 18:802–809

Howison M, Zapata F, Dunn CW (2013) Toward a statistically explicit understanding of de novo sequence assembly. Bioinformatics 29:2959–2963

Howison M, Zapata F, Edwards EJ, Dunn CW (2014) Bayesian genome assembly and assessment by Markov chain Monte Carlo sampling. PLoS One 9:e99497

Huang X, Madan A (1999) CAP3: a DNA sequence assembly program. Genome Res 9:868–877

Hunt M, Newbold C, Berriman M, Otto T (2014) A comprehensive evaluation of assembly scaffolding tools. Genome Biol 15:R42

Kelley D, Schatz M, Salzberg S (2010) Quake: quality-aware detection and correction of sequencing errors. Genome Biol 11:R116

Kircher M, Heyn P, Kelso J (2011) Addressing challenges in the production and analysis of Illumina sequencing data. BMC Genomics 12:382

Kircher M, Stenzel U, Kelso J (2009) Improved base calling for the Illumina genome analyzer using machine learning strategies. Genome Biol 10:R83

Koren S, Schatz M, Walenz B, Martin J, Howard J, Ganapathy G, Wang Z, Rasko D, McCombie W, Jarvis E, Phillippy A (2012) Hybrid error correction and de novo assembly of single-molecule sequencing reads. Nat Biotechnol 30:693–700

Koren S, Walenz BP, Berlin K, Miller JR, Phillippy AM (2016) Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. bioRxiv. doi.org/10.1101/071282.

Laver T, Harrison J, O'Neill PA, Moore K, Farbos A, Paszkiewicz K, Studholme DJ (2015) Assessing the performance of the Oxford nanopore technologies MinION. Biomol Detect Quantif 3:1–8

Li D, Luo R, Liu C-M, Leung C-M, Ting H-F, Sadakane K, Yamashita H, Lam T-W (2016) MEGAHIT v1.0: a fast and scalable metagenome assembler driven by advanced methodologies and community practices. Methods 102:3–11

Li H (2016) Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. Bioinformatics 32:2103–2110

Li R, Fan W, Tian G, Zhu H, He L, Cai J, Huang Q, Cai Q, Li B, Bai Y, Zhang Z, Zhang Y, Wang W, Li J, Wei F, Li H, Jian M, Nielsen R, Li D, Gu W, Yang Z, Xuan Z, Ryder O, Leung F-C, Zhou Y, Cao J, Sun X, Fu Y (2010) The sequence and de novo assembly of the giant panda genome. Nature 463:311–317

Lin Y, Yuan J, Kolmogorov M, Shen MW, Pevzner PA (2016) Assembly of long error-prone reads using de Bruijn graphs. Proc Natl Acad Sci USA 113:E8396-E8405 (In press)

Luo R, Liu B, Xie Y, Li Z, Huang W, Yuan J, He G, Chen Y, Pan Q, Liu Y (2012) SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. GigaScience 1:18

MacCallum I, Przybylski D, Gnerre S, Burton J, Shlyakhter I, Gnirke A, Malek J, McKernan K, Ranade S, Shea TP, Williams L, Young S, Nusbaum C, Jaffe DB (2009) ALLPATHS 2: small genomes assembled accurately and with high continuity from short paired reads. Genome Biol 10:R103

Marçais G, Kingsford C (2011) A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. Bioinformatics 27:764–770

Martin JA, Wang Z (2011) Next-generation transcriptome assembly. Nat Rev Genet 12:671–682

Miller J, Koren S, Sutton G (2010) Assembly algorithms for next-generation sequencing data. Genomics 95:315

Myers EW, Sutton GG, Delcher AL, Dew IM, Fasulo DP, Flanigan MJ, Kravitz SA, Mobarry CM, Reinert KHJ, Remington KA, Anson EL, Bolanos RA, Chou H-H, Jordan CM, Halpern AL, Lonardi S, Beasley EM, Brandon RC, Chen L, Dunn PJ, Lai Z, Liang Y, Nusskern DR, Zhan M, Zhang Q, Zheng X, Rubin GM, Adams MD, Venter JC (2000) A whole-genome assembly of *Drosophila*. Science 287:2196–2204

Nagarajan N, Pop M (2013) Sequence assembly demystified. Nat Rev Genet 14:157–167

Nakamura K, Oshima T, Morimoto T, Ikeda S, Yoshikawa H, Shiwa Y, Ishikawa S, Linak MC, Hirai A, Takahashi H, Altaf-Ul-Amin M, Ogasawara N, Kanaya S (2011) Sequence-specific error profile of Illumina sequencers. Nucleic Acids Res 39:e90

Namiki T, Hachiya T, Tanaka H, Sakakibara Y (2012) MetaVelvet: an extension of velvet assembler to de novo metagenome assembly from short sequence reads. Nucleic Acids Res 40:e155

Peng Y, Leung HCM, Yiu S-M, Chin FYL (2010) IDBA—a practical iterative de Bruijn graph de novo assembler. In: Berger B (ed) Research in computational molecular biology, vol 6044. Springer, Berlin, pp 426–440

Peng Y, Leung HCM, Yiu S-M, Lv M-J, Zhu X-G, Chin FYL (2013) IDBA-Tran: a more robust de novo de Bruijn graph assembler for transcriptomes with uneven expression levels. Bioinformatics 29:326–334

Peng Y, Leung HCM, Yiu SM, Chin FYL (2011) Meta-IDBA: a de Novo assembler for metagenomic data. Bioinformatics 27:i94–i101

Pevzner P, Tang H, Waterman M (2001) An Eulerian path approach to DNA fragment assembly. Proc Natl Acad Sci 98:9748–9753

Renaud G, Kircher M, Stenzel U, Kelso J (2013) freeIbis: an efficient basecaller with calibrated quality scores for Illumina sequencers. Bioinformatics 29:1208–1209

Robertson G, Schein J, Chiu R, Corbett R, Field M, Jackman SD, Mungall K, Lee S, Okada HM, Qian JQ, Griffith M, Raymond A, Thiessen N, Cezard T, Butterfield YS, Newsome R, Chan SK, She R, Varhol R, Kamoh B, Prabhu A-L, Tam A, Zhao Y, Moore RA, Hirst M, Marra MA, Jones SJM, Hoodless PA, Birol I (2010) *De novo* assembly and analysis of RNA-seq data. Nat Methods 7:909–912

Salmela L, Rivals E (2014) LoRDEC: accurate and efficient long read error correction. Bioinformatics 30:3506–3514

Salzberg S, Phillippy A, Zimin A, Puiu D, Magoc T, Koren S, Treangen T, Schatz M, Delcher A, Roberts M, Marcais G, Pop M, Yorke J (2012) GAGE: a critical evaluation of genome assemblies and assembly algorithms. Genome Res 22:557–567

Schatz MC, Delcher AL, Salzberg SL (2010) Assembly of large genomes using second-generation sequencing. Genome Res 20:1165–1173

Schulz MH, Zerbino DR, Vingron M, Birney E (2012) *Oases*: robust de novo RNA-seq assembly across the dynamic range of expression levels. Bioinformatics 28:1086–1092

Simão FA, Waterhouse RM, Ioannidis P, Kriventseva EV, Zdobnov EM (2015) BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. Bioinformatics 31:3210–3212

Simpson J, Wong K, Jackman S, Schein J, Jones S, Birol I (2009) ABySS: a parallel assembler for short read sequence data. Genome Res 19:1117–1123

Smeds L, Kunstner A (2011) CONDETRI - A content dependent read trimmer for Illumina data. PLoS One 6:e26314

References

Smith-Unna R, Boursnell C, Patro R, Hibberd J, Kelly S (2016) TransRate: reference free quality assessment of de novo transcriptome assemblies. Genome Res 26:1134–1144

Sović I, Križanović K, Skala K, Šikić M (2016) Evaluation of hybrid and non-hybrid methods for de novo assembly of nanopore reads. Bioinformatics 32:2582–2589

Wang Z, Gerstein M, Snyder M (2009) RNA-seq: a revolutionary tool for transcriptomics. Nat Rev Genet 10:57–63

Zerbino D, Birney E (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res 18:821–829