BRIEFINGS IN BIOINFORMATICS. VOL 14. NO 2. 213–224
Advance Access published on 23 December 2011
doi:10.1093/bib/bbr074

Hawkeye and AMOS: visualizing and assessing the quality of genome assemblies

Michael C. Schatz, Adam M. Phillippy*, Daniel D. Sommer, Arthur L. Delcher, Daniela Puiu, Giuseppe Narzisi, Steven L. Salzberg and Mihai Pop*

Submitted: 5th August 2011; Received (in revised form): 10th October 2011

Abstract

Since its launch in 2004, the open-source AMOS project has released several innovative DNA sequence analysis applications including: Hawkeye, a visual analytics tool for inspecting the structure of genome assemblies; the Assembly Forensics and FRCurve pipelines for systematically evaluating the quality of a genome assembly; and AMOScmp, the first comparative genome assembler. These applications have been used to assemble and analyze dozens of genomes ranging in complexity from simple microbial species through mammalian genomes. Recent efforts have been focused on enhancing support for new data characteristics brought on by second- and now third-generation sequencing. This review describes the major components of AMOS in light of these challenges, with an emphasis on methods for assessing assembly quality and the visual analytics capabilities of Hawkeye. These interactive graphical aspects are essential for navigating and understanding the complexities of a genome assembly, from the overall genome structure down to individual bases. Hawkeye and AMOS are available open source at http://amos.sourceforge.net.

Keywords: *DNA Sequencing; genome assembly; assembly forensics; visual analytics*

INTRODUCTION AND CONTEXT

Despite dramatic advances in sequencing technologies, we are still far from automating the decoding of a complete sequence from a DNA sample [1]. Instead, sequencing machines generate large numbers of short DNA fragments that need to be algorithmically assembled together into a reconstruction of a genome. The development of genome assembly algorithms, started in the late 1980s, was critical to the success of genomics as it provided a way for researchers to explore the genomes of organisms despite the limitations of sequencing technologies.

Over the years, numerous genome assemblers have been developed, among which we highlight

Corresponding author. Michael C. Schatz. One Bungtown Road, Cold Spring Harbor NY, 11724. Tel: (516) 367 5218 Fax: (516) 367 8380; E-mail: mschatz@cshl.edu

*These authors contributed equally to this work.

Michael C. Schatz, PhD is an Assistant Professor of Quantitative Biology at Cold Spring Harbor Laboratory.

Adam M. Phillippy, PhD is a Principal Investigator of Genomics and Bioinformatics at the National Biodefense Analysis and Countermeasures Center.

Daniel D. Sommer is a bioinformatics engineer in the Center for Bioinformatics and Computational Biology at the University of Maryland.

Arthur L. Delcher, PhD is an Instructor at the Institute for Genome Sciences at the University of Maryland School of Medicine and Professor Emeritus of Computer Science at Loyola University Maryland.

Daniela Puiu is a senior software engineer in the Institute of Genetic Medicine at Johns Hopkins University School of Medicine.

Giuseppe Narzisi, PhD, is a Research Scientist at the NYU Bioinformatics Lab of Courant Institute of Mathematical Sciences after receiving a PhD in CS from NYU in May 2011.

Steven L. Salzberg, PhD, is a Professor of Medicine and Biostatistics in the Institute of Genetic Medicine at Johns Hopkins University. From 2005 to 2011 he was Director of the Center for Bioinformatics and Computational Biology at the University of Maryland.

Mihai Pop, PhD, is an Associate Professor and Interim Director of the Center for Bioinformatics and Computational Biology at the University of Maryland.

just a few of the most memorable: phrap—one of the most widely used assemblers of the first generation sequencing era; Celera Assembler [2, 3]—the software originally developed at Celera Genomics and used to assemble the human genome through a whole-genome shotgun sequencing approach (as opposed to the BAC-by-BAC approach employed by the Human Genome Consortium); Velvet [4]—one of the first among a number of assemblers developed specifically for second generation sequencing data; and ALLPATHS-LG [5]—perhaps the most effective genome assembler for second generation sequencing data today. These, and the many other assemblers used in the community, have contributed to the reconstruction of tens of thousands of viruses and bacteria, as well as hundreds of eukaryotes, including the genomes of many mammals (human [6], mouse [7], cow [8], panda [9], etc.), fishes (zebrafish [10], fugu [11], etc.) and plants (Arabidopsis [12], rice [13], papaya [14], etc.) to name a few.

These successes hide an important fact—genome assembly is a difficult computational problem, and no genome assembler can fully reconstruct a large genome from short fragments without errors. Genome assembly is also computationally demanding: it falls in a class of computational problems called NP-hard [15], or problems that can only be exactly resolved by exhaustively trying all possible combinations. This is simply not tractable for large genomes with billions of reads, so all genome assemblers employ heuristics to accelerate the computation based on certain assumptions about the underlying data. However, these heuristics may also introduce errors by applying incorrect simplifications.

After a genome assembler has produced a result, it is natural to ask if the assembly is correct and if one assembly is better than another. Answering these questions is complex, especially because they require consideration across a wide range of scales spanning from individual bases, to gene sequences and other localized regions, and finally to the large-scale chromosome structure. It would be preferable to measure quality across all scales simultaneously by a single metric, but no such metric exists. Instead, the most commonly used metric for reporting assembly quality is the N50 contig or scaffold size, which is a weighted median size such that half of the genome has been into assembled into contigs or scaffolds larger than the N50 value. This metric measures the overall connectivity of the assembly, but says nothing about the correctness. Scaffolds or contigs can be trivially inflated to any size if the assembly algorithm is allowed to include very low quality or conflicting information, and size measurements do not address if a given gene or a given chromosome has been assembled correctly. Indeed it may be impossible to simultaneously measure quality across the entire range of scales given the complexity of the data [16].

These challenges motivated us to develop the interactive tool, Hawkeye, for assessing genome assemblies [17]. One of the key strengths of Hawkeye is that it supports analysis at any scale: users are initially presented with visualizations and reports of the overall assembly characteristics, from which they can zoom and filter to inspect scaffolds, contigs, reads, or even individual bases. The views within Hawkeye are integrated to include all relevant information into the display, and whenever possible the views are interactive, allowing on-the-fly clustering or selective filtering of particular data types.

In addition, Hawkeye leverages the AMOS assembly forensics pipeline to systematically identify erroneous regions of an assembly based on statistical inconsistencies in the data, such as clusters of mate-pairs whose separation fall outside the expected library distribution [16]. Hawkeye can highlight these regions to guide users to the most likely mis-assembled regions, and aid users in fixing any problems. Moreover, we have developed a new visual representation of assembly quality called the Feature-Response Curve (FRCurve) [18] that allows users to visually compare and rank multiple assemblies of the same data, again leveraging the inconsistencies identified by our assembly forensics pipeline to assess relative quality in addition to connectivity.

The advantages of a visual analytics approach for these analyses are numerous. The data have very high dimension and an interactive display enables users to zoom selectively into different portions of the assembly to display details that would be lost at a fixed resolution. Hawkeye also employs semantic zooming so that the initial display provides an abstract overview, which progressively becomes more detailed and more concrete as users zoom in. The visual display is well suited for the highly advanced pattern recognition capabilities of the human mind, and empowers users to identify trends and unanticipated relationships that might otherwise be overlooked. For example, certain mis-assemblies are easily

identified by eye as clusters of mis-oriented mates. Similarly, technical artifacts of PCR duplicates, which create clusters of reads with nearly exactly the same coordinates, can be readily identified. Supporting this type of novel analysis is critical in light of evolving sequencing technologies and diverse data types, so that users can visually identify trends before a formal analysis pipeline has even been created to search for them.

METHODS

AMOS modular design

The development of AMOS was motivated by the fact that scientists had been increasingly using genome assembly algorithms for 'non-standard' or experimental applications. Some examples are the assembly of viral genomes, metagenomic data, or data from RNA-seq or CHIP-seq experiments. Each of these possesses qualities that violate some of the assumptions built in to standard genome assemblers. Most existing genome assemblers were not designed with flexibility in mind and are difficult to adapt to these new uses. In contrast, AMOS is designed to be flexible and provides well-defined APIs between assembly modules, allowing for new components to be added or replaced to meet the challenges of a new experiment or assay (Figure 1). As such, many of the AMOS pipelines extensively reuse the components developed under a different context. For example, the *de novo* assembler Minimus and the comparative assembler AMOScmp use several of the same components, except that Minimus has its own read overlapping module, while AMOScmp infers overlaps by mapping reads to a reference genome. The components of these pipelines interact by iteratively refining data stored within a central database called a 'bank', which stores the primary assembly objects (reads, contigs, scaffolds, etc.) and associated information.

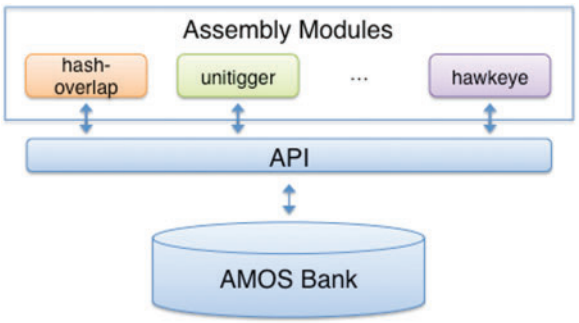

Figure 1: AMOS modular design. Individual assembly modules, such as for computing overlaps between reads (hash-overlap), for refining the overlaps between reads into contigs (unitigger), and the GUI Hawkeye, interact using well defined APIs through a lightweight central database called an AMOS bank.

The AMOS package includes a number of assembly tools, including:

- Minimus [19]—an assembler designed for simple assembly tasks such as the assembly of viral genomes or individual genes. Minimus was independently enhanced by users of our software into a new assembly package called Minimo that was also incorporated into the AMOS package.
- AMOScmp [20]—a comparative assembler that can use the reference of a closely related genome as a template during assembly.
- Bambus [21]—a standalone scaffolding package, recently extended to include features necessary for assembling metagenomic datasets.
- Hawkeye [17], AMOSvalidate [16] and the FRCurve [18]—assembly visualization and assembly validation packages that can detect misassemblies using a variety of internal consistency checks, and described in more detail below.

Conversion utilities are a critical, but often underappreciated component of a bioinformatics package that allows the software to interact with the diverse file types and data representations developed by the community. AMOS includes a number of conversion utilities that allow it to process data from a variety of input sources (e.g. fasta, fastq, TraceArchive xml files, etc.) and to output the data in commonly used assembly formats (fasta, ace, agp, sam/bam, etc.) These input/output conversion utilities also allow users to take the full output information from another genome assembler and load it into the central AMOS bank to either improve the assembly through AMOS-specific utilities (e.g. Bambus could be used to scaffold the data produced by another assembler), or to validate its quality as explained below.

The AMOS source code, manual, mailing lists and bug tracker are available at the project website hosted at SourceForge (http://amos.sourceforge.net). The source code is primarily written in C++, with selected components written in Perl and Python. As such, AMOS runs in almost any Unix environment including Linux, Mac OS X and Windows (under Cygwin). Several components of AMOS require the Boost C++ libraries

(http://boost.org), AMOScmp and AMOSvalidate rely on the Nucmer aligner [22], and Hawkeye requires the graphics library Qt 4.x (http://qt.nokia.com). Unlike some other graphical assembly viewers, AMOS and Hawkeye do not require a web or database server and can be run as standalone applications on a desktop.

Assessing assembly quality: Hawkeye, forensics and the FRCurve

The initial development of AMOS was spurred by the need for more flexible assembly pipelines, but the visualization and validation components of AMOS came from a need to understand what was happening within assemblies. The term 'assembly forensics' was coined after tracking down subtle mis-assembles contained in the draft genomes of *Bacillus anthracis* that were sequenced as part of the Amerithrax investigation [23]. In cases where the sequence output by an assembler did not agree with expectation, a deeper view than simple contig metrics was required. To this end, Hawkeye was developed to give developers and technicians the tools needed to decipher how a genome was reconstructed. To cope with the often daunting complexity of genome assemblies, the guiding design principle behind Hawkeye is to simplify through visualization so that assembly problems are made obvious.

Hawkeye presents all aspects of a genome assembly to users through interactive displays. This begins with the initial assembly 'Launch Pad', which summarizes contig sizes and overall assembly contiguity. From here, users may explore the assembly at either the scaffold or contig level. At the scaffold level the emphasis is on large-scale assembly quality, especially depth of coverage, repeat content and read pairing constraints. Unusually high or low coverage can often indicate a repeat copy number error, while mis-paired reads can be further evidence of copy number problems or indicate other structural errors. Paired reads that are in an unexpected orientation or separation are flagged by the display (either via color or placement) to draw the users eye to groups of violating reads. Groups of pairs that trend either too short or two long are automatically flagged using the CE statistic, which compares the distribution of insert sizes spanning a given position in the assembly to the global library distribution [24].

At the contig level, a detailed read tiling is displayed, and the focus is shifted to single base-call correctness. The strength of this display is the identification of low quality sequence, trimming problems and correlated based-call errors. A single disagreeing base is almost always the result of an instrument-based sequencing error, but when multiple reads support an alternate variant it can signal a non-random error, such as a heterozygous base, or more significantly a collapsed repeat or the merging of distinct sequences from a non-clonal sample (e.g. polyploidy, metagenomic).

The key functionality of Hawkeye lies not in one specific component, but rather in its ability to display multiple aspects of the data simultaneously. A mis-assembly often manifests as multiple anomalies in the data. For example, a collapsed repeat is one of the most common mis-assemblies. Figure 2 shows the multiple lines of evidence that can be investigated to identify this common problem. Among the typical indications are increased depth of coverage, paired-end compressions, correlated base variants and chimeric read alignments at the boundaries. All Hawkeye displays are dynamically synchronized, so that the location displayed in the current contig view is highlighted correspondingly in the scaffold view.

Visualizations are essential for understanding individual mis-assemblies, but for large assemblies it is not feasible to inspect the entire reconstruction manually. In order to direct users attention to the most likely regions of mis-assembly, the AMOS package includes an automated assembly forensics pipeline, AMOSvalidate. This tool identifies anomalous 'features' of an assembly that contradict the constraints of the input sequence data. A perfect assembler would make no such mistakes, but since computational complexity forces heuristic approaches, subtle mistakes are commonly made. Five important aspects of assembly quality considered by AMOSvalidate are based on the constraints that (i) the sequences of overlapping reads must agree, (ii) the distance between paired reads must be consistent with their size-selected parent fragments, (iii) paired reads must be oriented in their expected manner, (iv) read placement throughout the assembly must be consistent with the random shearing process and (v) all reads provided to the assembler must be consistent with the resulting assembly. Individual violation of these constraints is often due to random laboratory error, but multiple violations at the same location are deemed mis-assembly signatures. Both the individual mis-assembly *features*, and the more serious mis-assembly

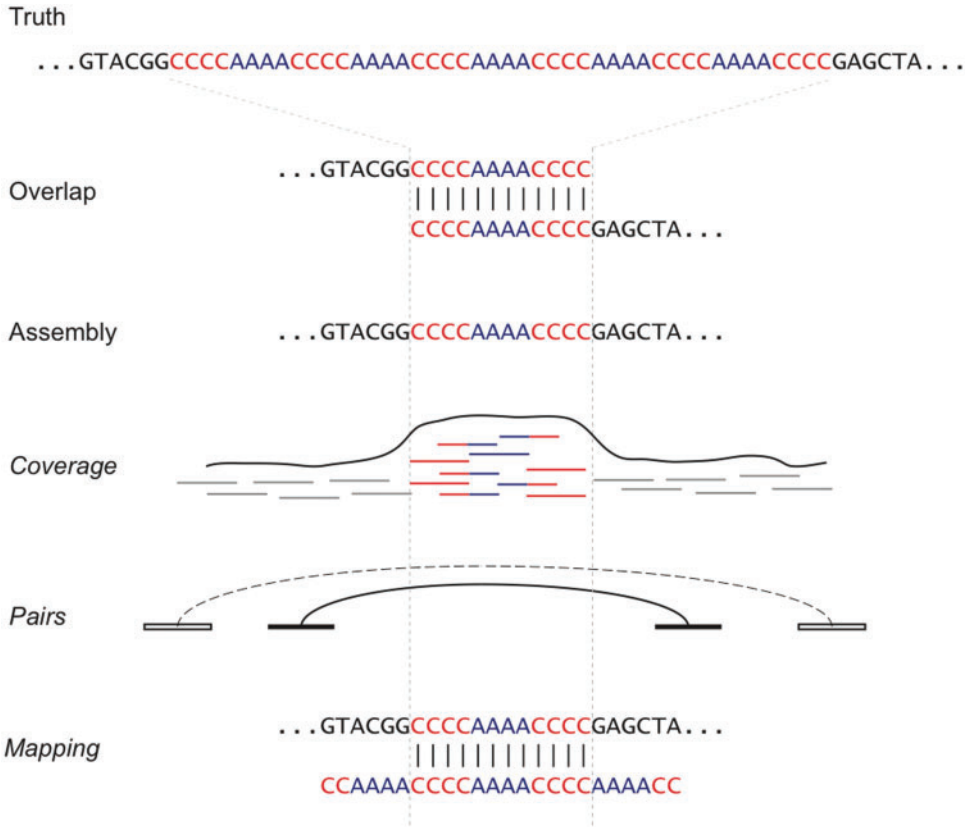

Figure 2: Assembly artifacts resulting from a tandem repeat collapse style mis-assembly. In this small illustration, the motif **{**CCCCAAAA**}** repeats in tandem in the original genome sequence. No single read spans the repetitive sequence and so the assembler has created a minimal reconstruction from two overlapping reads. This results in three signatures of mis-assembly: increased coverage, compressed pairs and partial read mappings. Increased coverage is caused by short reads internal to a repeat copy 'piling up' within the remaining copies. This can be observed both in the read tiling, or with the K* *k*-mer statistic introduced by AMOSvalidate. Compressed pairs (black) are anchored in neighboring unique sequence and are compressed relative to their expected separation (dotted). Finally, reads discarded by the assembler for apparent chimerism can sometimes be remapped to the assembly to identify the exact boundary of the collapsed repeat. The tails of these reads will fail to align, or will wrap around to the other side of the repeat.

signatures, can be loaded into Hawkeye to prioritize the most likely regions of mis-assembly for manual inspection (Figures 3–5).

The above methods detail the validation of a single assembly, but these techniques can also be very effective at comparing the performance of multiple assemblers. The recently introduced FRCurve is a novel assembly metric developed at NYU and contributed to AMOS to overcome many of the challenges of comparing multiple assemblies [18]. By visually displaying the frequency of mis-assembly features identified by AMOSvalidate, the FRCurve is able to evaluate and compare multiple assemblies in greater detail than typical contiguity metrics, like N50. Inspired by the receiver operating characteristic (ROC) curve, the FRCurve captures the *trade-off* between contiguity (genome coverage) and quality (number of features/errors) of the assembled contigs. The FRCurve is one of very few self-contained assembly metrics that captures both contiguity and quality. Using AMOSvalidate, each contig is assigned a number of features that correspond to constraint violations in the reconstruction. Given such a set of features, the FRCurve analyzes the response (quality) of the assembler output as a function of the maximum number of possible errors (features) in the contigs. Specifically, all contigs are sorted by size and, from longest to shortest, and their features are tallied until this sum exceeds a chosen feature threshold. For this set of contigs, the corresponding genome coverage is computed, leading to a single point of the FRCurve (Figure 6). Multiple assemblies

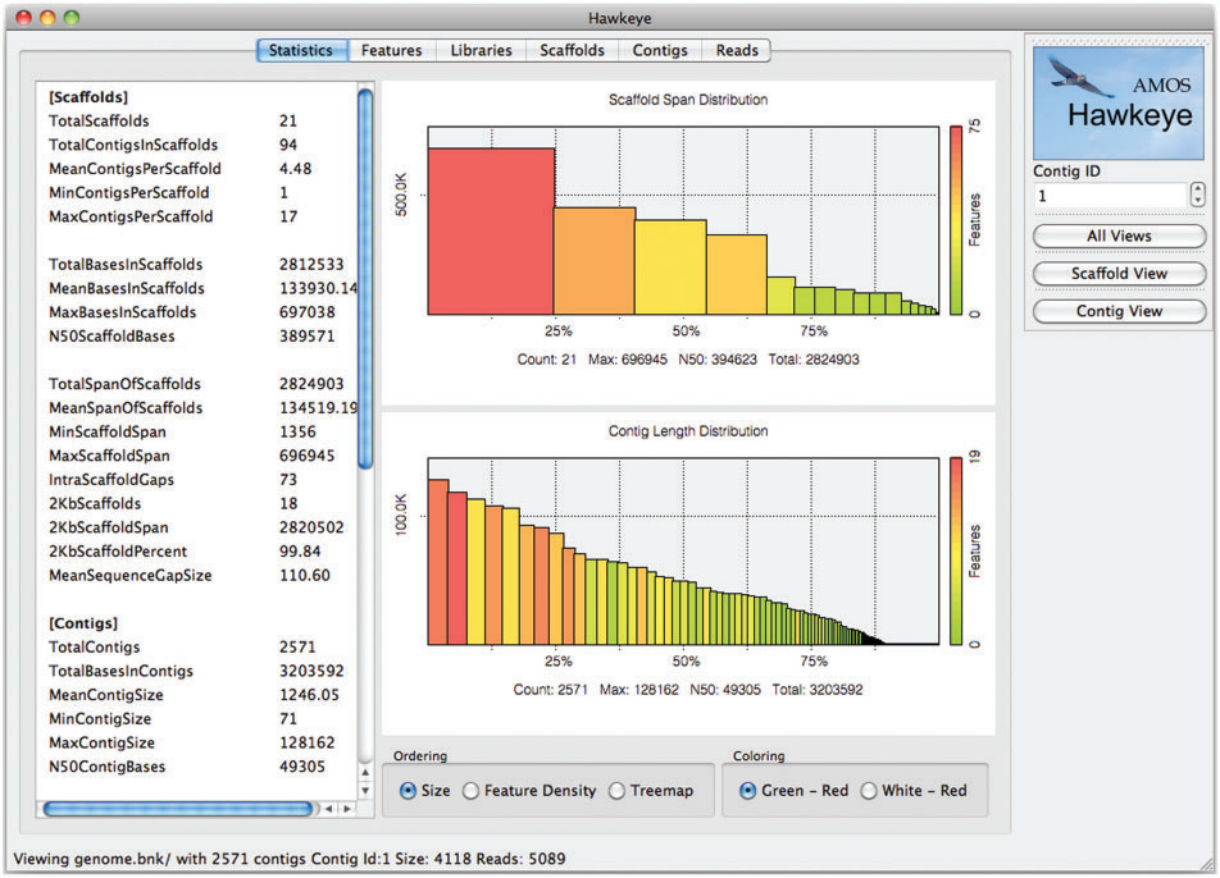

Figure 3: The Hawkeye LaunchPad displays the overall summary of the assembly using an interactive barchart of the scaffold (top) and contig (bottom) sizes and quality. The left panel displays common statistics such as counts and sizes of those sequences. Additional tabs display statistics and histograms for the features, libraries, scaffolds, contigs and reads.

can then be visually and intuitively compared by inspecting the relative position of each assembly's curve drawn in the same plot.

Recent innovations: analysis of second and third generation sequences

A recent effort spanning all aspects of AMOS has been to improve support for second generation and now third generation sequencing. In particular, second generation sequencing projects generally use much higher coverage and much shorter reads. For instance, 30–100× coverage of 50- to 100-bp reads for second-generation sequencing is commonly used compared to 8–10× of 500- to 1000-bp reads for first generation Sanger sequencing. Consequently, the total volume of sequence data used for an assembly of a given genome has grown dramatically, as has the shear number of reads in second-generation sequencing project compared to first generation sequencing project. This dramatic rise has necessitated optimizations of the AMOS data structures to minimize per-read overhead and provide better indexing to rapidly query for particular reads given a larger collection.

AMOS also includes new conversion utilities to support the new formats used with short reads, such as importing/exporting reads from/to fastq format, and importing/exporting contigs from/to SAM format, etc. Furthermore, the parameters and thresholds used within the assembly pipelines have been revised to accommodate much deeper coverage, short reads and a different error model. The AMOS components are currently under reevaluation again to optimize support for third generation sequencing, including long high error reads. Because of its versatility, AMOS is well suited for these emerging technologies, and the new hybrid assembler AHA developed by Pacific Biosciences is largely based upon the AMOS package (http://www.pacbiodevnet.com/).

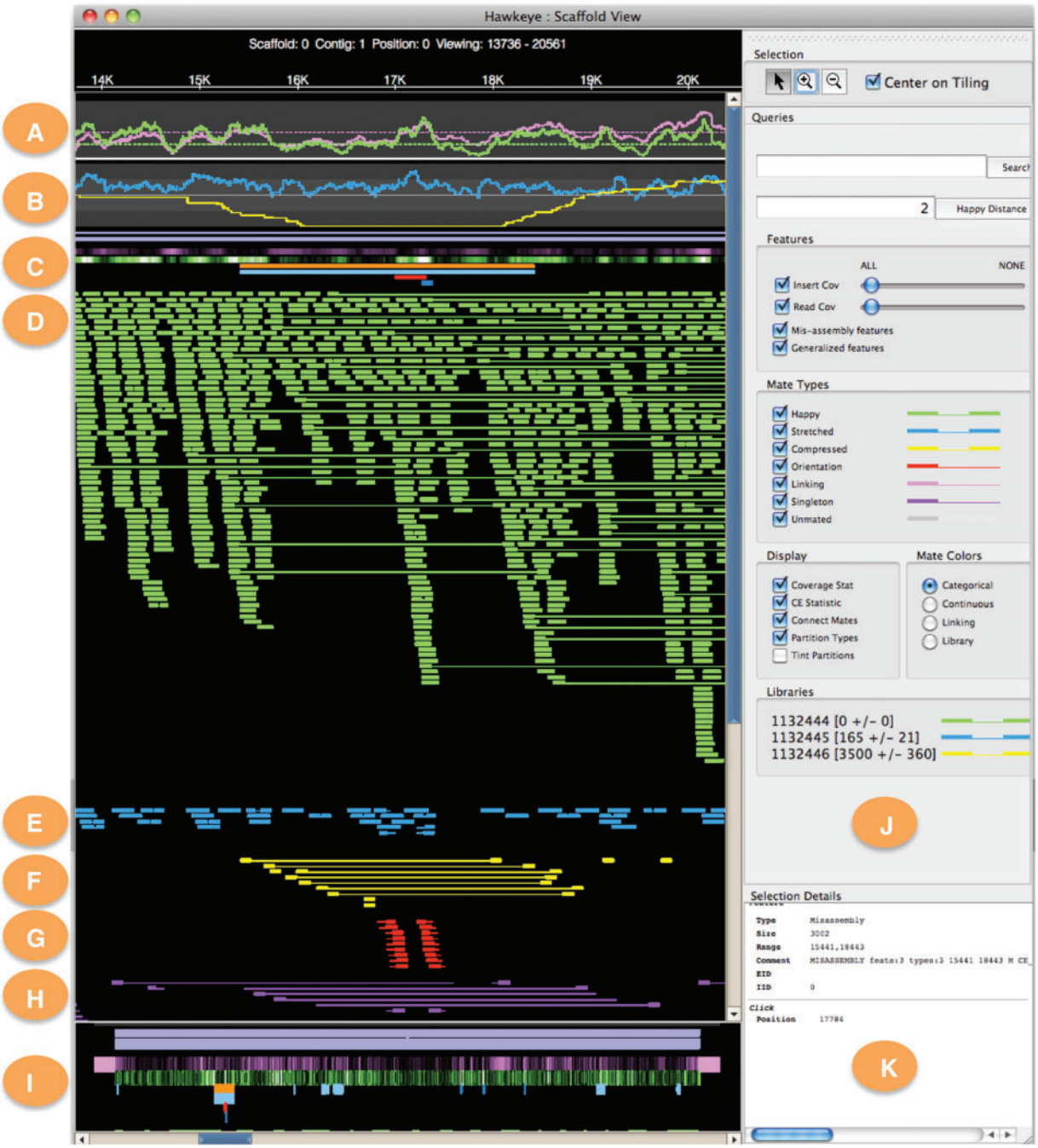

Figure 4: The Hawkeye Scaffold View displays the placement of individual reads and mates within the contigs (D–H), along with coverage statistics (A), mate statistics (B) and other features (C).

Hawkeye has also been substantially improved since its publication in 2007 in light of these challenges. As an AMOS component, Hawkeye immediately benefits from the general improvements to the AMOS infrastructure, such as reducing the per-read overhead and memory requirements. The visual display of Hawkeye has also been substantially revised and rewritten to leverage the most up-to-date version of the graphics library Qt used for drawing menus, windows, buttons and the assembly displays. In addition to generally simplifying installation and polishing the overall display, updating to Qt 4 has also made Hawkeye more responsive, more stable, and expanded operating system support to fully support Mac OS X. Finally, several of the views have been enhanced to better display deeper

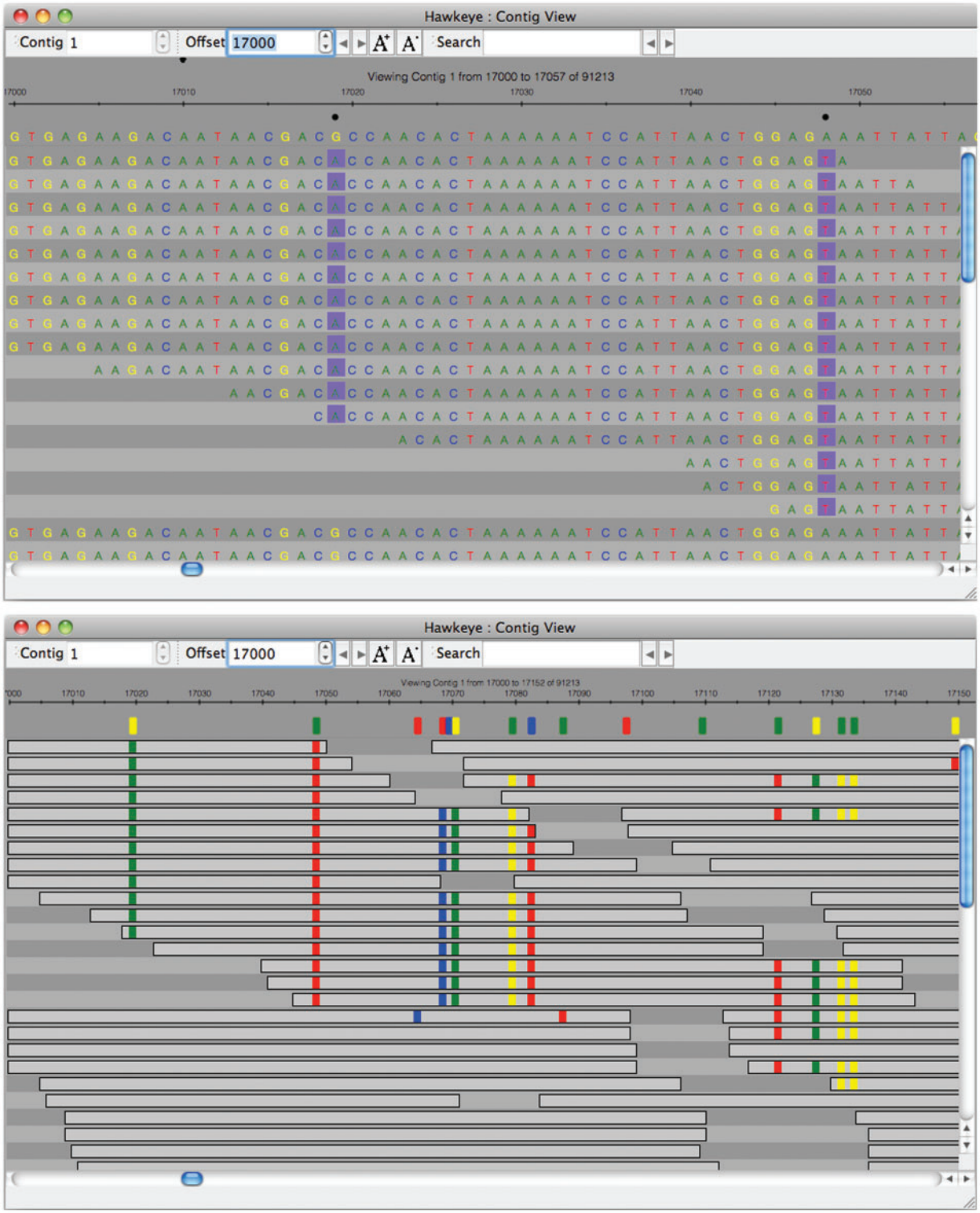

Figure 5: The Hawkeye Contig View (top) displays the individual bases in the assembly. Once the font size becomes smaller than would be visible (bottom), Hawkeye switches to an abstract view using colored rectangles to indicate bases that disagree with the consensus.

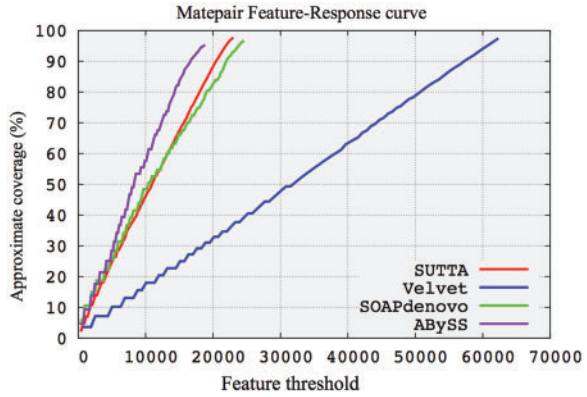

Figure 6: Feature-Response Curve comparison using the mate-pair feature type for *Escherichia coli*. The data set consists of 20.8 million paired-end 36-bp Illumina reads from a 200-bp insert *E. coli* strain K12 MG1655.

coverage of short reads. For example, the Contig View now displays multiple reads on the same horizontal row to make better use of the screen dimensions for short reads.

TECHNOLOGIES AND FEATURES

Visual assembly analytics with Hawkeye

The initial Hawkeye display is called the Assembly LaunchPad, and provides an overview to the assembly results and the data characteristics. As an example, Figure 3 shows the Hawkeye LaunchPad for an assembly of the 2.9-Mb *Staphylococcus aureus* genome available on the AMOS website (http://sourceforge.net/projects/amos/files/sample_data/). The genome was assembled using the pre-assembly error correction program Quake [25] and the Celera Assembler [3] from ~25× coverage of 100-bp reads sequenced at the Broad Institute using an Illumina Genome Analyzer II (SRA study SRP001086). These mated reads were sequenced using a combination of ~20× coverage of a 165-bp fragment library, and ~5× coverage of a 3.5-kb jumping library.

The left side of the LaunchPad is a table showing common summary statistics such as the number of contigs and scaffolds, and the N50 and total sizes. The middle panel is an interactive barchart of the scaffolds and contigs. Each scaffold (top) or contig (bottom) in the assembly is represented by a bar whose size is proportional to the size of the sequence. The height of each bar is an absolute size measured in base-pairs, and the width is the relative fraction of the genome size. The color of each bar is determined by the number of mis-assembly features discovered by AMOSvalidate. Red bars indicate the sequence has a high number of features, and are likely to be mis-assembled, while green bars indicate few or no features found. On the right are buttons to select particular contigs and display the scaffold and contig views described below. Contigs and scaffolds of interest can also be selected by double-clicking on a bar in the barchart.

Other tabs of the LaunchPad display summaries and enable queries of the features, libraries, scaffolds, contigs and reads in the assembly. These more detailed panels allow users to list particular assembly features, compute insert size histograms, examine sequence GC content distributions, and a variety of other high-level quality checks and summaries. Of particular relevance is the feature browser, which allows users to systematically explore every mis-assembly feature discovered in the genome.

The next most detailed display is the Scaffold View (Figure 4), which shows an interactive display of the placement of the reads within the contigs and scaffolds. At the top of the display is a line plot (A) of the read (green) and fragment (purple) coverage for a ~7-kb region of the assembly, showing a peak near 17 kb in the contig. The next track (B) is a line plot of the CE-statistic computed along the scaffold. Here the blue curve represents the fragment library and remains near zero indicating no significant compression or expansion, while the yellow curve represents the jumping library dips strongly negative indicating a potential compression. Track C shows the placement of the contig in the scaffold, a color gradient of the insert coverage (purple), read coverage (green) and the mis-assembly features. The orange bar indicates AMOSvalidate found a 'suspicious region' (orange) as indicated by a CE-statistic compression (light blue), along with the high read coverage (bright blue), and an unusually high density of heterogeneous SNPs (red). Tracks D–H show the placement of the individual reads, as thick rectangles connected by thin lines to connect mated reads. The green inserts (D) are separated and oriented as expected, while blue (E) are further than expected, yellow (F) are closer than expected, red (G) are mis-oriented, and purple (H) are reads that have mates, but those mates are not in the contig at the expected location. The bottom panel (I) gives an overview of the entire scaffold, showing this is the only predicted mis-assembly location in the scaffold.

The right panel contains several filters and search boxes (J–K). These controls allow users to selectively display different classes of inserts, or to search for specific reads or features. All of the elements in the scaffold view are interactive, and clicking on a read or feature displays the details for that item in the details panel (K).

The final Hawkeye display is the Contig view (Figure 5). When this view is first displayed, the sequences of the individual reads are displayed (top). However, as users decrease the font size, Hawkeye uses semantic zooming to switch to an abstract display of the reads and uses colored rectangles to highlight where the read bases disagree with the consensus. This display is synchronized with the Scaffold view and shows a high density of SNPs present in this region of the genome, including 9 bases with multiple reads that agree with each other but disagree with the consensus. Clicking on the consensus sorts the reads by the bases at that position and is very effective as an on-the-fly clustering of the reads. Users can also optionally display other read details such as color-coded quality values or trimmed bases, or search for particular sequence motifs or read names.

Hawkeye integrates all available evidence to highlight the mis-assembly presented above: the read coverage is abnormally high; clusters of jumping mates are compressed, mis-oriented and absent; and finally the individual reads disagree with each other. In this case we can confirm the assembly by aligning the contigs to the reference genome, and indeed the predicted mis-assembly is confirmed. However, a user can draw this same conclusion without a reference given the abundance of evidence presented by Hawkeye. After identifying this mis-assembly, the user may wish to adjust assembly parameters, or use the assembly correction tools available within AMOS to fix the region.

Multiple assembly comparison with the FRCurve

Complementary to the single assembly analysis capabilities of Hawkeye, the FRCurve excels at visualizing and comparing the overall quality of multiple alternate assemblies of the same data. These assemblies may be created using different assemblers, or even using the same assembler with different parameters. For example, a critical parameter of many short read assemblers is the *k*-mer word size used for constructing the de Bruijn graph, and it is now a common practice to systematically perform multiple assemblies using multiple *k*-mer sizes. The simplest way to evaluate multiple assemblies, and select the best, is to compare the sizes of the contigs and the scaffolds, such as by comparing the maximum size or the N50 size. However, this can be a poor metric, as it does not measure the accuracy of the sequences.

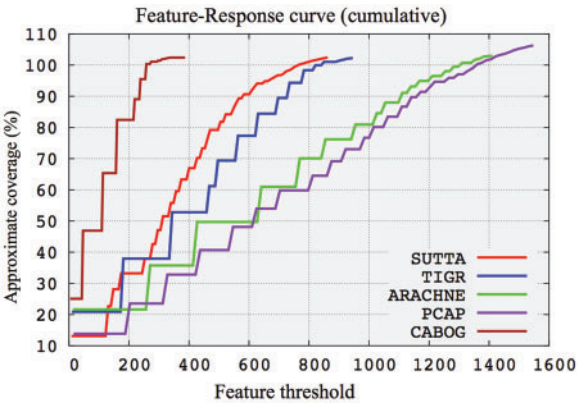

Figure 7: Feature-Response Curve comparison using the mate-pair feature type for *Staphylococcus epidermidis*. The data set consists of 60 761 paired-end Sanger reads of 900 bp average length for a total coverage of ~20×.

The FRCurve fills this void by simultaneously measuring the connectivity and the quality of multiple assemblies at once, using the mis-assembly features identified by AMOSvalidate to measure quality in the absence of a reference sequence. The visualizations generated by the FRCurve are highly intuitive: faster growing curves correspond to better assemblies since a larger fraction of a genome can be covered with contigs containing fewer features/errors. For example, Figures 6 and 7 show two examples where the FRCurve is used to compare the assembly quality of some well-known assemblers both for short next-generation Illumina data and long Sanger data respectively (part of these results have been previously published [18]). In the top figure one can immediately see that with this short read data set and for the parameters used, ABySS [26] produced the best assembly overall followed by SUTTA [27], SOAPdenovo [28] and Velvet. In the second figure, one can similarly see with this long read data set, CABOG [3] produced the best assembly overall, followed by SUTTA [27], the TIGR Assembler [29], Arachne [30, 31] and PCAP [32].

DISCUSSION AND FUTURE DIRECTIONS

The recent dramatic decreases in sequencing costs have driven a corresponding increase in the number and types of genomes sequenced. Many sequencing projects, such as the 1000 Genomes projects or the Cancer Genome Atlas, are resequencing many human genomes to identify common genetic variations or variations associated with various diseases. However, there are also a considerable number of projects underway for which no reference genome is available and *de novo* assembly is the only option available. Notably, the Genome 10 K project (http://www.genome10k.org/) aims to sequence 10 000 vertebrate genomes, the i5K initiative (http://arthropodgenomes.org/wiki/i5K) aims to sequence 5000 insects and other arthropods, and the Human Microbiome Project (https://commonfund.nih.gov/hmp/) aims to assemble and characterize the thousands of microbial species that live on and within the human body. Assembling these genomes will require substantial computational effort and easy-to-use software packages for assessing their quality. Hawkeye coupled with our assembly forensics and FRCurve pipelines provides exactly these features.

Interactive and visual analytics tools for these analyses provide many advantages, especially the ability to zoom and filter from high-level overviews down to individual bases. Furthermore, semantic zooming progressively provides more details of the assembly until all relevant details are displayed to allow inspection and interpretation that cannot be captured by a single metric. This enables users to literally see unusual or interesting events without necessarily knowing exactly what they are looking for. For example, users that see an unusual pileup of reads with the exact same coordinates will realize that they need to filter PCR duplicates. In addition to detecting errors, the *absence* of mis-assembly features can lend credence to important regions of the genome under study. This level of intuitive analysis will hopefully advance genome assembly from being a specialized task that only a few experts can understand to something that is accessible to anyone.

Future work remains to enhance Hawkeye and all of AMOS to support emerging data types and assays. Already AMOS forms the basis for the new hybrid second and third generation assembler created by Pacific Biosciences, and the AMOS data structures support the concept of multiple subreads sampled from a parent fragment, such as 'strobed' reads. However, fully supporting them will require generalizing our storage methods to support more than two subreads, and reengineering of the components that analyze paired-end data. Similar enhancements are needed to, for example, represent transcript assemblies derived from RNA-seq reads overlaid with the reference genome, or develop on-the-fly quantification of binding sites from ChIP-seq reads. Improved support for even larger datasets and even larger genomes continues as well, with plans in place for more compact representations of individual reads and better indexes of the assembly so that less data needs to be stored in memory without sacrificing performance. Finally, we aim to provide more analysis and statistical functionality into the visual displays so that users can cluster or filter in ad-hoc ways in support of emerging assays in which no established analysis methods exists.

Key Points

- Commonly used assembly metrics such the N50 size do not address the quality of an assembly, especially when assessing if a particular gene or chromosome has been assembled correctly.
- Many mis-assemblies can be identified by assembly forensics, which statistically and systematically evaluates the consistency of the reads and paired-end constraints.
- Visual analytics is well suited for assessing assembly quality by empowering users to interactively examine the evidence across resolutions, and for identifying trends before formal analysis pipeline are created.

Acknowledgements

We would like to thank all of the AMOS contributors, especially Sergey Koren, Todd Treangen, Florent Angly, David Kelly, James White, Niranjan Nagarajan, Ted Gibbons, Atif Memon, Dung Ta, Ben Shneiderman and Bud Mishra and our enthusiastic users. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security. The Department of Homeland Security does not endorse any products or commercial services mentioned in this publication.

FUNDING

This work was supported, in part, by NIH grants R01-LM006845 (S.L.S.), R01-HG006677 (S.L.S.) and R01-HG004885 (M.P.) and by NSF grant IIS-0812111(M.P.). This publication was developed in part under Agreement No. HSHQDC-07-C-00020 awarded by the US Department of Homeland Security for the management and

operation of the National Biodefense Analysis and Countermeasures Center (NBACC), a Federally Funded Research and Development Center.

References

1. Schatz MC, Delcher AL, Salzberg SL. Assembly of large genomes using second-generation sequencing. *Genome Res* 2010;**20**:1165–73.
2. Myers EW, Sutton GG, Delcher AL, *et al*. A whole-genome assembly of Drosophila. *Science* 2000;**287**(5461): 2196–204.
3. Miller JR, Delcher AL, Koren S, *et al*. Aggressive assembly of pyrosequencing reads with mates. *Bioinformatics* 2008; **24**(24):2818–24.
4. Zerbino DR, Birney E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.* 2008;**18**(5):821–9.
5. Gnerre S, Maccallum I, Przybylski D, *et al*. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proc Natl Acad Sci USA* 2011;**108**(4): 1513–8.
6. Venter JC, Adams MD, Myers EW, *et al*. The sequence of the human genome. *Science* 2001;**291**(5507):1304–51.
7. Waterston RH, Lindblad-Toh K, Birney E, *et al*. Initial sequencing and comparative analysis of the mouse genome. *Nature* 2002;**420**(6915):520–62.
8. Zimin AV, Delcher AL, Florea L, *et al*. A whole-genome assembly of the domestic cow, Bos taurus. *Genome Biol* 2009;**10**(4):R42.
9. Li R, Fan W, Tian G, *et al*. The sequence and de novo assembly of the giant panda genome. *Nature* 2010; **463**(7279):311–7.
10. Jekosch K. The zebrafish genome project: sequence analysis and annotation. *Methods Cell Biol* 2004;**77**:225–39.
11. Aparicio S, Chapman J, Stupka E, *et al*. Whole-genome shotgun assembly and analysis of the genome of Fugu rubripes. *Science* 2002;**297**(5585):1301–10.
12. The Arabidopsis Initiative. Analysis of the genome sequence of the flowering plant Arabidopsis thaliana. *Nature* 2000; **408**(6814):796–815.
13. Yu J, Hu S, Wang J, *et al*. A draft sequence of the rice genome (Oryza sativa L. ssp. indica). *Science* 2002; **296**(5565):79–92.
14. Ming R, Hou S, Feng Y, *et al*. The draft genome of the transgenic tropical fruit tree papaya (Carica papaya Linnaeus). *Nature* 2008;**452**(7190):991–6.
15. Nagarajan N, Pop M. Parametric complexity of sequence assembly: theory and applications to next generation sequencing. *J Comput Biol* 2009;**16**(7):897–908.
16. Phillippy AM, Schatz MC, Pop M. Genome assembly forensics: finding the elusive mis-assembly. *Genome Biol* 2008;**9**(3):R55.
17. Schatz MC, Phillippy AM, Shneiderman B, *et al*. Hawkeye: an interactive visual analytics tool for genome assemblies. *Genome Biol* 2007;**8**(3):R34.
18. Narzisi G, Mishra B. Comparing de novo genome assembly: the long and short of it. *PloS One* 2011;**6**(4):e19175.
19. Sommer DD, Delcher AL, Salzberg SL, *et al*. Minimus: a fast, lightweight genome assembler. *BMC Bioinformatics* 2007;**8**:64.
20. Pop M, Phillippy A, Delcher AL, *et al*. Comparative genome assembly. *Brief Bioinformatics* 2004;**5**(3):237–48.
21. Pop M, Kosack DS, Salzberg SL. Hierarchical scaffolding with Bambus. *Genome Res* 2004;**14**(1):149–59.
22. Kurtz S, Phillippy A, Delcher AL, *et al*. Versatile and open software for comparing large genomes. *Genome Biol* 2004; **5**(2):R12.
23. Rasko DA, Worsham PL, Abshire TG, *et al*. Bacillus anthracis comparative genome analysis in support of the Amerithrax investigation. *Proc Natl Acad Sci USA* 2011; **108**(12):5027–32.
24. Zimin AV, Smith DR, Sutton G, *et al*. Assembly reconciliation. *Bioinformatics* 2008;**24**(1):42–5.
25. Kelley DR, Schatz MC, Salzberg SL. Quake: quality-aware detection and correction of sequencing errors. *Genome Biol* 2010;**11**(11):R116.
26. Simpson JT, Wong K, Jackman SD, *et al*. ABySS: a parallel assembler for short read sequence data. *Genome Res* 2009;**19**: 1117–23.
27. Narzisi G, Mishra B. Scoring-and-unfolding trimmed tree assembler: concepts, constructs and comparisons. *Bioinformatics* 2011;**27**(2):153–60.
28. Li R, Zhu H, Ruan J, *et al*. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Res* 2009;**20**:265–72.
29. Sutton GG, White O, Adams MD, *et al*. TIGR Assembler: a new tool for assembling large shotgun sequencing projects. *Genome Sci Technol* 1995;**1**(1):9–19.
30. Batzoglou S, Jaffe DB, Stanley K, *et al*. ARACHNE: a whole-genome shotgun assembler. *Genome Res* 2002; **12**(1):177–89.
31. Jaffe DB, Butler J, Gnerre S, *et al*. Whole-genome sequence assembly for mammalian genomes: Arachne 2. *Genome Res* 2003;**13**(1):91–6.
32. Huang X, Wang J, Aluru S, *et al*. PCAP: a whole-genome assembly program. *Genome Res* 2003;**13**(9):2164–70.