BMC Bioinformatics

SOFTWARE   **Open Access**

Automated ensemble assembly and validation of microbial genomes

Sergey Koren[1*], Todd J Treangen[1], Christopher M Hill[2], Mihai Pop[2] and Adam M Phillippy[1]

Abstract

Background: The continued democratization of DNA sequencing has sparked a new wave of development of genome assembly and assembly validation methods. As individual research labs, rather than centralized centers, begin to sequence the majority of new genomes, it is important to establish best practices for genome assembly. However, recent evaluations such as GAGE and the Assemblathon have concluded that there is no single best approach to genome assembly. Instead, it is preferable to generate multiple assemblies and validate them to determine which is most useful for the desired analysis; this is a labor-intensive process that is often impossible or unfeasible.

Results: To encourage best practices supported by the community, we present iMetAMOS, an automated *ensemble assembly pipeline*; iMetAMOS encapsulates the process of running, validating, and selecting a single assembly from multiple assemblies. iMetAMOS packages several leading open-source tools into a single binary that automates parameter selection and execution of multiple assemblers, scores the resulting assemblies based on multiple validation metrics, and annotates the assemblies for genes and contaminants. We demonstrate the utility of the ensemble process on 225 previously unassembled *Mycobacterium tuberculosis* genomes as well as a *Rhodobacter sphaeroides* benchmark dataset. On these real data, iMetAMOS reliably produces validated assemblies and identifies potential contamination without user intervention. In addition, intelligent parameter selection produces assemblies of *R. sphaeroides* comparable to or exceeding the quality of those from the GAGE-B evaluation, affecting the relative ranking of some assemblers.

Conclusions: Ensemble assembly with iMetAMOS provides users with multiple, validated assemblies for each genome. Although computationally limited to small or mid-sized genomes, this approach is the most effective and reproducible means for generating high-quality assemblies and enables users to select an assembly best tailored to their specific needs.

Background

Genome assembly reconstructs a genome from many shorter sequencing reads as faithfully as possible [1-3]. Since reasonable formulations of the problem are NP-hard [2,4], practical implementations often return an approximate solution that contains errors. Recent assembly evaluations like GAGE and the Assemblathon [5-8] have highlighted the chaotic nature of genome assembly, in which assembler performance varies widely across datasets and small parameter changes can have drastic effects. In GAGE-B [8] for example, each dataset required

a different k-mer parameter, the best assembler was not consistent across datasets, and the continuity difference between best and second best was often two-fold.

Although genome assembly is a complex problem, validating assemblies is more straightforward. The quality of a genome assembly can be confirmed by verifying that the layout of reads is consistent with the sequencing process used to generate the data [9]. Multiple tools have been recently developed for validating genome assemblies both with and without the use of a reference genome [10-15]. Thus, given the chaotic nature of assemblers and the relative ease of validation, it is recommended to generate multiple assemblies and use validation to determine the most appropriate one. This is akin to a "hypothesis generation" view of assembly [16], which can be most

* Correspondence: korens@nbacc.net
[1]National Biodefense Analysis and Countermeasures Center, 110 Thomas Johnson Drive, Frederick, MD 21702, USA
Full list of author information is available at the end of the article

easily implemented as an ensemble of independent methods. Unfortunately, running multiple assemblers is a time consuming, non-trivial task requiring substantial installation, learning, and maintenance costs.

There exist a limited set of tools that integrate automated parameter selection and validation into the assembly process. The A5 pipeline [17,18] automates the microbial assembly process, but is limited to a single assembler and includes limited validation. CG-Pipeline [19] is targeted to 454 sequencing. VelvetOptimizer [20] automates a parameter sweep of k-mer sizes for the Velvet assembler [21], but uses contig N50 size as the optimization metric, which is not always representative of assembly quality [7,22]. More recently, a number of assembly methods have been developed that incorporate assembly likelihood estimates into the primary assembly algorithm [23-25]. However, none of these tools robustly automate the execution of multiple assembly methods and validation metrics to achieve the best possible assembly. Here we present iMetAMOS, which automates the process of ensemble assembly and validation.
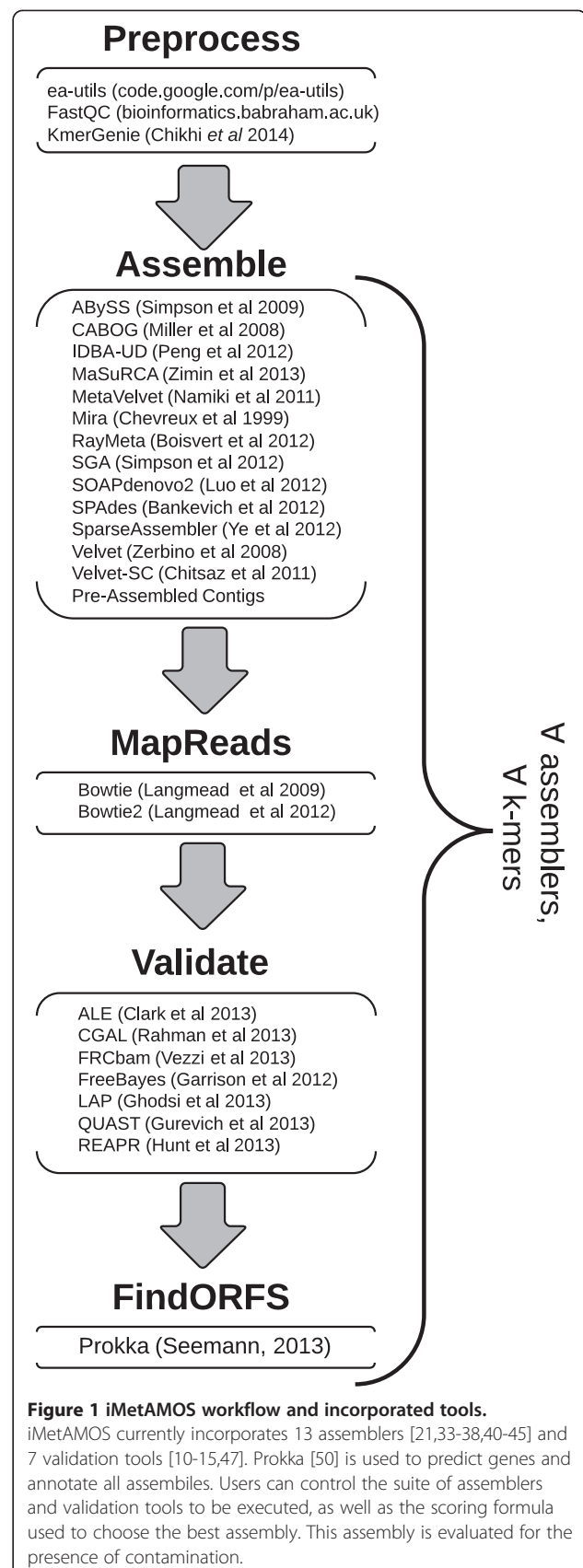
Implementation

Whereas MetAMOS [26] was developed for metagenomic assembly, iMetAMOS is an isolate-focused extension that encapsulates the current best practices for microbial genome assembly using Illumina [27], 454 [28], Ion [29], or PacBio [30] sequencing data. Building on the conclusions of GAGE and the Assemblathon, iMetAMOS runs multiple, independent tools to generate and validate assemblies. Uniquely, iMetAMOS automates the entire ensemble assembly process including automated parameter selection and sweeps, execution of multiple assembly and validation tools, preliminary gene annotation, and identification of potential contaminating sequences. This ensemble approach is robust to individual tool failures and reliably generates high-quality assemblies with minimal user input.

Pipeline design

Figure 1 details the iMetAMOS pipeline, which treats each assembly project as a competition among multiple assemblers. As an added benefit to the ensemble approach, iMetAMOS is robust to failure; if any tool or parameter combination fails, iMetAMOS will continue the analysis using only the ones that succeeded. A "winning" assembly is then automatically selected based on the combined validation results. However, users may also browse the validation metrics to choose their own preferred assembly, such as one that optimizes consensus accuracy without regard to continuity.

iMetAMOS is primarily written in Python and builds upon Ruffus [31] for pipeline management. However, it incorporates many freely available tools written in a variety

Figure 1 iMetAMOS workflow and incorporated tools.
iMetAMOS currently incorporates 13 assemblers [21,33-38,40-45] and 7 validation tools [10-15,47]. Prokka [50] is used to predict genes and annotate all assemblies. Users can control the suite of assemblers and validation tools to be executed, as well as the scoring formula used to choose the best assembly. This assembly is evaluated for the presence of contamination.

of languages. To simplify installation, iMetAMOS is distributed as 64-bit OS X and Linux binaries, including all supported assemblers, tools, and required databases. On 32-bit systems, iMetAMOS automatically downloads and installs the required dependencies, as needed, which significantly simplifies installation.

Modular infrastructure

To support future extensibility, iMetAMOS includes a generic framework to add new tools to the pipeline. Currently supported modules are for assembly and classification. When a new tool is available, no code modification is need. Instead, a configuration is written to specify parameters for the tool and required inputs and outputs. iMetAMOS will automatically load this configuration and run the requested tool. When an external tool is executed, a corresponding citation is output to ensure users of iMetAMOS properly credit the tools on which it relies.

Reproducibility

iMetAMOS enables reproducible analysis by recording all commands, software versions (via an MD5 hash), and intermediate inputs and outputs. The single, comprehensive binary is generated via PyInstaller [32], which also serves to fix and archive the exact version of all programs used. Reproducibility of custom analyses is supported via workflows. A workflow defines the software required for an analysis, as well as optional parameters and input data. Workflows support both local and remote file names, as well as SRA run identifiers, and can inherit their parameters from other workflows, allowing users to easily add or modify input data or parameters. Given a workflow, iMetAMOS will download any required remote data and run the analysis using pre-specified parameters. For further reproducibility, a workflow is automatically created for every iMetAMOS run, which can be easily shared with remote collaborators. If the data are available on the Internet, the entire analysis can be reproduced by two simple commands.

Assembly

Assembly is treated as a hypothesis generation and testing problem. Multiple assembly tools are run to ensure robustness to failure and a thorough exploration of the hypothesis space. The following assemblers are currently supported: ABySS [33], CABOG [34], IDBA-UD [35], MaSuRCA [36], MetaVelvet [37], MIRA [38], Ray/RayMeta [39,40], SGA [41], SOAPdenovo2 [42], SPAdes [43], SparseAssembler [44], Velvet [21], and Velvet-SC [45]. For De Bruijn assemblers, a k-mer size is automatically selected using KmerGenie [46]. Alternatively, users can specify a list of k-mers and iMetAMOS will run each assembly with each specified k-mer. In this mode, iMetAMOS can operate

similarly to VelvetOptimizer [20], but for multiple assemblers and with more appropriate validation measures.

Validation and annotation

Each assembly is treated as a hypothesis subject to validation. The following validation tools are supported: ALE [10], CGAL [11], FRCbam [15], FreeBayes [47], LAP [14], QUAST [13], and REAPR [12]. Both reference-based and reference-free validations are performed. For reference-based validation, a MUMi distance [48] is used to recruit the most similar reference genome from RefSeq [49] to calculate reference-based metrics. For reference-free validation, the input reads and read pairs are verified to be in agreement with the resulting assembly using both likelihood-based methods and mis-assembly features. In addition, to provide an initial annotation and comparison between gene content, the assemblies are automatically annotated using Prokka [50].

From the ensemble, the "winning" assembly is selected using the consensus of the validation tools. For each selected metric, the assemblies are assigned an order from best to worst (with 1 being best). By default, the top assemblies are selected as those that are in the top 10% for at least half the metrics. The best assembly is then selected as the top scoring assembly with the highest count of best scores. A user can select a single metric (i.e. consensus accuracy) or an arbitrarily weighted combination of metrics for validation. Importantly, this allows users to customize the validation process to suit their downstream project goals. For example, studies focused on phylogenetic tree reconstruction may prefer to minimize consensus errors, while structural variation studies may instead focus on maximizing continuity and minimizing long-range errors.

Contamination detection

Although iMetAMOS focuses on single-genome assembly, all inputs are considered as a metagenome to control against possible contamination. The winning assembly's contigs and unassembled reads are analyzed by a taxonomic classification program. By default, iMetAMOS uses the k-mer based Kraken [51] tool, but the alternative methods of FCP [52], PhyloSift [53], PHMMER [54], and PhymmBL [55] are also supported. Contigs are partitioned into separate, taxon-specific directories (genus by default) according to their classification, so that contaminating sequence can be easily identified and removed. This process also serves as an initial species identification when assembling novel organisms.

The classification result is dependent on the classifier and database used, and serves as only a preliminary species identification or indicator of potential contamination. Manual follow-up is recommended to confirm the classification. For example, recently acquired genomic

elements, such as phage integrations, may be incorrectly classified. Nevertheless, this initial binning facilitates rapid identification of the assembled organism and easier contaminant removal before downstream analysis or submission to a nucleotide archive.

Results display
The final output of iMetAMOS is a self-contained HTML5 summary page. Here, users can browse the output files as well as drill down to detailed results from any step in the pipeline. This includes FastQC [56] reports for the pre-process step, QUAST [13] graphs and metrics from the validation step, and an interactive Krona [57] display of the taxonomic classifications.

Results

Automated assembly evaluation
With iMetAMOS it is possible to automatically recreate an assembler evaluation for every sample. We used iMetAMOS to perform ensemble assembly of the *Rhodobacter sphaeroides* 2.4.1 MiSeq dataset from the recent GAGE-B evaluation [8]. In addition, our automated evaluation included four additional assemblers (IDBA-UD [35], SparseAssembler [44], Velvet-SC [45], and Ray [39]) and validation metrics not utilized by GAGE-B (e.g. consensus accuracy).

The GAGE-B evaluation included assemblers run with multiple, manually selected k-mers for each assembler that ranged from a minimum of 31 to a maximum of 85. For a thorough comparison, the *R. sphaeroides* dataset was downloaded from the GAGE-B website (http://ccb.jhu.edu/gage_b/), and two iMetAMOS runs generated, both with an auto-selected k of 35 as well as all k-mer sizes divisible by 5 from 25–125. The original GAGE scripts [7] were used to calculate both corrected and raw N50s on all iMetAMOS assemblies and those from GAGE-B. In some cases, iMetAMOS ran a more recent assembler than GAGE-B. Repeating the experiment using the exact software versions as GAGE-B produced the same results, with the exception of SPAdes, which showed assembly improvement in version 2.5. The entire auto-selected k-mer iMetAMOS run can be reproduced using the following commands:

```
> initPipeline –d test_gageb –W
isolate_gageb
> runPipeline –d test_gageb –p 16
```

The best assembly of this dataset, as selected by iMetAMOS with an automatically chosen kmer-size of 35, was MaSuRCA, matching the GAGE-B result. However, the corrected N50 of the iMetAMOS MaSuRCA assembly increased to 139 Kbp from the 120 Kbp using the
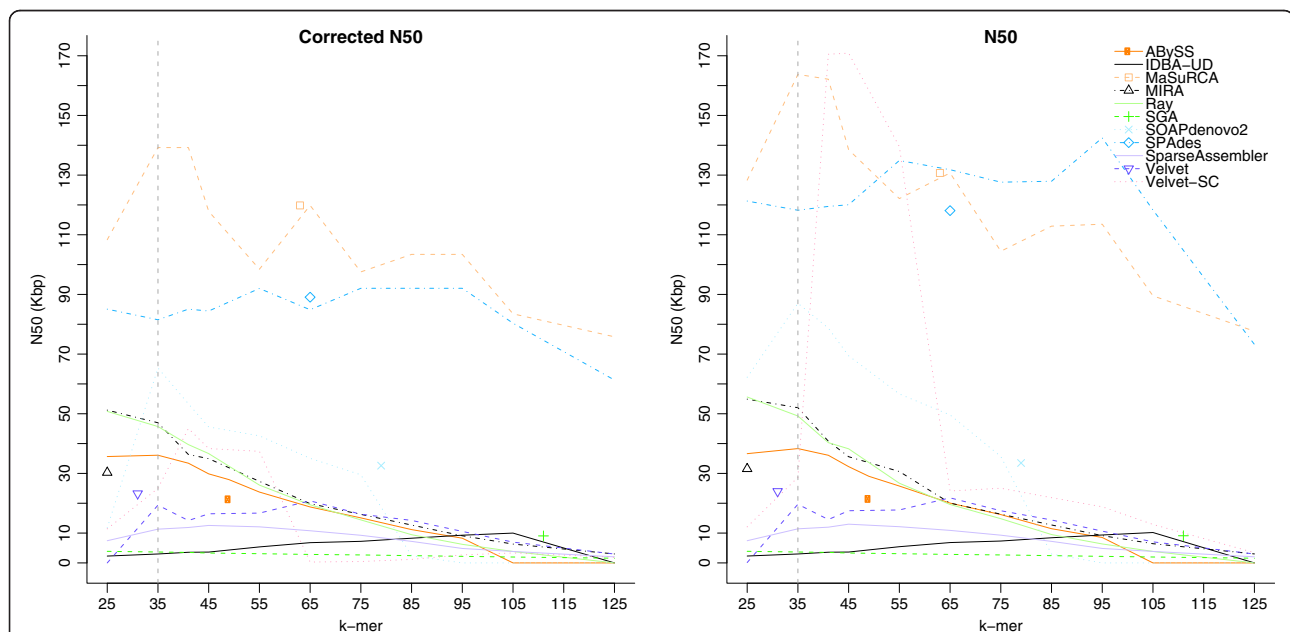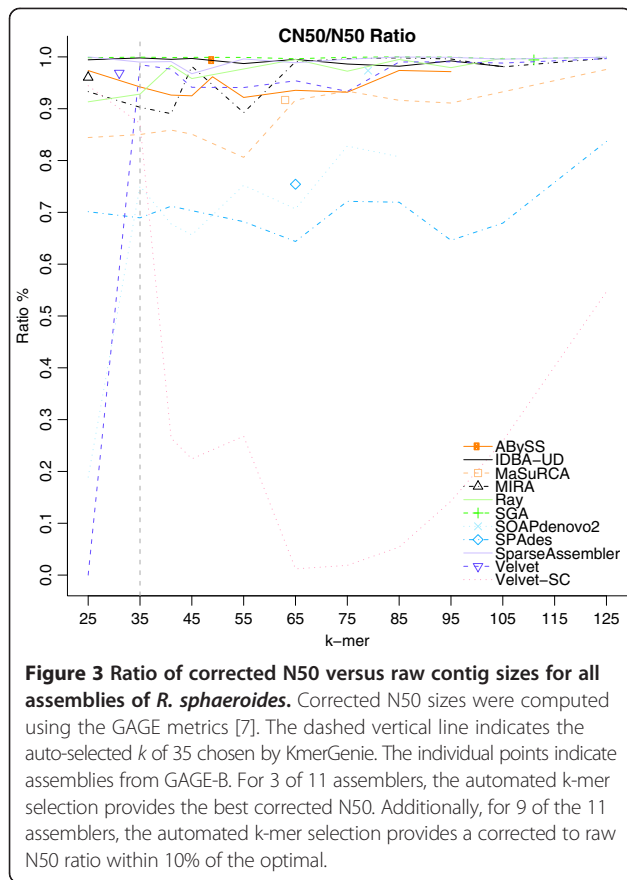

Figure 2 Comparison of corrected and raw N50 contig sizes for all assemblies of *R. sphaeroides*. Corrected N50 sizes were computed using the GAGE metrics [7]. The dashed vertical line indicates the auto-selected k of 35 chosen by KmerGenie. The individual points indicate assemblies from GAGE-B. The auto-selected k-mer provides the best overall corrected N50 on this dataset. One notable exception is SPAdes, for which the auto-selected k produced an N50 13% lower than the best. This is likely caused by SPAdes use of multiple k-mers for assembly, something that KmerGenie does not currently take into account. In all other cases, except when GAGE-B used EA-UTILS [58] to trim the input sequences, the automatically selected k-mer outperforms the k-mer choice from GAGE-B.

Figure 3 Ratio of corrected N50 versus raw contig sizes for all assemblies of *R. sphaeroides*. Corrected N50 sizes were computed using the GAGE metrics [7]. The dashed vertical line indicates the auto-selected *k* of 35 chosen by KmerGenie. The individual points indicate assemblies from GAGE-B. For 3 of 11 assemblers, the automated k-mer selection provides the best corrected N50. Additionally, for 9 of the 11 assemblers, the automated k-mer selection provides a corrected to raw N50 ratio within 10% of the optimal.

using raw sequencing data generated the best assemblies in GAGE-B, so pre-processing was disabled.

Figure 2 shows the raw and corrected N50s for all assemblers with *k* ranging from 25–125. This example illustrates the power of iMetAMOS, with each run being equivalent to a GAGE-style assembly evaluation. In addition, for 9 of 11 assemblers, the KmerGenie auto-selected *k* of 35 provided a ratio of corrected to raw N50 contig size within 10% of the optimal (Figure 3). Because the corrected assemblies are broken at each mis-assembly, the ratio of corrected to raw N50 size is a good indicator of error rate. Thus, these results suggest that simply running each assembler with an auto-selected value of *k* will produce high-quality assemblies without the computational expense of producing separate assemblies for multiple values of *k*.

The benefit of intelligently choosing a minimum overlap length (for overlap-based assemblers) or a value of *k* (for de Bruijn assemblers) is also obvious from Figure 2, as it changes the relative rankings of assemblies compared to the GAGE-B evaluation. In one example, MIRA goes from a corrected N50 of 15.19Kbp (ranked 6th) to 46.97Kbp (ranked 4th)—an increase of over 3-fold. The comparative ranking of assemblers on this dataset is given in Table 1.

Contaminant detection

The detection and removal of contaminating DNA sequences is an often-overlooked phase of assembly. For example, the Assemblathon 1 dataset included mock contaminant, which only a few teams attempted to detect and remove [6]. Failure to remove real contaminant from assemblies significantly affects the quality of public databases to which these genomes are submitted.

To assist in contaminant detection and removal, iMetAMOS supports multiple tools that taxonomically classify the assembled contigs and reads left unassembled. To test this process on a large scale, we downloaded and analyzed the raw sequencing data from 225 samples from a recently published study of *Mycobacterium tuberculosis*

manually selected k-mer size of 55 reported in GAGE-B. Similar improvements were observed for four assemblers when compared to the manually selected k-mer in GAGE-B. This improvement is the result of selecting a value of *k* to maximize assembly correctness, rather than the GAGE-B approach of maximizing the uncorrected contig N50 size. In cases where iMetAMOS did not outperform the GAGE-B results, GAGE-B had utilized EA-UTILS [58] to pre-process the data. While EA-UTILS is supported by iMetAMOS,

Table 1 GAGE-B assemblies versus iMetAMOS (iMA) assemblies on the *R. sphaeroides* dataset

Assembler	GAGE-B rank	GAGE-B N50	GAGE-B CN50	GAGE-B reference coverage	GAGE-B #short-range	GAGE-B #long-range	iMA rank	iMA N50	iMA CN50	iMA reference coverage	iMA #short-range	iMA #long-range
ABySS	5	21,441	21,307	99.64%	422	2	5	38,322	36,101	99.66%	674	0
MIRA	6	15,792	15,190	99.78%	424	15	4	52,034	46,977	99.84%	420	5
MaSuRCA	1	130,714	119,839	92.31%	532	2	1	163,762	139,231	92.95%	535	5
SGA	7	9,108	9,055	99.39%	295	0	7	3,657	3,655	91.96%	460	3
SOAPdenovo2	3	33,491	32,605	98.82%	343	1	3	87,036	65,337	99.86%	432	5
SPAdes	2	118,093	89,065	99.95%	605	1	2	118,214	81,505	99.76%	526	4
Velvet	4	23,979	23,230	99.10%	760	5	6	19,652	19,355	99.71%	1191	4

The table lists the relative ranking, assembly, and corrected N50 for each of the 7 assemblers common between GAGE-B and the iMetAMOS run using an automatically-selected k-mer. Assembly N50 was computed using the reference as the true genome size. Corrected N50 (CN50) was calculated as in GAGE [7]. The short-range errors are a sum of SNPs and short and long indels. The long-range errors are a sum of translocations, relocations, and inversions.

Table 2 iMetAMOS average runtime on 225 *M. tuberculosis* samples

Step	Average time (h)
Assemble	4.10
Validate	4.19
FindORFS	9.09
Classify	8.26
Total	32.25

The table lists the average run per sample for the longest-running steps within iMetAMOS. Timings were collected on a 16-core 2.4GHz AMD with 32G RAM. Full runtime details are available in Additional file 2.

[59]. All 225 runs corresponding to project ERP001731 were downloaded, and iMetAMOS was run using the ensemble of SPAdes, MaSuRCA, and Velvet. The iMetAMOS run for an individual sample can be reproduced using the following commands, which downloads the data directly from the SRA:

```
> initPipeline —d test_TB —W iMetAMOS —m
< RUN IDENTIFIER > —i 200:800
> runPipeline —d test_TB —a spades,
masurca,velvet —p 16
```

This also represents the first assembly of these samples, as the original publication focused only on read mapping (which is less affected by contamination issues). The dataset included paired-end and single-end data and sequence length ranged from 51 bp to 108 bp. The average number of contigs using unpaired data was 660, N50 was 17Kbp, and average k-mer size used was 26. For paired-end data, the average number of contigs was 333, N50 was 43Kbp, and average k-mer size used was 35. The SPAdes assembly was the best 88% of the time (Additional file 1). On average, an iMetAMOS analysis of a single sample took 32.25 hours using 16 cores. The Assembly, Validation, and FindORFS steps took approximately 50% of the runtime. Classification to identify contaminant took approximately 34% of the runtime (Table 2, Additional file 2).

While the majority of samples (218 of 225) indicated no contamination, several showed signs of having multiple organisms present, such as sample ERR233356. In this case, the assembly has significantly more bases than expected (6.24 Mbp versus 4.39 Mbp) and an unusually large number of contigs (>3,700). Figure 4 shows the HTML summary for sample ERR233356, with the validation tab selected. This tab provides information on the selected assembly, quality statistics for all assemblies, as well as an indication of contamination in the sample. Figure 5 summarizes the iMetAMOS classification output for sample ERR233356. *M. tuberculosis* is clearly
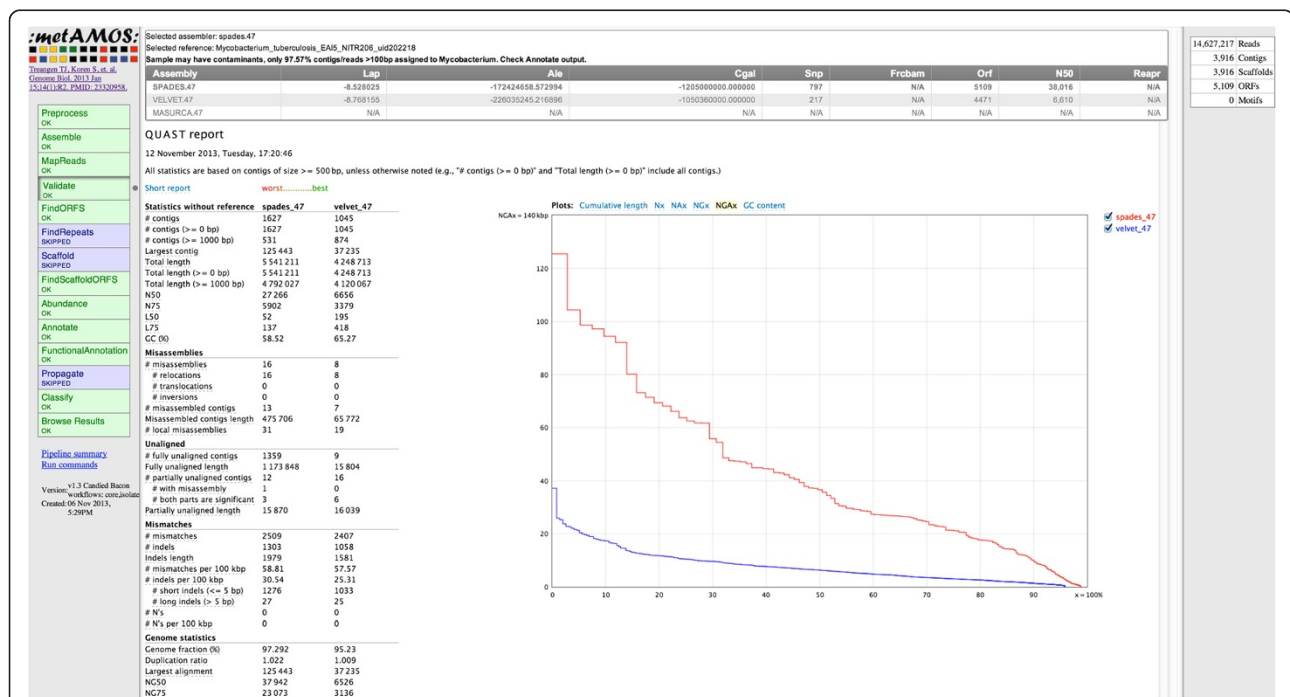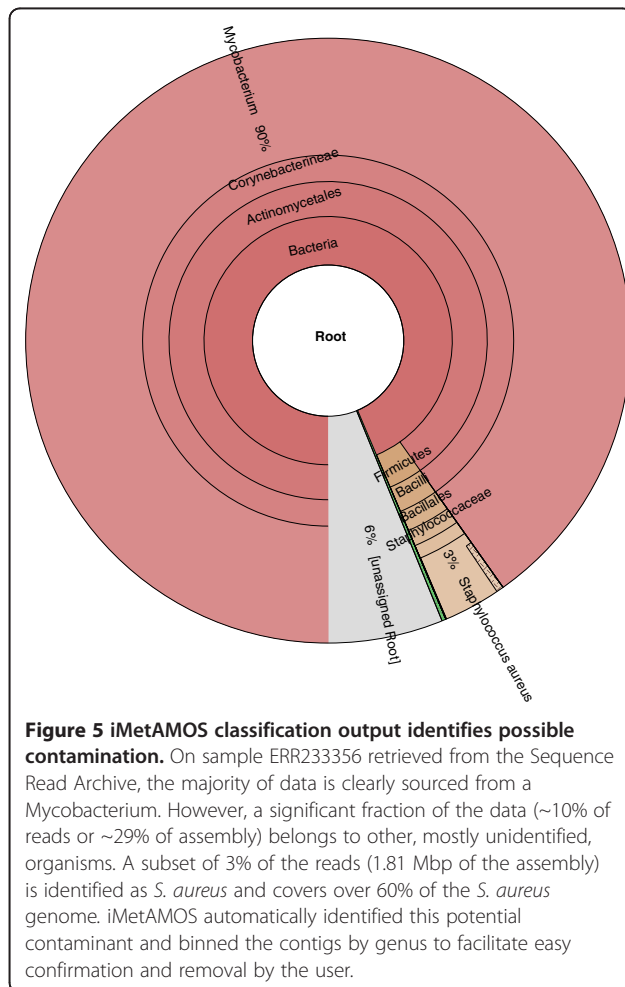

Figure 4 iMetAMOS validation output for an example dataset. The leftmost tab allows navigation to view the output of each pipeline step. The selected "Validation" tab results are shown in the main window. These include the validation metrics of all successful assemblies, including a comparison and QUAST [13] report against an automatically recruited reference genome from NCBI RefSeq. The validation tab also indicates the sample shows signs of contamination. Finally, the rightmost tab shows a quick summary of the winning assembly (# reads, #contigs, # orfs). MaSuRCA did not run on this sample because it requires paired-end input.

Figure 5 iMetAMOS classification output identifies possible contamination. On sample ERR233356 retrieved from the Sequence Read Archive, the majority of data is clearly sourced from a Mycobacterium. However, a significant fraction of the data (~10% of reads or ~29% of assembly) belongs to other, mostly unidentified, organisms. A subset of 3% of the reads (1.81 Mbp of the assembly) is identified as *S. aureus* and covers over 60% of the *S. aureus* genome. iMetAMOS automatically identified this potential contaminant and binned the contigs by genus to facilitate easy confirmation and removal by the user.

the dominant constituent, but a significant fraction of sequences are assigned to other bacteria or are left unclassified (because the Kraken classification database used included only microbial genomes, eukaryotic contamination appears as "unclassified"). To validate the iMetAMOS output, we aligned the classified sequences to *M. tuberculosis* NITR206 (4.39 Mbp) and *S. aureus* USA300 (2.87 Mbp) using dnadiff [9]. Sequences classified as Mycobacterium had an average identity of 99.87% and 98.93% coverage of the reference. Sequences classified as Staphylococus had an average identity of 99.85% and 61.65% coverage. Over half the *S. aureus* reference genome is present in the dataset, confirming the contamination as the most likely source of these reads. The difference in reference coverage between all assembled contigs and classified contigs was less than 1%, indicating the classifier accurately assigned contigs. All 7 *M. tuberculosis* samples identified by iMetAMOS as potentially contaminanted were manually examined and had similarly large assemblies with an unusual number of sequences classified as either microbial or unknown. Using BLAST [60] confirmed both eukaryotic (human) and microbial contaminant in these samples.

Discussion

We have developed an open-source microbial analysis pipeline, iMetAMOS, which automates the process of ensemble assembly. In addition, its modular architecture is extensible and able to incorporate additional analyses or alternative tools. A potential enhancement is assembly correction, or contig breaking, which iMetAMOS does not currently support. However, the infrastructure required to support this is largely in place. For example, REAPR [12] is included with iMetAMOS and capable of splitting assembled contigs at predicted mis-assemblies. Using this and other supplied validation tools, assembly breaking could be iteratively performed until the validation scores are no longer improving or no more corrections are possible. Alternatively, because iMetAMOS generates multiple assemblies, assembly reconciliation techniques [61-63] could be incorporated into the pipeline. However, in practice, we have found the simple process of running multiple assemblers with multiple parameters is capable of generating high-confidence assemblies on its own, while merging assemblies can increase the risk of mis-assembly without significantly improving continuity [8].

The iMetAMOS extensible framework also supports customizable workflows and parameters on a per-user or per-run basis. Because all components of iMetAMOS are open source, users and tool authors are able to contribute improved parameters to the repository. Users can also contribute custom workflows tailored for specific analyses. In this way, iMetAMOS can serve as a best-practice repository for multiple assemblers, data types, and analysis tools.

Conclusions

iMetAMOS enables accurate and reproducible genome assembly via a "GAGE-in-a-box" analysis, allowing non-expert users to run multiple assemblers, validation metrics, and annotations with a single command. Results are presented in a simplified and interactive HTML5 format, and reproducibility is enabled through detailed logging and workflows. The current implementation supports over thirteen assemblers and seven validation tools, and its modular architecture supports the easy addition of future tools. Ensemble assembly is more robust, reproducible, and accurate than manual assembly, even surpassing the quality of GAGE-B assemblies using the same data and tools. Most importantly, iMetAMOS provides users with a simple means to generate multiple assemblies and validation metrics, empowering them to choose the best assembly for their specific needs.

Availability and requirements

Project name: iMetAMOS.
Project home page: http://www.cbcb.umd.edu/software/ imetamos.

Operating systems: Linux/OS X.
Programming language: Python, C++, Perl, and Java.
Other requirements: Perl (5.8.8+), Python (2.7.3+), Java (1.6+), R (2.11.1+ with PNG support), gcc (4.7+ recommended), git, curl.
License: iMetAMOS and metAMOS-specific code are released open source under the Perl Artistic License [64].

All assemblies described here are available for download from http://www.cbcb.umd.edu/software/imetamos. The exact version of iMetAMOS used for analysis in this manuscript is available from ftp://ftp.cbcb.umd.edu/pub/data/metamos/imetamos_pub.tar.gz. However, we recommend using the latest release for all analyses.

Additional files

Additional file 1: Assembly statistics and reference-free validation scores for all assemblers used on the *M. tuberculosis* dataset.

Additional file 2: Computational time for the longest-running steps within iMetAMOS as well as total times per sample for the *M. tuberculosis* dataset.

Competing interests
The authors declare that they have no competing interest.

Authors' contributions
SK and AMP conceived the method and drafted the manuscript. SK, TJT, and CMH developed the software. MP led the development of the MetAMOS framework. All authors edited and approved the final manuscript.

Author details
[1]National Biodefense Analysis and Countermeasures Center, 110 Thomas Johnson Drive, Frederick, MD 21702, USA. [2]Center for Bioinformatics and Computational Biology, University of Maryland, College Park, MD 20742, USA.

References
1. Miller JR, Koren S, Sutton G: **Assembly algorithms for next-generation sequencing data.** *Genomics* 2010, **95**(6):315–327.
2. Nagarajan N, Pop M: **Parametric complexity of sequence assembly: theory and applications to next generation sequencing.** *J Comput Biol* 2009, **16**(7):897–908.
3. Nagarajan N, Pop M: **Sequence assembly demystified.** *Nat Rev Genet* 2013, **14**(3):157–167.
4. Myers EW: **Toward simplifying and accurately formulating fragment assembly.** *J Comput Biol* 1995, **2**(2):275–290.
5. 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: **Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species.** *GigaScience* 2013, **2**(1):10.
6. Earl D, Bradnam K, St John J, Darling A, Lin D, Fass J, Yu HO, Buffalo V, Zerbino DR, Diekhans M, Nguyen N, Ariyaratne PN, Sung WK, 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, *et al*: **Assemblathon 1: a competitive assessment of de novo short read assembly methods.** *Genome Res* 2011, **21**(12):2224–2241.
7. Salzberg SL, Phillippy AM, Zimin A, Puiu D, Magoc T, Koren S, Treangen TJ, Schatz MC, Delcher AL, Roberts M: **GAGE: a critical evaluation of genome assemblies and assembly algorithms.** *Genome Res* 2012, **22**(3):557–567.
8. Magoc T, Pabinger S, Canzar S, Liu X, Su Q, Puiu D, Tallon LJ, Salzberg SL: **GAGE-B: an evaluation of genome assemblers for bacterial organisms.** *Bioinformatics* 2013, **29**(14):1718–1725.
9. Phillippy AM, Schatz MC, Pop M: **Genome assembly forensics: finding the elusive mis-assembly.** *Genome Biol* 2008, **9**(3):R55–R55.
10. Clark SC, Egan R, Frazier PI, Wang Z: **ALE: a generic assembly likelihood evaluation framework for assessing the accuracy of genome and metagenome assemblies.** *Bioinformatics* 2013, **29**(4):435–443.
11. Rahman A, Pachter L: **CGAL: computing genome assembly likelihoods.** *Genome Biol* 2013, **14**(1):R8.
12. Hunt M, Kikuchi T, Sanders M, Newbold C, Berriman M, Otto TD: **REAPR: a universal tool for genome assembly evaluation.** *Genome Biol* 2013, **14**(5):R47.
13. Gurevich A, Saveliev V, Vyahhi N, Tesler G: **QUAST: quality assessment tool for genome assemblies.** *Bioinformatics* 2013, **29**(8):1072–1075.
14. Ghodsi M, Hill CM, Astrovskaya I, Lin H, Sommer DD, Koren S, Pop M: **De novo likelihood-based measures for assembly validation.** *BMC Res Notes* 2013, **6**(1):334.
15. Vezzi F, Narzisi G, Mishra B: **Reevaluating assembly evaluations with feature response curves: GAGE and assemblathons.** *PLoS One* 2012, **7**(12):e52210.
16. Howison M, Zapata F, Dunn CW: **Toward a statistically explicit understanding of de novo sequence assembly.** *Bioinformatics* 2013, **29**(23):2959–2963.
17. Tritt A, Eisen JA, Facciotti MT, Darling AE: **An integrated pipeline for de novo assembly of microbial genomes.** *PLoS One* 2012, **7**(9):e42304.
18. Coil D, Jospin G, Darling AE: **A5-miseq: an updated pipeline to assemble microbial genomes from Illumina MiSeq data.** arXiv preprint arXiv:1401.5130 2014.
19. Kislyuk AO, Katz LS, Agrawal S, Hagen MS, Conley AB, Jayaraman P, Nelakuditi V, Humphrey JC, Sammons SA, Govil D, Mair RD, Tatti KM, Tondella ML, Harcourt BH, Mayer LW, Jordan IK: **A computational genomics pipeline for prokaryotic sequencing projects.** *Bioinformatics* 2010, **26**(15):1819–1826.
20. Velvet Optimizer: http://bioinformatics.net.au/software.velvetoptimiser.shtml.
21. Zerbino DR, Birney E: **Velvet: algorithms for de novo short read assembly using de bruijn graphs.** *Genome Res* 2008, **18**(5):821–829.
22. Narzisi G, Mishra B: **Comparing de novo genome assembly: the long and short of it.** *PLoS One* 2011, **6**(4):17–17.
23. Medvedev P, Brudno M: **Maximum likelihood genome assembly.** *J Comput Biol* 2009, **16**(8):1101–1116.
24. Laserson J, Jojic V, Koller D: **Genovo: de novo assembly for metagenomes.** *J Comp Biol J Comp Mol Cell Biol* 2011, **18**(3):429–443.
25. Hayati A, Sato K, Sakakibara Y: **An extended genovo metagenomic assembler by incorporating paired-end information.** *PeerJ* 2013, **1**:e196.
26. Treangen TJ, Koren S, Sommer DD, Liu B, Astrovskaya I, Ondov B, Darling AE, Phillippy AM, Pop M: **MetAMOS: a modular and open source metagenomic assembly and analysis pipeline.** *Genome Biol* 2013, **14**(1):R2.
27. Bentley DR, Balasubramanian S, Swerdlow HP, Smith GP, Milton J, Brown CG, Hall KP, Evers DJ, Barnes CL, Bignell HR, Boutell JM, Bryant J, Carter RJ, Keira Cheetham R, Cox AJ, Ellis DJ, Flatbush MR, Gormley NA, Humphray SJ, Irving LJ, Karbelashvili MS, Kirk SM, Li H, Liu X, Maisinger KS, Murray LJ, Obradovic B, Ost T, Parkinson ML, Pratt MR, *et al*: **Accurate whole human genome sequencing using reversible terminator chemistry.** *Nature* 2008, **456**(7218):53–59.

28. Margulies M, Egholm M, Altman WE, Attiya S, Bader JS, Bemben LA, Berka J, Braverman MS, Chen Y-J, Chen Z, Dewell SB, Du L, Fierro JM, Gomes XV, Godwin BC, He W, Helgesen S, Ho CH, Irzyk GP, Jando SC, Alenquer MLI, Jarvie TP, Jirage KB, Kim J-B, Knight JR, Lanza JR, Leamon JH, Lefkowitz SM, Lei M, Li J, *et al*: Genome sequencing in microfabricated high-density picolitre reactors. *Nature* 2005, **437**(7057):376–380.

29. Rothberg JM, Hinz W, Rearick TM, Schultz J, Mileski W, Davey M, Leamon JH, Johnson K, Milgrew MJ, Edwards M, Hoon J, Simons JF, Marran D, Myers JW, Davidson JF, Branting A, Nobile JR, Puc BP, Light D, Clark TA, Huber M, Branciforte JT, Stoner IB, Cawley SE, Lyons M, Fu Y, Homer N, Sedova M, Miao X, Reed B, *et al*: An integrated semiconductor device enabling non-optical genome sequencing. *Nature* 2011, **475**(7356):348–352.

30. Eid J, Fehr A, Gray J, Luong K, Lyle J, Otto G, Peluso P, Rank D, Baybayan P, Bettman B, Bibillo A, Bjornson K, Chaudhuri B, Christians F, Cicero R, Clark S, Dalal R, Dewinter A, Dixon J, Foquet M, Gaertner A, Hardenbol P, Heiner C, Hester K, Holden D, Kearns G, Kong X, Kuse R, Lacroix Y, Lin S, *et al*: Real-time DNA sequencing from single polymerase molecules. *Science* 2009, **323**(5910):133–138.

31. Goodstadt L: Ruffus: a lightweight python library for computational pipelines. *Bioinformatics* 2010, **26**(21):2778–2779.

32. PyInstaller: http://www.pyinstaller.org/.

33. Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJM, Birol I: ABySS: a parallel assembler for short read sequence data. *Genome Res* 2009, **19**(6):1117–1123.

34. Miller JR, Delcher AL, Koren S, Venter E, Walenz BP, Brownley A, Johnson J, Li K, Mobarry C, Sutton G: Aggressive assembly of pyrosequencing reads with mates. *Bioinformatics* 2008, **24**(24):2818–2824.

35. Peng Y, Leung HC, Yiu SM, Chin FY: IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics* 2012, **28**(11):1420–1428.

36. Zimin AV, Marcais G, Puiu D, Roberts M, Salzberg SL, Yorke JA: The MaSuRCA genome assembler. *Bioinformatics* 2013, **29**(21):2669–2677.

37. Namiki T, Hachiya T, Tanaka H, Sakakibara Y: MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads. *Nucleic Acids Res* 2012, **40**(20):e155.

38. Chevreux B, Pfisterer T, Drescher B, Driesel AJ, Muller WE, Wetter T, Suhai S: Using the miraEST assembler for reliable and automated mRNA transcript assembly and SNP detection in sequenced ESTs. *Genome Res* 2004, **14**(6):1147–1159.

39. Boisvert S, Laviolette F, Corbeil J: Ray: simultaneous assembly of reads from a mix of high-throughput sequencing technologies. *J Comput Biol* 2010, **17**(11):1519–1533.

40. Boisvert S, Raymond F, Godzaridis E, Laviolette F, Corbeil J: Ray meta: scalable de novo metagenome assembly and profiling. *Genome Biol* 2012, **13**(12):R122.

41. Simpson JT, Durbin R: Efficient de novo assembly of large genomes using compressed data structures. *Genome Res* 2012, **22**(3):549–556.

42. Luo R, Liu B, Xie Y, Li Z, Huang W, Yuan J, He G, Chen Y, Pan Q, Liu Y, Tang J, Wu G, Zhang H, Shi Y, Liu Y, Yu C, Wang B, Lu Y, Han C, Cheung DW, Yiu SM, Peng S, Xiaoqian Z, Liu G, Liao X, Li Y, Yang H, Wang J, Lam TW, Wang J: SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience* 2012, **1**(1):18.

43. 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: SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *J Comput Biol* 2012, **19**(5):455–477.

44. Ye C, Ma ZS, Cannon CH, Pop M, Yu DW: Exploiting sparseness in de novo genome assembly. *BMC Bioinforma* 2012, **13**(Suppl 6):S1.

45. Chitsaz H, Yee-Greenbaum JL, Tesler G, Lombardo MJ, Dupont CL, Badger JH, Novotny M, Rusch DB, Fraser LJ, Gormley NA, Schulz-Trieglaff O, Smith GP, Evers DJ, Pevzner PA, Lasken RS: Efficient de novo assembly of single-cell bacterial genomes from short-read data sets. *Nat Biotechnol* 2011, **29**(10):915–921.

46. Chikhi R, Medvedev P: Informed and automated k-mer size selection for genome assembly. *Bioinformatics* 2014, **30**(1):31–37.

47. Garrison E, Marth G: Haplotype-based variant detection from short-read sequencing. 2012, arXiv preprint arXiv:1207.3907.

48. Deloger M, El Karoui M, Petit MA: A genomic distance based on MUM indicates discontinuity between most bacterial species and genera. *J Bacteriol* 2009, **191**(1):91–99.

49. NCBI RefSeq: ftp://ftp.ncbi.nih.gov/genomes/Bacteria/all.fna.tar.gz.

50. Seemann T: Prokka: rapid prokaryotic genome annotation. *Bioinformatics* 2014:btu153.

51. Wood D, Salzberg S: Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol* 2014, **15**(3):R46.

52. Parks DH, MacDonald NJ, Beiko RG: Classifying short genomic fragments from novel lineages using composition and homology. *BMC Bioinforma* 2011, **12**(1):328–328.

53. Darling AE, Jospin G, Lowe E, Matsen FAIV, Bik HM, Eisen JA: PhyloSift: phylogenetic analysis of genomes and metagenomes. *PeerJ* 2014, **2**:e243.

54. Eddy SR: Accelerated profile HMM searches. *PLoS Comput Biol* 2011, **7**(10):e1002195–e1002195.

55. Brady A, Salzberg SL: Phymm and PhymmBL: metagenomic phylogenetic classification with interpolated markov models. *Nat Methods* 2009, **6**(9):673–676.

56. FastQC: A quality control tool for high throughput sequence data: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/.

57. Ondov BD, Bergman NH, Phillippy AM: Interactive metagenomic visualization in a web browser. *BMC Bioinforma* 2011, **12**(1):385–385.

58. Command-line tools for processing biological sequencing data: https://code.google.com/p/ea-utils/.

59. Comas I, Coscolla M, Luo T, Borrell S, Holt KE, Kato-Maeda M, Parkhill J, Malla B, Berg S, Thwaites G, Yeboah-Manu D, Bothamley G, Mei J, Wei L, Bentley S, Harris SR, Niemann S, Diel R, Aseffa A, Gao Q, Young D, Gagneux S: Out-of-Africa migration and Neolithic coexpansion of Mycobacterium tuberculosis with modern humans. *Nat Genet* 2013, **45**(10):1176–1182.

60. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ: Basic local alignment search tool. *J Mol Biol* 1990, **215**(3):403–410.

61. Vicedomini R, Vezzi F, Scalabrin S, Arvestad L, Policriti A: GAM-NGS: genomic assemblies merger for next generation sequencing. *BMC Bioinforma* 2013, **14**(Suppl 7):S6.

62. Yao G, Ye L, Gao H, Minx P, Warren WC, Weinstock GM: Graph accordance of next-generation sequence assemblies. *Bioinformatics* 2012, **28**(1):13–16.

63. Sommer DD, Delcher AL, Salzberg SL, Pop M: Minimus: a fast, lightweight genome assembler. *BMC Bioinforma* 2007, **8**(1):64–64.

64. Perl Artistic License: http://dev.perl.org/licenses/artistic.html.